State Institute of Engineering and Technology Nilokheri.

DIGITAL ELECTRONICS LAB MANUAL

Department of Electronics and Communication Engineering

List of Experiment

1. Study of Logic gates – AND, OR, NOT, NAND, NOR, EX-OR, EXNOR.
2. Design and realize a given function using K-Map andverify its performance.
3. To design and implement multiplexer and demultiplexer using logic gates and study of IC 74150 and IC 74154.
4. To design and implement of 2-bit and 8-bit magnitude comparator.
5. Verification of state tables of SR, JK, D flip-flops using NAND & nor gates.
6. To verify the operation of bi-directional shift register.
7. To design and implement 3 bit synchronous up/down counter.
8. Design and verification of the truth tables of Half and Full adder circuits
9. Design and verification of the truth tables of Half and Full Subtractor circuits.
10. To verify the NAND and NOR gates using universal logic gates.
11. To verify Binary to Gray Code Conversion.
12. To verify Gray to Binary Code Conversion.

Experiment No. -01

Aim: To study about logic gates and verify their truth tables.

Apparatus:

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

Theory:

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

1. AND GATE:

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

2. OR GATE:

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

3. NOT GATE:

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

4. NAND GATE:

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

5. NOR GATE:

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

6. X-OR GATE:

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

Procedure:

(i) Connections are given as per circuit diagram.

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

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

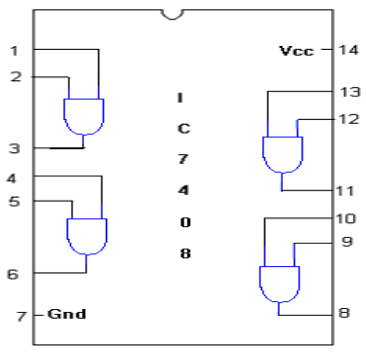
1. AND GATE :

SYMBOL:

TRUTH TABLE

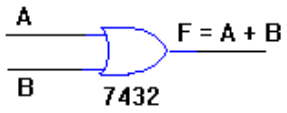
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

PIN DIAGRAM:

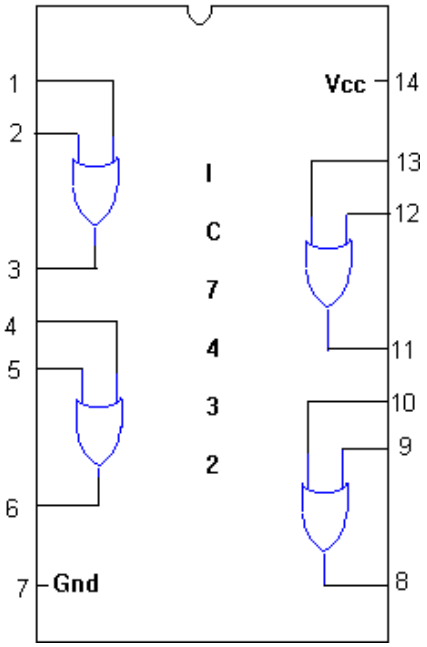


2. OR GATE:

SYMBOL :



PIN DIAGRAM :

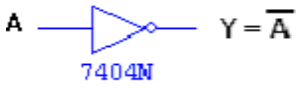


TRUTH TABLE

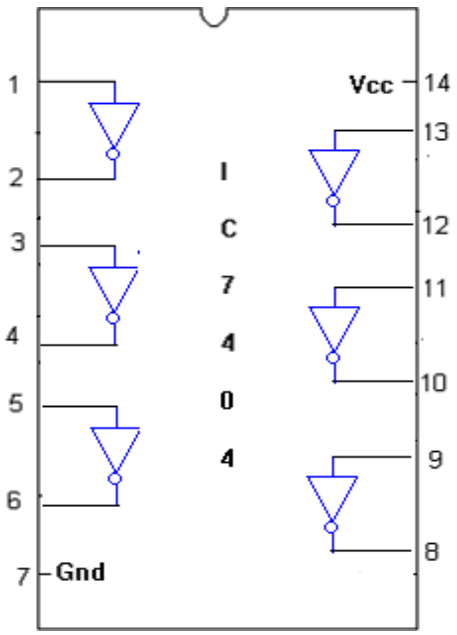
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT GATE:

SYMBOL:



PIN DIAGRAM:

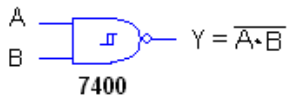


TRUTH TABLE :

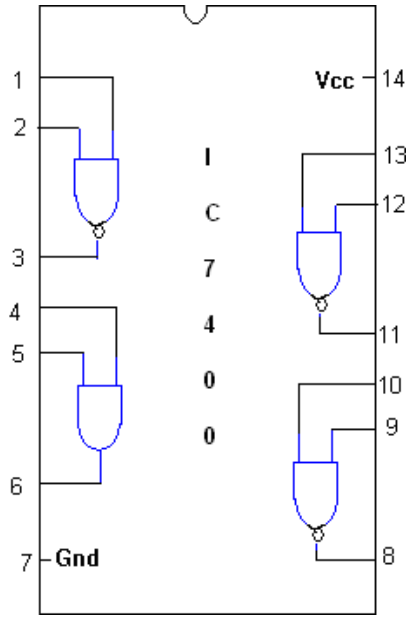
A	$\overline{A}$
0	1
1	0

4. NAND GATE:

SYMBOL:



PIN DIAGRAM

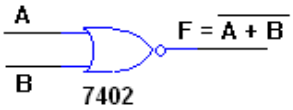


TRUTH TABLE

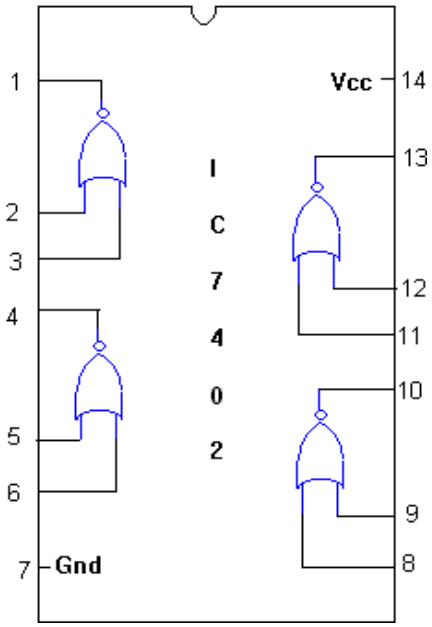
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR GATE:

SYMBOL :



PIN DIAGRAM :

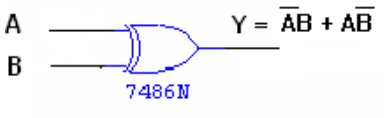


TRUTH TABLE

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

6. X-OR GATE :

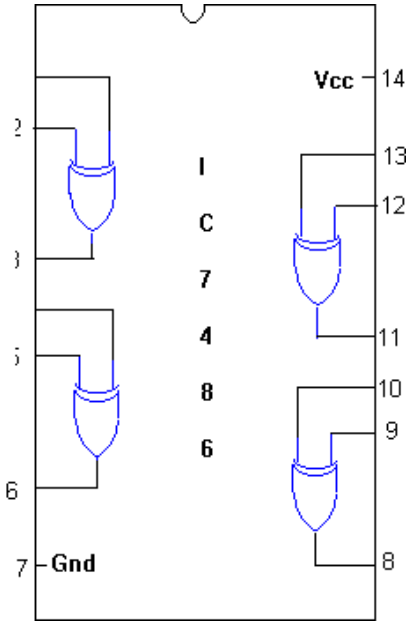
SYMBOL :



TRUTH TABLE :

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

PIN DIAGRAM :



Result:

All the logic gates verified successfully.

Experiment No.-02

Aim: Design and realize a given function using K-Map and verify its performance.

Appratus: IC's (7404,7408,7432), connecting wires, power supply.

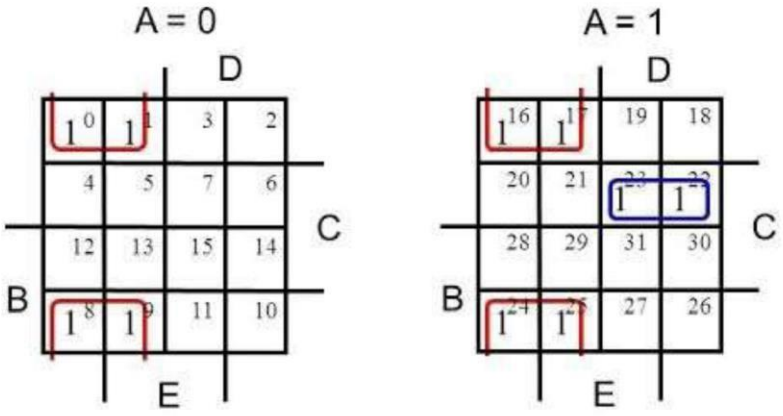
Theory:

A Karnaugh map (K-map) is a pictorial method used to minimize boolean expressions without having to use boolean algebra theorems and equation manipulations. A K-map can be thought of as a special version of a truth table. Using a K-map, expressions with two to four variables are easily minimized.

Canonical Forms (Normal Forms): Any Boolean function can be written in disjunctive normal 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.

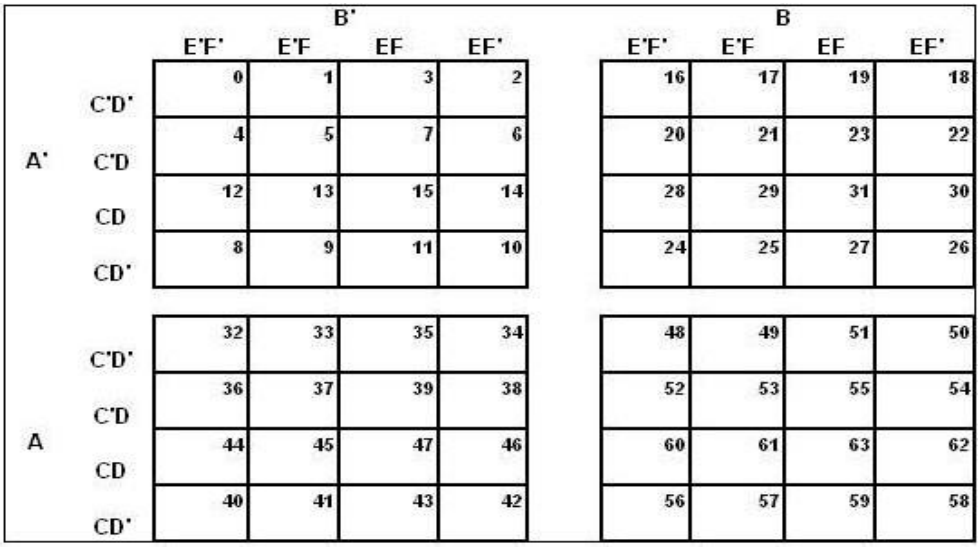
Example of a Five-Variable K-map

- $F(A, B, C, D, E) = \sum(0, 1, 8, 9, 16, 17, 22, 23, 24, 25)$
- $F = \overline{C}\,\overline{D} + A\,\overline{B}\,C\,D$



6-Variable K-Map

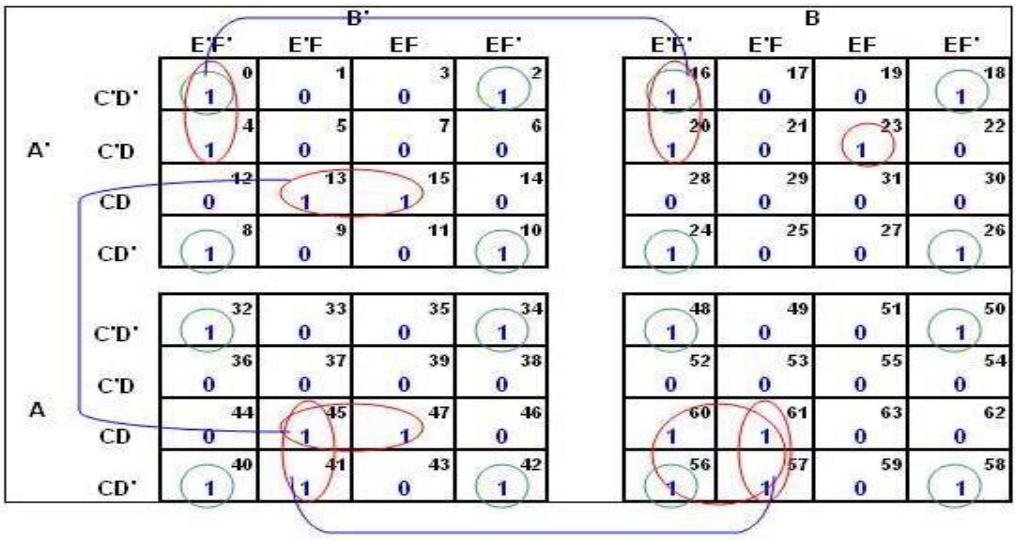
A 6-variable K-Map is drawn as below:



Given function,

F = Σ (0, 2, 4, 8, 10, 13, 15, 16, 18, 20, 23, 24, 26, 32, 34, 40, 41, 42, 45, 47, 48, 50, 56, 57, 58, 60, 61)

Let's draw K-Map for this function by writing 1 in cells that are present in function and 0 in rest of the cells.



Applying rules of simplifying K-Map , there is one loop which has 16 1's- containing 1's at all the corners of all 4 squares. We obtain it by visualizing all the 4 squares over one another but only in horizontal vertical directions and figuring its adjacent cells. All the 1's in corners

are circled in green. There are 4 pairs, one in fourth square at bottom-right and other 3 are between the squares and are highlighted by blue connecting line.

There are 4 pairs, one in fourth square at bottom-right and other 3 are between the squares and are highlighted by blue connecting line.

(0, 2, 8, 10, 16, 18, 24, 26, 32, 34, 40, 42, 48, 50, 56, 58) – D'F' (A, B, C and E are changing variables, so they are eliminated)

(41, 45, 57, 61) – ACE'F (B & D are changing variables, so they are eliminated)

(13, 15, 45, 47) – B'CDF (A & E are changing variables, so they are eliminated)

(0, 4, 16, 20) – A'C'E'F' (B & D are changing variables, so they are eliminated)

(56, 57, 60, 61) – ABCE' (D and F are changing variables, so they are eliminated)

There is 1 in cell 23, which cannot be looped with any adjacent cell, hence it cannot be simplified further and left as it is. 23 = A'BC'DEF

Thus, F = D'F' + ACE'F + B'CDF + A'C'E'F' + ABCE' + A'BC'DEF

Procedure:

(i) Connections are given as per circuit diagram.

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

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

Result:

This experiment perform successfully.

Experiment NO.-03

Aim : To design and implement multiplexer and demultiplexer using logic gates and study of IC 74150 and IC 74154.

Apparatus:

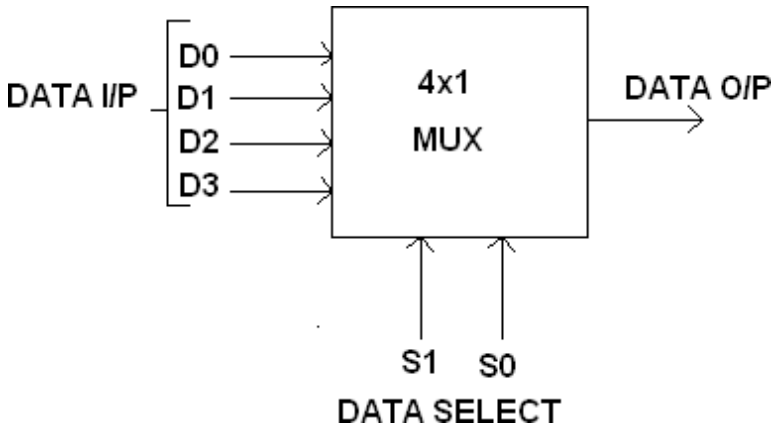
Sl.No.	COMPONENT	SPECIFICATION	QTY.
1.	3 I/P AND GATE	IC 7411	2
2.	OR GATE	IC 7432	1
3.	NOT GATE	IC 7404	1
2.	IC TRAINER KIT	-	1
3.	PATCH CORDS	-	32

Theory:

1. Multiplexer:

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

Block Diagram For 4:1 Multiplexer:

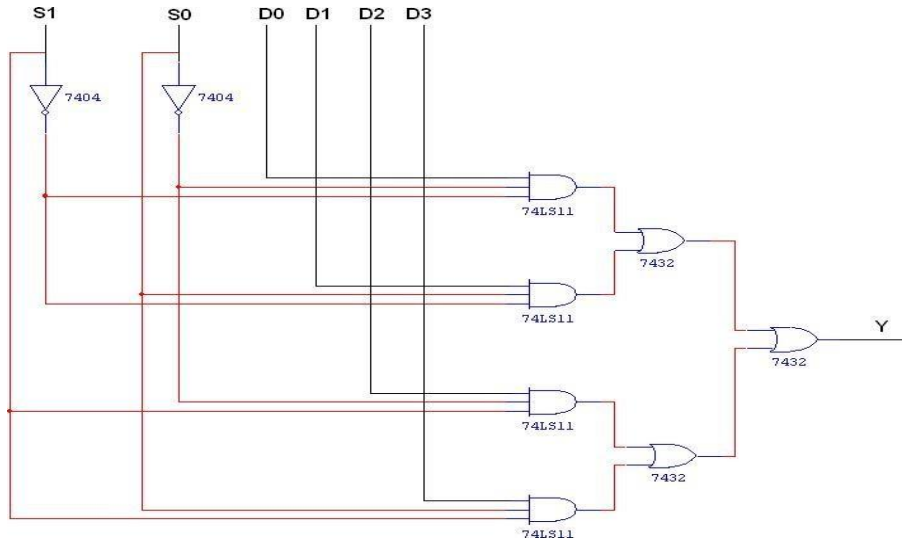


Function Table:

S1	S0	INPUTS Y
0	0	D0 → D0 S1' S0'
0	1	D1 → D1 S1' S0
1	0	D2 → D2 S1 S0'
1	1	D3 → D3 S1 S0

Y = D0 S1' S0' + D1 S1' S0 + D2 S1 S0' + D3 S1 S0

Circuit Diagram:



Truth Table:

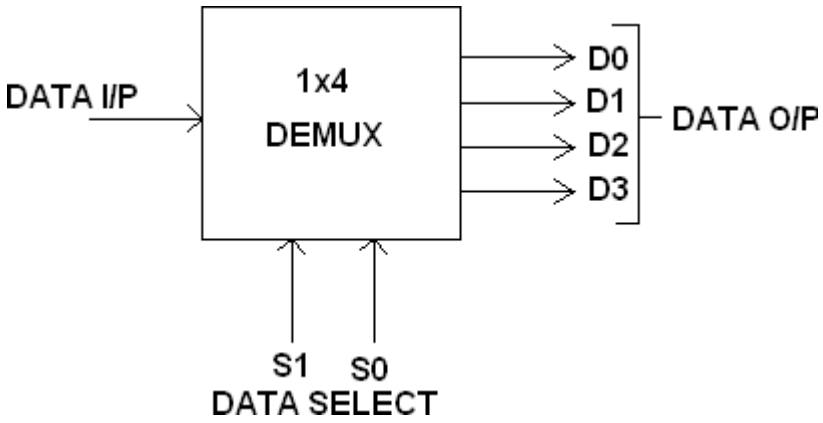
S1	S0	Y = OUTPUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

2. Demultiplexer:

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

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

Block Diagram For 1:4 Demultiplexer:

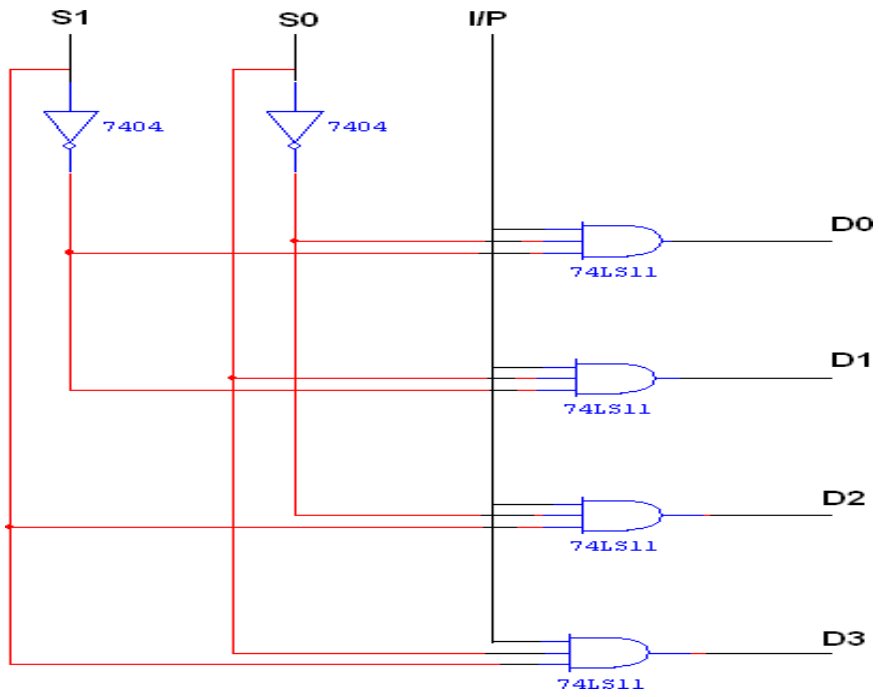


Function Table:

S1	S0	INPUT
0	0	X → D0 = X S1' S0'
0	1	X → D1 = X S1' S0
1	0	X → D2 = X S1 S0'
1	1	X → D3 = X S1 S0

Y = X S1' S0' + X S1' S0 + X S1 S0' + X S1 S0

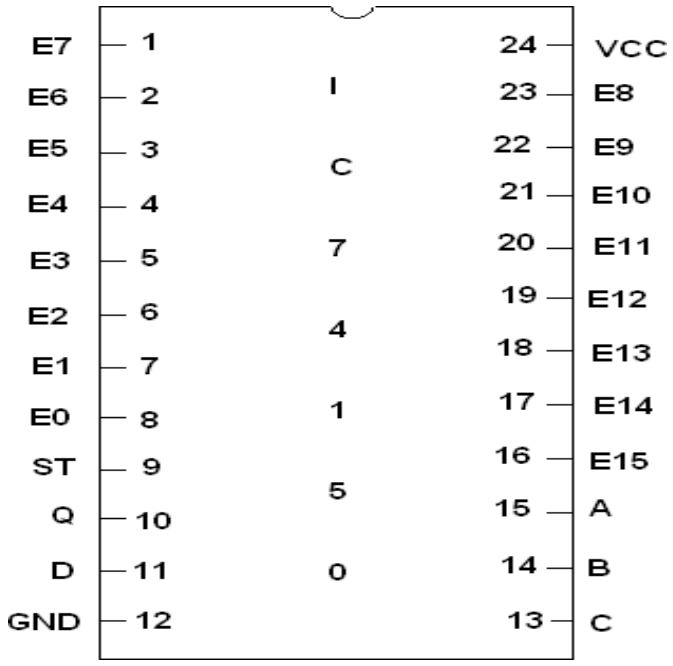
Logic Diagram Of Demultiplexer:



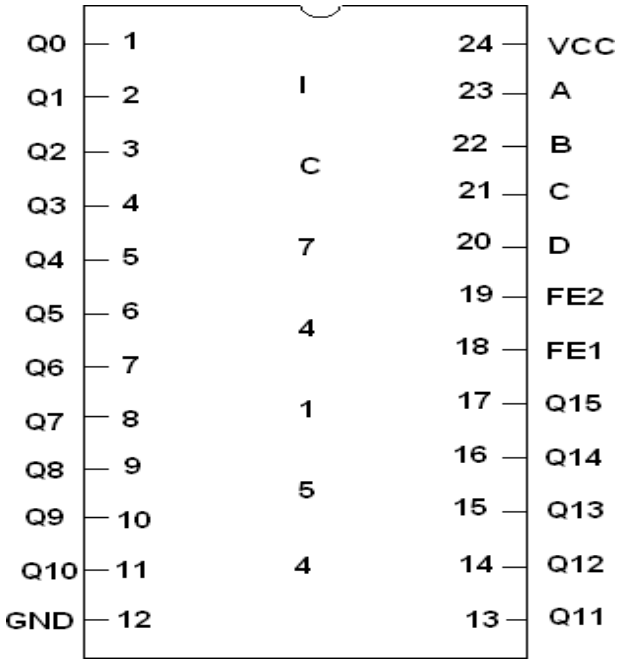
Truth Table:

INPUT			OUTPUT			
S1	**S0**	**I/P**	**D0**	**D1**	**D2**	**D3**
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

Pin Diagram For IC 74150(Multiplexer):



Pin Diagram For IC 74151 (Demultiplexer):



Procedure:

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

Result:

This experiment performed successfully.

Experiment No.04

Aim: To design and implement

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

(ii) 8 – bit magnitude comparator using IC 7485

Apparatus:

Sr. No.	COMPONENT	SPECIFICATION	QTY.
1.	AND GATE	IC 7408	2
2.	X-OR GATE	IC 7486	1
3.	OR GATE	IC 7432	1
4.	NOT GATE	IC 7404	1
5.	4-BIT MAGNITUDE COMPARATOR	IC 7485	2
6.	IC TRAINER KIT	-	1
7.	PATCH CORDS	-	30

Theory:

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

$A = A_3\ A_2\ A_1\ A_0$

$B = B_3\ B_2\ B_1\ B_0$

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

This indicates A greater than B, then inspect the relative magnitude of pairs of significant digits starting from most significant position. A is 0 and that of B is 0. The same circuit can be used to compare the relative magnitude of two BCD digits. Where,

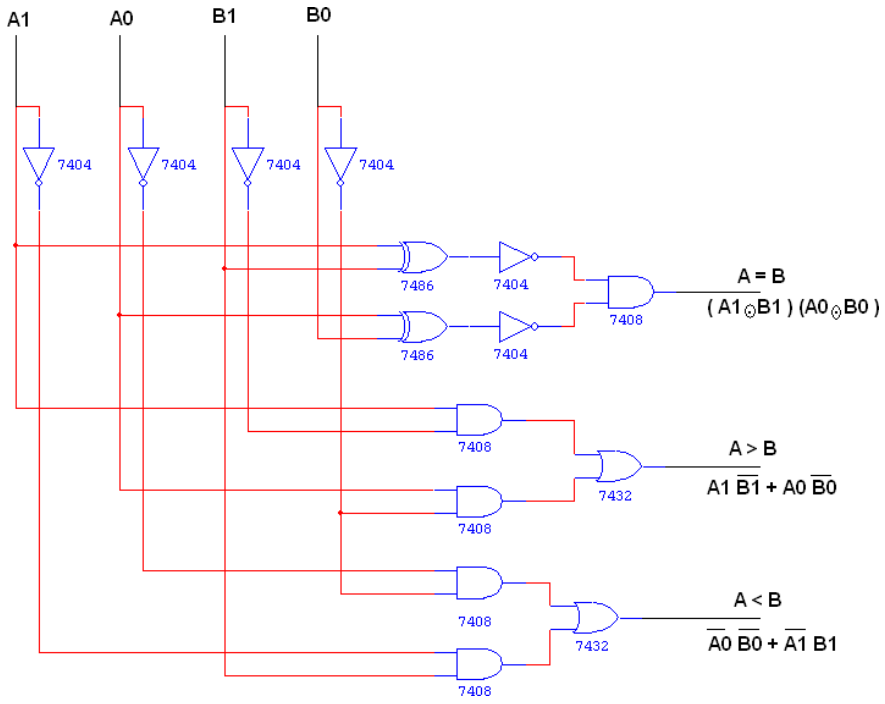
A = B is expanded as,

$A = B = (A_3 + B_3)\,(A_2 + B_2)\,(A_1 + B_1)\,(A_0 + B_0)$

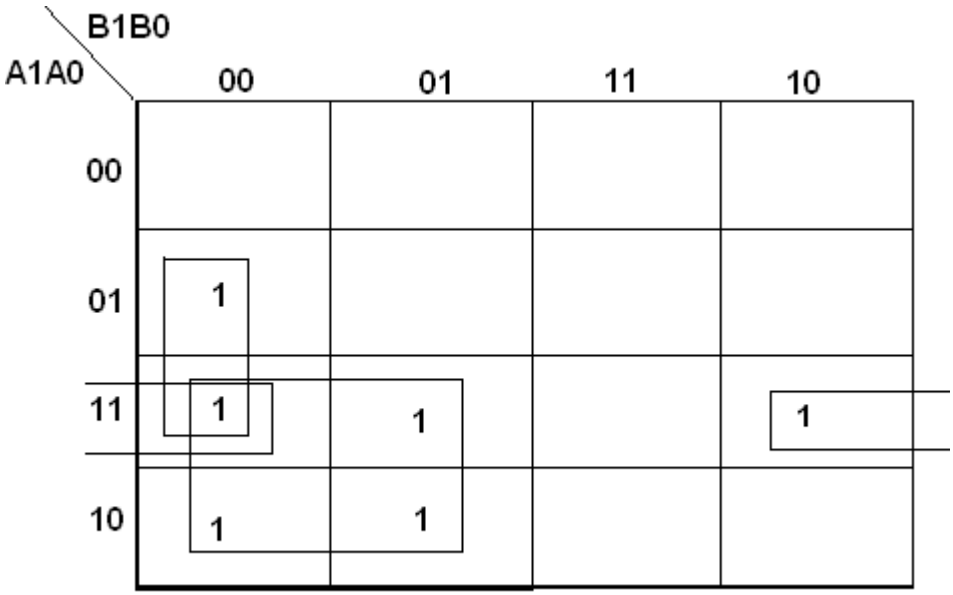
x_3 x_2 x_1 x_0

Logic Diagram:

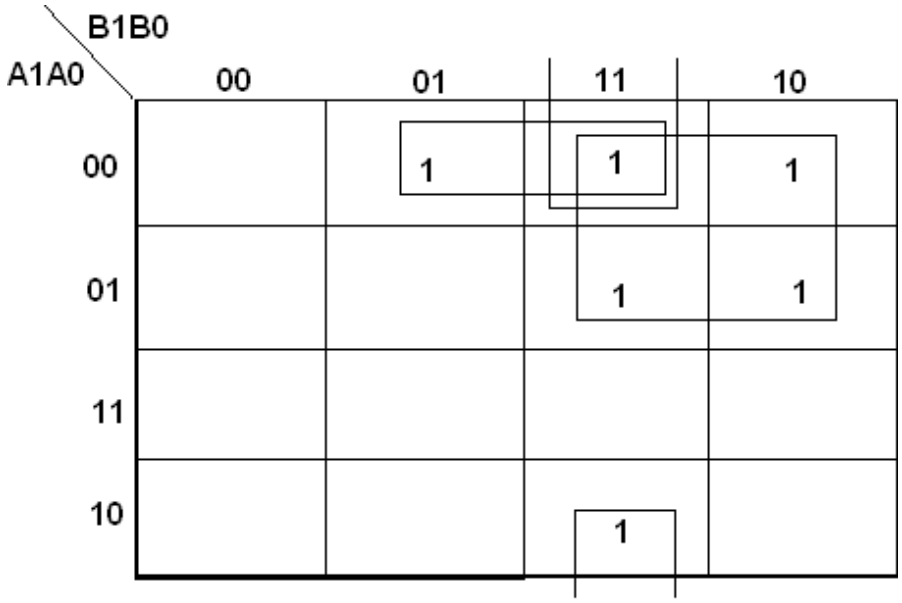
1. 2 Bit Magnitude Comparator



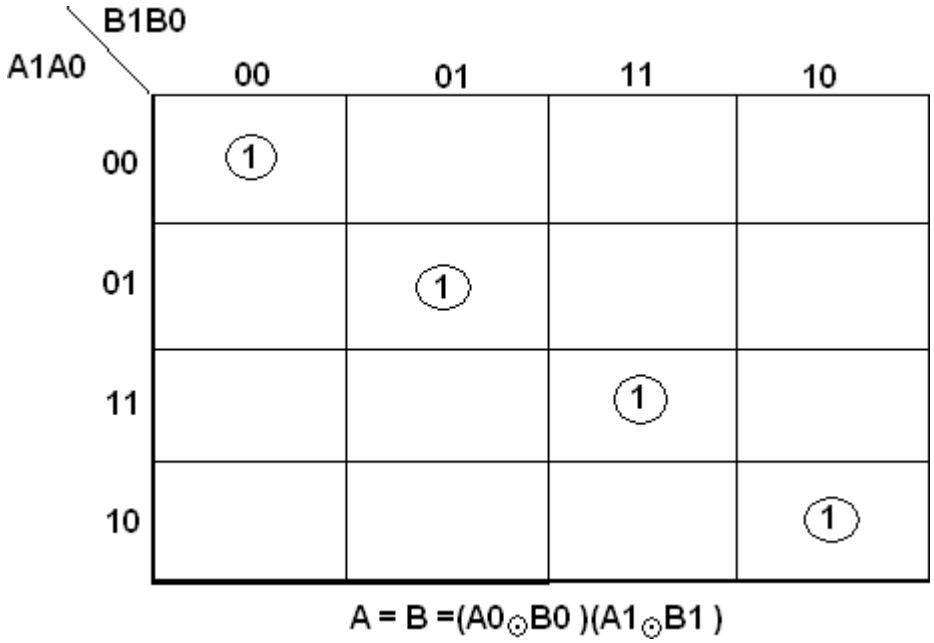
K MAP



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



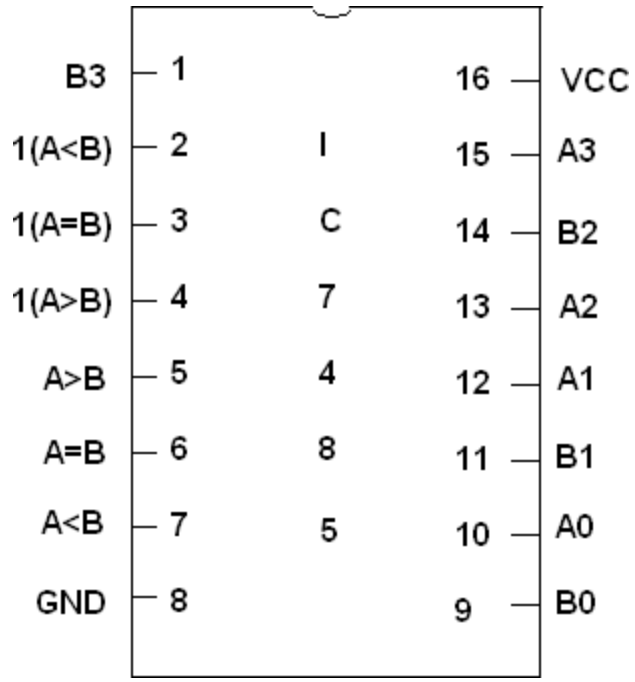
$A < B = \overline{A1}\ \overline{A0}\ B0 + \overline{A0}\ B0\ B1 + \overline{A1}\ B1$



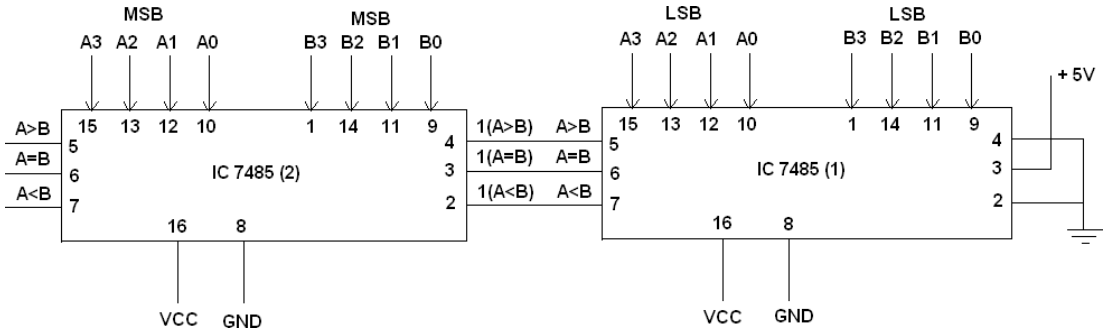
Truth Table:

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

Pin Diagram For IC 7485:



2. Logic Diagram of 8 Bit Magnitude Comparator



Truth Table:

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

Procedure:

(i) Connections are given as per circuit diagram.

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

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

Result:

This experiment perform successfully.

Experiment No.- 05

Aim: Verification of state tables of SR, JK, D flip-flops using NAND & nor gates.

Apparatus: IC's 7400,7402, Digital trainer & connecting leads

Theory:

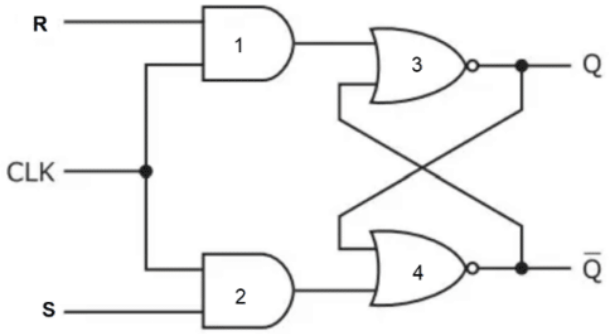
1. **RS Flip-Flop:** There are two inputs to the flip-flop defined as R and S. When i/p are R = 0 and S = 0 then o/p remains unchanged. When i/p are R = 0 and S = 1 then the flip-flop is switches to the stable state where o/p is 1 i.e. SET. The i/p condition is R = 1 and S =0 , the flip-flop is switched to the stable state where output is 0 i.e. RESET. The i/p condition is R = 1 and S = 1 the flip-flop is switched to the stable state where o/p is forbidden.

2. **JK Flip-Flop**: For purpose of counting, the JK flip-flop is the ideal element to use. The variable J and K are called control I/Ps because they determine what the flip- flop does when a positive edge arrives. When J and K are both 0s, both AND gates are disabled and Q retains its last value.

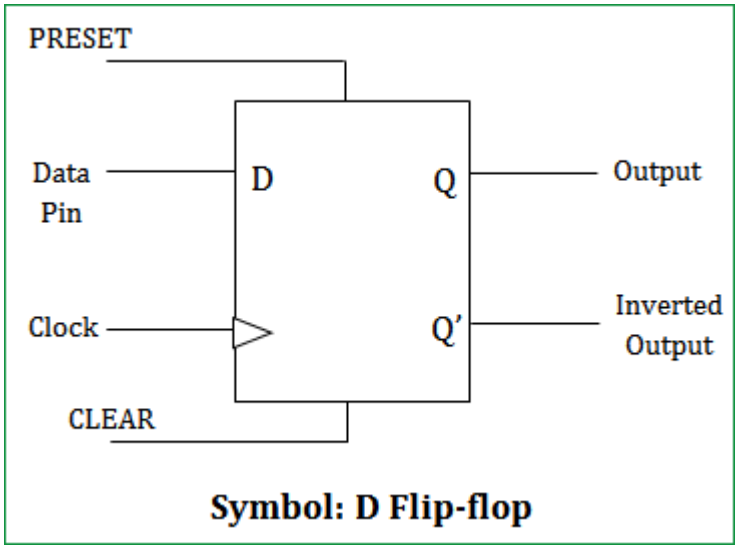
3. **D Flip-Flop**: This kind of flip flop prevents the value of D from reaching the Q output until clock pulses occur. When the clock is low, both AND gates are disabled D can change value without affecting the value of Q. On the other hand, when the clock is high, both AND gates are enabled. In this case, Q is forced to equal the value of D. When the clock again goes low, Q retains or stores the last value of D. a D flip flop is a bistable circuit whose D input is transferred to the output after a clock pulse is received.

Circuit Diagram:

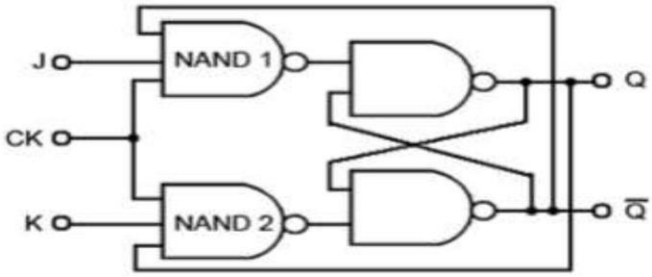
1. **SR Flip- Flop**



2. D Flip-Flop



JK Flip-Flop



Truth Table:

SR Flip- Flop:

CLOCK	S	R	Qn+1
1	0	0	NO CHANGE
1	0	1	0
1	1	0	1
1	1	1	?

D Flip- Flop:

INPUT	OUTPUT
0	0
1	1

JK Flip- Flop

CLOCK	S	R	Q_{N+1}
1	0	0	NO CHANGE
1	0	1	0
1	1	0	1
1	1	1	Qn '

Procedure:

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

Result:

Truth table is verified on digital trainer.

Experiment No. -06

Aim: To design and implement of shift register

1. **Serial in serial out**
2. **Serial in parallel out**
3. **Parallel in serial out**
4. **Parallel in parallel out**

Apparatus:

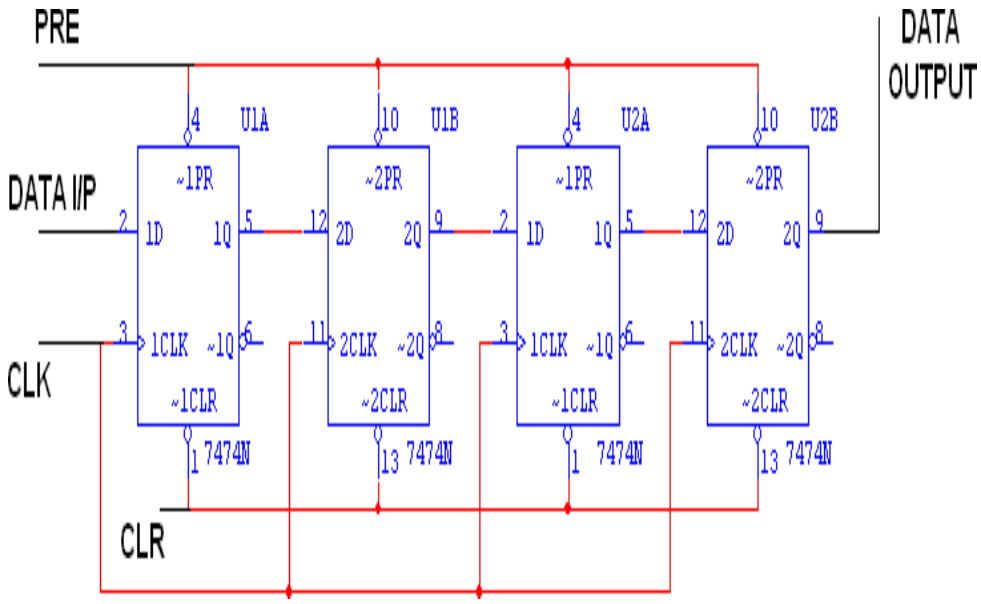
Sr. No.	COMPONENT	SPECIFICATION	QTY.
1.	D FLIP FLOP	IC 7474	2
2.	OR GATE	IC 7432	1
3.	IC TRAINER KIT	-	1
4.	PATCH CORDS	-	35

Theory:

A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only flip flop. The output of a given flip flop is connected to the input of next flip flop of the register. Each clock pulse shifts the content of register one bit position to right.

Circuit Diagram:

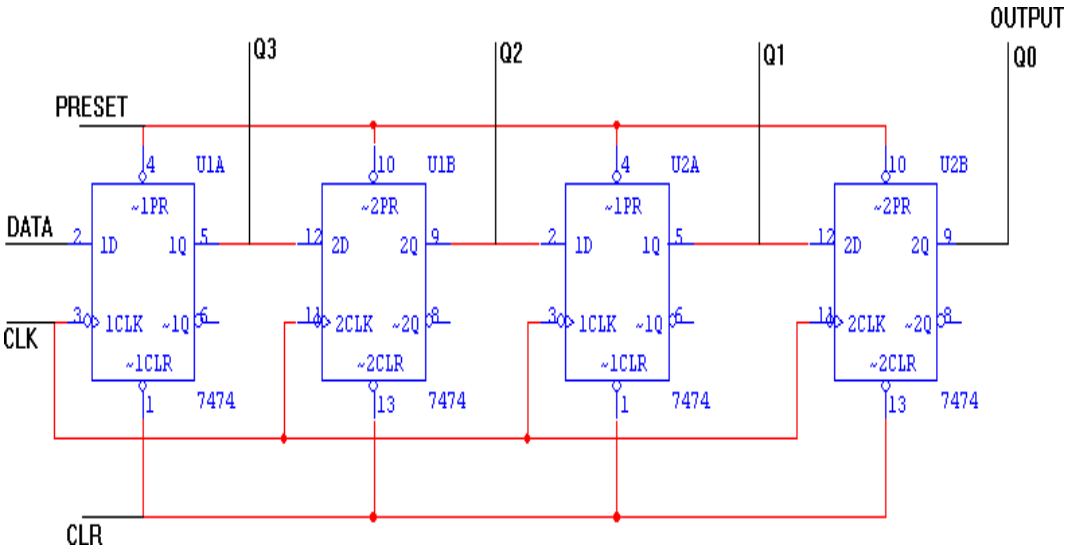
1 Serial In Serial Out:



Truth Table:

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

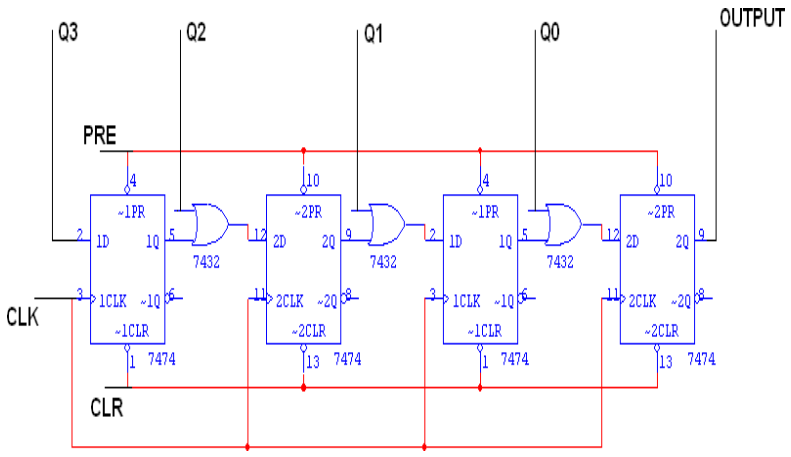
2 Serial In Parallel Out:



Truth Table:

CLK	DATA	OUTPUT			
		Q_A	Q_B	Q_C	Q_D
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	0	0	1

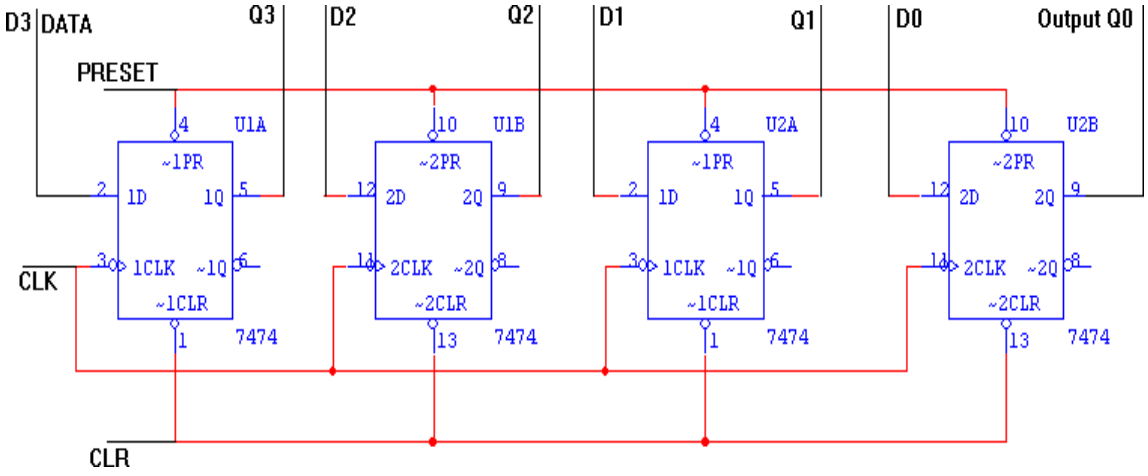
3 Parallel In Serial:



Truth Table:

CLK	Q3	Q2	Q1	Q0	O/P
0	1	0	0	1	1
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	1

4 Parallel In Parallel Out:



Truth Table:

CLK	DATA INPUT				OUTPUT			
	D_A	D_B	D_C	D_D	Q_A	Q_B	Q_C	Q_D
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

Procedure:

(i) Connections are given as per circuit diagram.

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

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

Result:

Experiment performed successfully.

Experiment No.-07

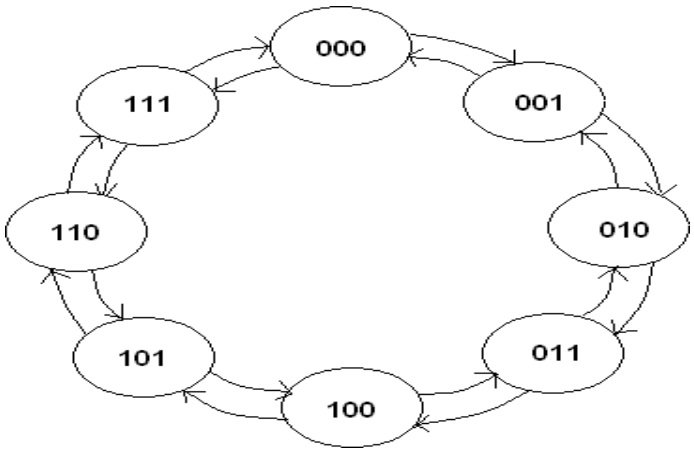
Aim: To design and implement 3 bit synchronous up/down counter.

Apparatus:

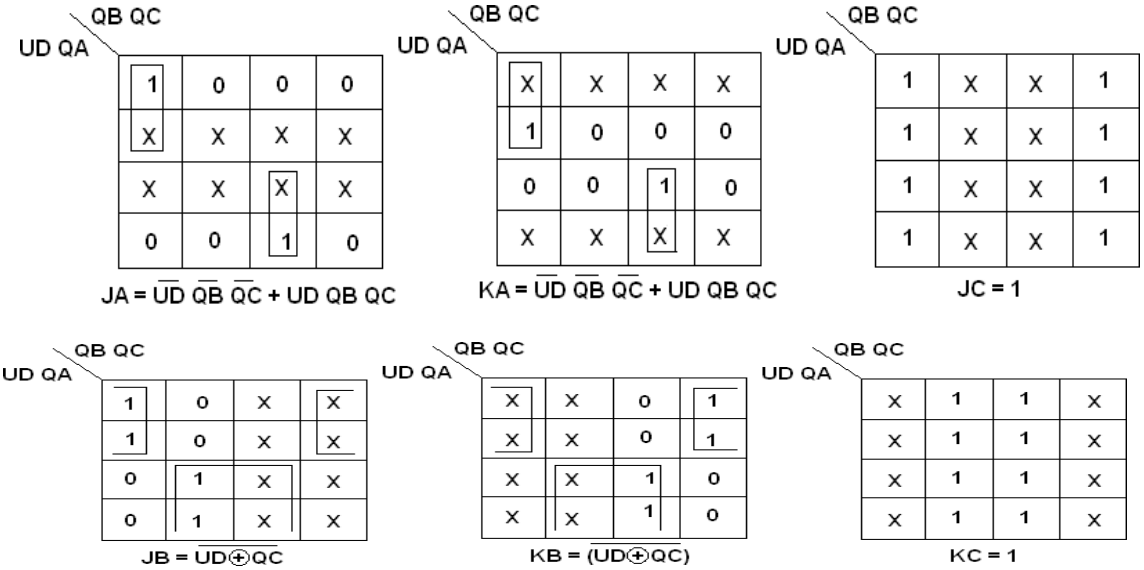
Sr No.	COMPONENT	SPECIFICATION	QTY.
1.	JK FLIP FLOP	IC 7476	2
2.	3 I/P AND GATE	IC 7411	1
3.	OR GATE	IC 7432	1
4.	XOR GATE	IC 7486	1
5.	NOT GATE	IC 7404	1
6.	IC TRAINER KIT	-	1
7.	PATCH CORDS	-	35

Theory:

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. An up/down counter is one that is capable of progressing in increasing order or decreasing order through a certain sequence. An up/down counter is also called bidirectional counter. Usually up/down operation of the counter is controlled by up/down signal. When this signal is high counter goes through up sequence and when up/down signal is low counter follows reverse sequence. For clear understanding state diagram is shown below:



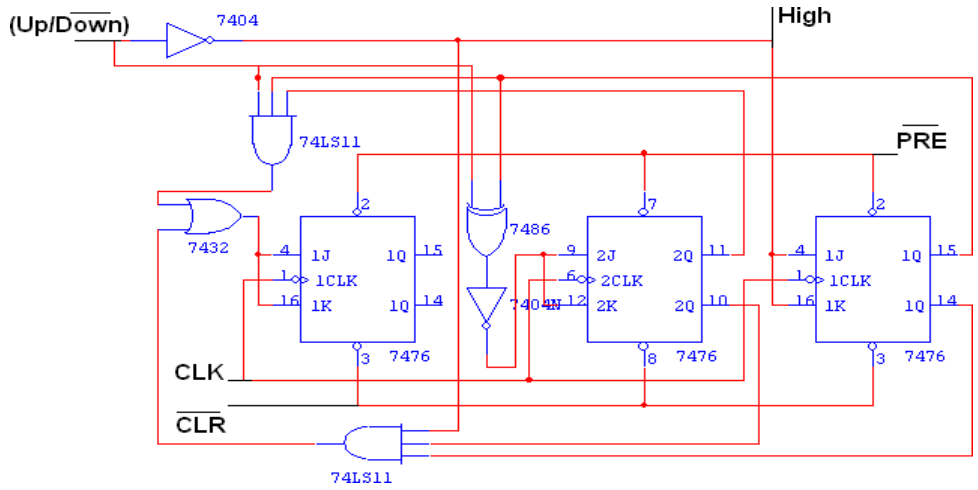
1. K Map:



Characteristic Table:

Q	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Circuit Diagram:



Truth Table:

TRUTH TABLE:

Input	Present State			Next State			A		B		C	
Up/Down	Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	1	1	1	1	X	1	X	1	X
0	1	1	1	1	1	0	X	0	X	0	X	1
0	1	1	0	1	0	1	X	0	X	1	1	X
0	1	0	1	1	0	0	X	0	0	X	X	1
0	1	0	0	0	1	1	X	1	1	X	1	X
0	0	1	1	0	1	0	0	X	X	0	X	1
0	0	1	0	0	0	1	0	X	X	1	1	X
0	0	0	1	0	0	0	0	X	0	X	X	1
1	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
1	0	1	0	0	1	1	0	X	X	0	1	X
1	0	1	1	1	0	0	1	X	X	1	X	1
1	1	0	0	1	0	1	X	0	0	X	1	X
1	1	0	1	1	1	0	X	0	1	X	X	1
1	1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	0	0	X	1	X	1	X	1

Procedure:

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

Result:

Experiment is performed successfully.

Experiment No.-08

Aim: Design and verification of the truth tables of Half and Full adder circuits .

Apparatus: logic trainer kit, NAND gates (IC 7400), XOR gates (IC 7486), AND gates (IC 7408), wires.

Theory:

1 Half adder:

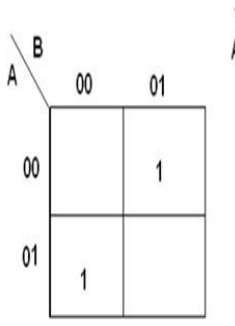
A half adder has two inputs for the two bits to be added and two outputs one from the sum ' S' and other from the carry 'C' into the higher adder position. Above circuit is called as a carry signal from the addition of the less significant bits sum from the X-OR Gate the carry out from the AND gate.

Truth Table:

Truth Table for Half Adder

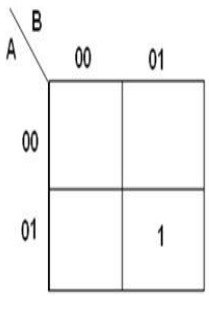
INPUT		OUTPUT	
A	**B**	**SUM**	**CARRY**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-Map for SUM:



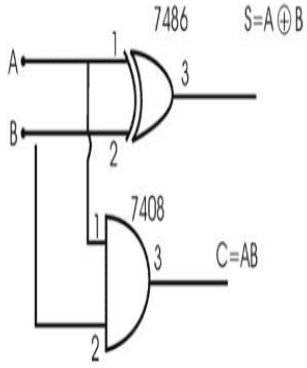
SUM = A'B + AB'

K-Map for CARRY:



CARRY = AB

Half Adder using basic gates:-



$S = \bar{A}B + A\bar{B}$

$S = A \oplus B$

$C = AB$

2 Full Adder:

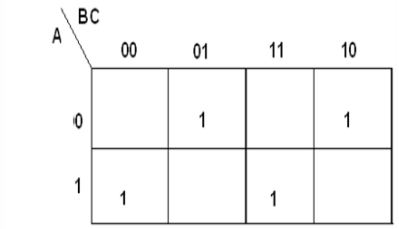
A full adder is a combinational circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from X-OR Gate, carry output will be taken from OR Gate.

Truth Table:

Truth Table for Full Adder

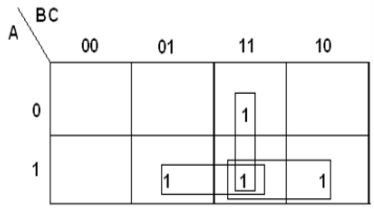
INPUT			OUTPUT	
A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K-Map for SUM:



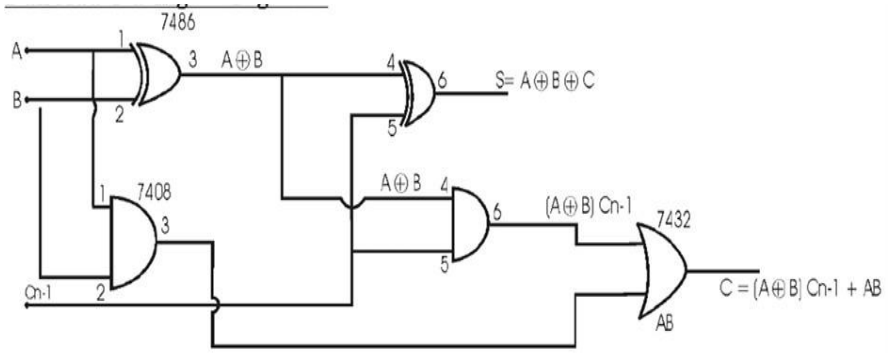
SUM = A'B'C + A'BC' + ABC' + ABC

K-Map for CARRY:



CARRY = AB + BC + AC

Full Adder Using Basic Gates:



Procedure:

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

Result:

Thus the half adder &full adder was designed and their truth table is verified.

Experiment No.-09

Aim: - Design and verification of the truth tables of Half and Full Subtractor circuits.

Apparatus:

Logic trainer kit, NAND gates (IC 7400), XOR gates (IC 7486), AND gates (IC 7408), NOT gates (IC 7404), connecting wires.

Theory:

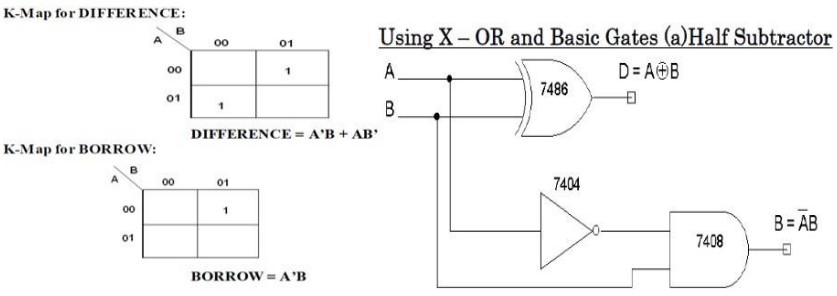
1. Half Subtractor:

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

Truth Table:

Truth Table for Half Subtractor

INPUT		OUTPUT	
A	B	Bo	Diff
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0



2. Full Subtractor:

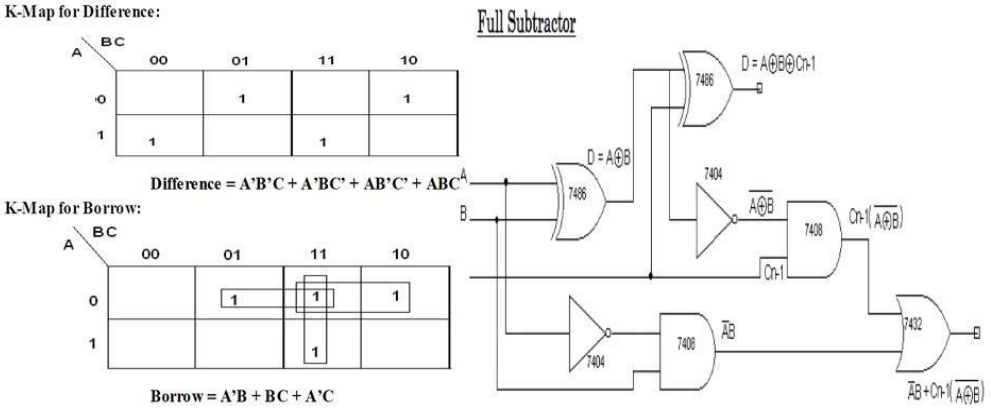
The full subtractor is a combination of X-OR, AND, OR, NOT Gates. In a full subtractor the logic circuit should have three inputs and two outputs. The two half subtractor put together gives a full subtractor .The first half subtractor will be C and A B.

The output will be difference output of full subtractor. The expression AB assembles the borrow output of the half subtractor and the second term is the inverted difference output of first X-OR.

Truth Table:

Truth Table for Full Subtractor

INPUT			OUTPUT	
A	B	C	Bo	Diff
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Procedure:

(i) Connections are given as per circuit diagram.

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

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

Result:

Thus the half subtractor and full subtractor was designed and their truth table is verified.

Experiment No.-10

Aim: - Verify the NAND and NOR gates as universal logic gates.

Apparatus:

Logic trainer kit, NAND gates (IC 7400), NOR gates (IC 7402), wires.

Theory: -

1. NAND gate is actually a combination of two logic gates: AND gate followed by NOT gate. So its output is complement of the output of an AND gate. This gate can have minimum two inputs; output is always one. By using only NAND gates, we can realize all logic functions: AND, OR, NOT, X-OR, X-NOR, NOR. So this gate is also called universal gate. NAND gates as NOT gate: A NOT produces complement of the input. It can have only one input, tie the inputs of a NAND gate together. Now it will work as a NOT gate. Its output is

   Y = (A.A)'

   Y = (A)'

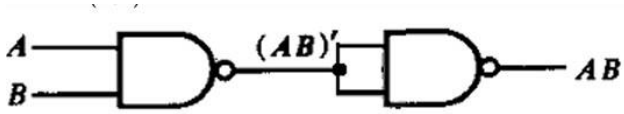
NOT (inverter)

(i) NAND gates as AND gate:

A NAND produces complement of AND gate. So, if the output of a NAND gate is inverted, overall output will be that of an AND gate.

Y = ((A.B)')'

Y = (A.B)



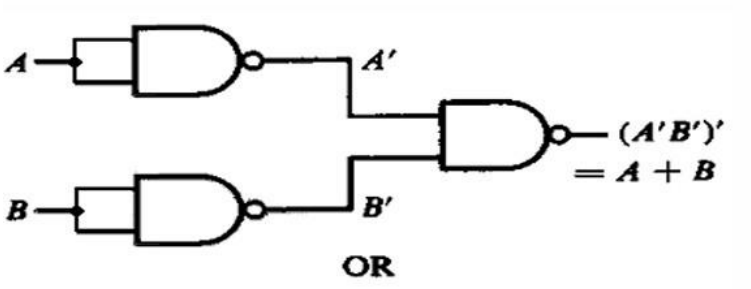
AND

(ii) NAND gates as OR gate:

From DeMorgan's theorems: (A.B)' = A' + B

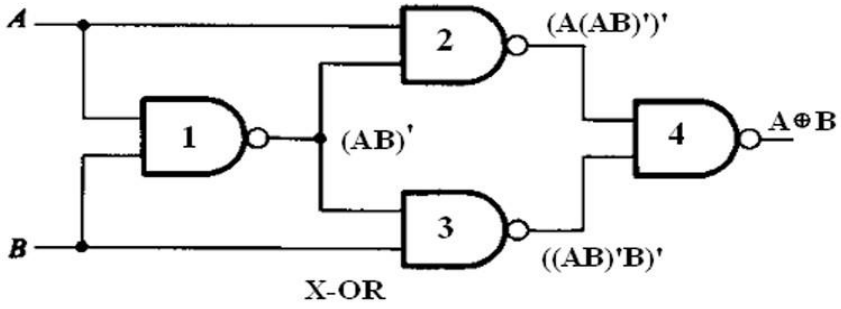
' (A'.B')' = A'' + B'' = A + B

So, give the inverted inputs to a NAND gate, obtain OR operation at output



(iii) NAND gates as X-OR gate:

The output of a two input X-OR gate is shown by: Y = A'B + AB'. This can be achieved with the logic diagram shown in the below.



Gate No.	Inputs	Output
1	A, B	(AB)'
2	A, (AB)'	(A (AB)')'
3	(AB)', B	(B (AB)')'
4	(A (AB)')', (B (AB)')'	A'B + AB'

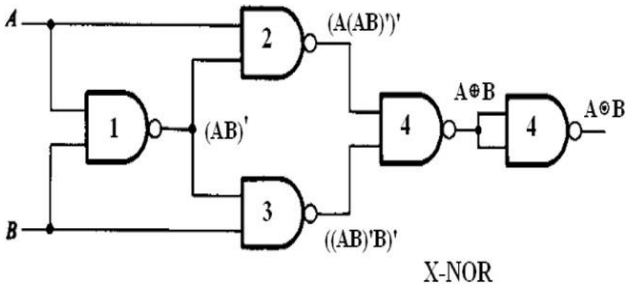
Now the output from gate no. 4 is the overall output of the configuration.

$$
\begin{aligned}
Y &= ((A\,(AB)')'\,(B\,(AB)')')' \\
&= (A(AB)'')' + (B(AB)'')' \\
&= (A(AB)') + (B(AB)') \\
&= (A(A' + B')) + (B(A' + B')) \\
&= (AA' + AB') + (BA' + BB') \\
&= (0 + AB' + BA' + 0) \\
&= AB' + BA' \\
Y &= \mathbf{AB' + A'B}
\end{aligned}
$$

(iv) NAND gates as X-NOR gate:

X-NOR gate is actually X-OR gate followed by NOT gate. So give the output of X-OR gate to a NOT gate, overall output is that of an X-NOR gate.

Y = AB+ A'B'



X-NOR

Procedure:

1. Connect the trainer kit to ac power supply.

2. Connect the NAND gates for any of the logic functions to be realized.

3. Connect the inputs of first stage to logic sources and output of the last gate to logic indicator.

4. Apply various input combinations and observe output for each one.

5. Verify the truth table for each input/ output combination.

6. Repeat the process for all logic functions.

7. Switch off the power supply.

2. NOR THEORY:

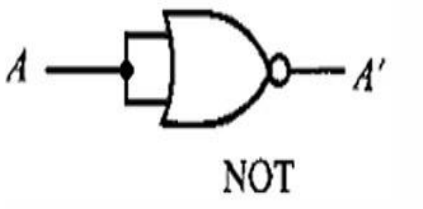
NOR gate is actually a combination of two logic gates: OR gate followed by NOT gate. So its output is complement of the output of an OR gate. This gate can have minimum two inputs; output is always one. By using only NOR gates, we can realize all logic functions: AND, OR, NOT, X-OR, X-NOR, NAND. So this gate is also called universal gate.

NOR gates as NOT gate:

A NOT produces complement of the input. It can have only one input, tie the inputs of a NOR gate together. Now it will work as a NOT gate. Its output is

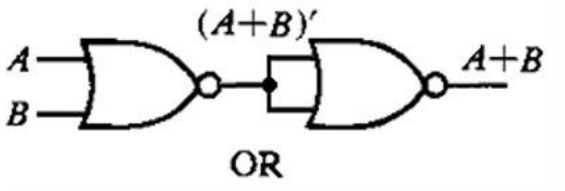
Y = (A+A)’

Y = (A)’



(i) NOR gates as OR gate:

A NOR produces complement of OR gate. So, if the output of a NOR gate is inverted, overall output will be that of an OR gate.

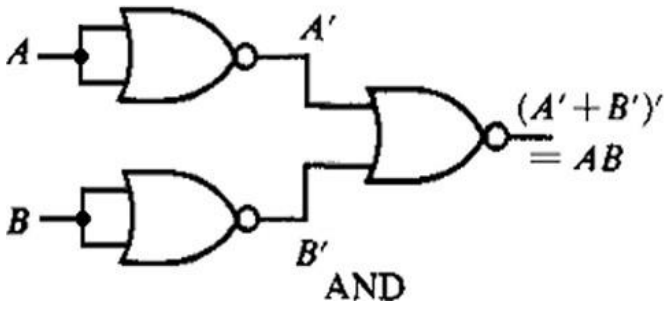


(ii) NOR gates as AND gate:

From DE Morgan's theorems: (A+B)' = A'B'

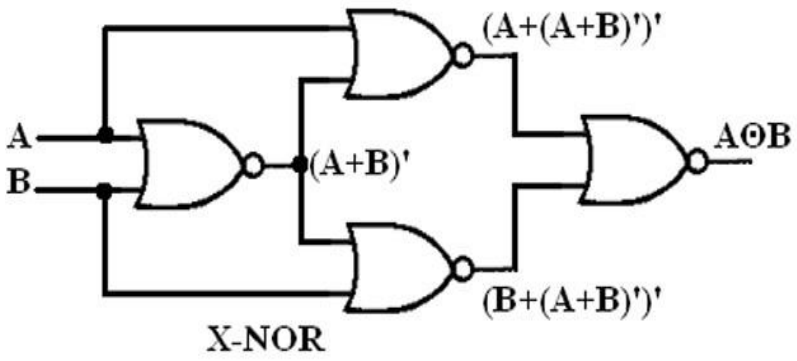
(A'+B')' = A''B'' = AB

So, give the inverted inputs to a NOR gate, obtain AND operation at output.



(iii) NOR gates as X-NOR gate:

The output of a two input X-NOR gate is shown by: Y = AB + A'B'. This can be achieved with the logic diagram shown in the left side.



Gate No.	Inputs	Output
1	A, B	(A + B)'
2	A, (A + B)'	(A + (A+B)')'
3	(A + B)', B	(B + (A+B)')'
4	(A + (A + B)')', (B + (A+B)')'	AB + A'B'

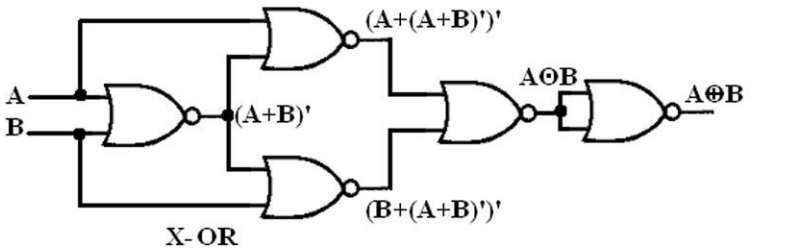
Now the output from gate no. 4is the overall output of the configuration.

$$
\begin{aligned}
Y &= ((A + (A+B)')' (B +(A+B)')')' \\
&= (A+(A+B)'')'.(B+(A+B)'')' \\
&= (A+(A+B)').(B+(A+B)') \\
&= (A+A'B').(B+A'B') \\
&= (A + A').(A + B').(B+A')(B+B') \\
&= 1.(A+B').(B+A').1 \\
&= (A+B').(B+A') \\
&= A.(B + A') +B'.(B+A') \\
&= AB + AA' +B'B+B'A' \\
&= AB + 0 + 0 + B'A' \\
&= AB + B'A' \\
\mathbf{Y} &= AB + A'B'
\end{aligned}
$$

(iv) NOR gates as X-OR gate:

X-OR gate is actually X-NOR gate followed by NOT gate. So give the output of X-NOR gate to a NOT gate, overall output is that of an X-OR gate.

Y = A'B+ AB'



Procedure:

1. Connect the trainer kit to ac power supply.

2. Connect the NOR gates for any of the logic functions to be realized.

3. Connect the inputs of first stage to logic sources and output of the last gate to logic indicator.

4. Apply various input combinations and observe output for each one.

5. Verify the truth table for each input/ output combination.

6. Repeat the process for all logic functions.

7. Switch off the ac power supply.

Result:

NAND & NOR are verified as universal gates successfully.

Experiment No.-11

Aim: To verify Binary to Gray code conversion.

Apparatus:

ST2611 Digital Circuit Development Platform trainer with power supply cord, DB06 – Code Conversion, Set of wires

Theory:

From the previous experiments, we studied that the binary code of data is represented by two values such as 0's and 1's, and it is mainly used in the world of the computer. Gray Code is a form of binary that uses a different method of incrementing from one number to the next. With Gray Code, only one-bit changes state from one position to another. This feature allows a system designer to perform some error checking. Gray Code is the most popular Absolute encoder output type because its use prevents certain data errors which can occur with Natural Binary during state changes.

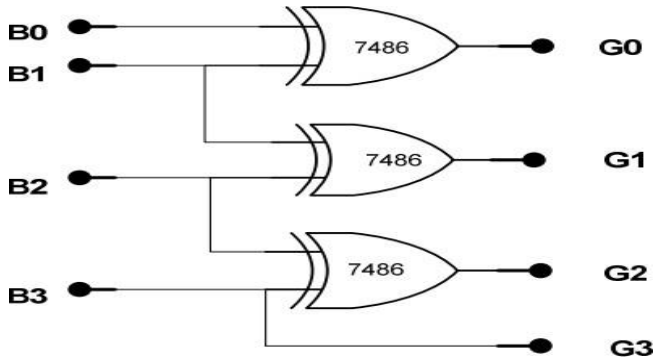
Code converter is a circuit that makes two systems compatible even though each uses different binary code. Code converters are used for protecting private information from spies. Moreover, they are used to enhance data portability and tractability. We will discuss in this experiment how to convert from binary to gray code and vice versa, in order to understand how the encoder and decoder are working.

1. **Binary to Gray Code Conversion**

The following example will be very useful for knowing the conversion of binary to gray code. In this conversion method, take down the MSB bit of the present binary number, as the primary bit or MSB bit of the gray code number is similar to the binary number. To get the straight gray coded bits for generating the corresponding gray coded digit for the given binary digits, look at the primary digit or the MSB digit of binary number and the second digit then note down 0 if they are similar to each other and 1 if they are different next to the primary bit of gray code. Do the same thing with the next binary bit and third bit then note down the product next to the 2nd bit of gray code.

Similarly, follow this procedure until the final binary bit as well as note down the outcomes to generate the corresponding gray coded binary digit. The truth table of binary to gray code conversion can be seen at Table (1).

Circuit Diagram Using X-OR Gate:



Truth Table:

Binary Code				Gray Code			
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

Procedure:

1. Place the DB06 panel shown in Figure 1 on the trainer.

2. To provide power to the board, connect +5 V pin from the trainer on the left side to +5V pin on the DB06 using a wire.

3. Connect GND pin from the trainer on the left side to ground symbol pin on the DB06.

4. Connect the input switches which are pins no. D0, D1, D2 and D3 on the bottom of the trainer to B0, B1, B2, and B3 pins of the binary to gray code circuit on the left of the panel respectively.

5. Connect the pin no. G0, G1, G2, and G3 of binary to gray code circuit on the left side of the panel to the output pin no. 0, 1, 2, and 3 of the 8-bit LED Display on the right side of the trainer respectively, in order to display the gray code from the binary inputs.

6. Make sure all your connections are right then turn on the power supply.

9. Verify Truth Table on Table (1).

Result:

Binary to gray code conversion verified successfully.

Experiment No.-12

Aim: To verify Gray to Binary code conversion.

Apparatus:

ST2611 Digital Circuit Development Platform trainer with power supply cord, DB06 – Code Conversion, Set of wires

Theory:

From the previous experiments, we studied that the binary code of data is represented by two values such as 0's and 1's, and it is mainly used in the world of the computer. Gray Code is a form of binary that uses a different method of incrementing from one number to the next. With Gray Code, only one-bit changes state from one position to another. This feature allows a system designer to perform some error checking.

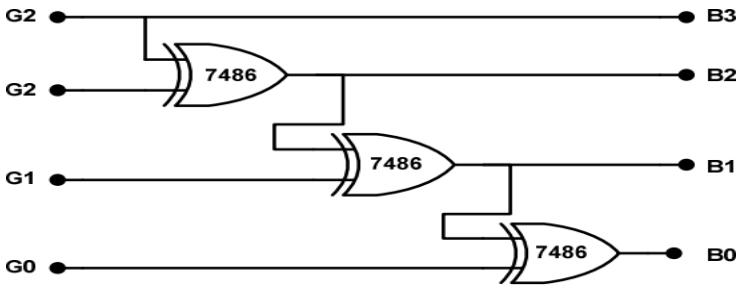
Gray Code is the most popular Absolute encoder output type because its use prevents certain data errors which can occur with Natural Binary during state changes. Code converter is a circuit that makes two systems compatible even though each uses different binary code. Code converters are used for protecting private information from spies. Moreover, they are used to enhance data portability and tractability. We will discuss in this experiment how to convert from binary to gray code and vice versa, in order to understand how the encoder and decoder are working.

1. **Gray to Binary Code Conversion**

This gray to binary conversion method also uses the working concept of binary to gray code conversion. The following example with step by step procedure may help to know the conversion concept of gray code to binary code. To change gray to binary code, take down the MSB digit of the gray code number, as the primary digit or the MSB of the binary code is similar to the gray digit. To get the next straight binary bit, look at the primary digit or the MSB digit of gray code and the second digit then note down 0 if they are similar to each other and 1 if they are different next to the primary bit of binary number.

Do the same thing with the next gray bit and third bit then note down the product next to the 2nd bit of binary code. Similarly, follow this procedure until the final gray bit as well as note down the outcomes to generate the corresponding binary number. The truth table of gray to binary code conversion can be seen at Table (2).

Circuit Diagram using X-OR gates:



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

Procedure:

1. Turn off the power supply and keep no. 1 through 3 of exercise 1 connections the same and take of the others.

2. Connect the input switches which are pins no. D0, D1, D2 and D3 on the bottom of the trainer to G0, G1, G2, and G3 pins of the gray to binary code circuit on the right of the panel respectively.

3. Connect the pin no. B0, B1, B2, and B3 of gray to binary code circuit on the right side of the panel to the output pin no. 0, 1, 2, and 3 of the 8-bit LED Display on the right side of the trainer respectively, in order to display the binary number from the gray inputs.

4. Make sure all your connections are right then turn on the power supply.

5. Turn D3 switch to position 1 and keep D0, D1 and D2 switches at position 0 then observe the output on 8-bit LED display (green light indicates 0 and red light indicates 1).

6. Observe the output for different input combination as shown in truth table of gray to binary code conversion on Table (2).

7. Verify Truth Table on Table (2).

Result:

Gray to Binary code conversion verified successfully.