EXPERIMENT: 1 ***DATE:***

VERIFICATION OF BASIC LOGIC GATES

AIM: To verify the truth tables of Basic Logic Gates

NOT, OR, AND, NAND, NOR, Ex-OR and Ex-NOR.

APPARATUS: ***mention the required IC numbers****, Connecting wires and* ***IC*** *Trainer Kit*

THEORY:

Logic gates are the digital circuits with one output and one or more inputs. They are the basic building blocks of any logic circuit.

Different logic gates are: **AND, OR, NOT, NAND, NOR, Ex-OR and Ex-NOR**

They work according to certain logic.

AND: The output of AND gate is true when the inputs A and B are True.

Logic equation: $Y = A.B$

Truth Table:

A	**B**	$Y = A.B$
0	0	0
0	1	0
1	0	0
1	1	1

Graphic Symbol:

OR: The output of OR gate is true when one of the inputs A and B or both the inputs are true.

Logic equation: $Y = A + B$

Truth Table:

A	B	$Y = A + B$
0	0	0
0	1	0
1	0	0
1	1	1

Graphic Symbol:

NOT: The output of NOT gate is complement of the input.

Logic equation $Y = \overline{A}$

Truth Table:

Graphic Symbol:

NAND: The output of NAND gate is true when one of the inputs or both the inputs are low level.

Logic Equation: $Y = \overline{A.B} = \overline{A} + \overline{B}$

Truth Table:

Graphic Symbol:

NOR: The output of NOR gate is true when both the inputs are low.

Logic Equation: $Y = \overline{A + B} = \overline{A}.\overline{B}$

Truth Table:

Graphic Symbol:

EX-OR: The output of EX-OR gate is true when both the inputs are unequal.

Logic Equation: $Y = \overline{A}B + A\overline{B} = A \oplus B$

Truth Table:

Graphic Symbol:

EX-NOR: The output of EX-NOR gate is true when both the inputs are equal.

Logic Equation: $Y = AB + \overline{A}\overline{B}$

Truth Table:

Graphic Symbol:

PROCEDURE:

RESULT:

EXPERIMENT: 2 *DATE:*

REALIZATION GIVEN BOOLEAN FUNCTION

AIM: To simplify the given expression using K-map and realize it using Basic gates and Universal gates.

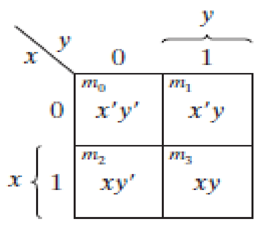
APPARATUS:(write the apparatus)

THEORY:

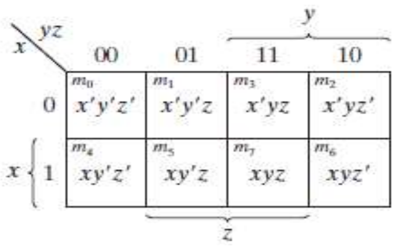
Canonical Forms (Normal Forms): Any Boolean function can be written in disjunctive normal form (sum of min-terms) or conjunctive normal form (product of max-terms). A Boolean function can be represented by a Karnaugh map in which each cell corresponds to a minterm. The cells are arranged in such a way 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.

Karnaugh Maps:

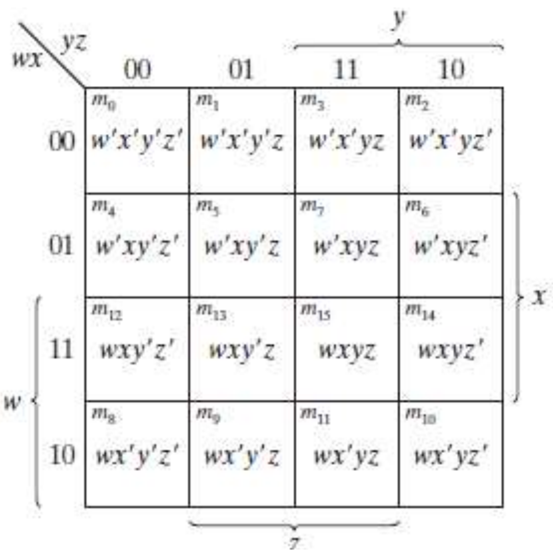
Two- variable K-Map



Three Variable K-Map:



Four variable K-Map:



wx \ yz	00	01	11	10
00	m_0 w'x'y'z'	m_1 w'x'y'z	m_3 w'x'yz	m_2 w'x'yz'
01	m_4 w'xy'z'	m_5 w'xy'z	m_7 w'xyz	m_6 w'xyz'
11	m_{12} wxy'z'	m_{13} wxy'z	m_{15} wxyz	m_{14} wxyz'
10	m_8 wx'y'z'	m_9 wx'y'z	m_{11} wx'yz	m_{10} wx'yz'

Any two adjacent squares in the map differ by only one variable, which is primed in one square and unprimed in the other. Therefore, any two minterms in djacent squares (vertically or horizontally, but not diagonally, adjacent) that are ORed together will cause a removal of the dissimilar variable.

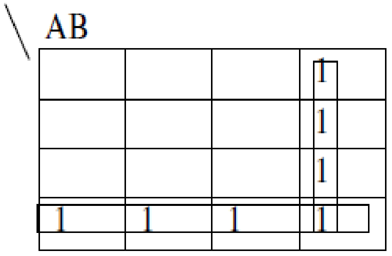
Simplification of given expression using K-Map:

Given expression is

$$F(A,B,C,D)=\sum(2,6,8,9,10,11,14)$$

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

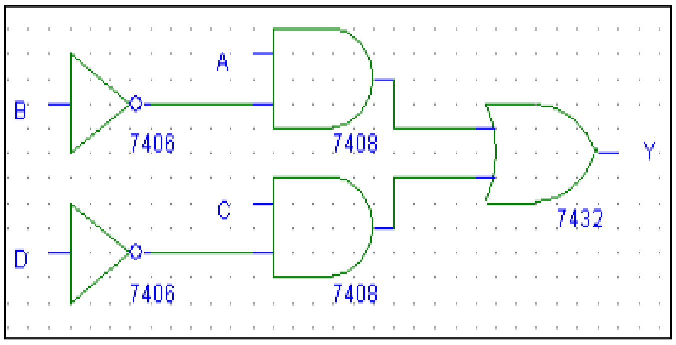
K-Map:



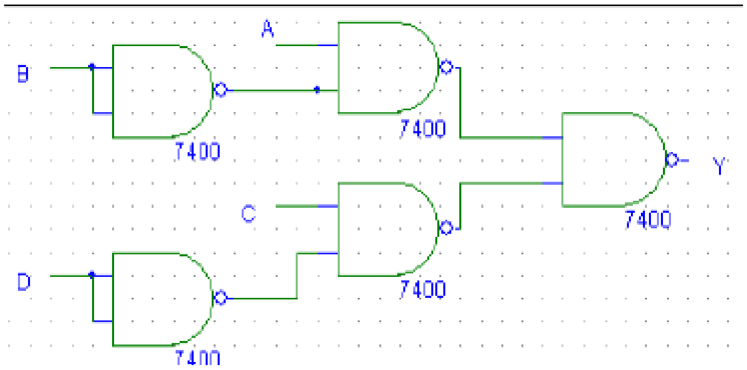
AB				
				1
				1
				1
	1	1	1	1

Simplified Expression is : $Y = A\bar{B} + C\bar{D}$

Realization using Basic gates:



Realization using NAND gates:



PROCEDURE:

1. Simplify the given Boolean expression using 4 Variable K-Map to minimize the number of literals in the given expression.
2. Design Logic circuit using Basic gates.
3. Check the components for their working.
4. Insert the appropriate IC into the IC base.
5. Make connections as shown in the circuit diagram.

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

RESULT:

EXPERIMENT: 3 *DATE:*

REALIZATION OF BASIC GATES USING NAND

AIM: To implement the basic gates(*NOT, AND* and *OR*), *Ex-OR* and *Ex-NOR* using universal NAND gates.

APPARATUS:

THEORY:

AND, OR, NOT are called basic gates as their logical operation cannot be simplified further. NAND and NOR are called universal gates as using only NAND or only NOR any logic function can be implemented. Using NAND and NOR gates and De Morgan's Theorems different basic gates & EX-OR gates are realized.

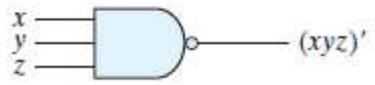
NAND : The output of NAND gate is true when one of the inputs or both the inputs are low level.

Logic Equation: $Y = \overline{A.B} = \overline{A} + \overline{B}$

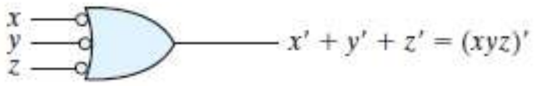
Truth Table:

A	**B**	$Y = \overline{A.B} = \overline{A} + \overline{B}$
0	0	1
0	1	1
1	0	1
1	1	0

Graphic Symbol:

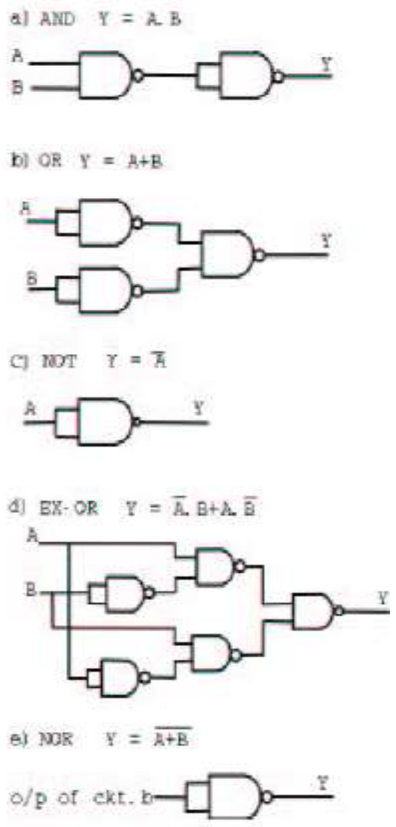


AND-Invert



Invert-OR

Realization of basic gates using NAND:


a) AND Y = A. B
A
B
Y
b) OR Y = A+B
A
B
Y
C) NOT Y = $\overline{A}$
A
Y
d) EX-OR Y = $\overline{A}$. B+A. $\overline{B}$
A
B
Y
e) NOR Y = $\overline{A+B}$
o/p of ckt. b
Y

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

RESULT:

EXPERIMENT: 4 *DATE:*

REALIZATION OF BASIC GATES USING NOR

AIM: To implement the basic gates(*NOT, AND* and *OR*), *Ex-OR* and *Ex-NOR* using universal NOR gates.

APPARATUS:

THEORY:

AND, OR, NOT are called basic gates as their logical operation cannot be simplified further. NAND and NOR are called universal gates as using only NAND or only NOR any logic function can be implemented. Using NAND and NOR gates and De Morgan's Theorems different basic gates & EX-OR gates are realized.

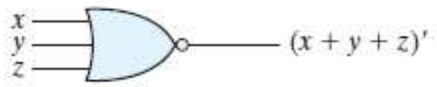
NOR : The output of NOR gate is true when both the inputs are low.

Logic Equation: $Y = \overline{A+B} = \overline{A}.\overline{B}$

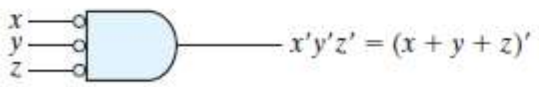
Truth Table:

A	**B**	$Y = \overline{A+B} = \overline{A}.\overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

Graphic Symbol:

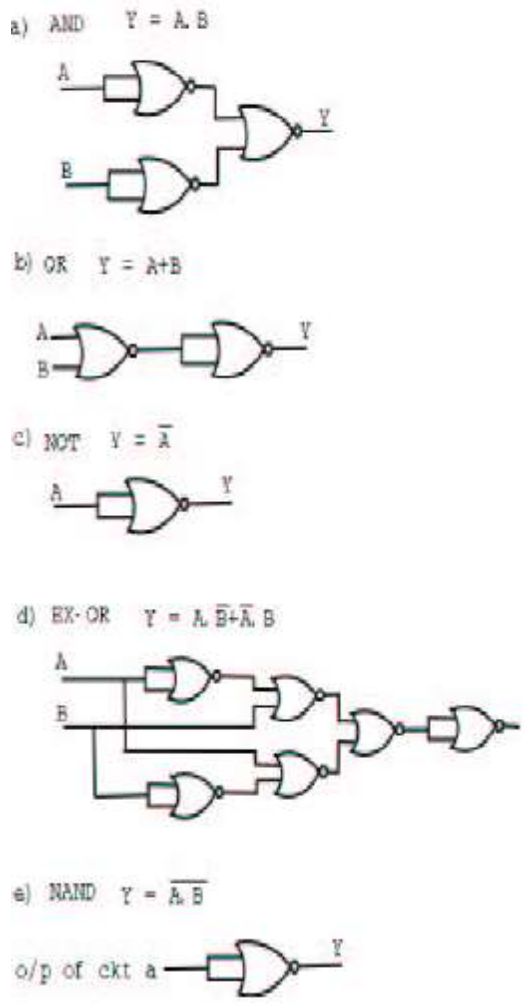


OR-Invert



Invert-AND

Realization of basic gates using NOR:


a) AND Y = A.B
A
B
Y
b) OR Y = A+B
A
B
Y
c) NOT Y = $\overline{A}$
A
Y
d) EX-OR Y = A.$\overline{B}$+$\overline{A}$.B
A
B
e) NAND Y = $\overline{A.B}$
o/p of ckt a
Y

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

RESULT:

EXPERIMENT: 5 ***DATE:***

DESIGN OF HALF ADDER AND HALF SUBTRACTOR

AIM: To design Half-Adder and Half Subtractor using basic logic gates and verification of truth table.

APPARATUS:

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:

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

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:

$$B\_out = A \oplus B$$
$$D = A'\,B$$

Realization of Half Adder Circuit:

TRUTH TABLE

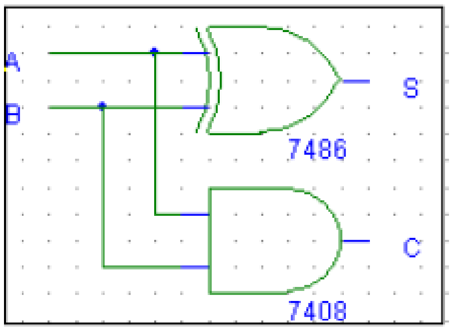
INPUTS		OUTPUTS	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

BOOLEAN EXPRESSIONS:

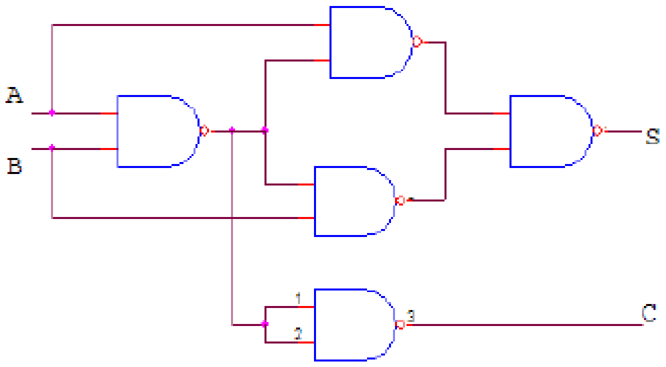
$S=A \oplus B$

$C=A\ B$

Basic Gates



ii) NAND Gates



Realization of Half-Subtractor Circuit:

TRUTH TABLE

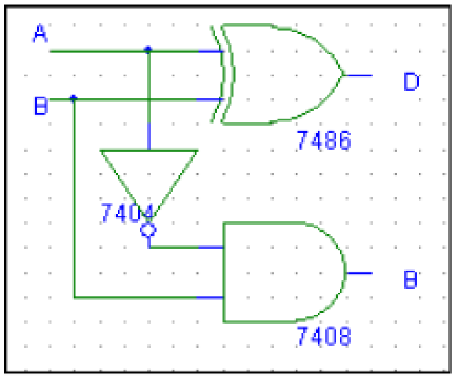
INPUTS		OUTPUTS	
A	B	D	Br
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

BOOLEAN EXPRESSIONS:

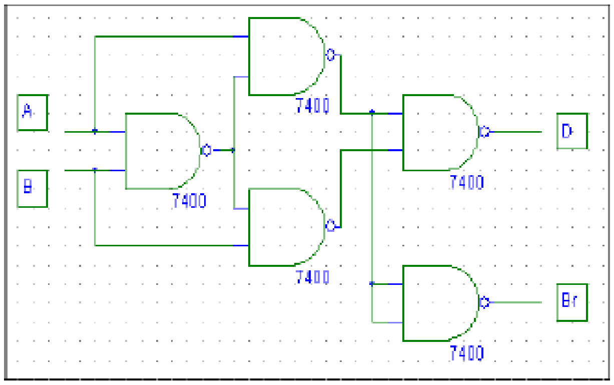
D = A $\oplus$ B

Br = $\bar{A}B$

i)BASIC GATES



ii) NAND Gates



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

RESULT:

EXPERIMENT: 6 ***DATE:***

DESIGN OF FULL ADDER AND FULL SUBTRACTOR

AIM: To design Full-Adder and Full-Subtractor using basic logic gates and verification of truth table.

APPARATUS:

THEORY:

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, C_{in} , is called a full-adder.

The Boolean functions describing the full-adder are:

$$S = A \oplus B \oplus C_{in}$$

$$C = AB + BC_{in} + C_{in}A$$

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-subtractor are:

$$D = A \oplus B \oplus C_{in}$$

$$Br = A'B + BC_{in} + C_{in}A'$$

Realization of Full-Adder:

With basic Gates:

II. FULL ADDER

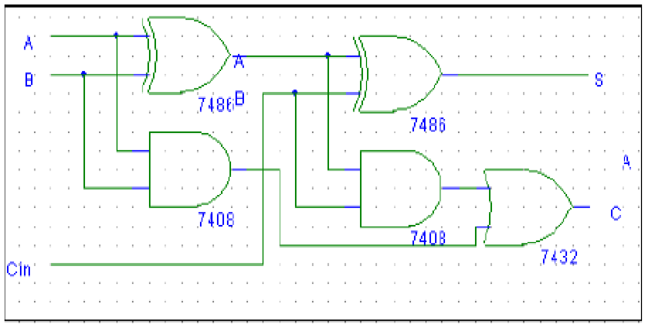
TRUTH TABLE

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

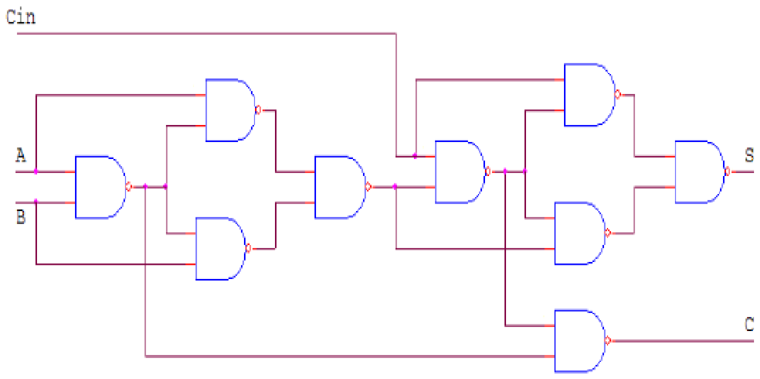
BOOLEAN EXPRESSIONS:

$S = A \oplus B \oplus C$

$C = A\ B + B\ Cin + A\ Cin$



ii) NAND GATES



Realization of Full-Subtractor:

IV. FULL SUBTRACTOR

TRUTH TABLE

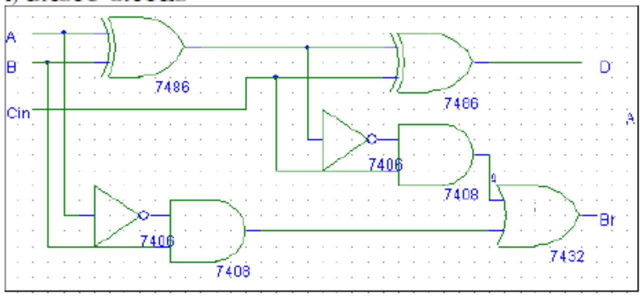
INPUTS			OUTPUTS	
A	B	Cin	D	Br
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

BOOLEAN EXPRESSIONS:

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

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

i) BASIC GATES



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

RESULT:

EXPERIMENT: 7 *DATE:*

BINARY TO GRAY CODE CONVERTER

AIM: To design Binary to Gray code converter and verification of truth table.

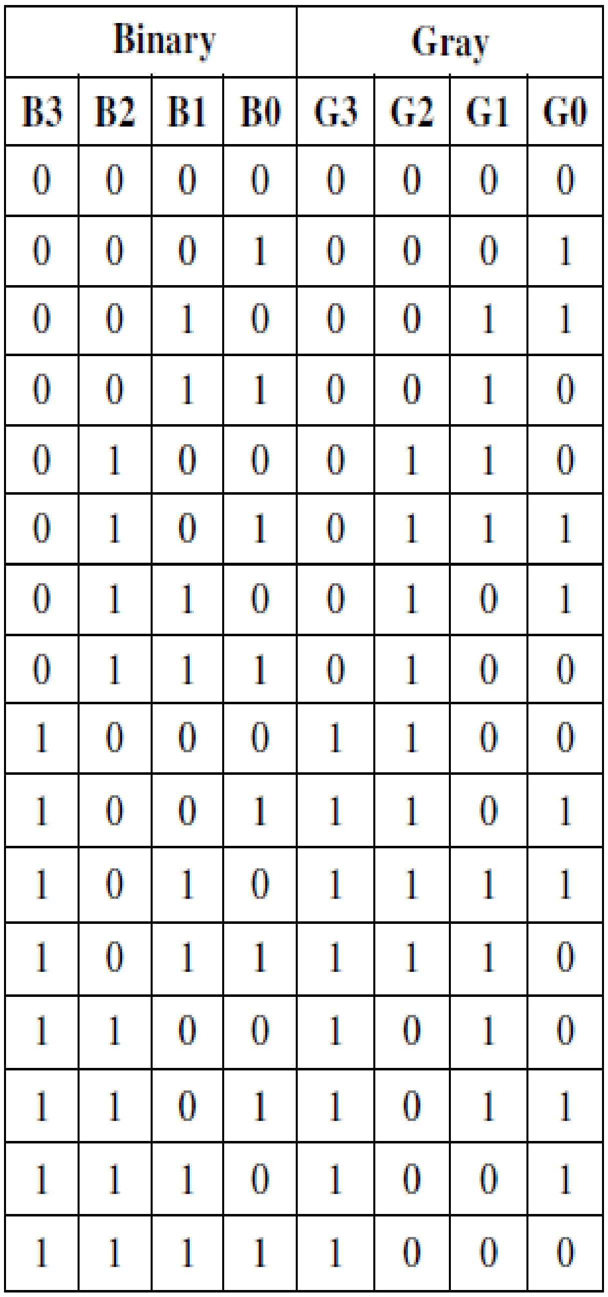
APPARATUS:

THEORY:

Code converter is a combinational circuit that translates the input code word into a new corresponding word.

Gray Code is one of the most important codes. It is a non-weighted code which belongs to a class of codes called minimum change codes. In this codes while traversing from one step to another step only one bit in the code group changes. In case of **Gray Code** two adjacent code numbers differs from each other by only one bit. The idea of it can be cleared from the table given below. As this code it is not applicable in any types of arithmetical operations but it has some applications in analog to digital converters and in some input/output devices.

Binary to Gray Code Conversion Table:



Decimal Number	Binary				Gray			
	B3	B2	B1	B0	G3	G2	G1	G0
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
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Binary to Gray Code Conversion from Conversion Table:

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

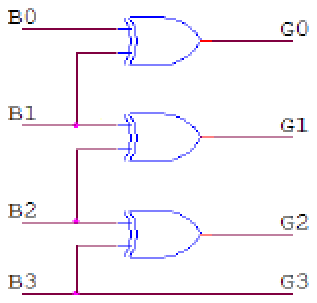
G1=B1 ⊕ B2

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

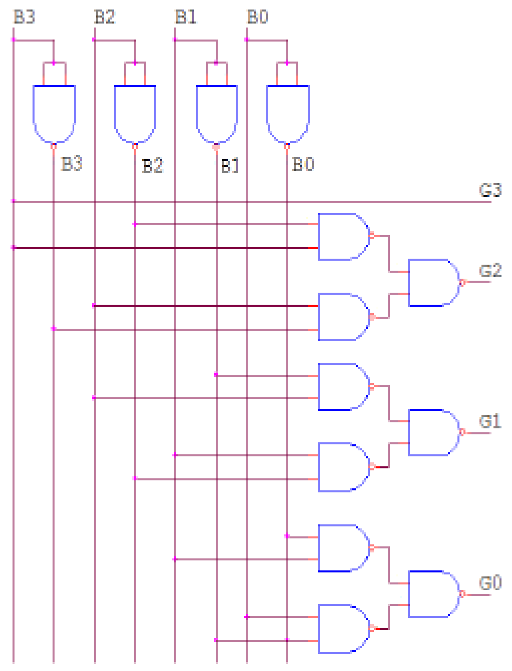
G0=B1 ⊕ B0

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

Realization Binary to Gray Code Converter Using Ex-OR Gates:



Realization of Binary to Gray Code Converter Using NAND Gates:



PROCEDURE:

1. Construct Binary to Gray code Conversion table as shown in Table.
2. Deriver Boolean Expression for each output variables(G_0,G_1,G_2 and G_3).
3. Check the components for their working.
4. Insert the appropriate IC into the IC base.
5. Make connections as shown in the circuit diagram.
6. Provide the input data via the input switches and observe the output on output LEDs

RESULT:

EXPERIMENT: 8 ***DATE:***

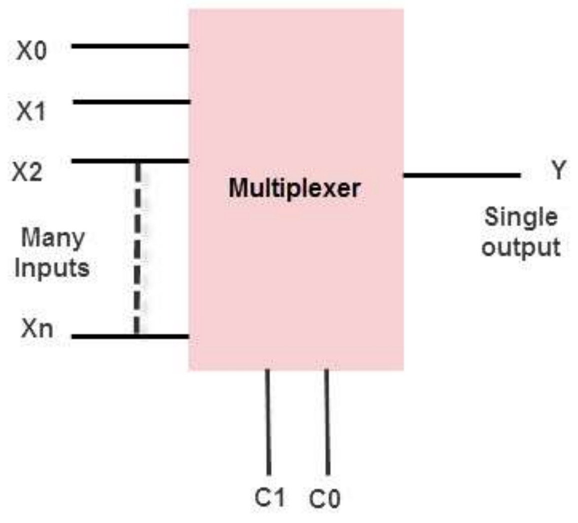
DESIGN OF MULTIPLEXER CIRCUIT

AIM: To design a combinational circuit for 4X1 Multiplexer using NAND gates and verify the truth table

APPARATUS:

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.

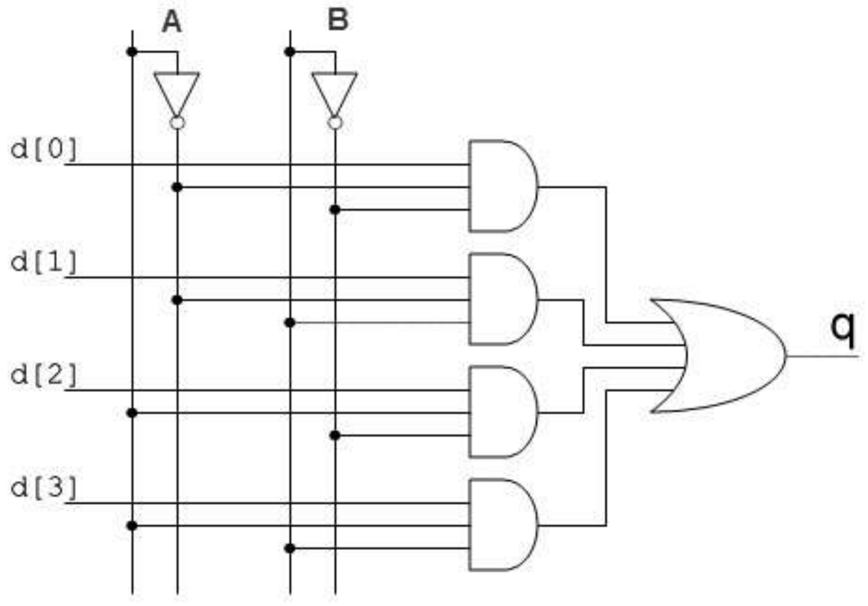


The 4X1 multiplexer comprises 4-input bits, 1- output bit, and 2- Selection lines. The four input bits are namely D_0, D_1, D_2 and D_3, respectively; only one of the input bit is transmitted to the output. The out 'q' depends on the value of selection input AB. The selection bit pattern AB decides which of the input data bit should transmit the output. The following figure shows the 4X1 multiplexer circuit diagram using AND gates. For example, when the control bits AB =00, then the higher AND gate are allowed while remaining AND gates are restricted. Thus, data input D_0 is transmitted to the output 'q"

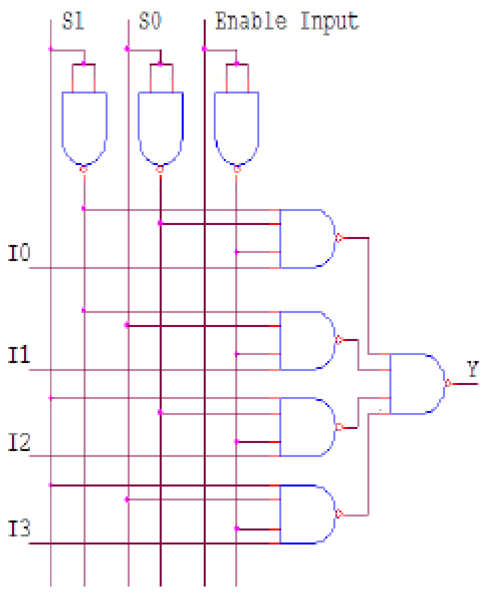
Input Selection Table:

A	***B***	***Output(q)***
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Realization of 4X1 Multiplexer using Basic gates:



Realization of 4X1 Multiplexer using NAND gates with Enable Input :



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

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

RESULT:

EXPERIMENT: 9 ***DATE:***

DESIGN OF DEMULTIPLEXER CIRCUIT

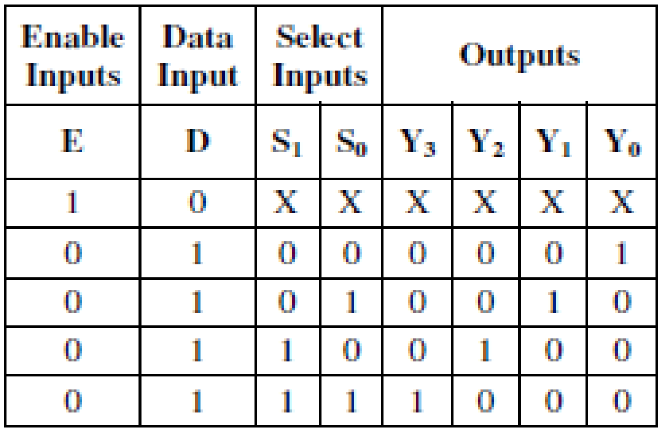
AIM: To design a combinational circuit for 1X4 Demultiplexer and verify its truth table.

APPARATUS:

THEORY:

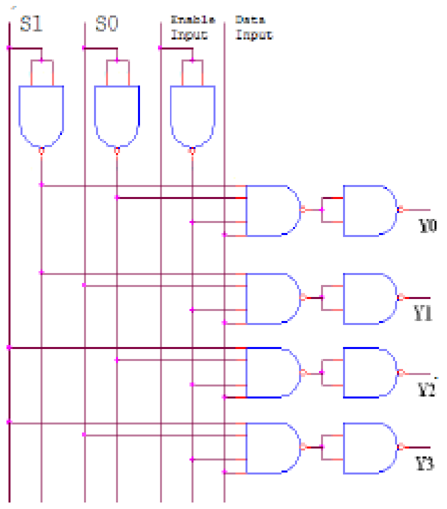
De-multiplexers perform the opposite function of multiplexers. They transfer a small number of information units (usually one unit) over a larger number of channels under the control of selection signals. The general de-multiplexer circuit has 1 input signal, n control/select signals and 2^n output signals.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

Truth Table for 1X4 Demultiplexer using Enable Input:



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

Realization of 1X4 Demultiplexer using Enable Input



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

RESULT:

EXPERIMENT:10 ***DATE:***

DESIGN OF FLIPFLOPS

AIM: To Construct the basic SR and D Flip-Flips and verify their truth tables.

APPARATUS:

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.

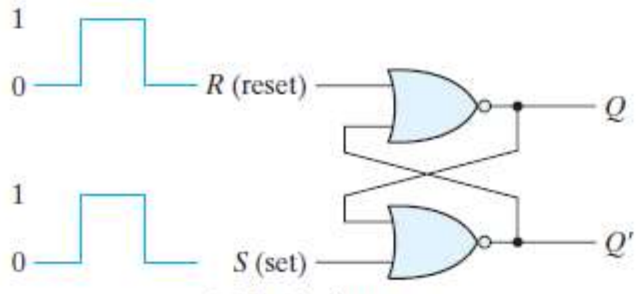
SR LATCH:

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 in the figures.

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.

SR Latch:

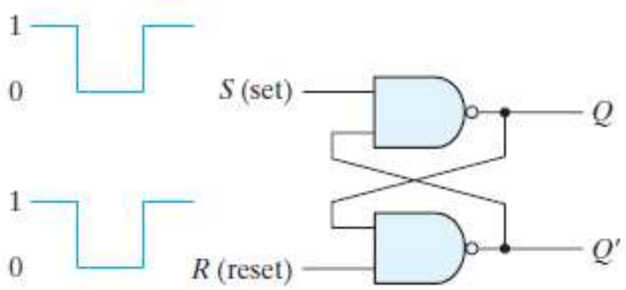


(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	(forbidden)

(b) Function table

SR Latch with NOR gates



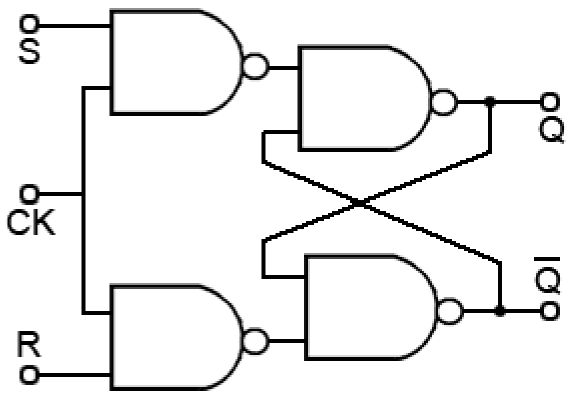
(a) Logic diagram

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

(b) Function table

SR Latch with NAND gates

SR Flip Flop



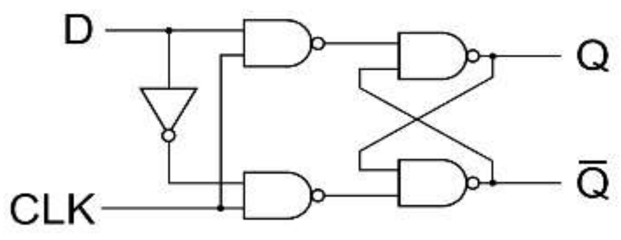
Functional Table of SR Flip flop:

Initial Conditions	Inputs (Pulsed)		Final Output
Q	S	R	Q (t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

Excitation Table for SR FF:

S	R	Q
0	0	No Change
0	1	Reset (0)
1	0	Set (1)
1	1	Indeterminate

D(Delay) -Flip Flop:



Functional Table of D- Flip flop:

S	R	Q(t + 1)
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table for D- FF:

D	Q
0	0
1	1

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

RESULT:

EXPERIMENT:10 ***DATE:***

DESIGN OF DECODERS

AIM: To design 2x4 Decoder circuit using basic logic gates and verify its truth table

APPARATUS:

THEORY:

A decoder is a combinational circuit that connects the binary information from ‘n’ input lines to a maximum of 2^n unique output lines. Decoder is also called a min-term generator/maxterm generator. A min-term generator is constructed using AND and NOT gates. The appropriate output is indicated by logic 1 (positive logic). Max-term generator is constructed using NAND gates. The appropriate output is indicated by logic 0 (Negative logic).

2:4 DECODER (MIN TERM GENERATOR):
Truth Table:

INPUT		OUTPUT			
A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

BOOLAEN EXPRESSIONS:

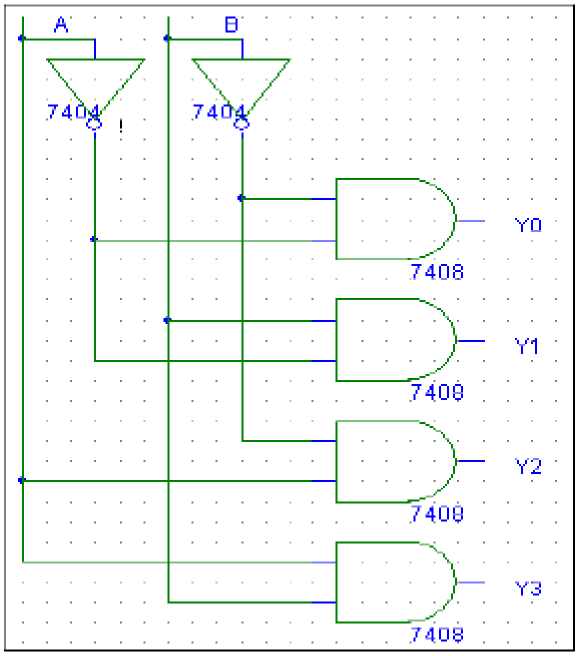
$$Y0 = \overline{AB}$$

$$Y1 = \overline{A}B$$

$$Y2 = A\overline{B}$$

$$Y3 = AB$$

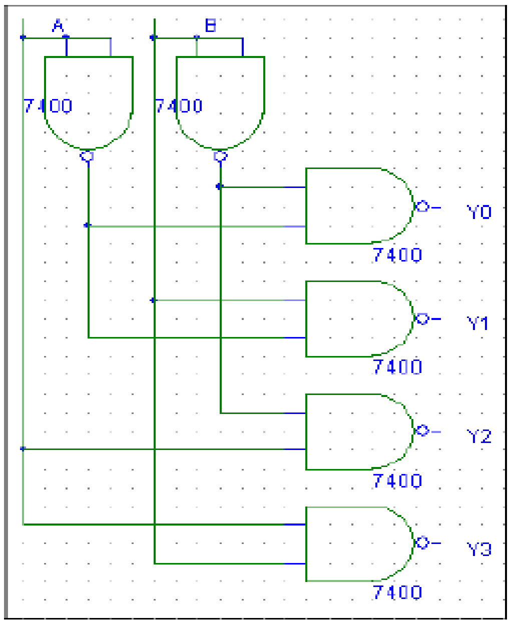
Realization of 2X4 Decoder using basic gates:



2:4 DECODER (MAX TERM GENERATOR):
Truth Table:

INPUT		OUTPUT			
A	B	Y0	Y1	Y2	Y3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

CIRCUIT DIAGRAM:



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

RESULT: