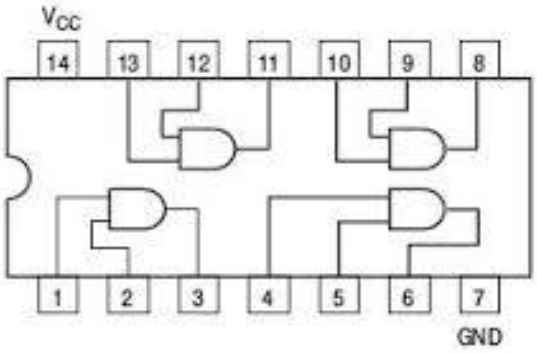
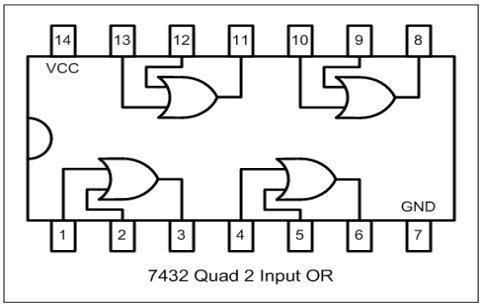
Pin configuration or connection diagrams for 7408, 7432,7404,7400,7402 and 7486

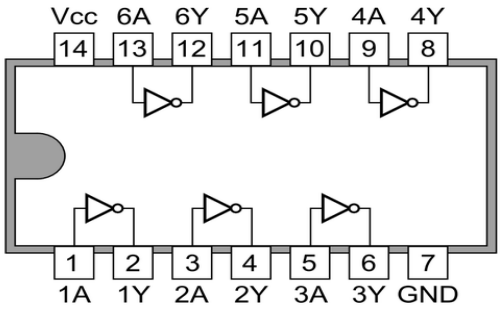
7408 AND gate



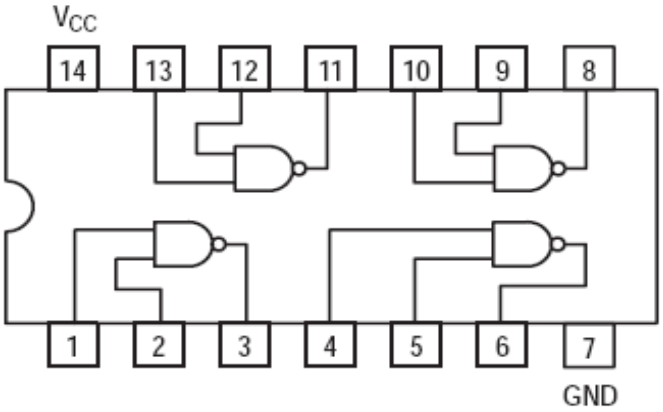
7432 OR gate

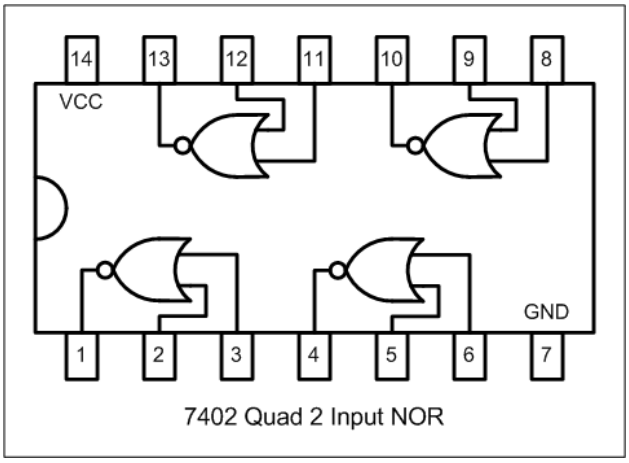


7404 Hex Inverters

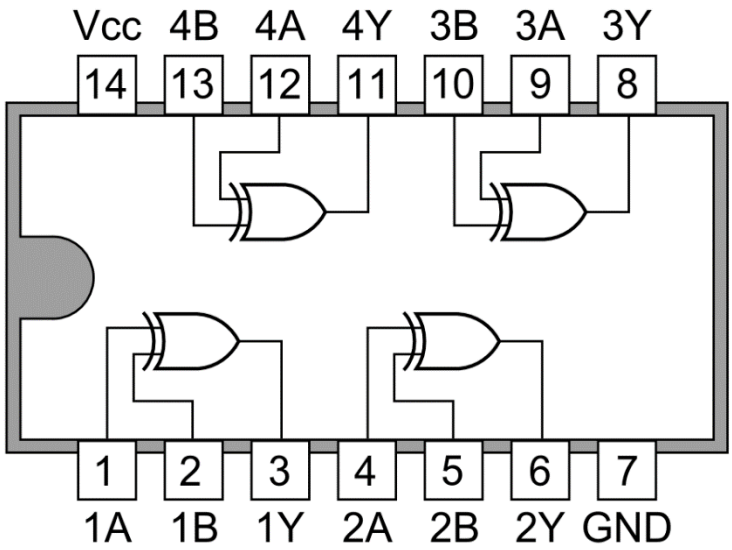


7400 NAND gate





7486 Quad 2-input ExOR Gates



Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester -** I

Experiment No: - 01

Aim: - To study and verify the Truth Table of various Logic Gates using IC's and realize Boolean expressions using gates.

Theory: -

Verification of basic gates:

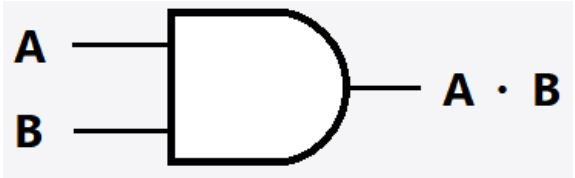
1) *Basic Gates:*

1) AND Gate:
Output of AND gate is HIGH or logic 1 only when it's both inputs are 1, otherwise its o/p is 0.

Truth Table:

A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Logic Symbol:

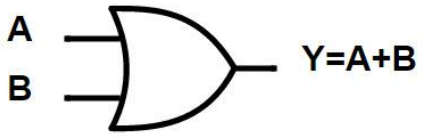


2) OR Gate:
Output of OR gate is LOW or logic 0 only when it's both inputs are 0, otherwise its o/p is 1.

Truth Table:

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

Logic Symbol:

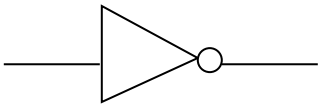


3) NOT Gate:
NOT gate has single i/p. o/p of NOT gate is inverse of its i/p .

Truth Table:

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

Logic Symbol:



2) <u>Universal Gates:</u>

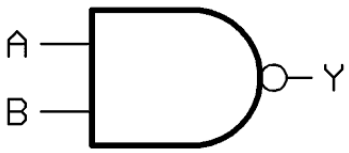
NAND & NOR are the universal gates because with the help of these gates we can implement all the gates. Also we can realize any Boolean expression with the same.

1) NAND Gate:
Output of AND gate is LOW or logic 0 only when it's both inputs are 1, otherwise its o/p is 1.

Truth Table:

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

Logic Symbol:

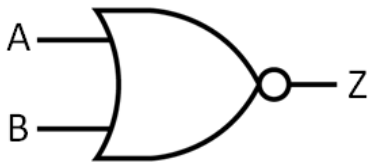


2) NOR Gate:
Output of NOR gate is LOW or logic 0 only when it's both inputs are 0, otherwise its o/p is 1.

Truth Table:

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

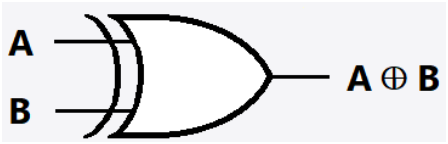
Logic Symbol:



3) <u>Special Gates:</u>

1) XOR Gate:
Output of XOR gate is LOW or logic 0 only when it's both inputs are same, otherwise its o/p is 1.

Logic Symbol:



Truth Table:

A	B	Y=A ⊙B
0	0	0
0	1	1
1	0	1
1	1	0

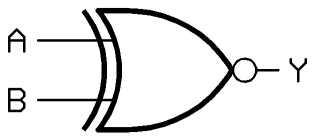
2) XNOR Gate:

Output of XNOR gate is LOW or logic 0 only when it's both inputs are different, otherwise it's o/p is 1.

Truth Table:

A	B	$Y=\overline{A \odot B}$
0	0	1
0	1	0
1	0	0
1	1	1

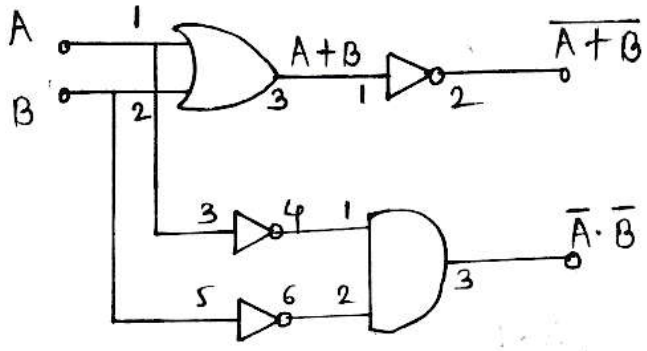
Logic Symbol:



Verification of Demorgan's Law:

1. $\overline{A + B} = \overline{A} \cdot \overline{B}$

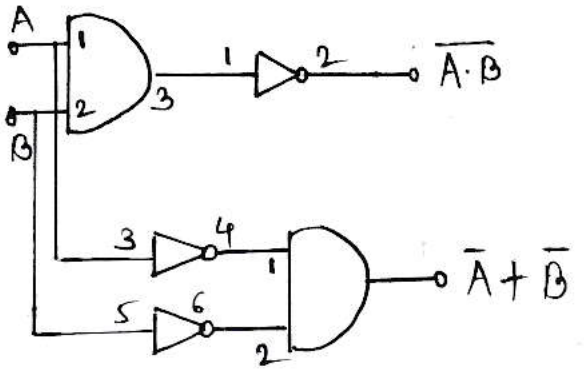
   Complement of Sum is equal to product of individual complements



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

2. $\overline{A \cdot B} = \overline{A} + \overline{B}$

   Complement of product is equal to sum of individual complements



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

Realization of Boolean expressions using gates:

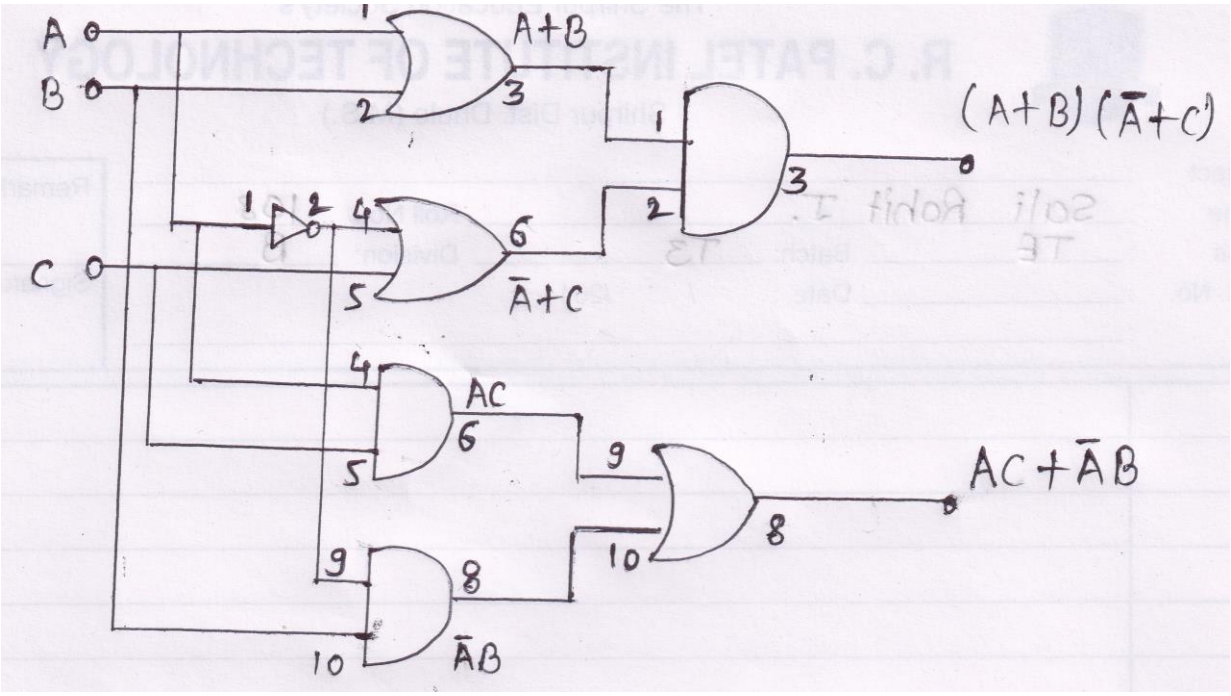
$$(A+B)(\bar{A}+C) = AC + \bar{A}B$$

$$\begin{aligned}
LHS &= (A+B)(\bar{A}+C) \\
&= A\bar{A} + AC + \bar{A}B + BC \\
&= 0 + AC + \bar{A}B + BC \\
&= AC + \bar{A}B + BC \\
&= AC + \bar{A}B + BC \cdot (1) \\
&= AC + \bar{A}B + BC(A + \bar{A}) \\
&= AC + \bar{A}B + ABC + \bar{A}BC \\
&= AC + ABC + \bar{A}B + \bar{A}BC \\
&= AC(1+B) + \bar{A}B(1+C) \\
&= AC(1) + \bar{A}B(1) \\
&= AC + \bar{A}B \\
&= RHS
\end{aligned}$$

Truth Table:

A	B	C	$\bar{A}$	A + B	$\bar{A}$ + C	(A + B).($\bar{A}$ + C)	AC	$\bar{A}$B	AC + $\bar{A}$B
0	0	0	1	0	1	0	0	0	0
0	0	1	1	0	1	0	0	0	0
0	1	0	1	1	1	1	0	1	1
0	1	1	1	1	1	1	0	1	1
1	0	0	0	1	0	0	0	0	0
1	0	1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0	0	0
1	1	1	0	1	1	1	1	0	1

Circuit Diagram:



Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester -** I

Experiment No: - 02

Aim: - To realize Basic Gates using Universal Gates.

Theory: -

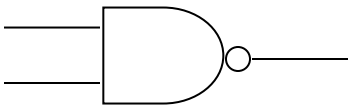
Implementation of basic gates by using NAND gate:

NAND Gate:
Output of AND gate is LOW or logic 0 only when it's both inputs are 1, otherwise its output is 1.

Truth Table:

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

Logic Symbol:



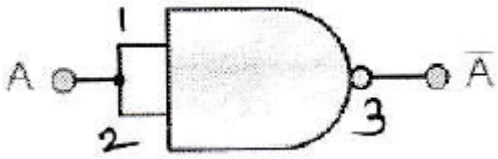
1) **NOT Gate:**

NOT gate has single input. The output of NOT gate is inverse of its input.

Truth Table:

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

Implementation:



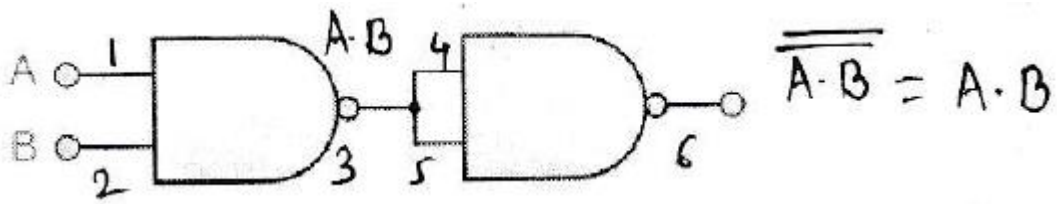
2) AND Gate:

Output of AND gate is HIGH or logic 1 only when both of its inputs are 1, otherwise its output is 0.

Truth Table:

A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Implementation:



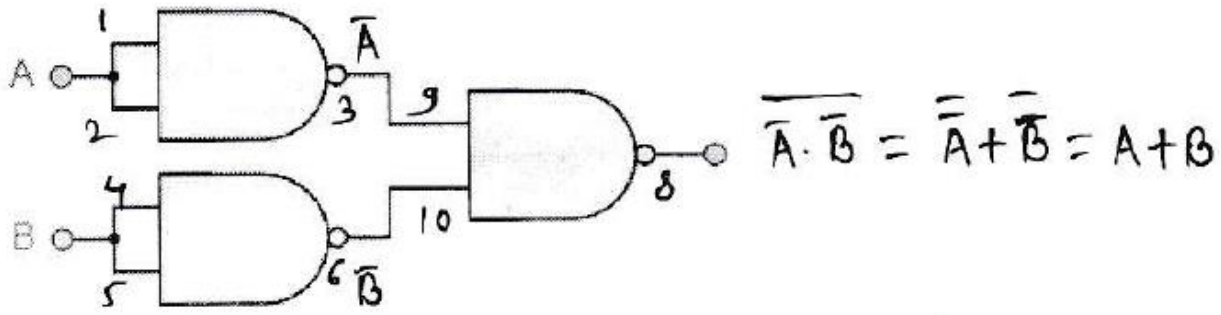
3) OR Gate:

Output of OR gate is LOW or logic 0 only when it's both inputs are 0, otherwise its output is 1.

Truth Table:

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

Implementation:



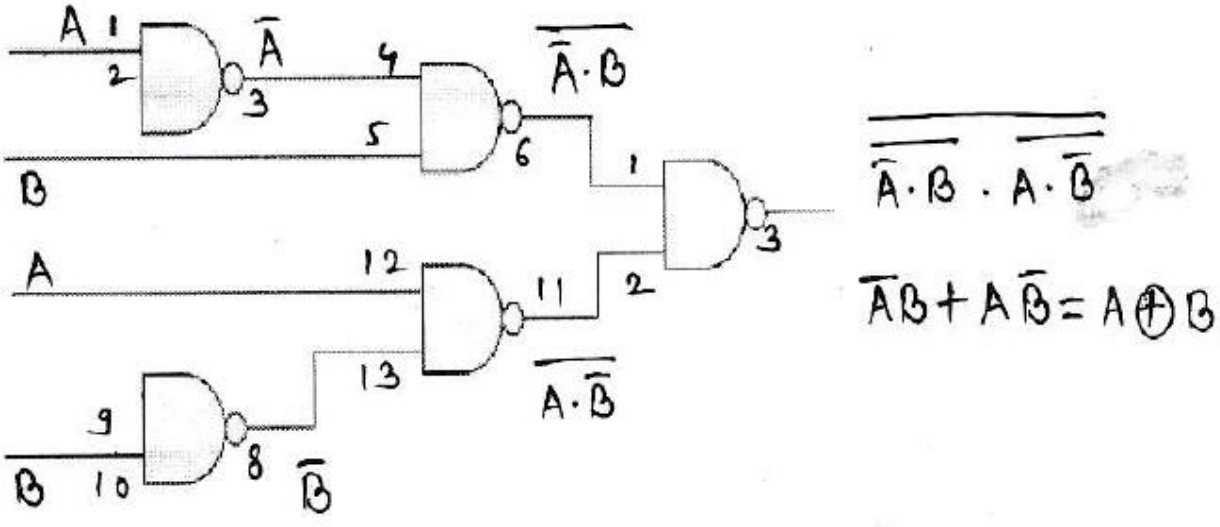
4) XOR Gate:

Output of XOR gate is LOW or logic 0 only when it's both inputs are same, otherwise its o/p is 1.

Truth Table:

A	B	Y=A⊙B
0	0	0
0	1	1
1	0	1
1	1	0

Implementation:



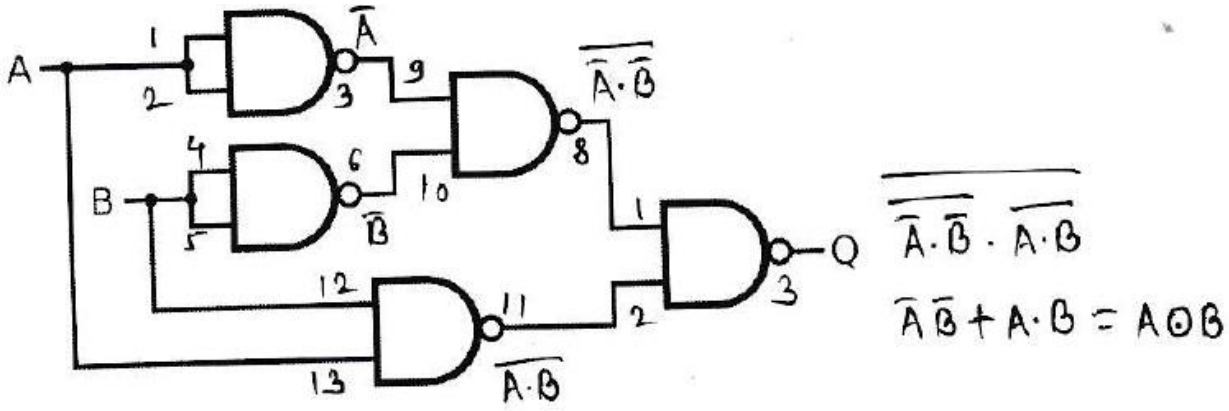
5) XNOR Gate:

Output of XNOR gate is LOW or logic 0 only when it's both inputs are different, otherwise its output is 1.

Truth Table:

A	B	$Y=\overline{A \odot B}$
0	0	1
0	1	0
1	0	0
1	1	1

Implementation:



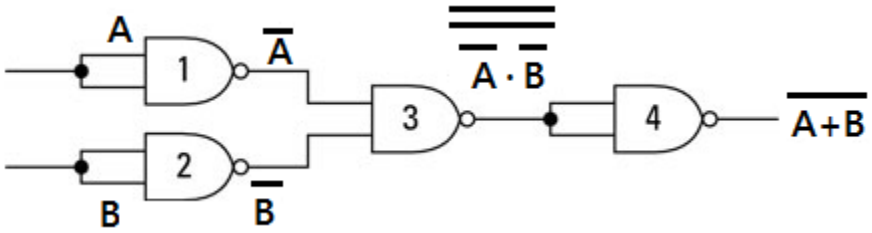
6) NOR Gate:

Output of NOR gate is LOW or logic 0 only when it's both inputs are 0, otherwise its o/p is 1.

Truth Table:

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

Implementation:



Implementation of basic gates by using NOR gate:

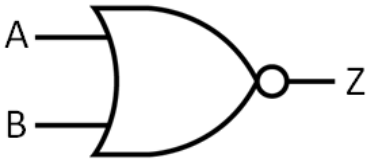
NOR Gate:

Output of NOR gate is LOW or logic 0 only when it's both inputs are 0, otherwise its output is 1.

Truth Table:

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

Logic Symbol:



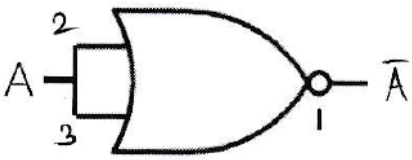
1) NOT Gate:

NOT gate has single input. The Output of NOT gate is inverse of its input.

Truth Table:

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

Implementation:



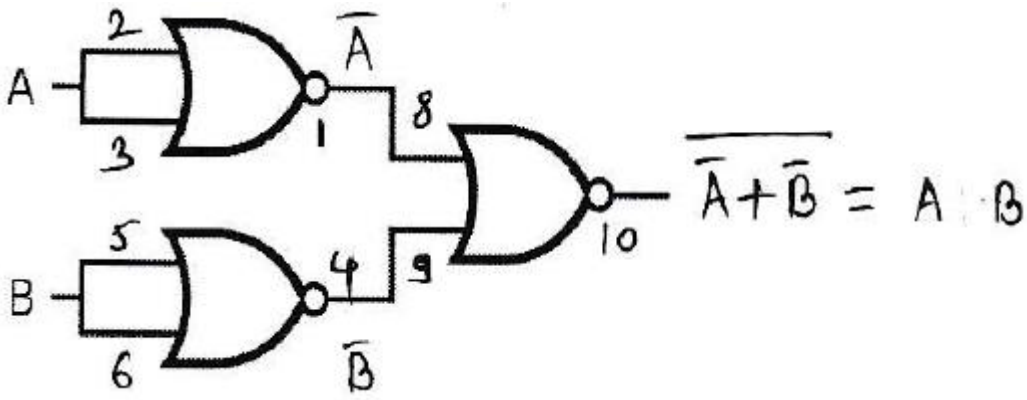
2) AND Gate:

Output of AND gate is HIGH or logic 1 only when it's both inputs are 1, otherwise its output is 0.

Truth Table:

A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Implementation:



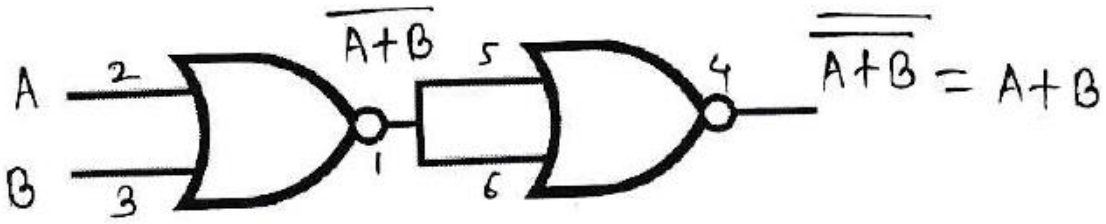
3) OR Gate:

Output of OR gate is LOW or logic 0 only when it's both inputs are 0, otherwise its output is 1.

Truth Table:

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

Implementation:



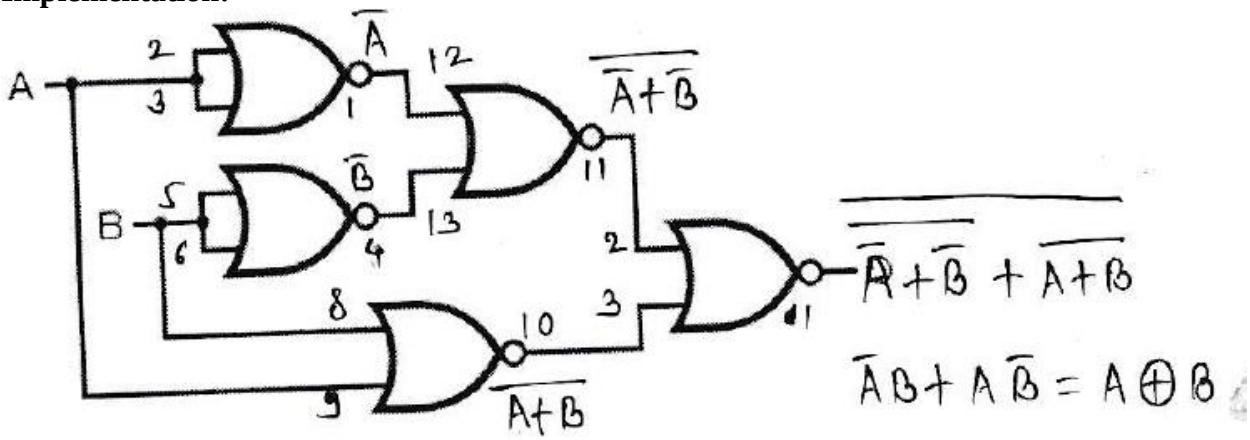
4) XOR Gate:

Output of XOR gate is LOW or logic 0 only when it's both inputs are same, otherwise its o/p is 1.

Truth Table:

A	B	Y=A⊙B
0	0	0
0	1	1
1	0	1
1	1	0

Implementation:



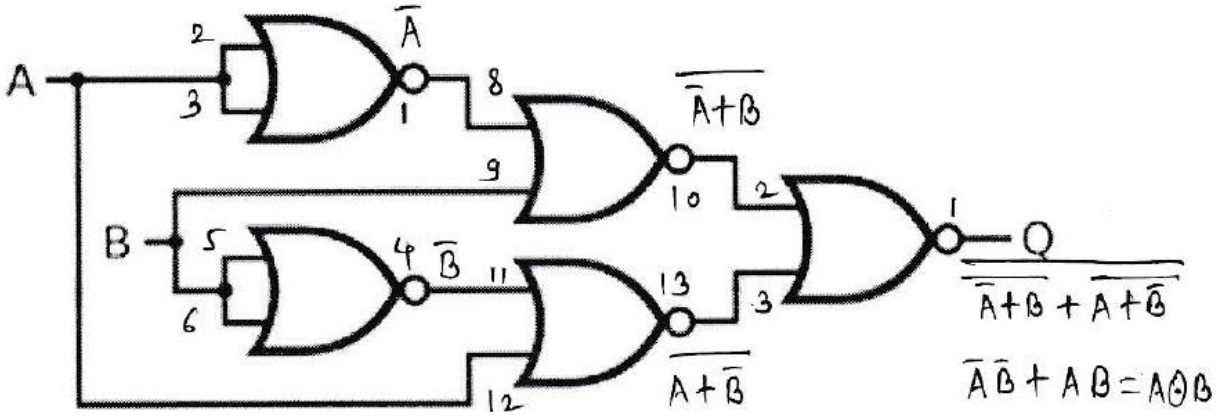
5) XNOR Gate:

Output of XNOR gate is LOW or logic 0 only when it's both inputs are different, otherwise its output is 1.

Truth Table:

A	B	$Y=\overline{A \odot B}$
0	0	1
0	1	0
1	0	0
1	1	1

Implementation:



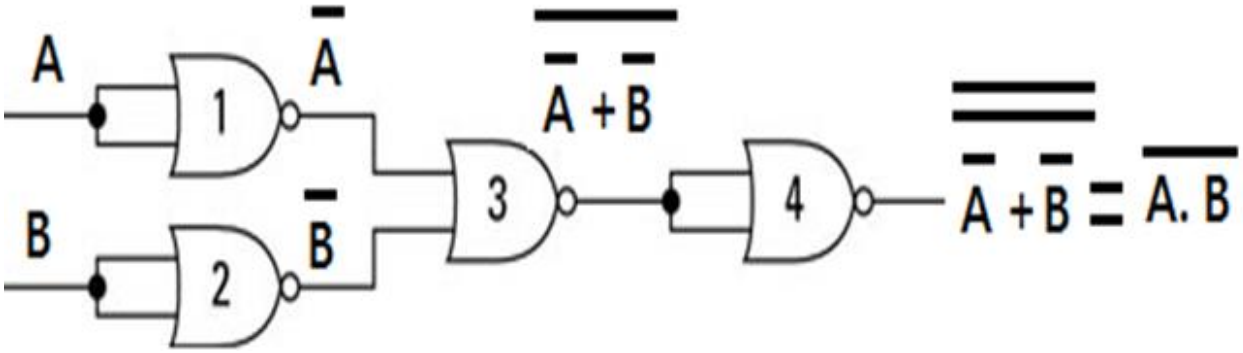
6) NAND Gate:

Output of AND gate is LOW or logic 0 only when it's both inputs are 1, otherwise its o/p is 1.

Truth Table:

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

Implementation:



Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester** - I

Experiment No: - 03

Aim: - To realize arithmetic circuits i) Half Adder ii) Full Adder iii) Half Subtractor iv) Full Subtractor.

Theory: -

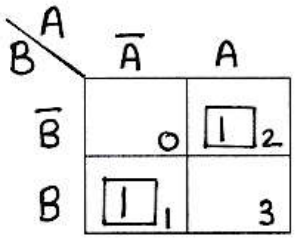
Implementation of Half Adder:

Half adder is a logic circuit used to perform addition of two single bit numbers.

Truth Table:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

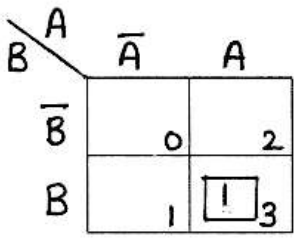
Kmap for Sum:



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

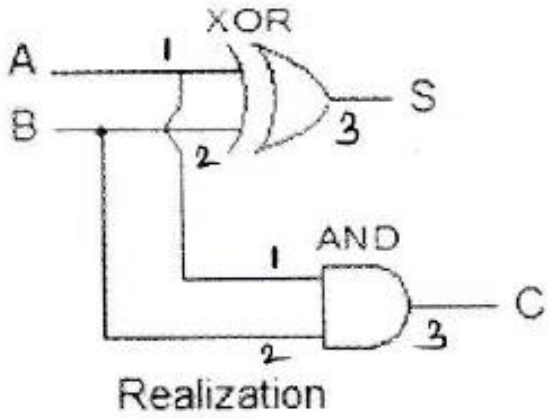
$\boxed{Sum = A \oplus B}$

Kmap for Carry



$\boxed{Carry = AB}$

Circuit Diagram:

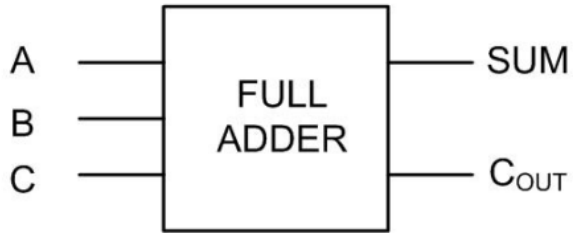


Implementation of Full Adder:

Full adder is a logic circuit used to perform addition of multi bit numbers. To overcome over the drawback of half adder (which doesn't perform multibit addition) the full adder is used. It considers the previous carry i.e. carry generated from previous addition.

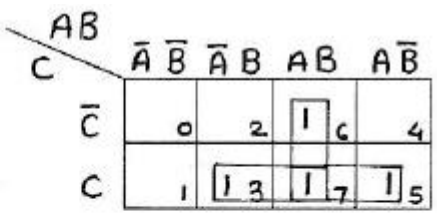
Truth Table:

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



Kmap for Sum:

Kmap for Carry:

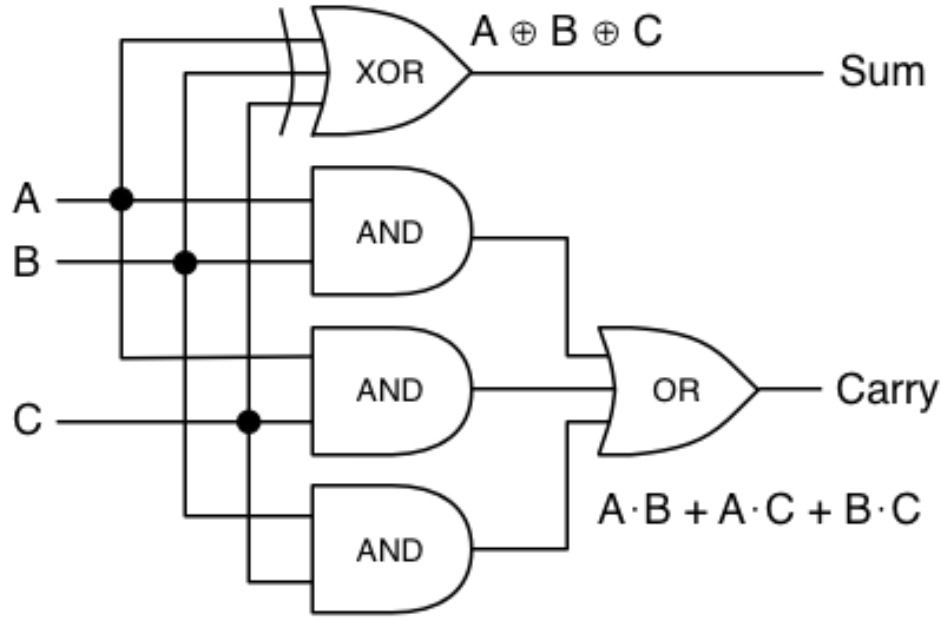


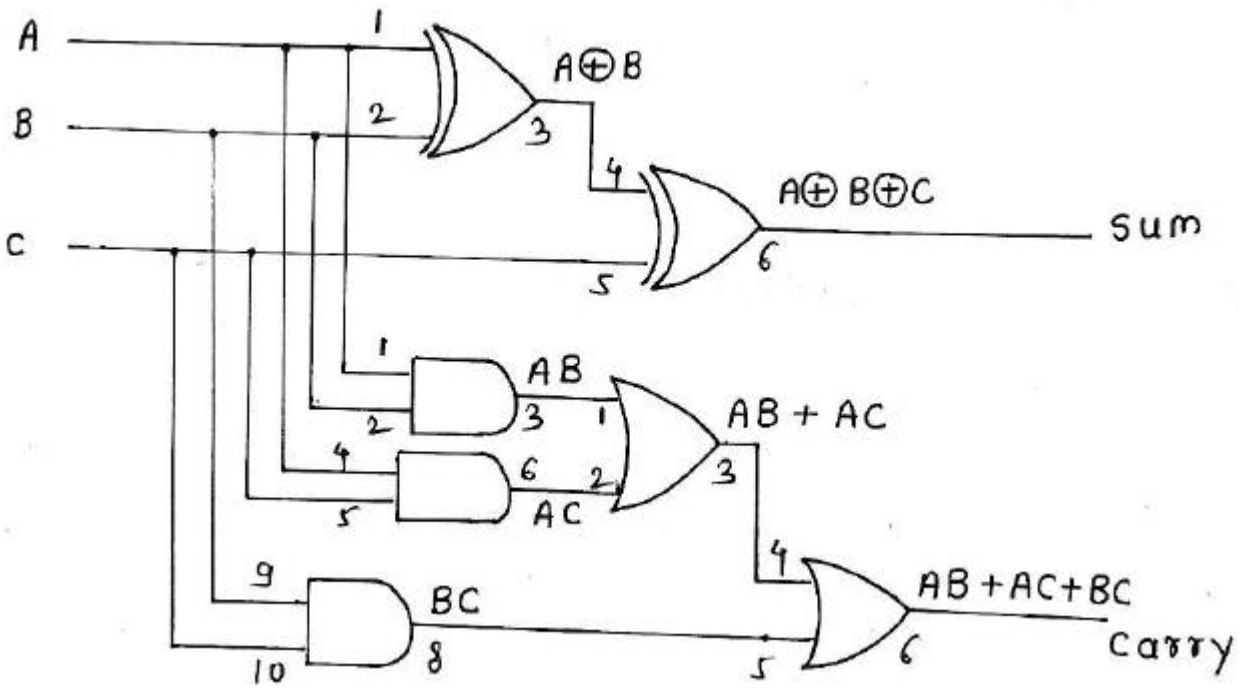
$$\text{Sum: } \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$$

$$\boxed{\text{Sum} = A \oplus B \oplus C}$$

$$\boxed{\text{Carry} = AB + AC + BC}$$

Circuit Diagram:





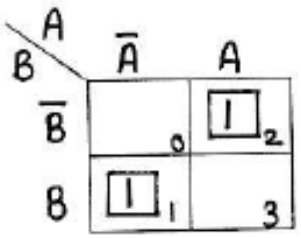
Implementation of Half Substractor:

Half subtractor is a logic circuit used to perform subtraction of two single bit numbers.

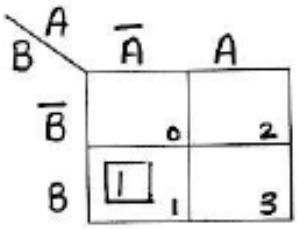
Truth Table:

A	B	Diff.	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Kmap for Difference



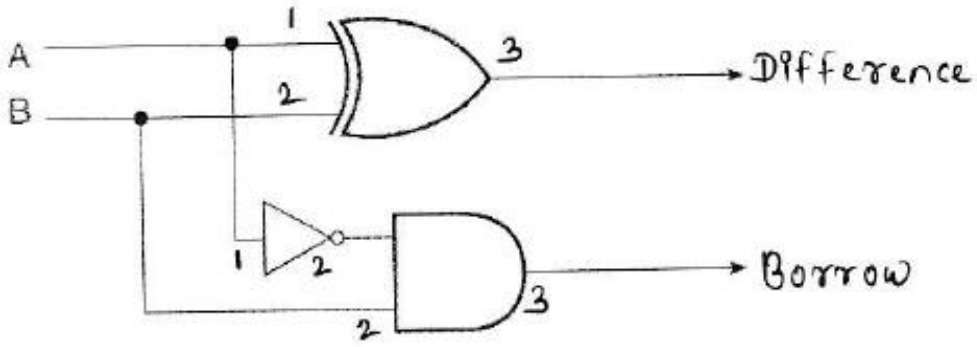
Kmap for Borrow



Difference = $A \oplus B$

Borrow = $\bar{A}B$

Circuit Diagram:

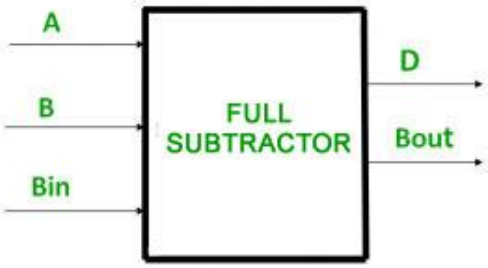


Implementation of Full Substractor:

Full Substractor is a logic circuit which performs subtraction of multibit numbers. To overcome over the drawback of half Substractor (which doesn't perform multibit subtraction) the full adder is used. It considers the previous borrow i.e. borrow generated from previous subtraction.

Truth Table:

A	B	C	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



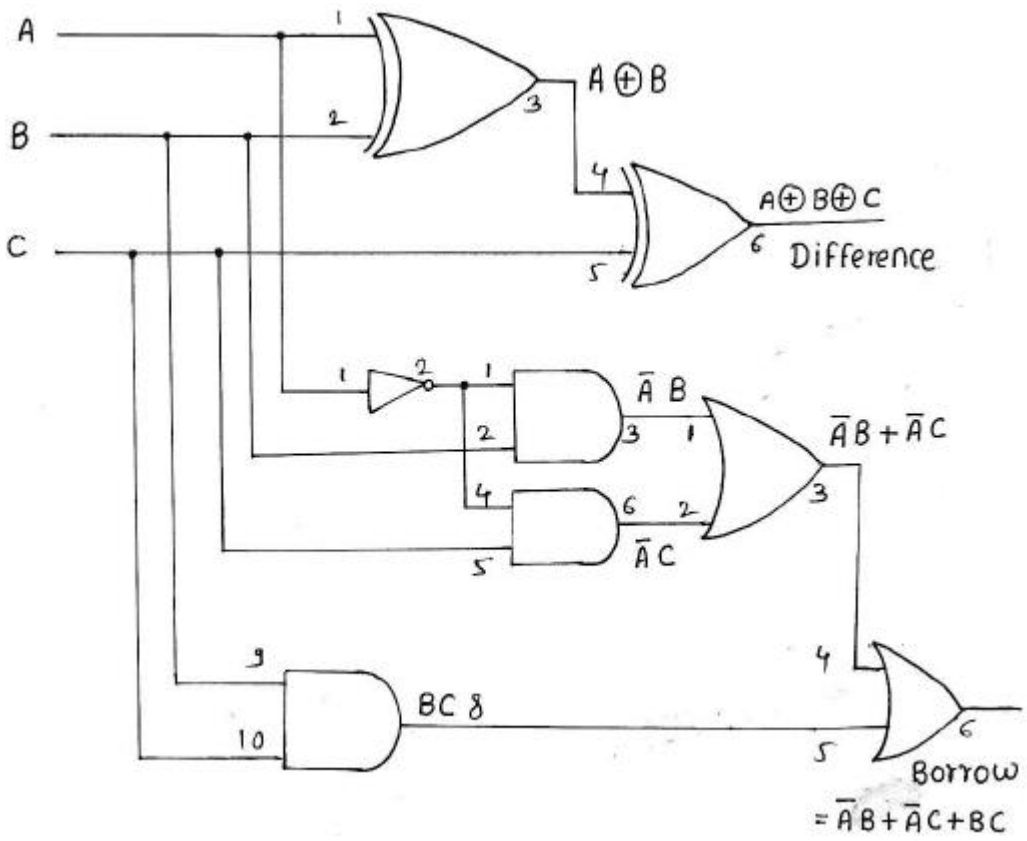
Kmap for Difference:

Kmap for Borrow:

Difference = $A \oplus B \oplus C$

Borrow = $\bar{A}C + \bar{A}B + BC$

Circuit Diagram:



Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester** - I

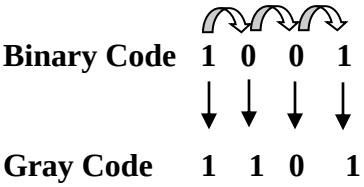
Experiment No: - 04

Aim: - To realize Binary to Gray Code and Gray to Binary Code Converter.

Theory: -

Binary to Gray Code Converter:

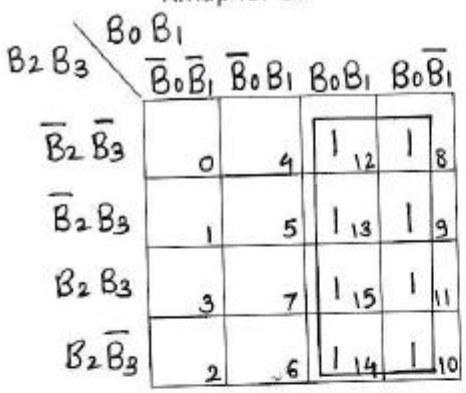
For E.g. Convert $(1001)_2$ to Gray Code



Truth Table:

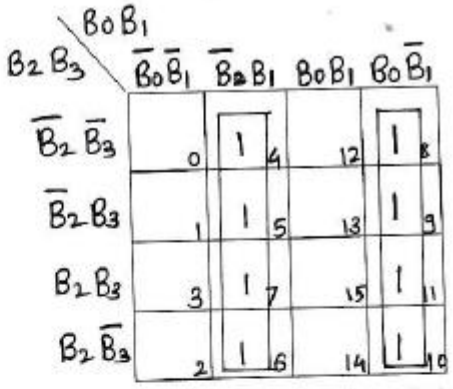
Dec No.	B_0	B_1	B_2	B_3	G_0	G_1	G_2	G_3	Dec No.
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1
2	0	0	1	0	0	0	1	1	3
3	0	0	1	1	0	0	1	0	2
4	0	1	0	0	0	1	1	0	6
5	0	1	0	1	0	1	1	1	7
6	0	1	1	0	0	1	0	1	5
7	0	1	1	1	0	1	0	0	4
8	1	0	0	0	1	1	0	0	12
9	1	0	0	1	1	1	0	1	13
10	1	0	1	0	1	1	1	1	15
11	1	0	1	1	1	1	1	0	14
12	1	1	0	0	1	0	1	0	10
13	1	1	0	1	1	0	1	1	11
14	1	1	1	0	1	0	0	1	9
15	1	1	1	1	1	0	0	0	8

Kmap for G_0:



$G_0 = B_0$

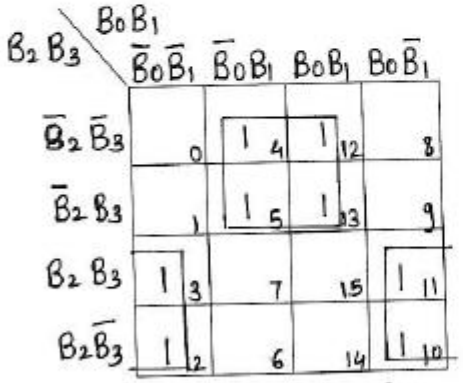
Kmap for G_1:



$G_1 = \bar{B}_0 B_1 + B_0 \bar{B}_1$

$G_1 = B_0 \oplus B_1$

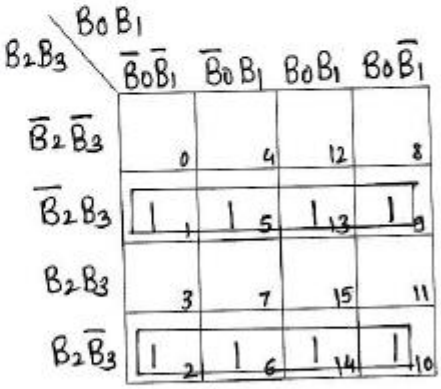
Kmap for G_2:



$G_2 = B_1 \bar{B}_2 + \bar{B}_1 B_2$

$G_2 = B_1 \oplus B_2$

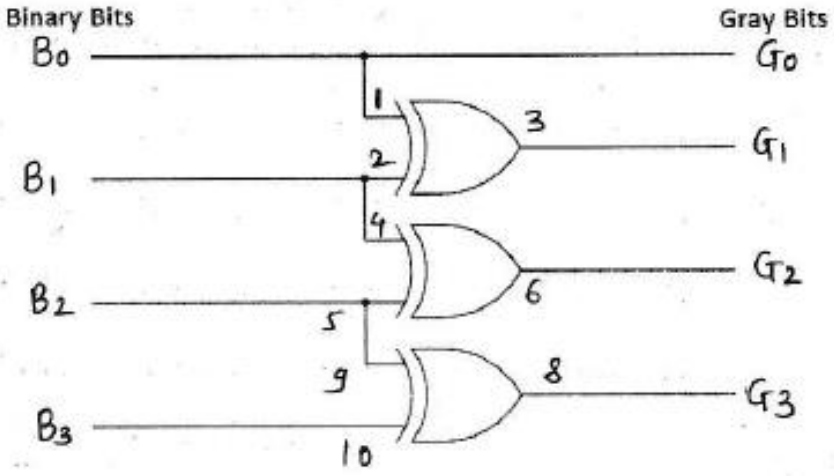
Kmap for G_3:



$G_3 = \bar{B}_2 B_3 + B_2 \bar{B}_3$

$G_3 = B_2 \oplus B_3$

Circuit Diagram:



Gray to Binary Code Converter:

For E.g. Convert (1110) Gray to Binary Code

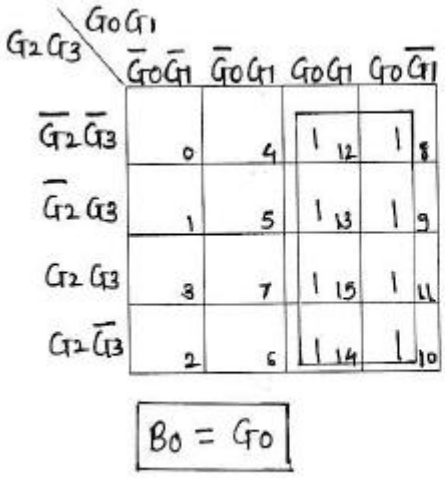
Gray Code 1 1 1 0

Binary Code 1 0 1 1

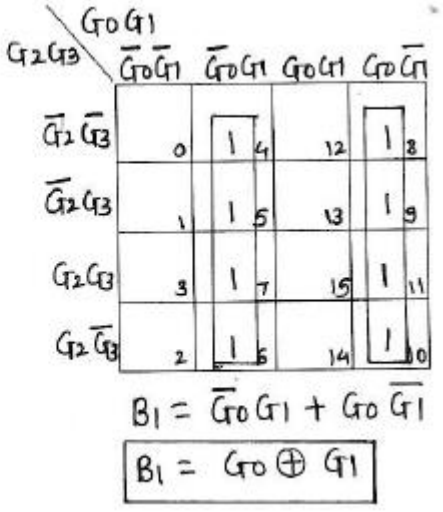
Truth Table:

Dec No.	G_0	G_1	G_2	G_3	B_0	B_1	B_2	B_3
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	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	0	0
7	0	1	1	1	0	1	0	1
8	1	0	0	0	1	1	1	1
9	1	0	0	1	1	1	1	0
10	1	0	1	0	1	1	0	0
11	1	0	1	1	1	1	0	1
12	1	1	0	0	1	0	0	0
13	1	1	0	1	1	0	0	1
14	1	1	1	0	1	0	1	1
15	1	1	1	1	1	0	1	0

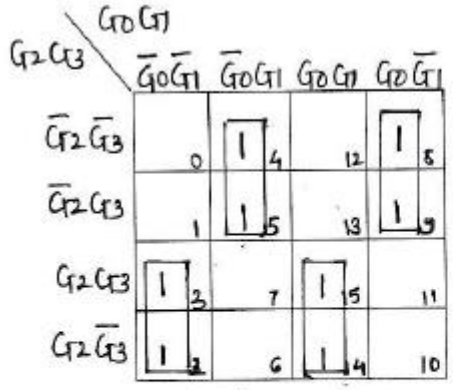
Kmap for B_0



Kmap for B_1



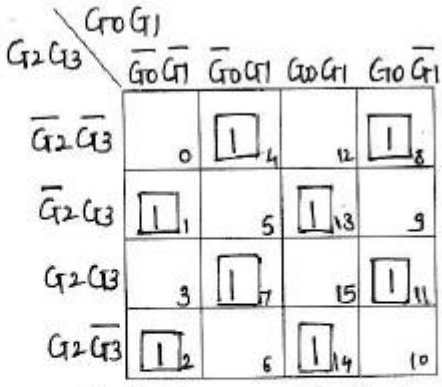
Kmap for B_2



$B_2 = \bar{G}_0\bar{G}_1G_2 + \bar{G}_0G_1\bar{G}_2 + G_0G_1G_2 + G_0\bar{G}_1\bar{G}_2$

$B_2 = G_0 \oplus G_1 \oplus G_2$

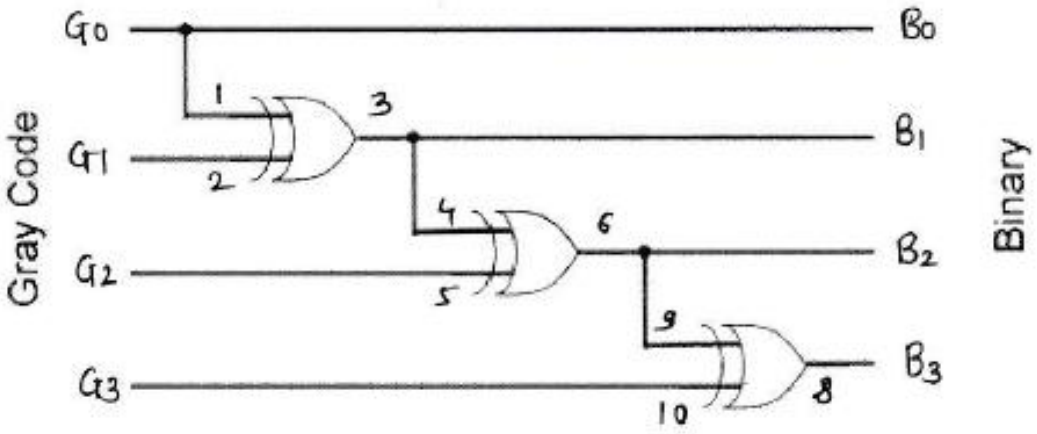
Kmap for B_3



$B_3 = \bar{G}_0\bar{G}_1\bar{G}_2G_3 + \bar{G}_0\bar{G}_1G_2\bar{G}_3 + \bar{G}_0G_1\bar{G}_2\bar{G}_3 + \bar{G}_0G_1G_2G_3 + G_0G_1\bar{G}_2G_3 + G_0G_1G_2\bar{G}_3 + G_0\bar{G}_1\bar{G}_2\bar{G}_3 + G_0\bar{G}_1G_2G_3$

$B_3 = G_0 \oplus G_1 \oplus G_2 \oplus G_3$

Circuit Diagram:



Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester** - I

Experiment No: - 05

Aim: - To realize 1 bit and 2 bit comparator

Theory: -

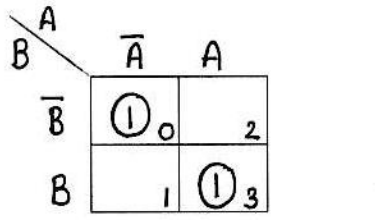
1 bit Comparator:

Truth Table:

Input		Output		
A	B	$Y_{A=B}$	$Y_{A>B}$	$Y_{A<B}$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

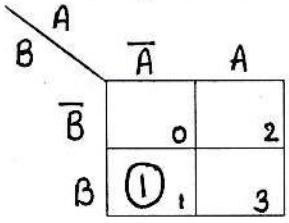
Kmap:

Kmap for $Y_{A=B}$



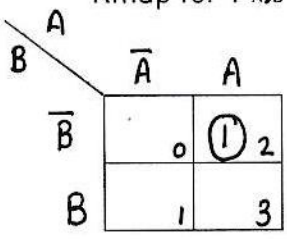
$Y_{A=B} = \overline{A}\,\overline{B} + AB$

Kmap for $Y_{A<B}$



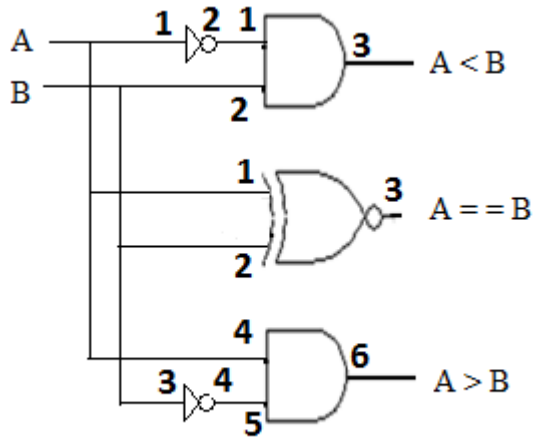
$Y_{A<B} = \overline{A}\,B$

Kmap for $Y_{A>B}$



$Y_{A>B} = A\overline{B}$

Circuit Diagram:



1-bit magnitude comparator

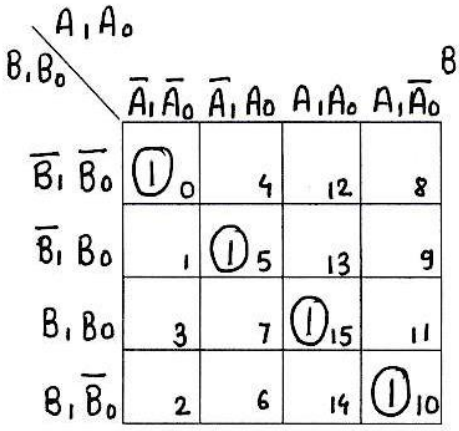
2 bit Comparator:

Truth Table:

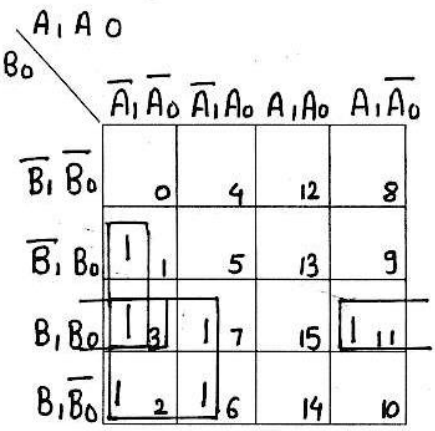
Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Kmap:

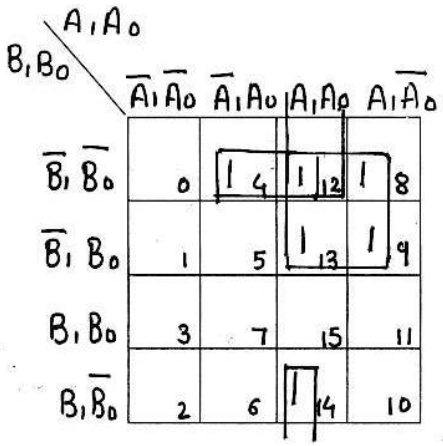
Kmap for $Y_{A=B}$



Kmap for $Y_{A<B}$



Kmap for $Y_{A>B}$



$$Y_{A<B} = \bar{A}_1 \cdot B_1 + \bar{A}_1 \bar{A}_0 B_0 + \bar{A}_0 B_0 B_1$$

$$Y_{A>B} = A_1 \bar{B}_1 + A_0 \bar{B}_0 \bar{B}_1 + A_1 \bar{B}_1 \bar{B}_0$$

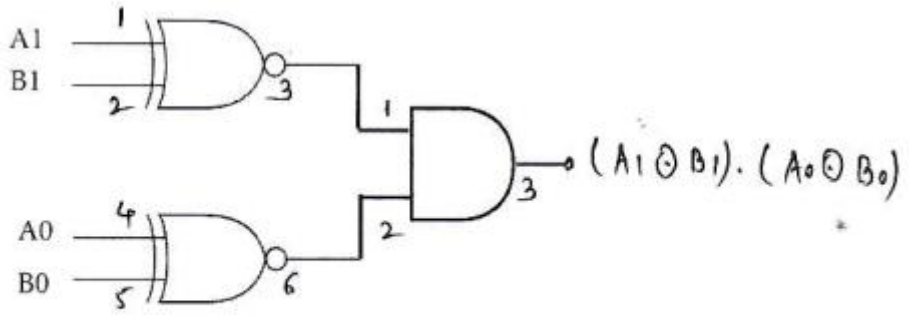
$$Y_{A=B} = \bar{A}_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0$$

$$= \bar{A}_1 \bar{B}_1 (\bar{A}_0 \bar{B}_0 + A_0 B_0) + A_1 B_1 (A_0 B_0 + \bar{A}_0 \bar{B}_0)$$

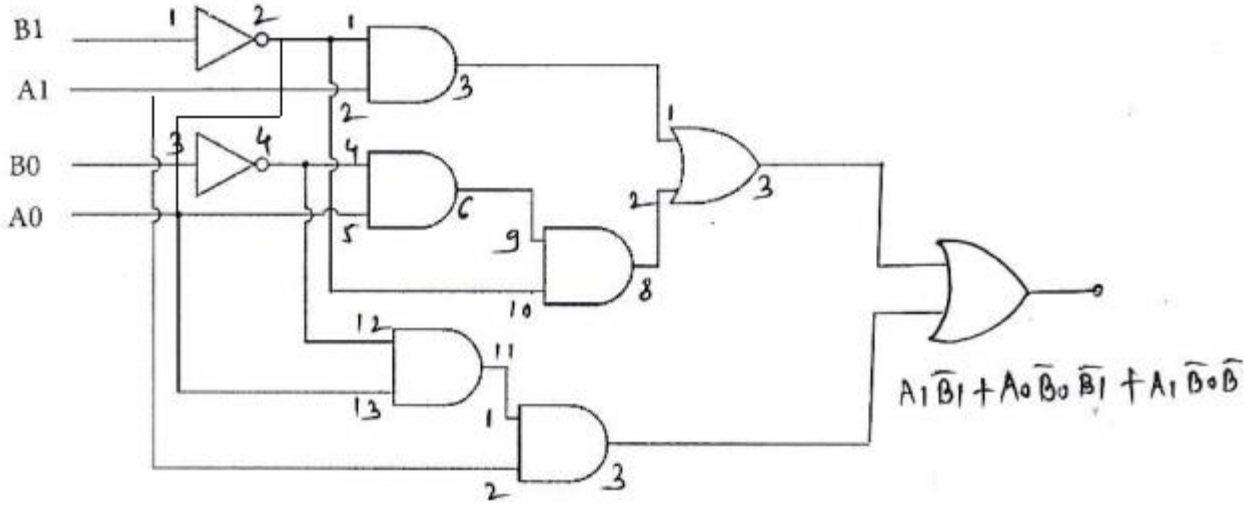
$$= \bar{A}_1 \bar{B}_1 (A_0 \odot B_0) + A_1 B_1 (A_0 \odot B_0)$$

$$= (A_1 \odot B_1) \cdot (A_0 \odot B_0)$$

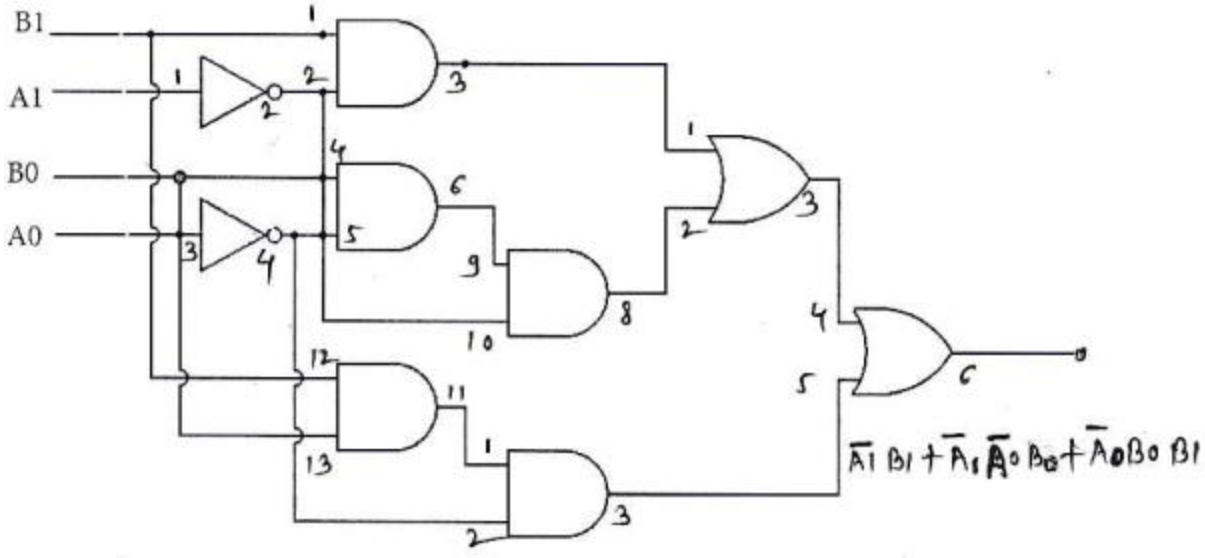
Circuit Diagram:



Circuit diagram for Y A=B



Circuit diagram for Y A>B



Circuit diagram for Y A<B

Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester -** I

Experiment No: - 06

Aim: - To realize Parity Generator and Detector.

Theory: -

Parity Generator and Checker:

A Parity Generator is a combinational logic circuit that generates the parity bit in the transmitter. On the other hand, a circuit that checks the parity in the receiver is called Parity Checker. A combined circuit or device of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data.

Even Parity and Odd Parity

The sum of the data bits and parity bits can be even or odd. In even parity, the added parity bit will make the total number of 1s an even number, whereas in odd parity, the added parity bit will make the total number of 1s an odd number.

Parity Generator

It is combinational circuit that accepts an n-1 bit data and generates the additional bit that is to be transmitted with the bit stream. This additional or extra bit is called as a Parity Bit.

In even parity bit scheme, the parity bit is ‘0’ if there are even number of 1s in the data stream and the parity bit is ‘1’ if there are odd number of 1s in the data stream.

In odd parity bit scheme, the parity bit is '1' if there are even number of 1s in the data stream and the parity bit is '0' if there are odd number of 1s in the data stream. Let us discuss both even and odd parity generators.

Even Parity Generator

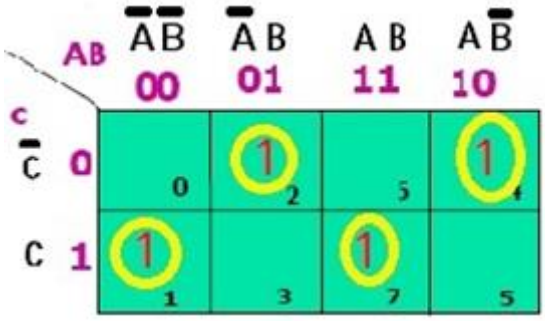
Let us assume that a 3-bit message is to be transmitted with an even parity bit. Let the three inputs A, B and C are applied to the circuit and output bit is the parity bit P. The total number of 1s must be even, to generate the even parity bit P.

The figure below shows the truth table of even parity generator in which 1 is placed as parity bit in order to make all 1s as even when the number of 1s in the truth table is odd.

Truth Table:

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Kmap:



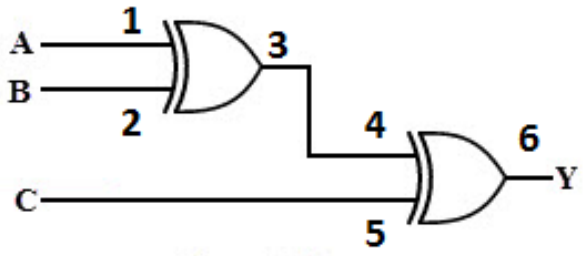
$$P = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$

$$= \bar{A}\,(\bar{B}\,C + B\,\bar{C}) + A\,(\bar{B}\,\bar{C} + B\,C)$$

$$= \bar{A}\,(B \oplus C) + A\,(\overline{B \oplus C})$$

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

Circuit Diagram:



Even Parity Detector (Checker)

Consider that three input message along with even parity bit is generated at the transmitting end. These 4 bits are applied as input to the parity checker circuit, which checks the possibility of error on the data. Since the data is transmitted with even parity, four bits received at circuit must have an even number of 1s.

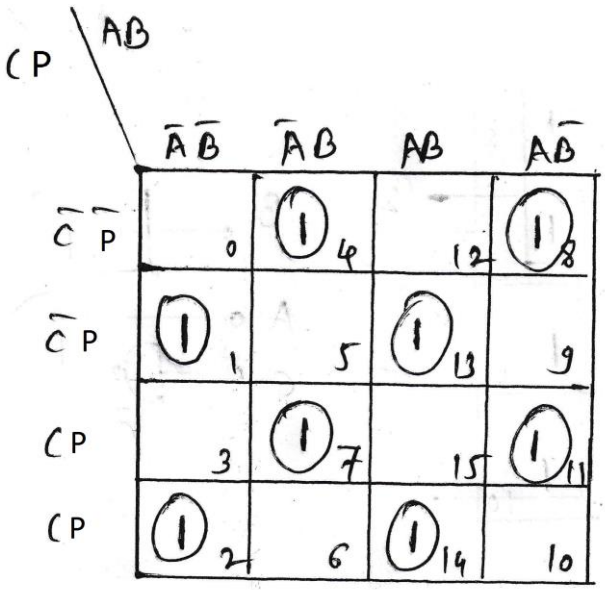
If any error occurs, the received message consists of odd number of 1s. The output of the parity checker is denoted by PEC (Parity Error Check).

The below table shows the truth table for the Even Parity Checker in which PEC = 1 if the error occurs, i.e., the four bits received have odd number of 1s and PEC = 0 if no error occurs, i.e., if the 4-bit message has even number of 1s.

Truth Table:

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Kmap:



$$F = \bar{A}\bar{B}(\bar{C}P + C\bar{P}) + \bar{A}B(\bar{C}\bar{P} + CP) + AB(\bar{C}P + C\bar{P}) + A\bar{B}(\bar{C}\bar{P} + CP)$$

$$= \bar{A}\bar{B}(C \oplus P) + \bar{A}B\overline{(C \oplus P)} + AB(C \oplus P) + A\bar{B}\overline{(C \oplus P)}$$

$$= (C \oplus P)(\bar{A}\bar{B} + AB) + \overline{(C \oplus P)}(\bar{A}B + A\bar{B})$$

$$= (C \oplus P)\overline{(A \oplus B)} + \overline{(C \oplus P)}(A \oplus B)$$

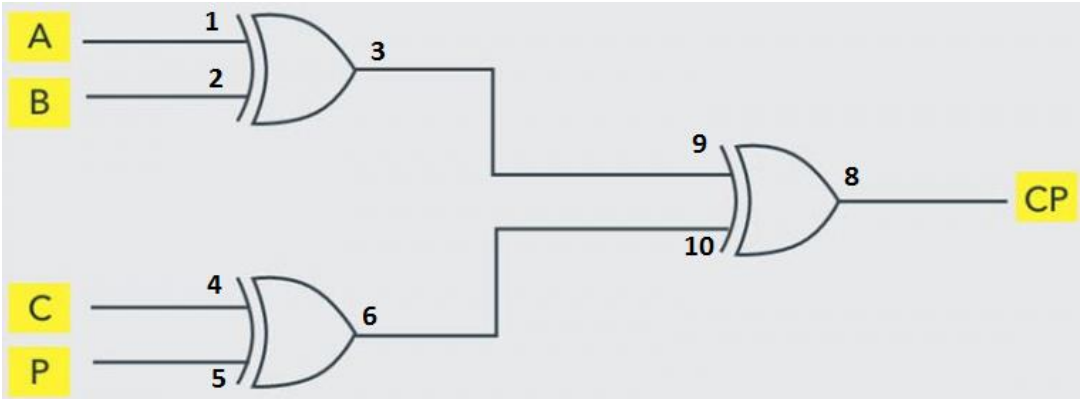
$$= E\bar{F} + \bar{E}F \quad (\text{Assume } E = C \oplus P \text{ and } F = A \oplus B)$$

$$= E \oplus F$$

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

$$F = A \oplus B \oplus C \oplus P$$

Circuit Diagram:



Odd Parity Generator

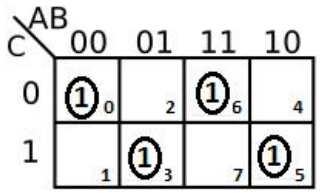
Let us consider that the 3-bit data is to be transmitted with an odd parity bit. The three inputs are A, B and C and P is the output parity bit. The total number of bits must be odd in order to generate the odd parity bit.

In the given truth table below, 1 is placed in the parity bit in order to make the total number of bits odd when the total number of 1s in the truth table is even.

Truth Table:

3-bit message			Odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	0	0

Kmap:



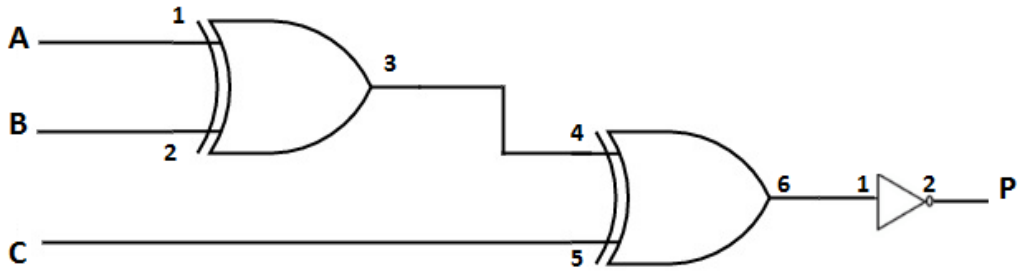
$$P = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,B\,C + A\,B\,\bar{C} + \bar{A}\,B\,C$$

$$= \bar{A}\,(\bar{B}\,\bar{C} + B\,C) + B\,(A\bar{C} + \bar{A}C)$$

$$= \bar{A}\,\overline{(B \oplus C)} + A\,(B \oplus C)$$

$$P = \overline{A \oplus B \oplus C}$$

Circuit Diagram:



Odd Parity Detector (Checker)

Consider that a three bit message along with odd parity bit is transmitted at the transmitting end. Odd parity checker circuit receives these 4 bits and checks whether any error are present in the data.

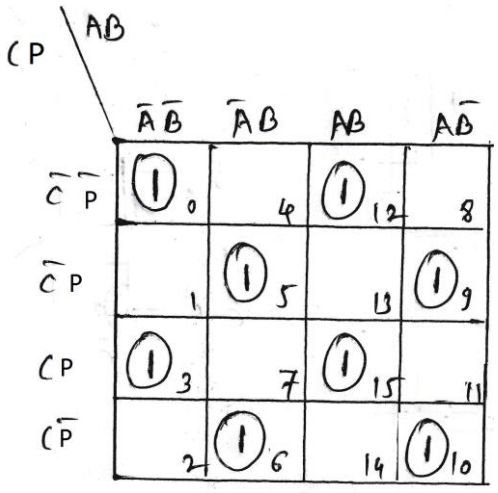
If the total number of 1s in the data is odd, then it indicates no error, whereas if the total number of 1s is even then it indicates the error since the data is transmitted with odd parity at transmitting end.

The below figure shows the truth table for odd parity generator where PEC =1 if the 4-bit message received consists of even number of 1s (hence the error occurred) and PEC= 0 if the message contains odd number of 1s (that means no error).

Truth Table:

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Kmap:



$$F = \bar{A}\bar{B}(\bar{C}\bar{P} + CP) + \bar{A}B(\bar{C}P + C\bar{P}) + AB(\bar{C}\bar{P} + CP) + A\bar{B}(\bar{C}P + C\bar{P})$$

$$= \bar{A}\bar{B}(\overline{C \oplus P}) + \bar{A}B(C \oplus P) + AB(\overline{C \oplus P}) + A\bar{B}(C \oplus P)$$

$$= (\overline{C \oplus P})(\bar{A}\bar{B} + AB) + (C \oplus P)(\bar{A}B + A\bar{B})$$

$$= (\overline{C \oplus P})(\overline{A \oplus B}) + (C \oplus P)(A \oplus B)$$

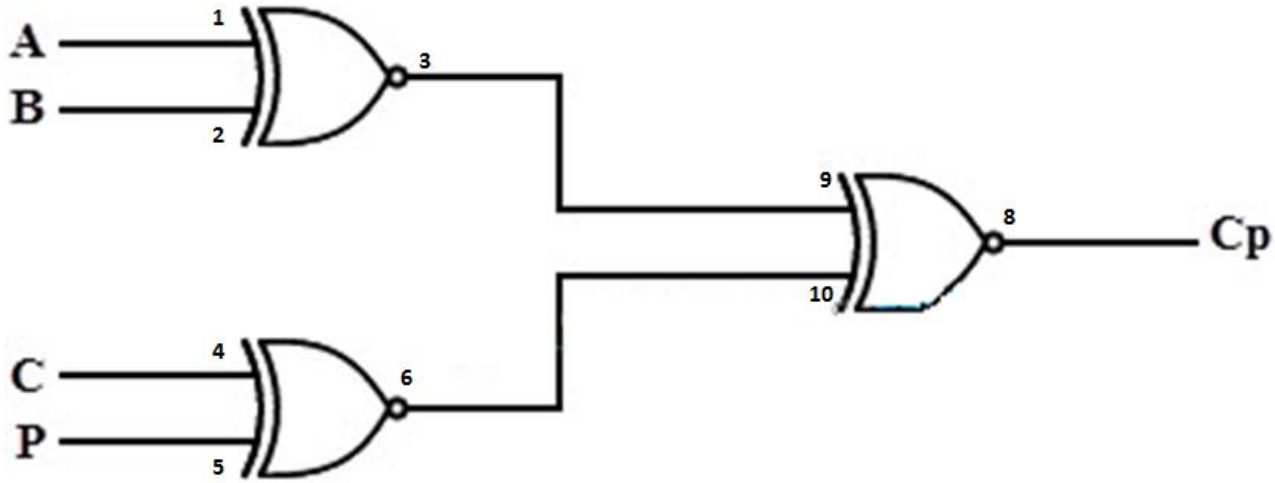
$$= \bar{E}\bar{F} + EF \quad (\text{Assume } E = C \oplus P \text{ and } F = A \oplus B)$$

$$= \overline{E \oplus F}$$

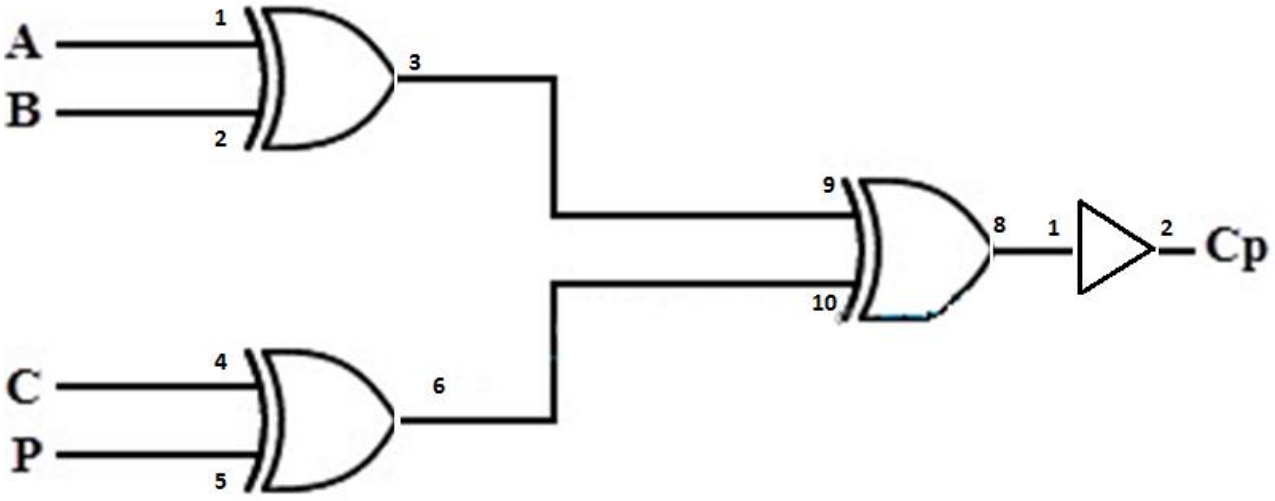
$$= \overline{C \oplus P \oplus A \oplus B}$$

$$F = \overline{A \oplus B \oplus C \oplus P}$$

Circuit Diagram:



OR



Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester -** I

Experiment No: - 07

Aim: - To study Multiplexer IC and realization of Full Adder using Multiplexer IC.

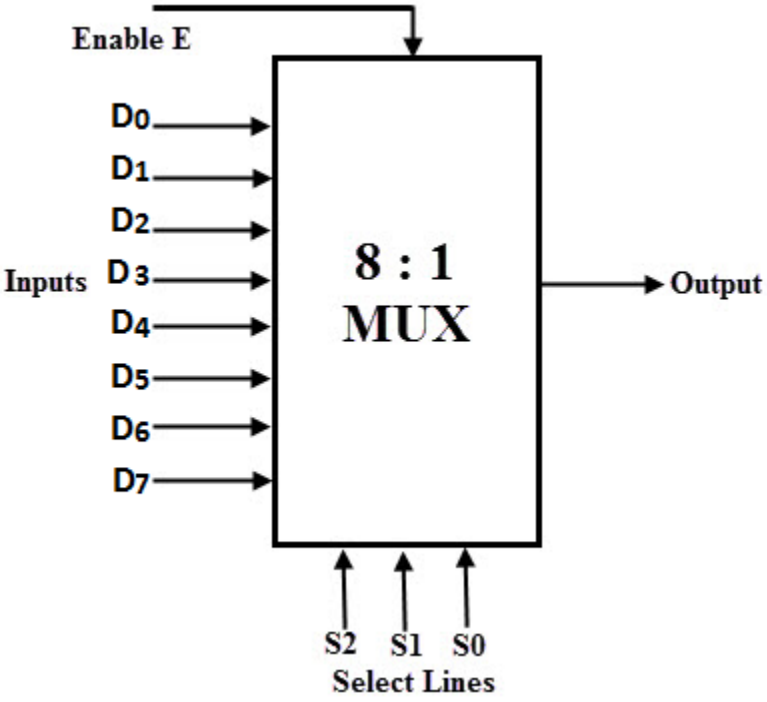
Theory: -

Verification of Multiplexer:

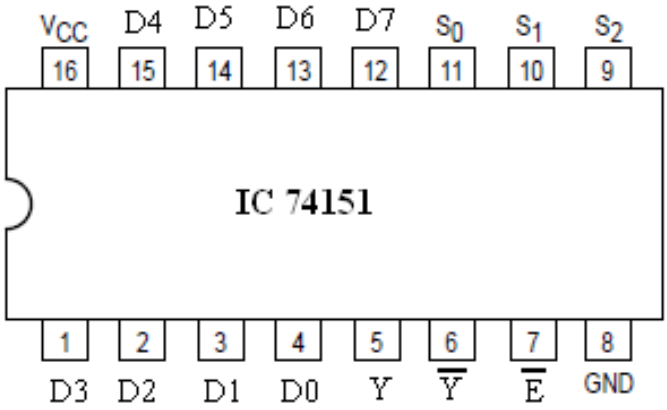
Multiplexer is a combinational logic circuit that performs multiplexing; it selects one of many analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2^n inputs has n select bits, which are used to select which input line to send to the output. An electronic multiplexer makes it possible for several signals to share one device or resource.

Truth Table:

Dec No.	S_2	S_1	S_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1



General symbol for 8: 1 Mux



Pin Configuration for IC 74151

$D_0 – D_7$: Input lines

S0 - S2: Select Lines

G or E: Strobe / Enable

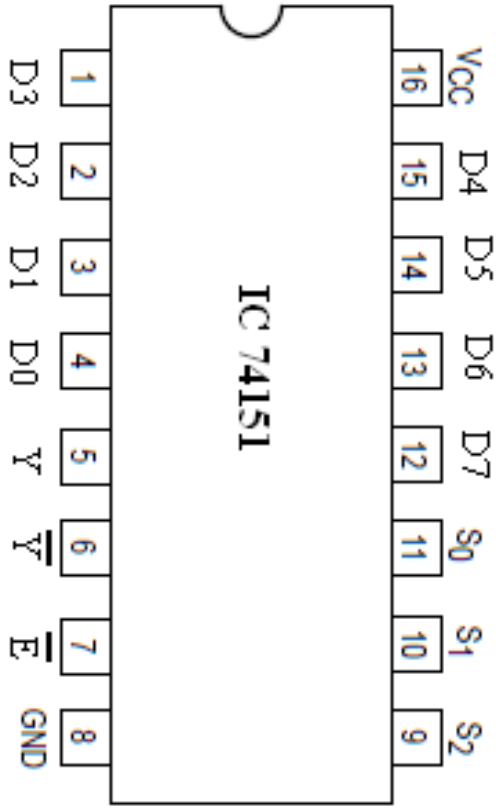
Y / $\overline{Y}$: Output Lines

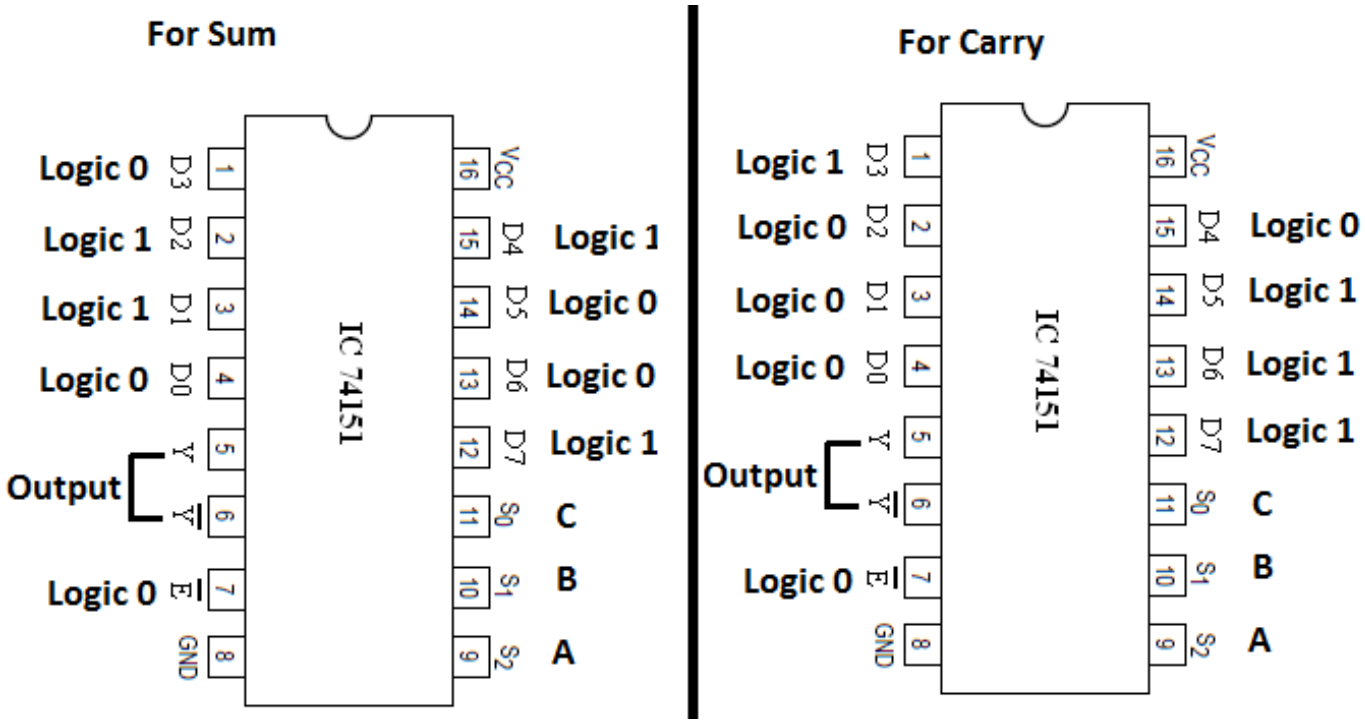
Realization of Full Adder using Multiplexer IC

Truth Table of Full Adder:

Inputs			Outputs	
A	B	C_{in}	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

Sum = $\sum$ m (1, 2, 4, 7) and Carry = $\sum$ m (3, 5, 6, 7)




For Sum
Logic 0
Logic 1
Logic 1
Logic 0
Output
Logic 0
D3
D2
D1
D0
Y
Y
E
GND
IC 74151
VCC
D4
D5
D6
D7
S0
S1
S2
Logic 1
Logic 0
Logic 0
Logic 1
C
B
A
For Carry
Logic 1
Logic 0
Logic 0
Logic 0
Output
Logic 0
Logic 0
Logic 1
Logic 1
Logic 1
C
B
A

Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester -** I

Experiment No: - 08

Aim: - To study Decoder IC and realization of Combinational Logic using Decoder IC.

Theory: -

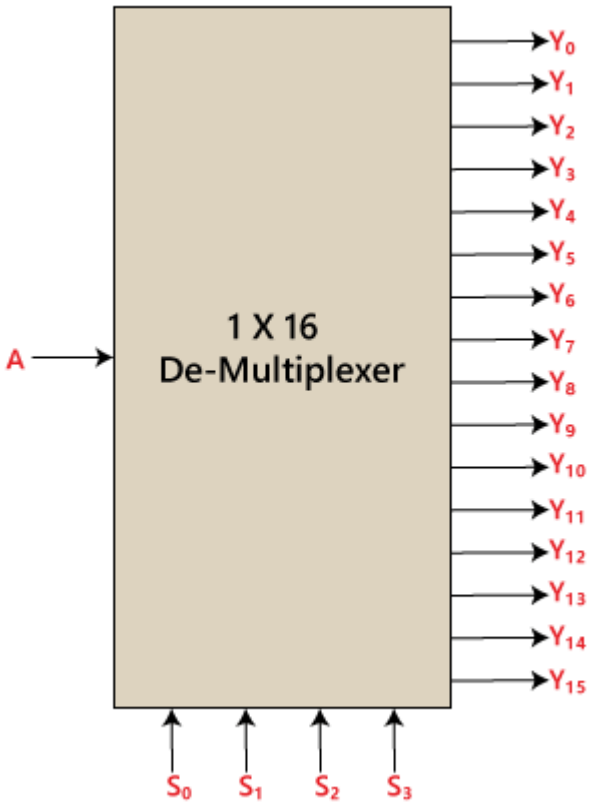
Verification of Demultiplexer:

A Demultiplexer (also known as a Demux or Data Distributor) is defined as a circuit that can distribute or deliver multiple outputs from a single input. ... The function of a Demultiplexer circuit essentially the reverse of the multiplexer.

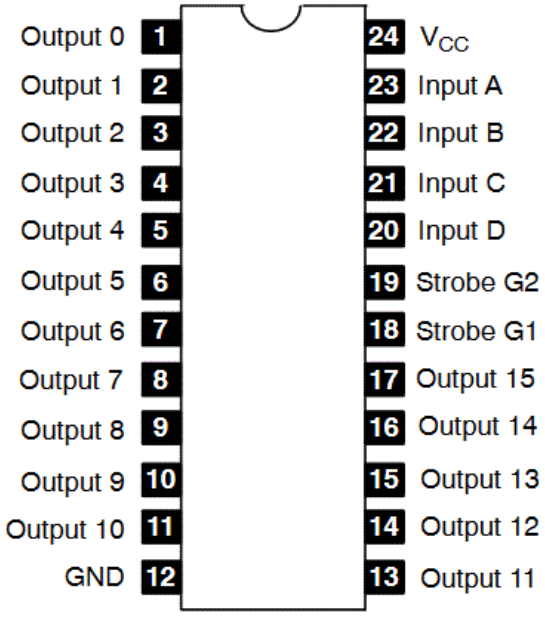
Truth Table:

Dec No	G1	G2	D	C	B	A	Output
0	0	0	0	0	0	0	Y0
1	0	0	0	0	0	1	Y1
2	0	0	0	0	1	0	Y2
3	0	0	0	0	1	1	Y3
4	0	0	0	1	0	0	Y4
5	0	0	0	1	0	1	Y5
6	0	0	0	1	1	0	Y6
7	0	0	0	1	1	1	Y7
8	0	0	0	0	0	0	Y8
9	0	0	1	0	0	1	Y9
10	0	0	1	0	1	0	Y10
11	0	0	1	0	1	1	Y11
12	0	0	1	1	0	0	Y12
13	0	0	1	1	0	1	Y13
14	0	0	1	1	1	0	Y14
15	0	0	1	1	1	1	Y15

General Symbol for 1:16 Demux:



Pin Configuration for IC 74154:

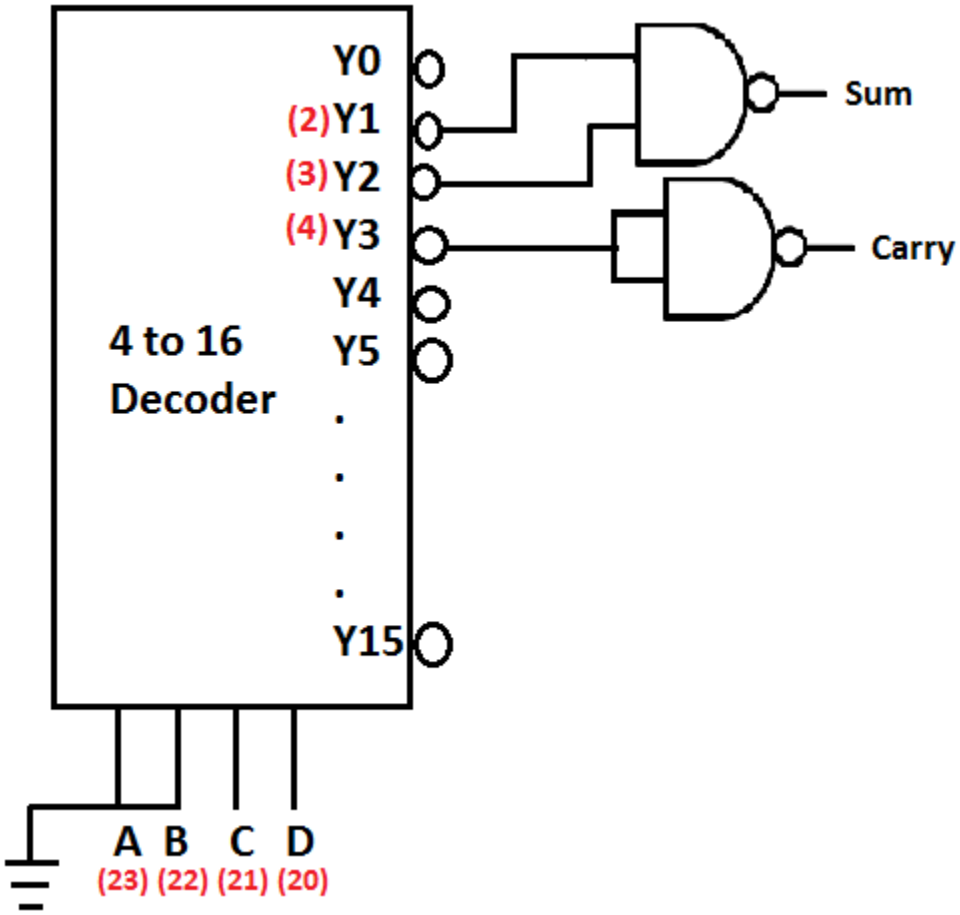


$Y_0 - Y_{15}$: Output lines
A, B, C, D : Select Lines
G : Strobe / Enable

Realization of Combinational Logic Circuit (Half Adder) using Decoder IC:
Truth Table:

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Implementation:



Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester -** I

Experiment No: - 09

Aim: - To study of J-K Flip-Flop using IC.

Theory: -

Flip-Flop:

In electronics, a Flip-flop or Latch is a circuit that has two stable states and can be used to store 1 bit of information at a time.

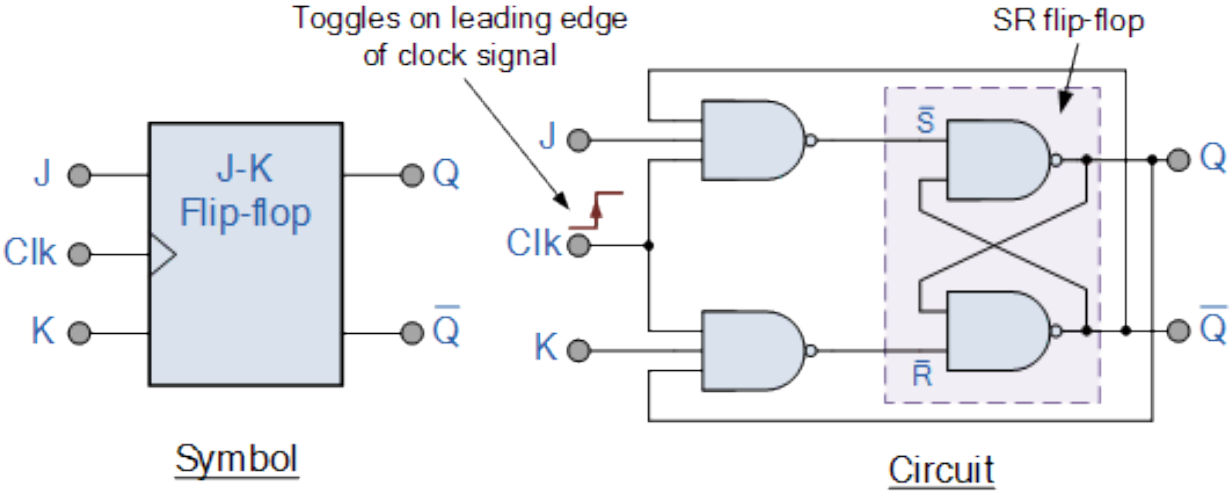
Types of Flip-Flop:

1. S-R Flip Flop
2. J-K Flip Flop
3. D (Delay) Flip Flop

T (Toggle) Flip Flop

The **JK Flip Flop** is the most widely used flip flop. It is considered to be a universal flip-flop circuit. The sequential operation of the JK Flip Flop is the same as for the SR flip-flop with the same **SET** and **RESET** input.

Logical Symbol: **Circuit Diagram:**



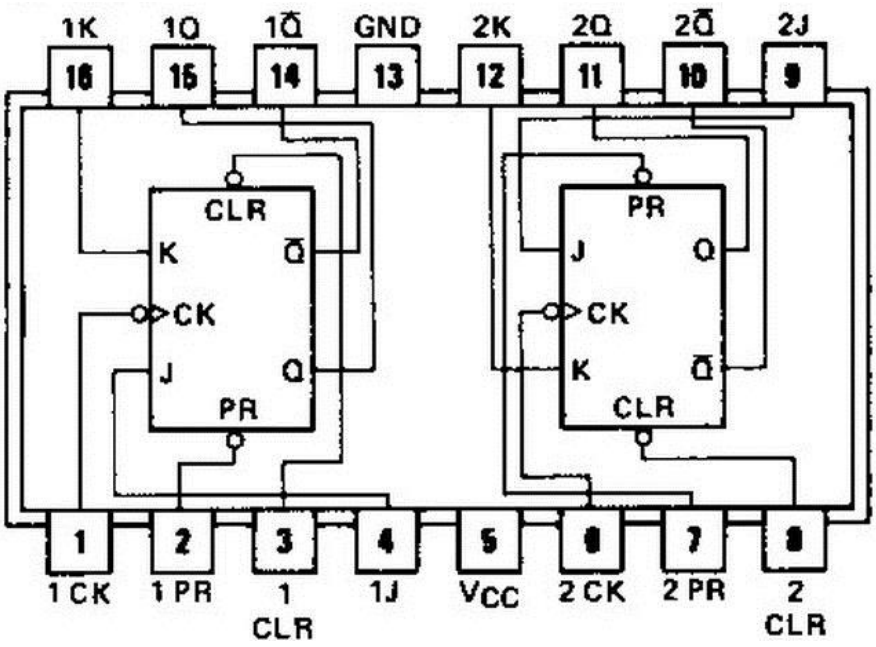
Truth Table:

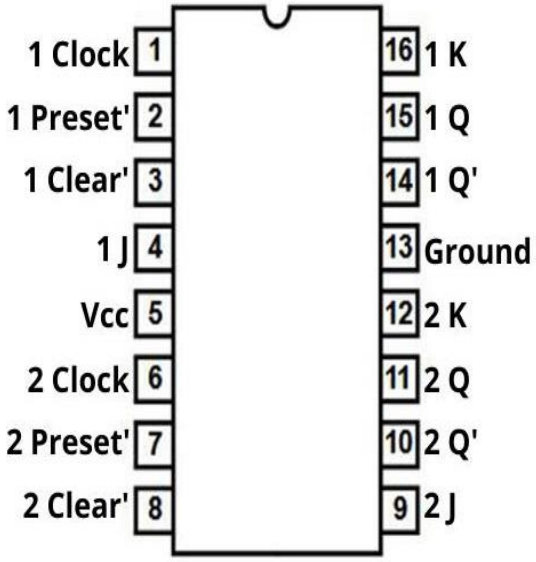
CLK	J	K	Q	Q'	State
1	0	0	Q	Q'	No Change
1	0	1	0	1	Reset
1	0	0	1	0	Set
1	0	1	X	X	Toggle

Function Table:

Inputs					Outputs	
PR	CLR	CLK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H (Note 1)	H (Note 1)
H	H	⎍	L	L	Q_0	$\overline{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	Toggle	

Pin Configuration for IC 7476:





Laboratory Manual

Subject: - Digital Electronics
Class: - S Y BTech Computer Engineering **Semester -** I

Experiment No: - 10

Aim: - To realize BCD Mod 10 Counter.

Theory: -

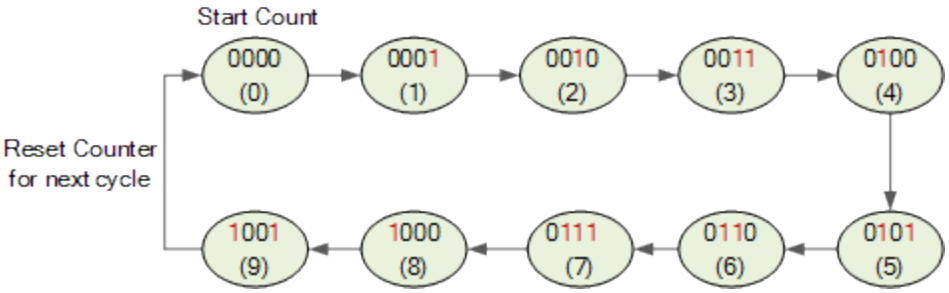
Asynchronous Counters use flip-flops which are serially connected together so that the input clock pulse appears to ripple through the counter. An Asynchronous counter can have 2n-1 possible counting states.

Such counters are generally referred to as Decade Counters. A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010 and to do this we need to feed this condition back to the reset input. A counter with a count sequence from binary "0000" (BCD = "0") through to "1001" (BCD = "9") is generally referred to as a BCD binary-coded-decimal counter because its ten state sequence is that of a BCD code but binary decade counters are more common.

To make a digital counter which counts from 1 to 10, we need to have the counter count only the binary numbers 0000 to 1001. That is from 0 to 9 in decimal and fortunately for us, counting circuits are readily available as integrated circuits with one such circuit being the Asynchronous 74LS90 Decade Counter.

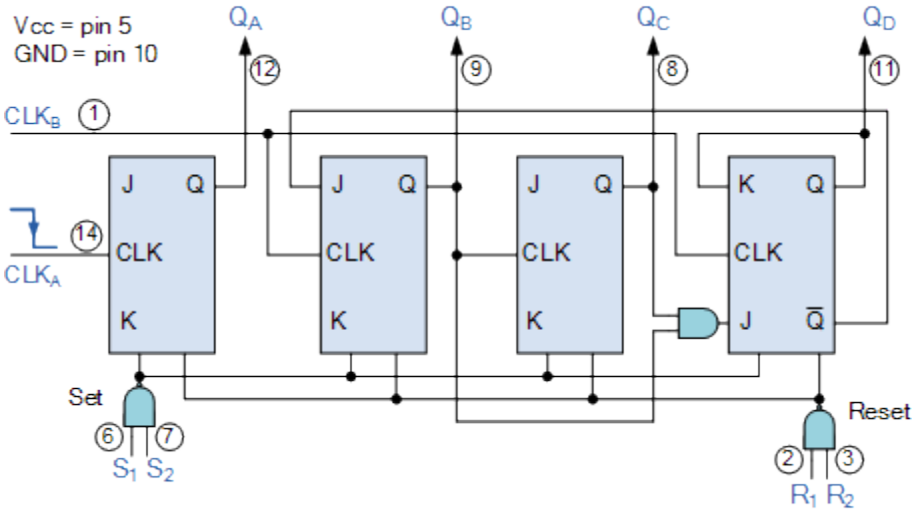
BCD Counter State Diagram:

A decade counter counts in a sequence of ten and then returns back to zero after the count of nine. Obviously to count up to a binary value of nine, the counter must have at least four flip-flops within its chain to represent each decimal digit as shown.

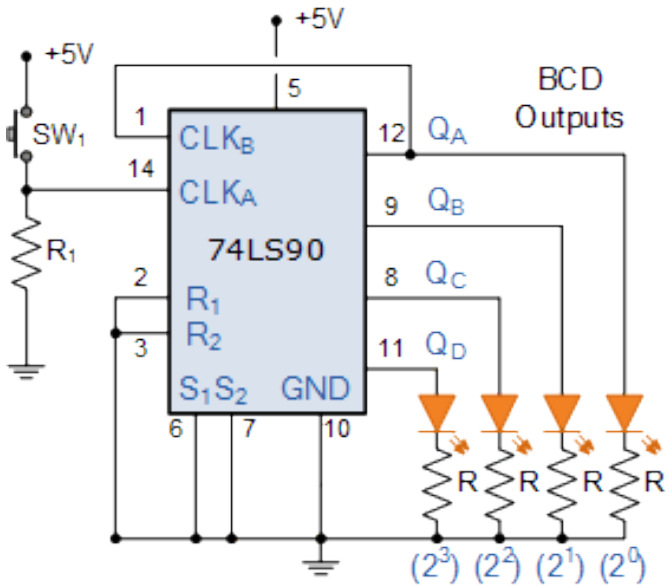


Then a decade counter has four flip-flops and 16 potential states, of which only 10 are used.

74LS90 BCD Counter



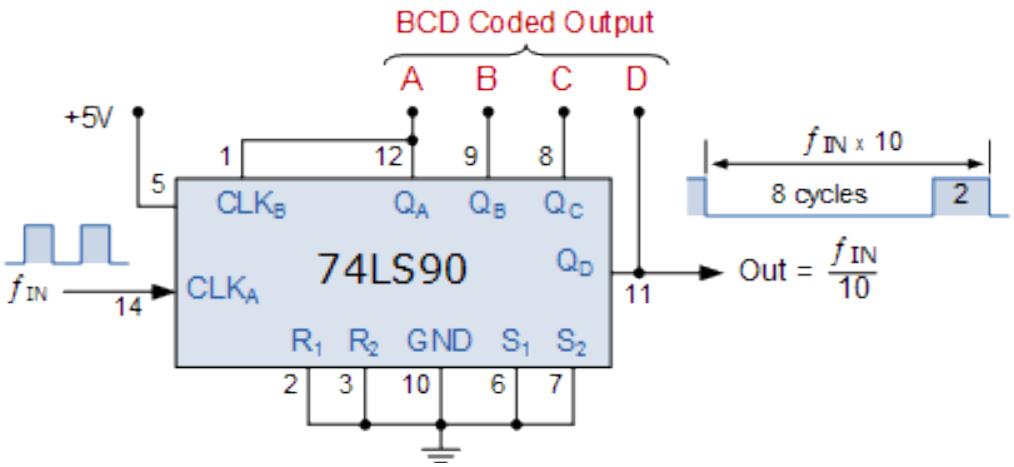
Truth Table:

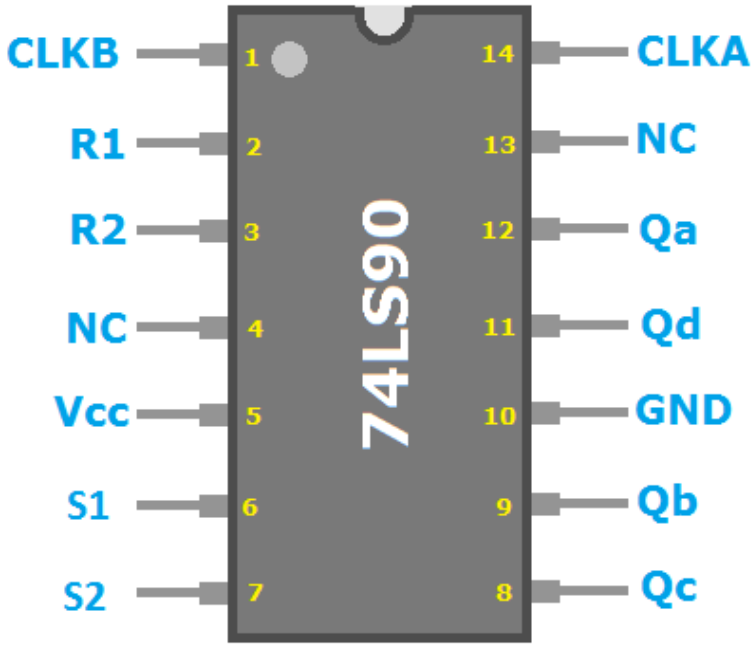


Truth Table

count	Q_D	Q_C	Q_B	Q_A
0 [start]	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 [new cycle]	0	0	0	0

Pin Configuration for IC 7490:





IC 7490 Pin Diagram