9 Pgs

S16120 37

Experiment No. 1

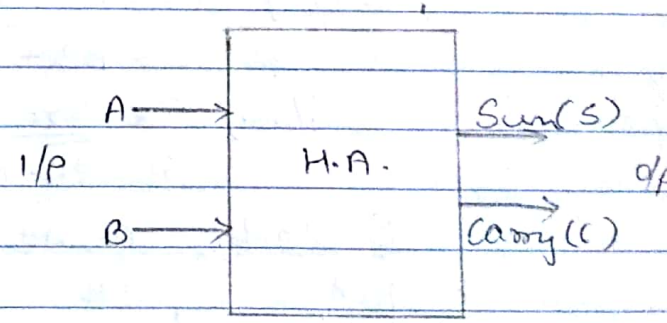
Aim: Realize half and full adder and subtractor using : a] Basic gates and b] Universal gates.

Theory: Binary Adders.

Half Adder :-

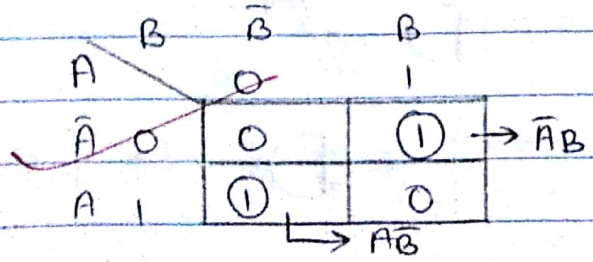
- Half adder is a combinational logic circuit with 2 inputs & 2 outputs. It is the basic building block for addition of two 'single' bit numbers. This circuit has 2 outputs namely 'sum' & 'carry'.
- The half adder circuit is supposed to add 2 single bit binary numbers A & B. Therefore the truth table of a half adder is as follows :-

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



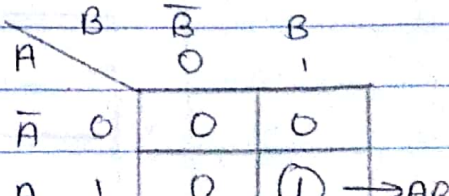
Block diagram.

a] K-map for sum output



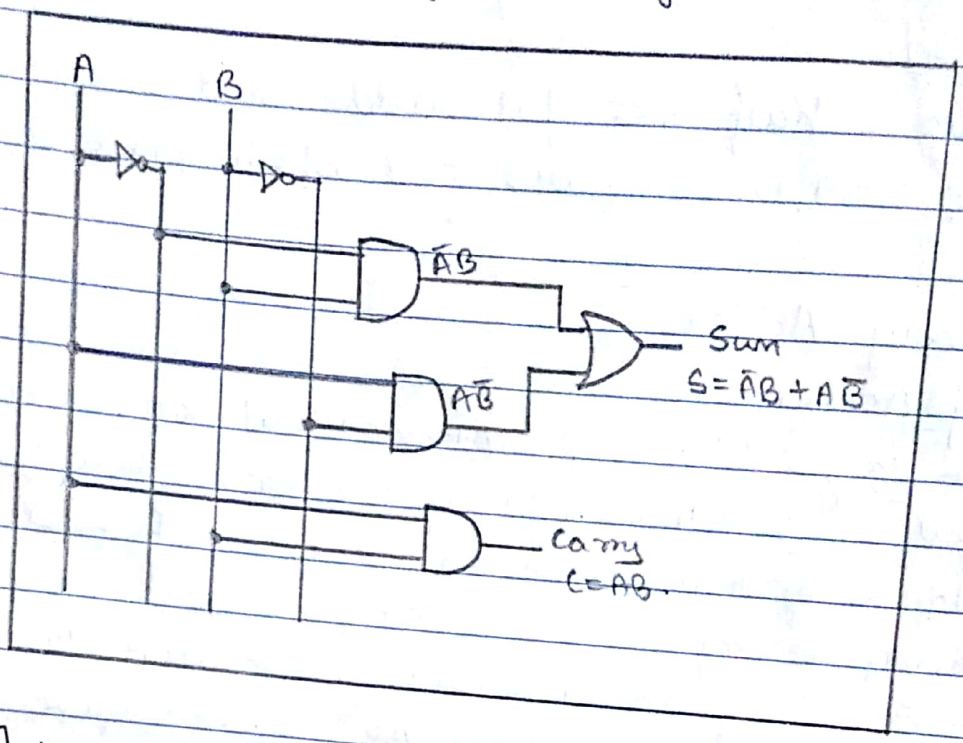
$\therefore S = \bar{A}B + A\bar{B} = A \oplus B.$

b] K map for carry output



$\therefore C = AB.$

a] Half Adder using basic gates.



b] Half adder using Universal gates:-

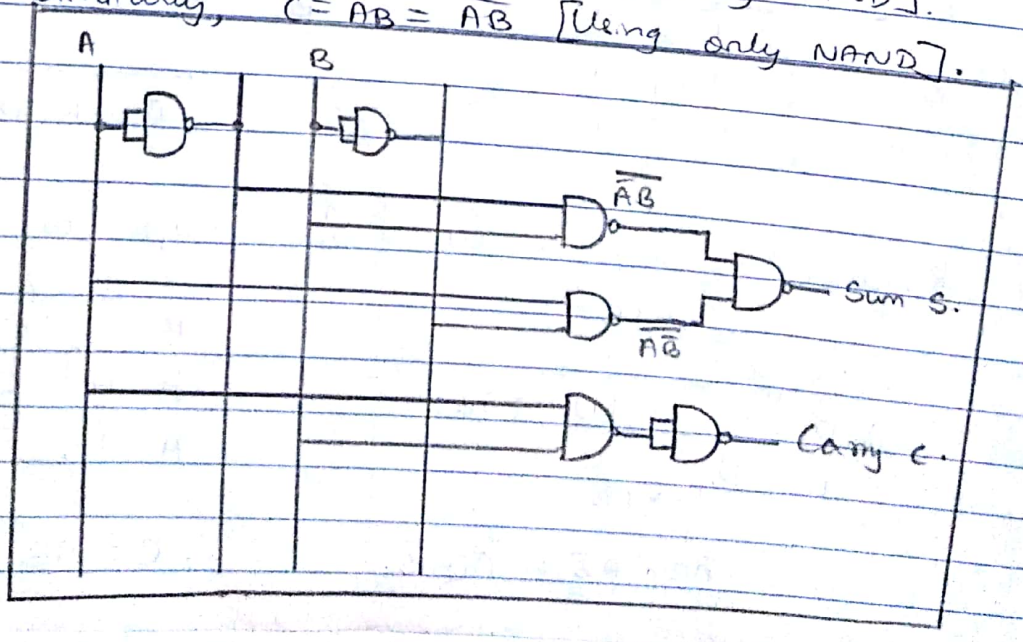
Taking double inversion on equation for S, we get

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

Using DeMorgan's theorem,

$$S = \overline{\bar{A}B} \cdot \overline{A\bar{B}} \quad \text{[Using only NAND]}.$$

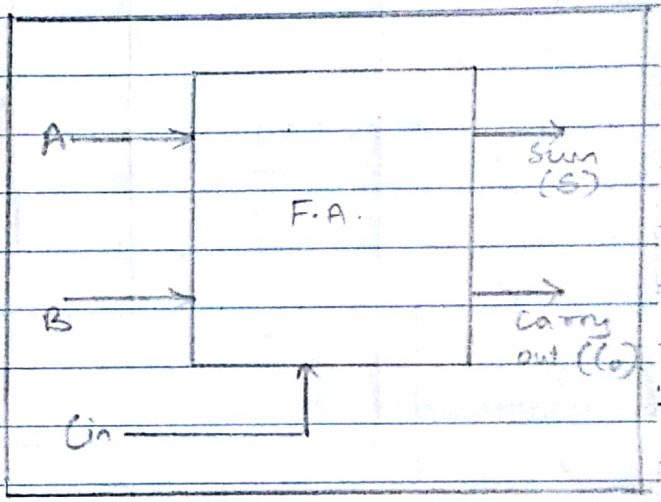
Similarly, $C = AB = \overline{\overline{AB}}$ [Using only NAND].



Full Adder :-

- A 3 single bit adder circuit called full Adder is developed to add 2 one-bit numbers A and B and carry C_{in}.
- It is a 3 input and 2 output combinational circuit.

Inputs			Outputs	
A	B	C_{in}	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



K-map for sum:-

A \ BCin	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

Expression:- (Sum)

$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + AB C_{in} + A\bar{B}\bar{C}_{in}$

$S = C_{in}(\underbrace{\bar{A}\bar{B} + AB}_{\text{EX-NOR}}) + \bar{C}_{in}(\underbrace{\bar{A}B + A\bar{B}}_{\text{EX-OR}})$

$\therefore S = C_{in}(\overline{\bar{A}B + A\bar{B}}) + \bar{C}_{in}(\bar{A}B + A\bar{B})$

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

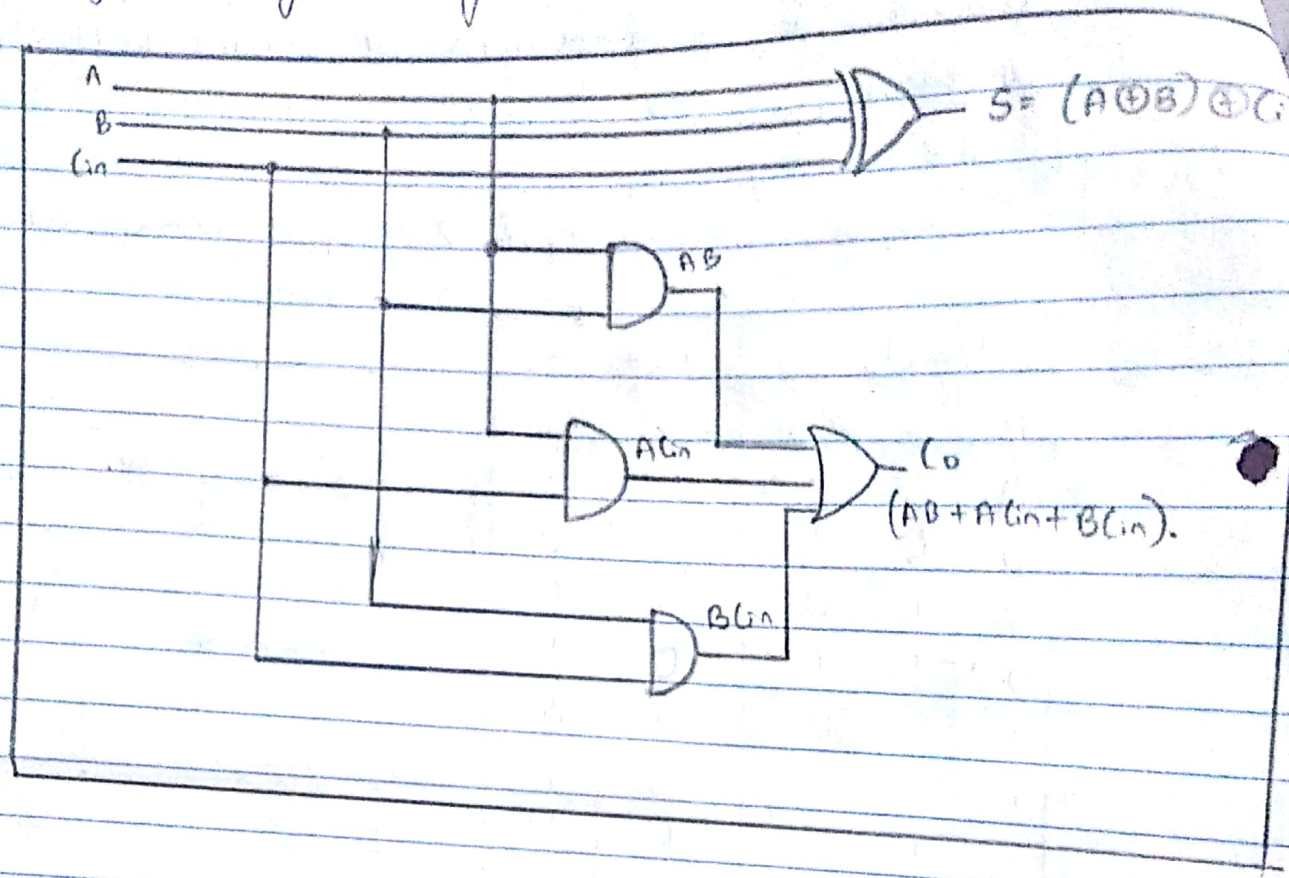
Kmap for carry:-

A \ BCin	00	01	11	10
0	0	0	(1)	0
1	0	(1)	(1)	(1)

Expression :- (carry)

$C_o = AB + AC_{in} + BC_{in}$.

Logic Diagram for Full Adder:-

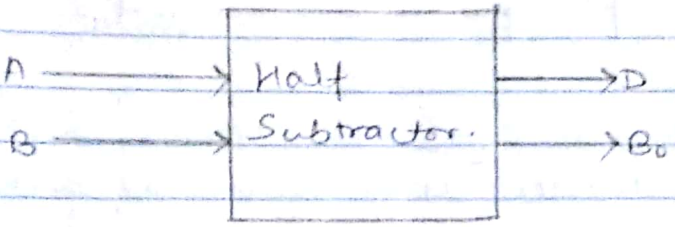


Theory: Binary Subtractors:-

Half Subtractor:-

- Half subtractor is a combinational circuit with 2 inputs & 2 outputs (difference & borrow).
- It produces the difference between the 2 binary bits @ the input and also produces an output (borrow) to indicate if a 1 has been borrowed.
- In the subtraction (A-B), A is called as minuend bit and B is called as subtrahend bit.
- The truth table & block diagram are as follows:-

Inputs		Outputs	
A	B	Difference D (A-B)	Borrow (B_o)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Kmap for difference

A \ B	0	1
0	0	(1)
1	(1)	0

Difference:-

$D = \bar{A}B + A\bar{B} = A \oplus B$.

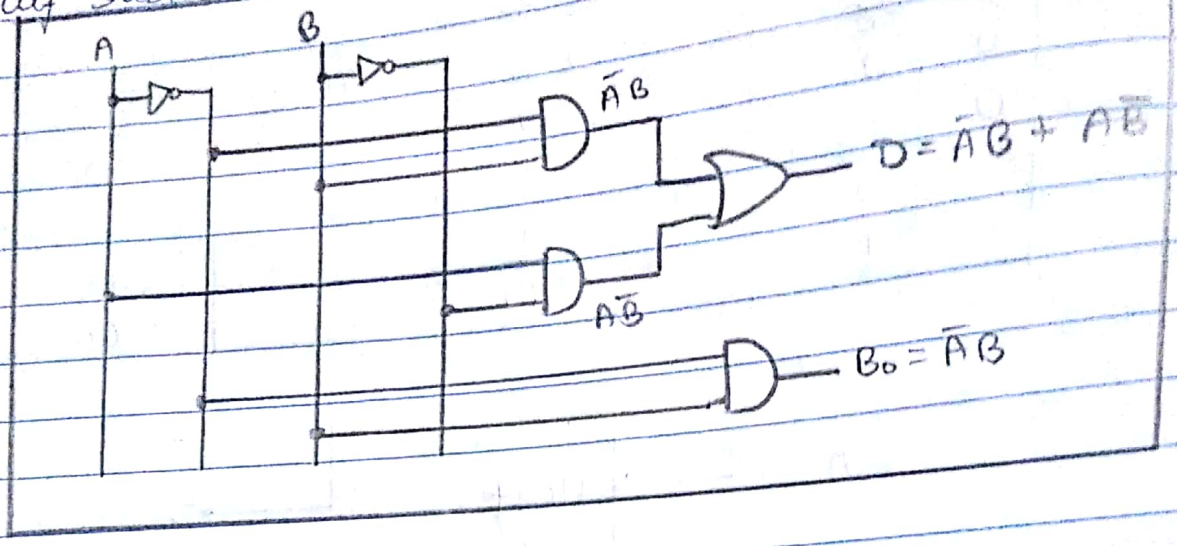
Kmap for borrow:-

A \ B	0	1
0	0	(1)
1	0	0

Borrow:-

$B = \bar{A}B$.

Half subtractor using basic gates:-



Half a subtractor using universal gates:-

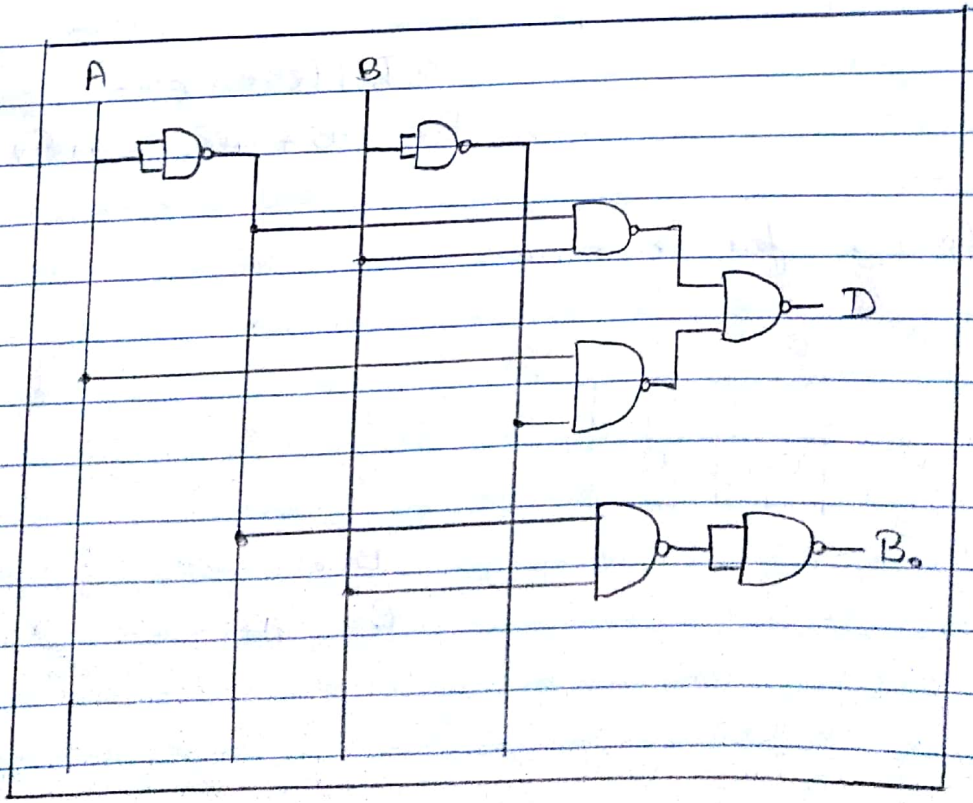
-Taking double inversion of equation of D, we get

$$D = \overline{\overline{\bar{A}B + A\bar{B}}}$$

$$\therefore D = \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}}$$

Similarly;

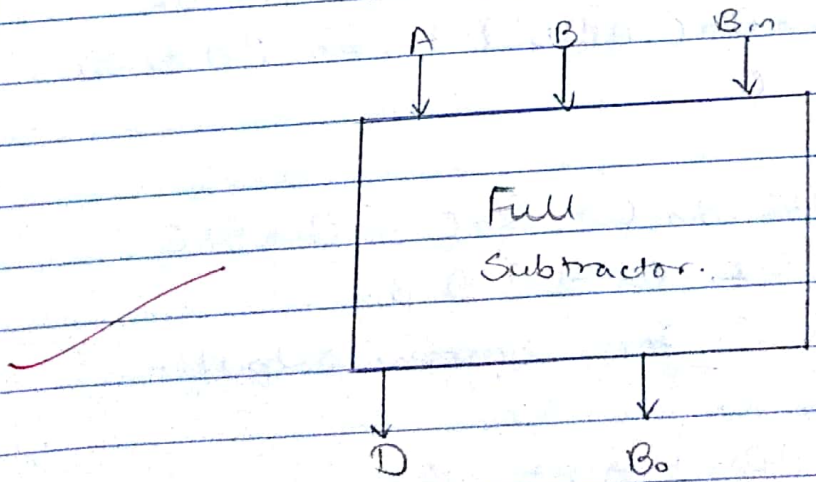
$$B_o = \bar{A}B = \overline{\overline{\bar{A}B}}$$



Full Subtractors:-

- The full subtractor is a combinational circuit with 3 inputs A, B and B_{in} & 2 outputs D and B_o.
- A is the minuend, B is the subtrahend, B_{in} is the borrow produced by previous stage, D is the difference output and B_o is the borrow output.

Inputs.			Outputs.	
A (Minuend)	B (Subtrahend)	B_{in} (Previous borrow)	$(A-B-B_{in})$	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Kmap for difference:-

A \ BBin	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

$\therefore D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B_{in}} + A\bar{B}\bar{B_{in}} + ABB_{in}$

K map for boorow:-

A \ BBin	00	01	11	10
0	0	(1	(1)	1)
1	0	0	1	0

$\therefore B_0 = \bar{A}B_{in} + \bar{A}B + BB_{in}$.

Simplification for difference output:-

$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B_{in}} + A\bar{B}\bar{B_{in}} + ABB_{in}$

$= B_{in}(\underbrace{\bar{A}\bar{B} + AB}_{\text{EX-NOR}}) + \bar{B_{in}}(\underbrace{\bar{A}B + A\bar{B}}_{\text{EX-OR}})$

$\therefore D = B_{in}(\overline{A \oplus B}) + \bar{B_{in}}(A \oplus B)$

Let $A \oplus B = C$,

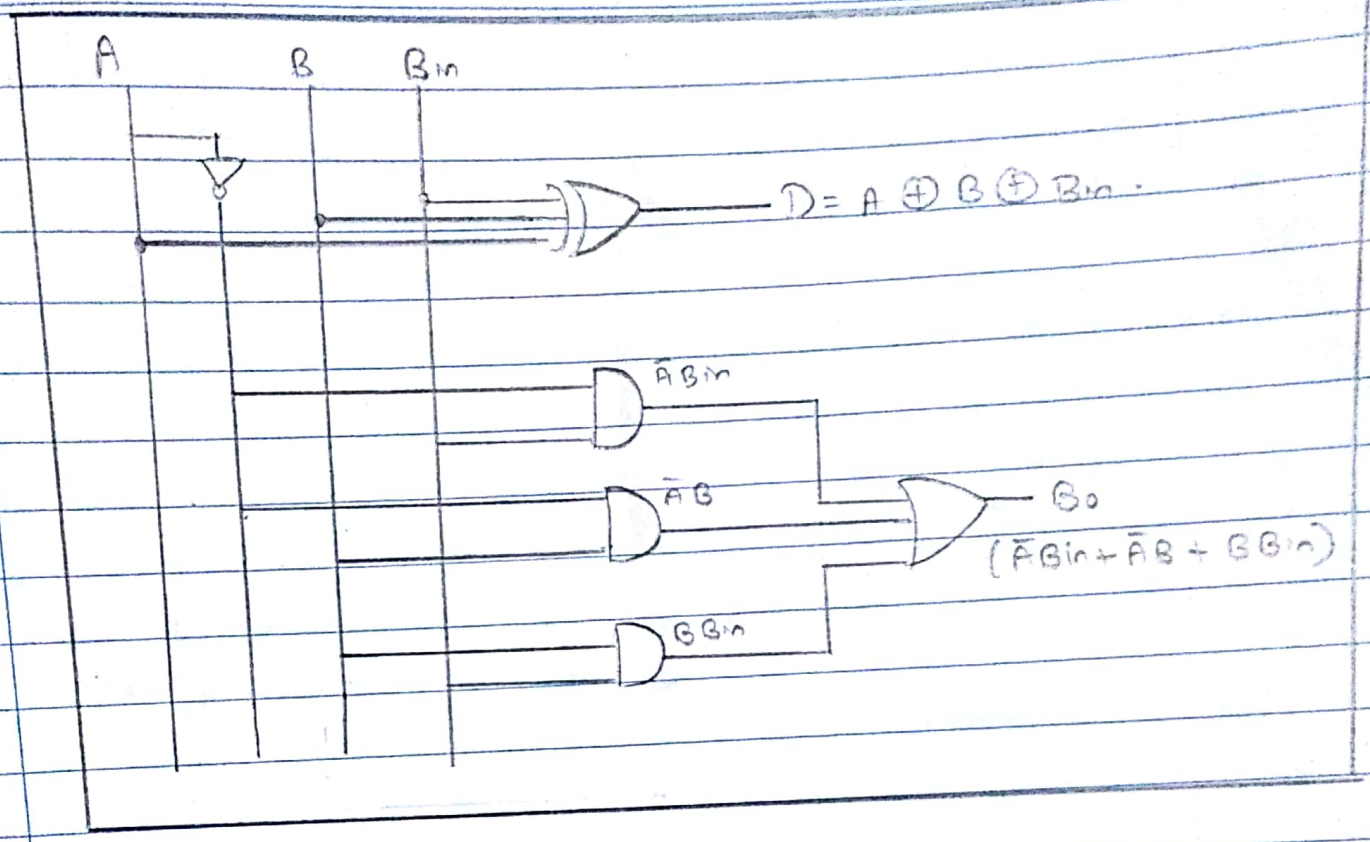
$\therefore D = B_{in}\bar{C} + \bar{B_{in}}C = B_{in} \oplus C$

$\therefore D = B_{in} \oplus A \oplus B$.

Simplification for boorow output:-

$B_0 = \bar{A}B_{in} + \bar{A}B + BB_{in}$

No furthes simplification is possible.



Conclusion:-

Hence we studied the logical circuits for binary adders & subtractors using basic gates as well as universal gates.

3+4+3+3 = 13|15

26/6/17

S1612037

ZEAL COER
Page No.:
www.zealeducation.com

Experiment No 2.

Aim: Design and implement code converter binary to gray and BCD to excess 3.

Requirement: IC 7486, Power supply, wires, IC 7404, IC 7432, LED and resistors 220 ohm, IC, 7408.

Theory:

Binary To Gray Conversion.

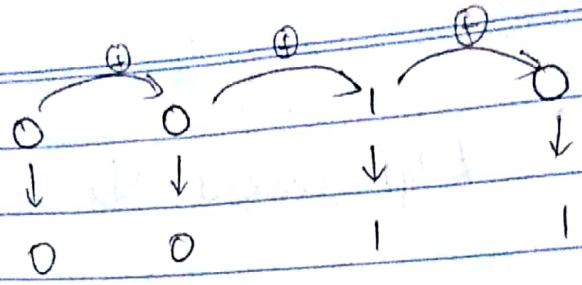
1) The reflected binary code also known as Gray code after Frank Gray, is a binary numerical system where 2 successive values differ in only 1 bit.
2) The reflected binary code was originally designed to prevent spurious output from electromechanical switches. Today, Gray codes are quickly & widely used to facilitate error correction in digital communications. Such as digital terrestrial television and some cable TV system.
3) For design binary to gray, we require EX-OR gate. Following is the truth table of EX-OR gate.

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

4) By using truth table of EX-OR we can convert binary codes to gray codes. Following are some steps of conversion.

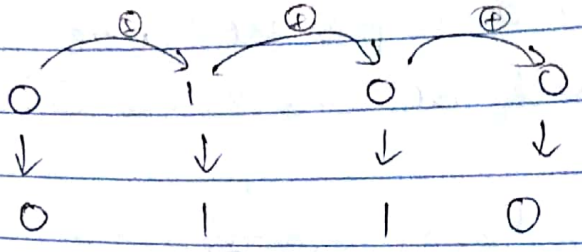
Example) a) Convert $(0010)_2$ into gray code.
b) Convert (0100)

a)



= Gray code.

b)



= Gray code.

Binary				Gray.			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
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

• K-map for G_3.

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$G_3 = B_3$.

• Kmap for G_2.

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$G_2 = \bar{B_3}B_2 + B_3\bar{B_2}$

$G_2 = B_2 \oplus B_3$

• Kmap for G_1

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$G_1 = \bar{B_2}B_1 + B_2\bar{B_1}$

$G_1 = B_2 \oplus B_1$

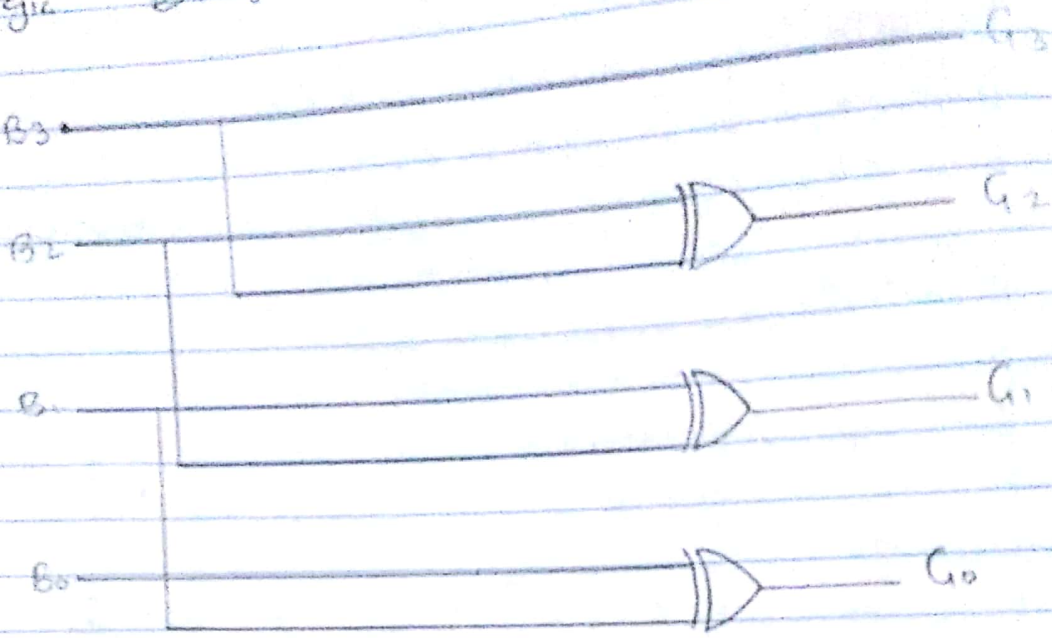
• Kmap for G_0.

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$G_0 = \bar{B_1}B_0 + B_1\bar{B_0}$

$G_0 = B_1 \oplus B_0$

• Logic Diagram -



BCD to Excess - 3

1) The term BCD refers to representing the 10 decimal digits in binary forms.
⇒ BCD = Binary coded decimal.
2) The Excess - 3 system simply adds 3 to each numbers to make the codes look different.

The Excess - 3 BCD system has some properties that made it useful in early computers.

The Excess - 3 BCD system is formed by adding 0011 to each BCD value.

For example:-

The decimal no 7, which is coded as 0111 in BCD is coded as

0111 + 0011 = 1010 in Excess - 3 BCD.

BCD				Excess 3			
B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	0	1	0	0

Kmap for E_3.

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$E_3 = B_3 + B_1B_2 + B_0B_2$

$E_3 = B_3 + B_2(B_1 + B_0)$

• Kmap for E_2

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

$E_2 = \overline{B_3}B_1 + B_3\overline{B_2} + \overline{B_2}\cdot\overline{B_1}B_0 + \overline{B_1}\,\overline{B_0}\,B_2$

$E_2 = \overline{B_0}\,\overline{B_1}B_2 + \overline{B_2}(B_0 + B_1)$

$E_2 = \overline{B_0}\,\overline{B_1}B_2 + (B_1 + B_0)\overline{B_2}$.

• K map for E_1

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

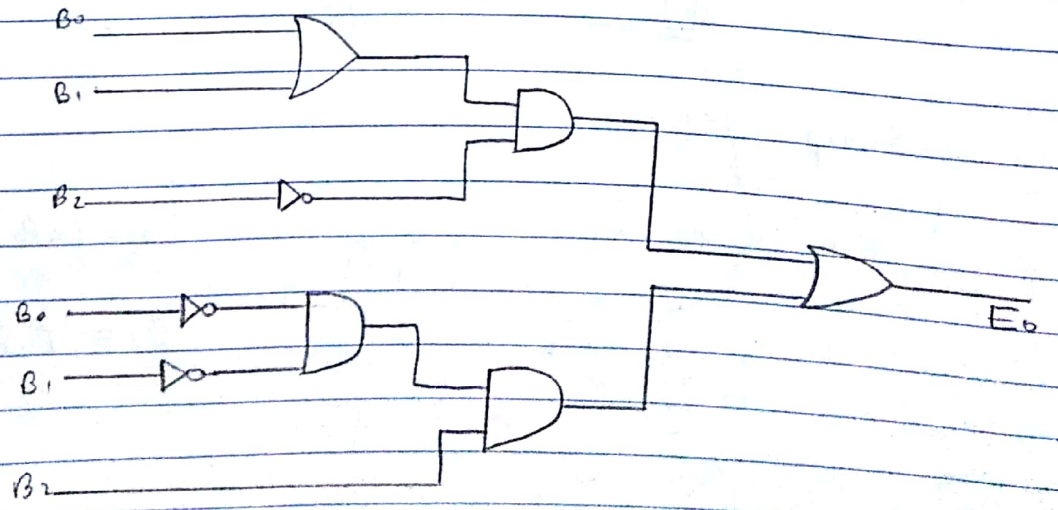
$E_1 = \overline{B_0 \oplus B_1}$

• K map for E_0

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$E_0 = \overline{B_0}$

• Logic Diagram:-



Conclusion:-

From the above experiment we studied, design & implementation of code converters.

i) Binary to Gray.

ii) BCD to Excess-3.

C
3/7/17

3+3+4+4 = 14/15

S1612037

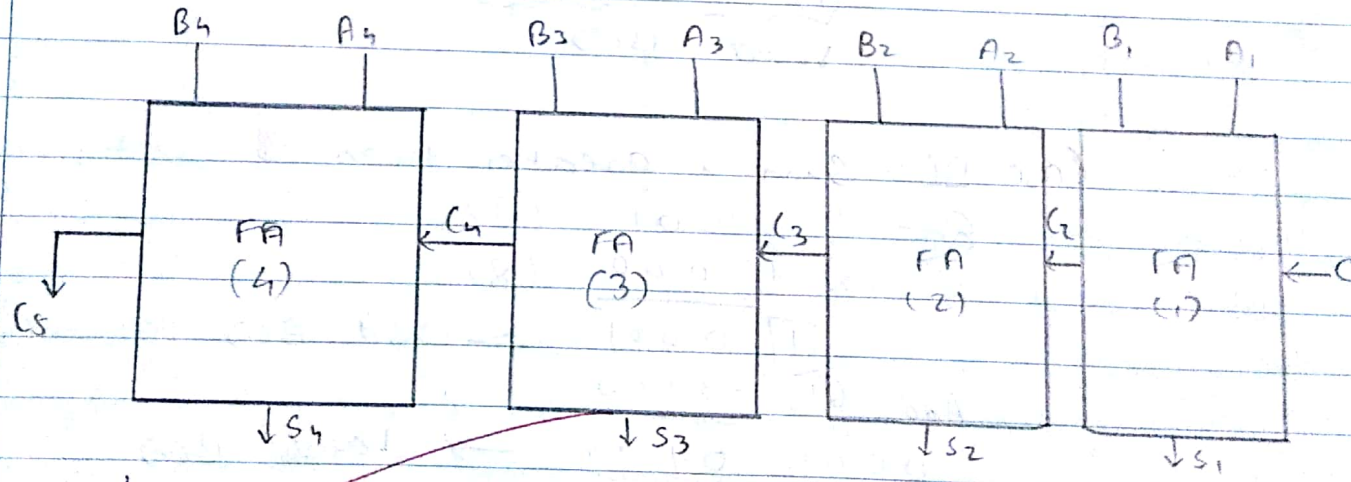
Experiment No. 3

Aim: Design of n-bit Carry Slave Adder (CSA) and propagation Adder (CPA). Design and realization of BCD adder using 4-bit Binary Adder (IC 7483).

Theory:-

The BCD Adder can be used to add 4 bit BCD equivalent of 2 single digit decimal numbers. It can be implemented using a binary adder IC 7483. The binary adder IC can add 2 4-bit binary numbers along with a carry generator.

The binary adder is nyait a 4 bit parallel adder also called as parallel binary adder.



In the above schematic, every block is a full adder (FA). Each FA has 2 single bit inputs and one output. All the full adders add the subsequent bits in the 2 4-bit binary number input. The carry from each full adder is passed to the next full adder of higher stage order.

Certain cases needed to be consider while adding 2 one digit BCD numbers using a binary adder are as follows:-

Case I: Sum is less than 9

Ex 1 :-

$$\begin{array}{rll} 0101 & (5)_{10} \\ +\ 0011 & (3)_{10} \\ \hline 1000 & (8)_{10} \end{array}$$

Case II: Sum is greater than 9 with carry 0.

Example:-

$$\begin{array}{rll} 1000 & (8)_{10} \\ +\ 0100 & (4)_{10} \\ \hline 1100 & (12)_{10} \leftarrow \text{Correct answer, invalid BCD} \end{array}$$

Add $(6)_{10}$ $+0110$

$$\begin{array}{ll} 0001 & 0010 \\ \text{valid} & \text{BCD} \end{array} \qquad (12)_{BCD}.$$

Case III :- Sum is greater than 9 with carry 1.

Ex:-

$$\begin{array}{rll} 1001 & (9)_{10} \\ +\ 1000 & (8)_{10} \\ \hline \boxed{1}\ 0001 & \rightarrow \text{Valid BCD, Incorrect answer.} \end{array}$$

Add $(6)_{10}$ $+0110$

$$\begin{array}{ll} 0001 & 0111 \\ 1 & 7 \end{array} \rightarrow \text{Valid BCD.}$$

Taking case I and II into account, an addition combinational circuit needs to be implemented with the binary adder circuits to get an accurate BCD result.

We need to build an additional circuit to deal with the cases, which considers the arising problem.

Truth Table:-

Z_3	Z_2	Z_1	Z_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

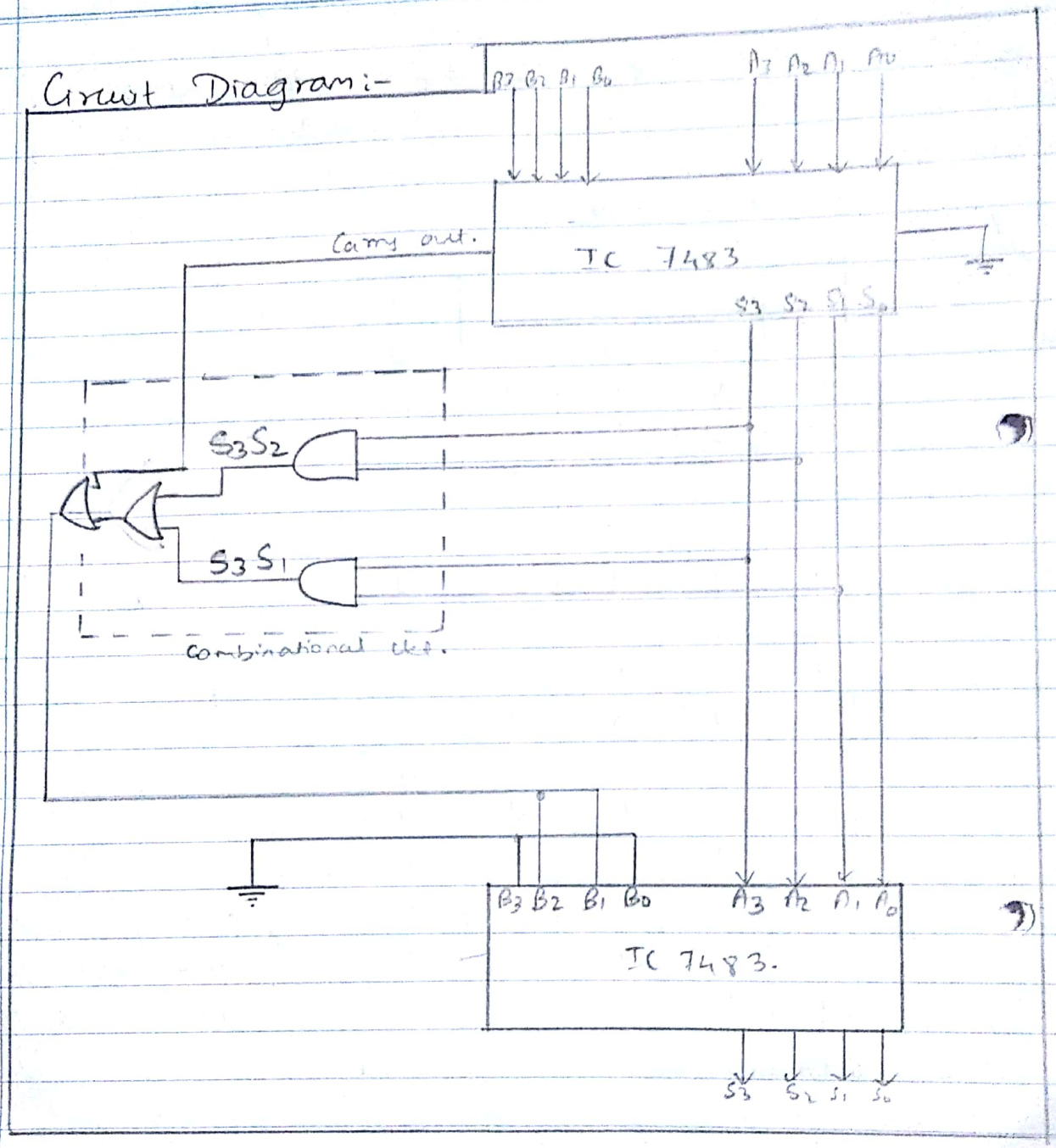
To design a circuit for these conditions, especially case 2 & 3.

K-map for Y :-

Z_3Z_2 \ Z_1Z_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$\therefore Y = Z_3Z_2 + Z_3Z_1$.

Circuit Diagram:-



Conclusion: In this way, we studied BCD Adder using 4 bit Binary Adder (IC 7483).

3+3+4+3 = 13/15

28/12/17

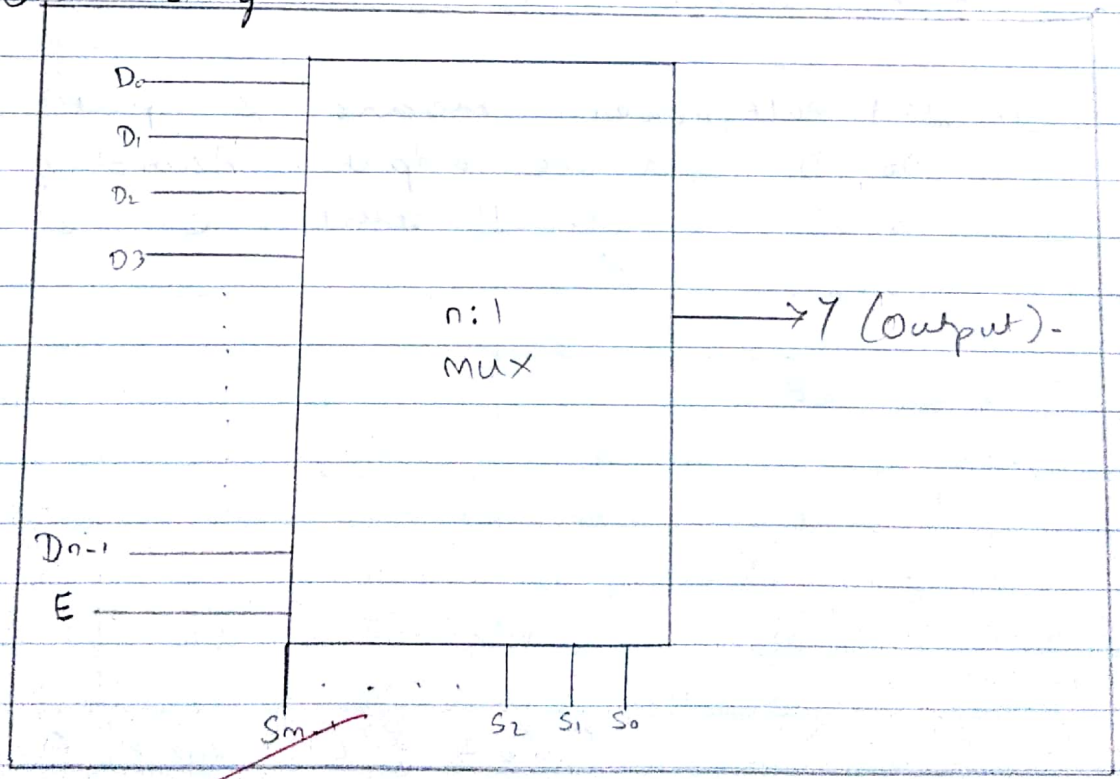
S1612037.

Experiment No 4

Aim:- To realize Boolean expression for suitable combination logic using 74151 MUX.

Theory:- Multiplexer means many input and one output. It is special type of combinational circuit which select one of the n data inputs and routes (connected) it to the output. The selection of one of the inputs is done with the help of select inputs.

Block diagram:-



D0 – Dn-1 = Input lines.

E = Enable line.

Y = output.

S0 ... Sm-1 = Select lines.

n:1 = n input and 1 output.

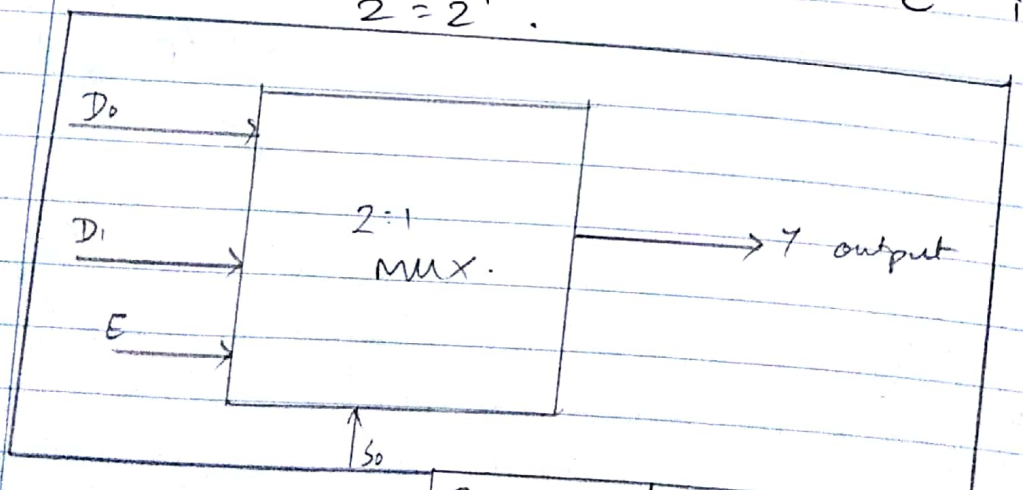
$$n = 2^m$$

where n = no. of inputs.
m = no of select lines.

There are many types of multiplexers as follows.

① 2:1 mux.
② 4:1 mux.
③ 8:1 mux.
④ 16:1 mux.
⑤ 32:1 mux.

① 2:1 Multiplexer:- means 2 input named as D_0, D_1 and one output. According to relation $n = 2^m$ it has 1 select line i.e $2 = 2^1$.



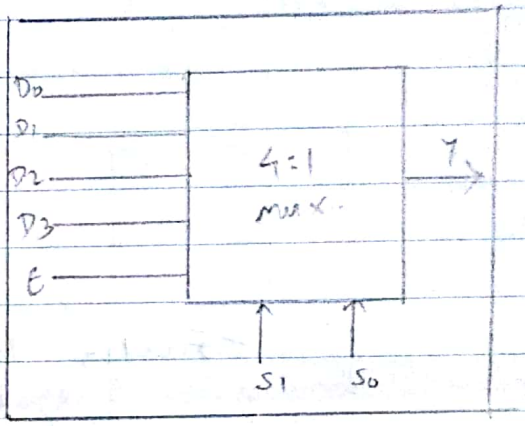
Enable	Select input	Output (Y)
0	X	0
1	0	D_0
1	1	D_1

② 4:1 MUX:-

It has 4 input and 1 output according to relation $n = 2^m$.

It has 4 input and 2 select lines

$4 = 2^2$.



Truth Table:-

Enable	Select Input		Output
	S_0	S_1	Y
0	X	X	0
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

③ 8:1 MUX.

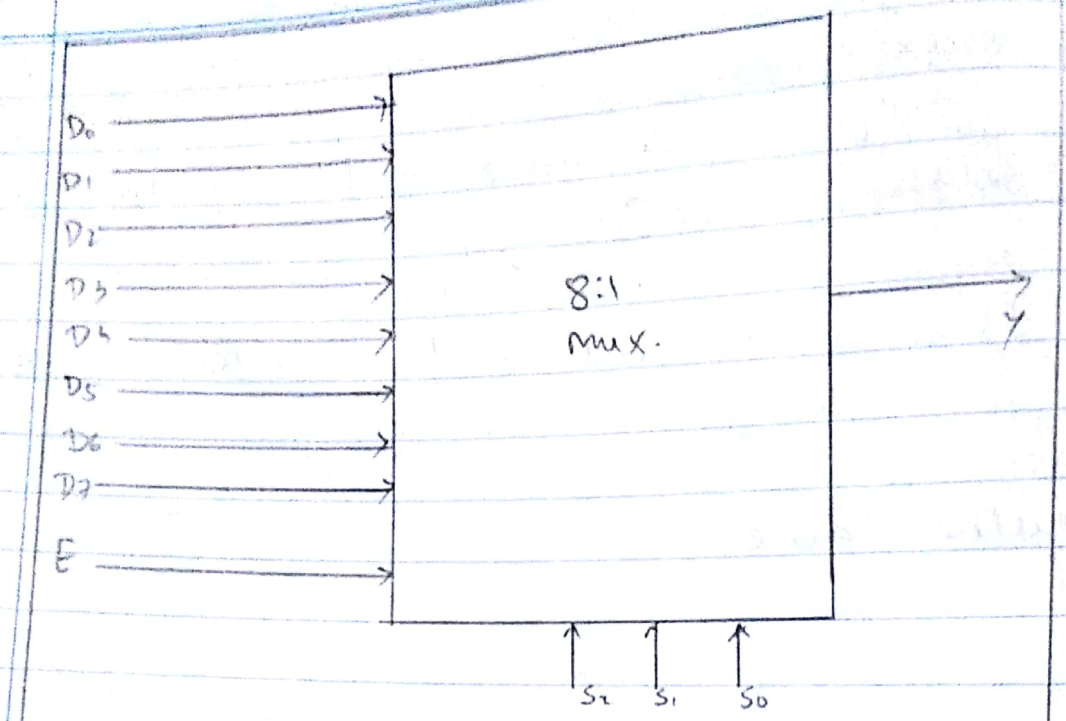
It has 8 input & 1 output.

According to relation

$$n = 2^m.$$

It has 8 input and 3 select lines.

$$8 = 2^3$$



Truth Table:-

Enable	Select inputs.			Output
E	S_2	S_1	S_0	Y.
0	X	X	X	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7

(4) 16:1 Mux

It has 16 input & 1 output

According to the relation

$n = 2^m$.

It has 16 input and 4 select lines.

$\therefore 16 = 2^4$.

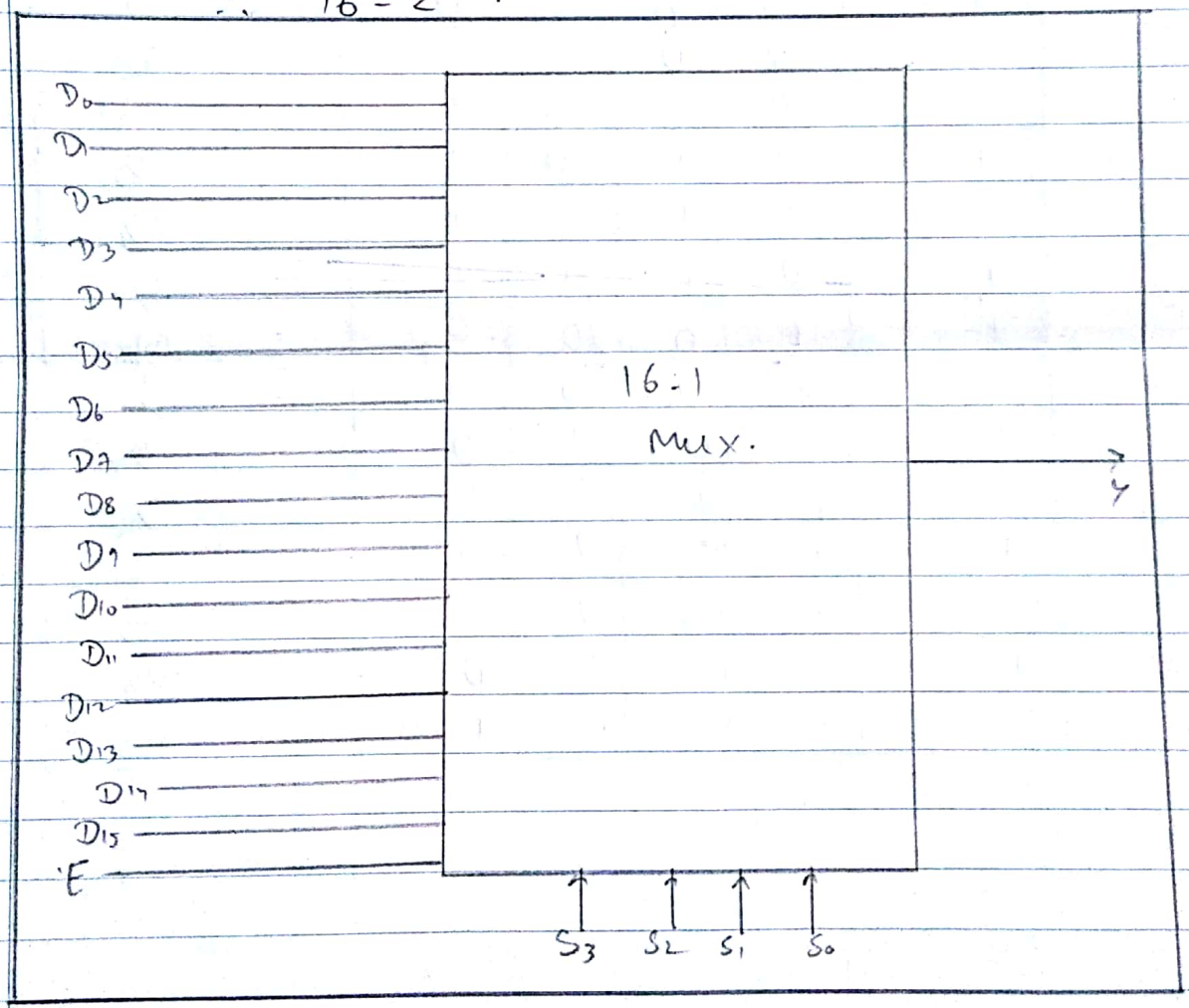
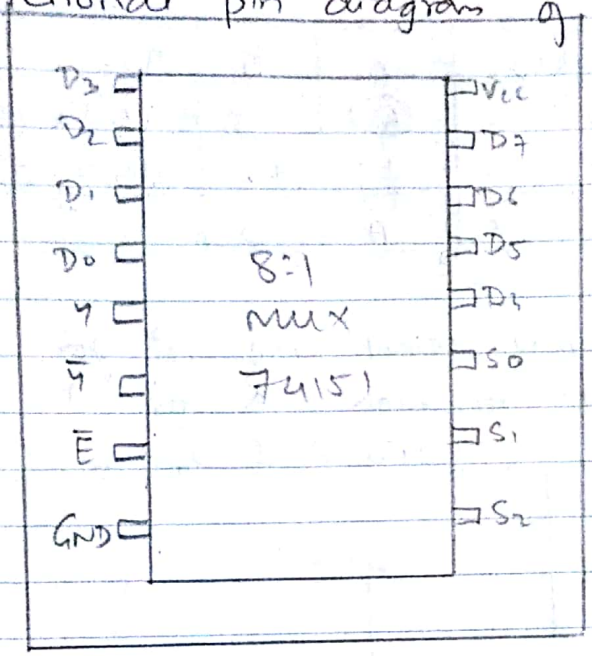


Fig:- Block diagram of 16:1 MUX.

Truth Table:-

Enable	Select Input lines.				Output
E	S_3	S_2	S_1	S_0	Y
X	X	X	X	X	0
1	0	0	0	0	D_0
1	0	0	0	1	D_1
1	0	0	1	0	D_2
1	0	0	1	1	D_3
1	0	1	0	0	D_4
1	0	1	0	1	D_5
1	0	1	1	0	D_6
1	0	1	1	1	D_7
1	1	0	0	0	D_8
1	1	0	0	1	D_9
1	1	0	1	0	D_{10}
1	1	0	1	1	D_{11}
1	1	1	0	0	D_{12}
1	1	1	0	1	D_{13}
1	1	1	1	0	D_{14}
1	1	1	1	1	D_{15}

Functional pin diagram of mux 74151 (8:1 mux).



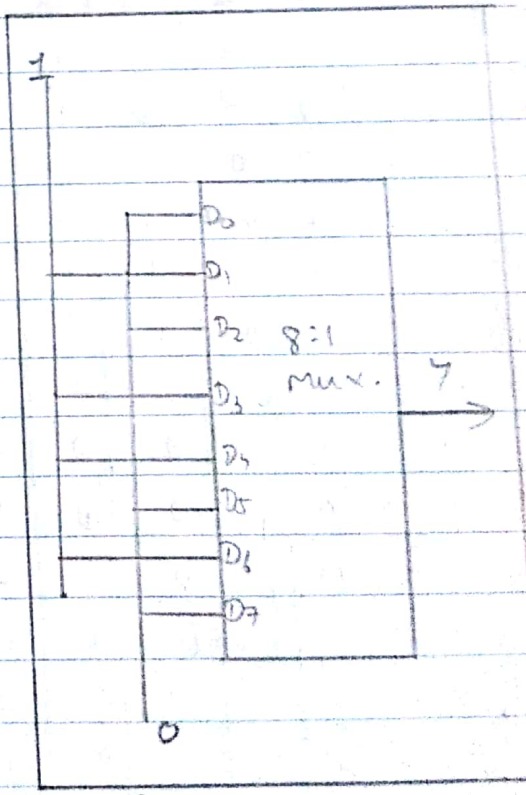
D_0-D_{15} = Data input.
Y = Output $\bar{Y}$ = Complemented o/p
$\bar{E}$ = Enable pin.
GND = Ground.
S_0-S_2 = Select lines.
V_{cc} = +5V supply.

For eg:-

$F(A,B,C) = \Sigma m(1,3,4,6)$.

Truth Table:-

	Inputs			Outputs	I/P lines
Sr.no.	S_2	S_1	S_0	Y	
0	0	0	0	0	D_0
1	0	0	1	1	D_1
2	0	1	0	0	D_2
3	0	1	1	1	D_3
4	1	0	0	1	D_4
5	1	0	1	0	D_5
6	1	1	0	1	D_6
7	1	1	1	0	D_7



Connect D_1, D_3, D_4, D_6 to high & other inputs D_0, D_2, D_5 & D_7 to Ground, so we get high output for 1, 3, 4, 6 input lines.

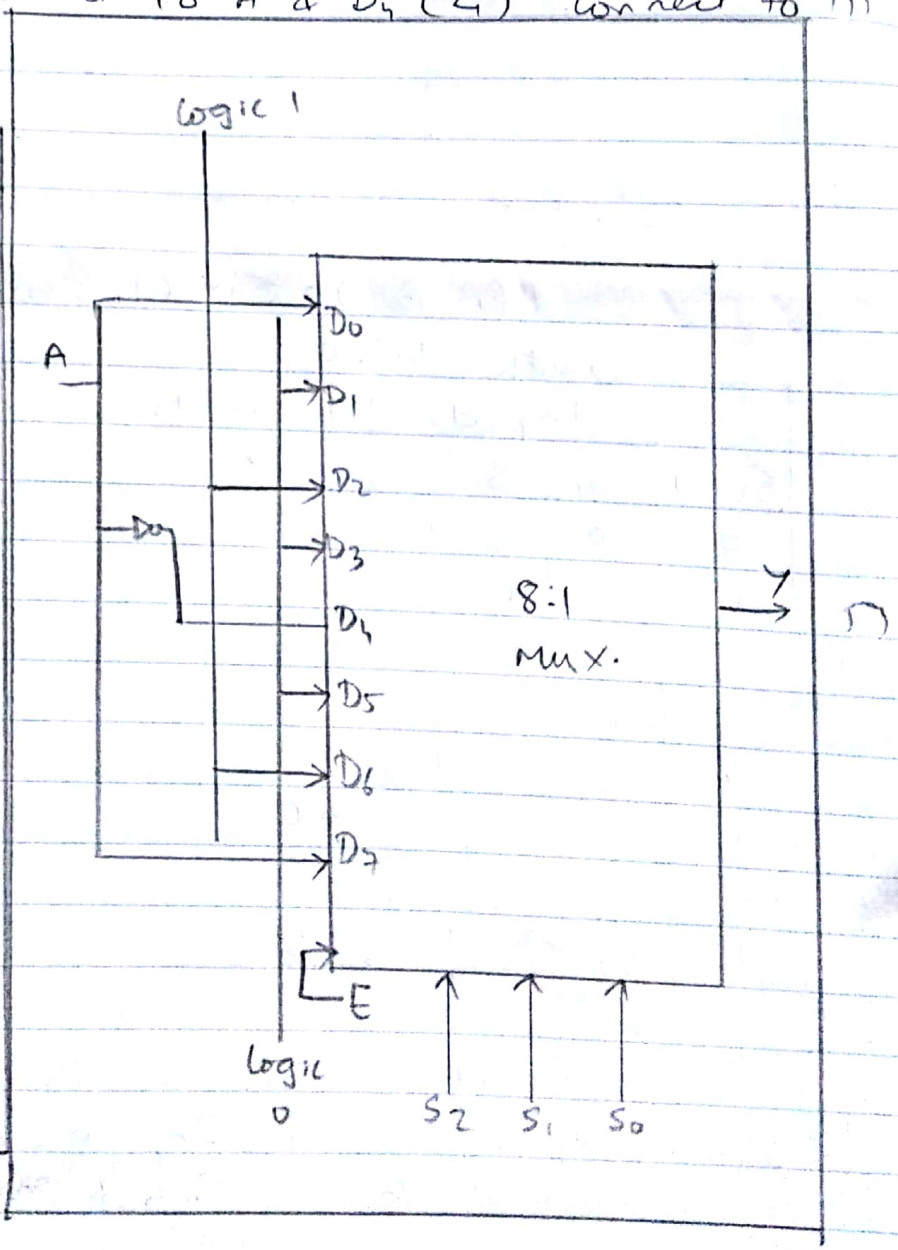
Q2) $f(A, B, C) = \Sigma M(2, 4, 6, 8, 10, 14)$.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	(2)	3	(4)	5	(6)	7
A	(8)	9	(10)	11	12	13	(14)	(15)
*	A	0	1	0	$\bar{A}$	0	1	A

Takes extra input select line A* for D_2 (2, 10) & D_6 (6, 14) connect to high, & for D_0 (8), connect A & D_7 (15) connect to A & D_4 (4) connect to $\bar{A}$.

Truth Table:-

	A	S_2	S_1	S_0	Y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1



Conclusion:-

In this way, we have studied functional pin diagram of 74151 (8:1 MUX) and its working.

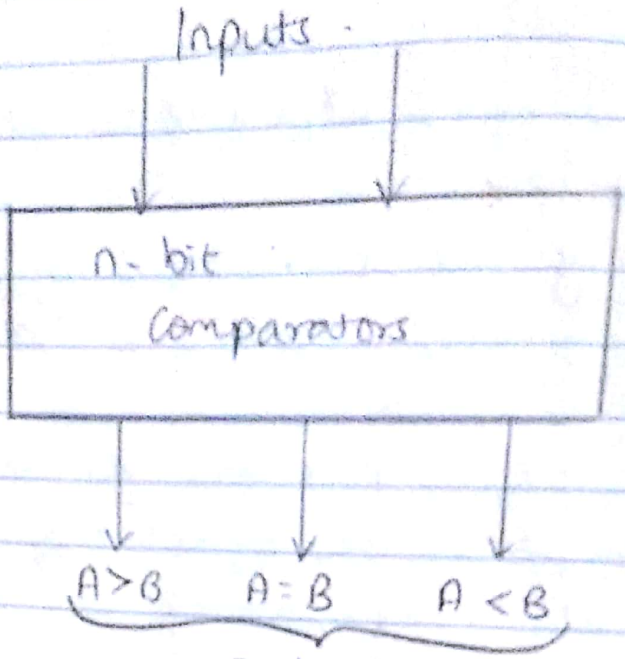
Experiment No 5

Aim: Verify the truth table of 1 bit & 2 bit comparator using logic gates & comparator IC.

Theory:

- Digital comparators are combinational circuits which compares the 2 n-bit binary words applied at its inputs.
- A comparator has 3 inputs namely $A > B$, $A = B$ and $A < B$. Depending on the result of comparison, one of these outputs will go high.
- Digital comparators are of 2 types.
- Identity comparators: Comparators that have only one output terminal and produces the output either low or high are identity comparators.
- Magnitude comparators:- comparators with 3 output terminal and checks for 3 condition i.e greater than or less than or equal to is magnitude comparators.

The below fig shows the block diagram of a n-bit comparator which compares the 2 ~~bit~~ numbers of n-bit length and generates their relation between themselves.



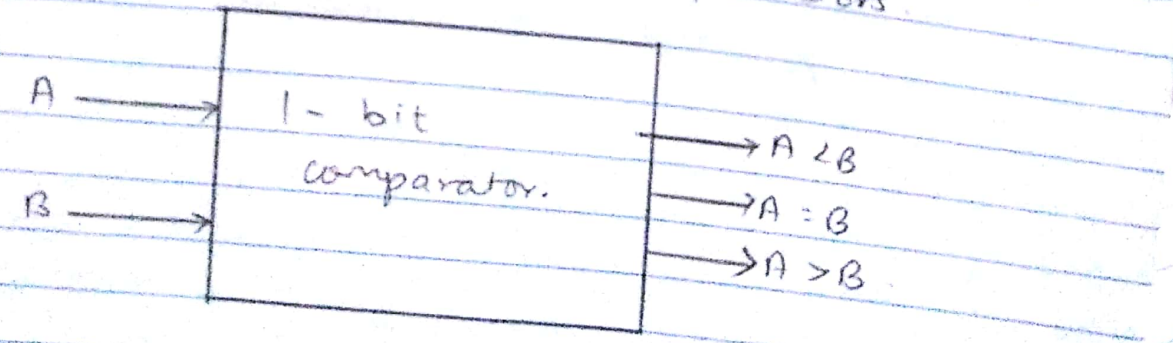
These comparators can compare 2 bit, 4 bit, 8 bit no. depending on the application requirement

One bit comparators:-

A comparator used to compare 2 bits, i.e 2 numbers each of single bit is called a single bit comparators.

It consists of 2 inputs for following 2 single bit numbers and 3 outputs to generate less than, equal & greater than comparison outputs.

The figure shows the block diagram of a single bit magnitude comparators.



Truth Table:-

When AB = 00 & 11, both inputs are equal. Therefore, A = B output will be high.
When AB = 01, B is more than A & hence A<B is active therefore A<B.

Inputs.		Outputs.		
A	B	$Y_1 = A<B$	$Y_2 = A=B$	$Y_3 = A>B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

K map for each other.

K map for $Y_1 = A<B$

A \ B	$\bar{B}$	B
$\bar{A}$	0	(1)
A	0	0

$Y_1 = (A<B)$
$= \bar{A}B$

K map for $Y_2 = A=B$

A \ B	$\bar{B}$	B
A	(1)	0
A	0	(1)

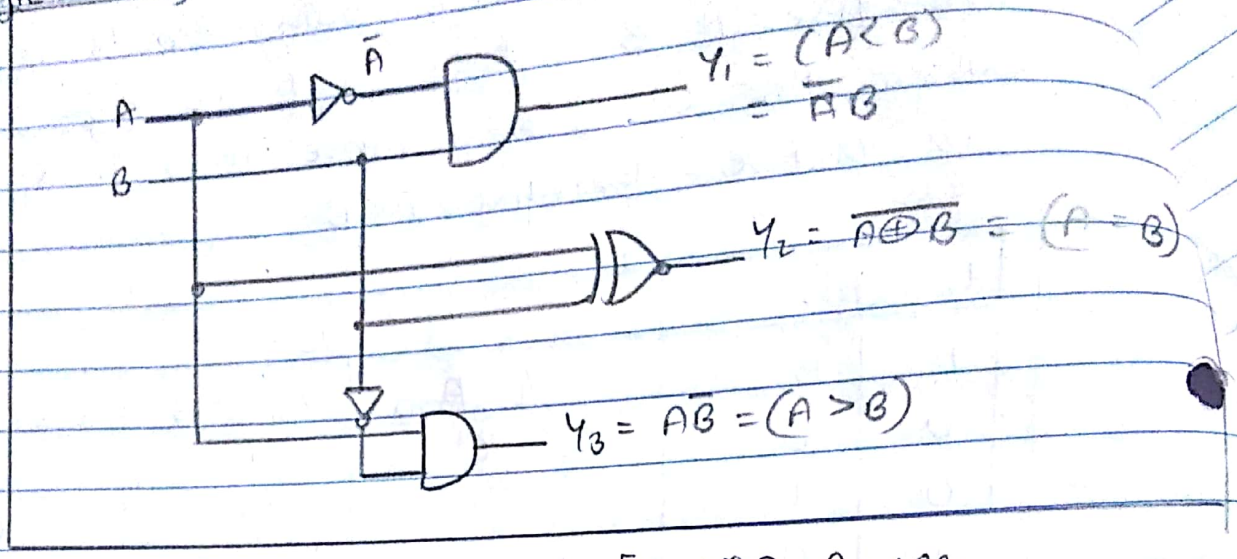
$Y_2 = (A=B) = \bar{A}\bar{B} + AB$
$= \overline{A \oplus B}$

K map for $Y_3 = A>B$

A \ B	$\bar{B}$	B
A	0	0
A	(1)	0

$Y_3 = A\bar{B}$.

Logic diagram of 1 bit comparators :-

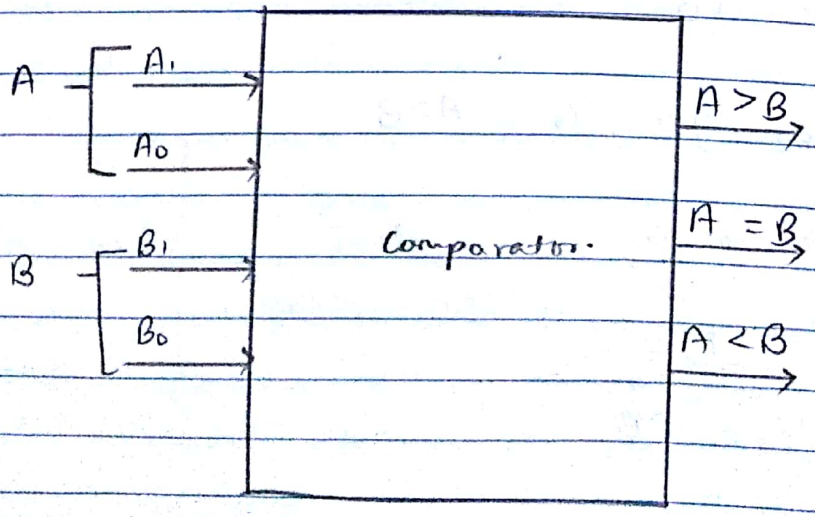


(b) Using AND and EX-NOR Gates.

2-bit comparators -

2 bit comparators compares 2 binary numbers each of 2 bits and produces their relation such as one number is equal or greater than or less than other.

The figure shows the block diagram of a 2-bit comparators which has four inputs and three outputs.



Truth Table:-

The first no. A is designed as $A = A_1 A_0$ and The second no is designed as $B = B_1 B_0$.

This comparators produces three outputs $(A > B)$, $(A = B)$, $(A < B)$.

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

K maps for each outputs -

K map for $A < B$:

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

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

K-map for $A > B$:

	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

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

K-map for $A = B$:

	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

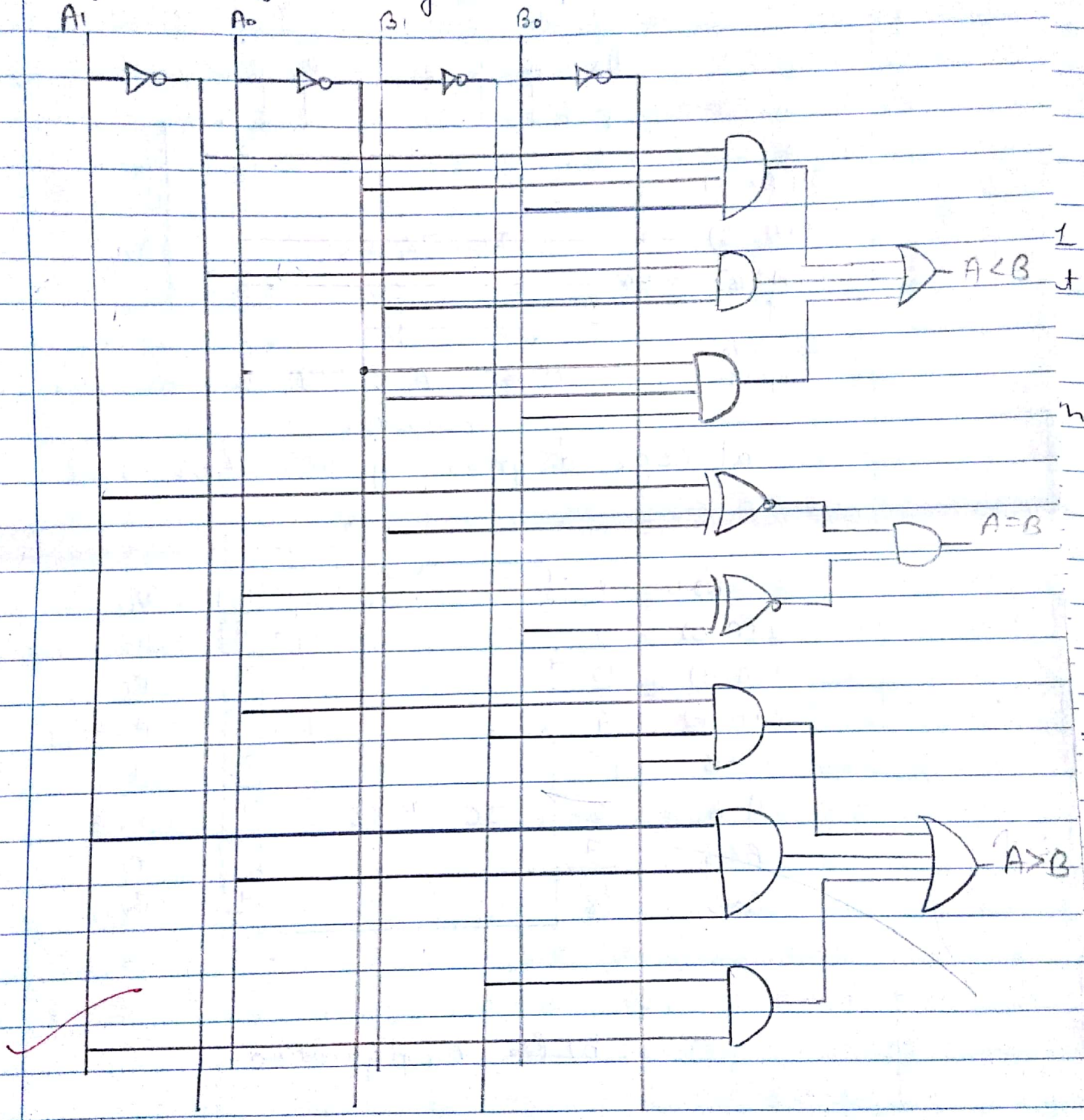
$(A=B) = \bar{A}_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0$

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

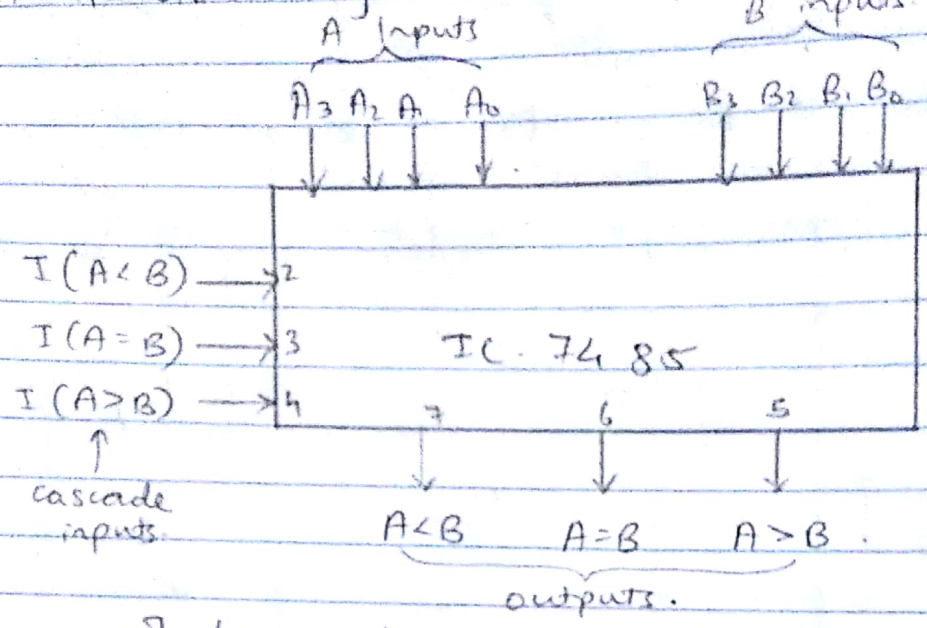
$= (\overline{A_1 \bar{B}_1} + A_1 B_1) (\overline{\bar{A}_0 \bar{B}_0} + A_0 B_0)$

$(A=B) = (A_1 \odot B_1)(A_0 \odot B_0)$, where, $\odot$ EX-NOR.

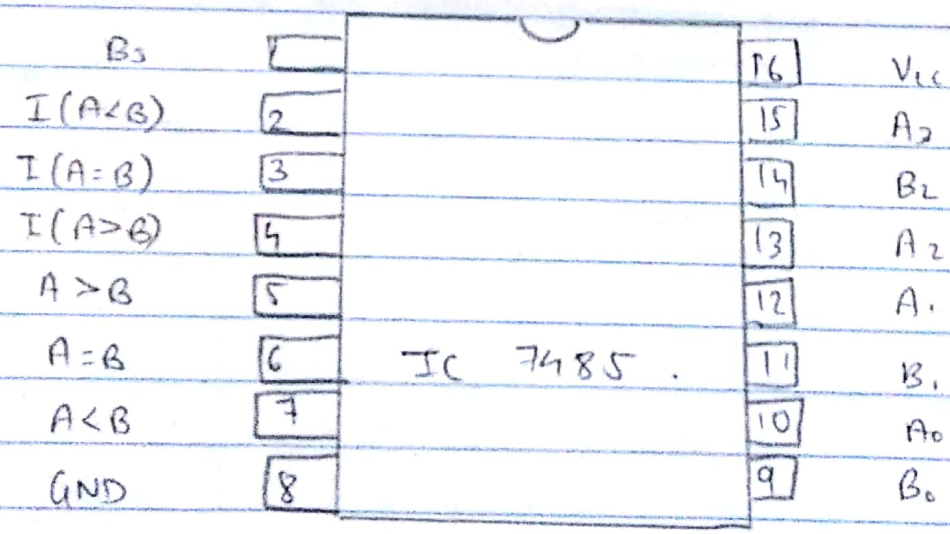
Logic diagram of 2 bit comparator:-



* A 4-bit magnitude comparators (IC 7485)



a] Logic diagram of IC 7485.



b) Pin configuration.

- IC 7485 is a 4 bit comparator in the integrated circuit form. It compares 2 4 bit words A ($A_3 - A_0$) & B ($B_3 - B_0$).
- It is possible to cascade more than one IC 7485 to compare words of almost any length.
- Above figure shows the pin configuration & logic symbol of IC 7485.

Pin names & their functions:-

Pin name	Pin number	Function.
A_0 to A_3	10, 12, 13, 15	Binary input active high (operand 1)
B_0 to B_3	9, 11, 14, 1	Binary input active high (operand 2)
I(A<B)	2	These lines are used for cascading a no of IC 7485 output of the previous stage are fed as input to this stage.
I (A=B)	3	
I (A>B)	4	
(A<B)	7	These are the outputs. When IC 7485 are cascaded, these o/ps are applied to cascading inputs of the next stage.
(A=B)	6	
(A>B)	5	

Truth Table of IC 7485:-

Comparing inputs.				Cascading i/P's.			Outputs		
$A_3 B_3$	$A_2 B_2$	$A_1 B_1$	$A_0 A_0$	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$Y_{A>B}$	$Y_{A<B}$	$Y_{A=B}$
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 > B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	L	H	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	X	X	H	L	L	H
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	H	L	L	L	L

Function table:-

- The function table tells us that if I (A<B) or I (A>B) is equal to 1 and if the other 2 cascading inputs are at logic 0 then the corresponding outputs (A<B or A>B) will be active.
- But if A=B, then the corresponding output will be active if only if I(A=B) is at logic 1 irrespective of the other 2 inputs.

Comparing inputs.	Cascading inputs.			Outputs.		
	$I_{A<B}$	$I_{A=B}$	$I_{A>B}$	$Y_{A<B}$	$Y_{A=B}$	$Y_{A>B}$
$A<B$	X	X	X	1	0	0
$A=B$	0	0	0	1	0	1
	0	0	1	0	0	1
	1	0	0	1	0	0
	1	0	1	0	0	0
	X	1	X	0	1	0
$A>B$	X	X	X	0	0	1

General description:-

As shown in truth table, the MSB are compared first, i.e. A_3 is compared with B_3 depending on whether $A_3 > B_3$ or $A_3 < B_3$ the o/p $Y(A>B)$ or $Y(A<B)$ are activated.

If $A_3A_2A_1A_0 = B_3B_2B_1B_0$ then IC 7485 will check the cascading inputs. The decision making will then take place as follows:-

Case 1: If $I(A>B)=1$ and $I(A=B)=I(A<B)=0$.

This indicates that at the previous stage, LSB of A is greater than LSB of B.

Case 2: If $I(A=B)=1$.

If $I(A=B)=1$ then the chip will not check status of $I(A<B)$ and $I(A>B)$ inputs and will understand that the LSB of A is equal to LSB of B.

Conclusion:-

In this way we studied the truth table of one bit and two bit comparators using logic gates and comparator IC.

3+3+4+3 = 13/15

4 pgs.

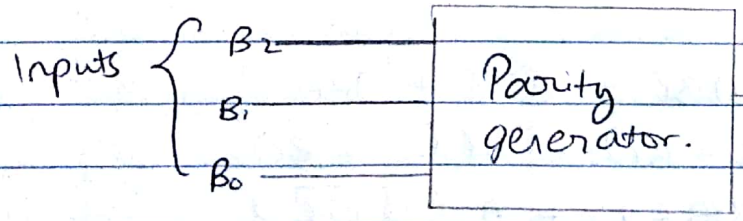
S1612037

Experiment No 6

Aim: Design & implement parity generator using EX-OR gates.

Theory:

- It is a combinational circuit that accepts an n-1 bit stream data & generates the additional bit that is to be transmitted with the bit stream. This additional or extra bit is termed as parity bit.
- A parity generator is a combinational logic circuit that generates the parity bit in the transmitter.
- The basic principle involved in the implementation of parity circuit is the sum of odd no of 1s is always 1 & sum of even no of 1s is always zero. Such error detecting & correction can be implemented by using EX-OR gates.
- To produce 2 bit sum, one EX-OR gate is sufficient. whereas for adding 3 bits, 2 EX-OR gates are required as shown below.



Block diagram of parity generator.

* Even Parity Generators:-

Let us assume that 3-bit message is to be transmitted with an even parity bit. Let 3 i/p's A, B, C are applied to the circuit & o/p bit is parity bit P. The total no. of 1's must be even to generate even parity bit P.

* Truth table of even parity generator:-

3-bit message.			Even parity bit generator
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

* The K-map for 3-bit even parity generator:

A \ BC	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

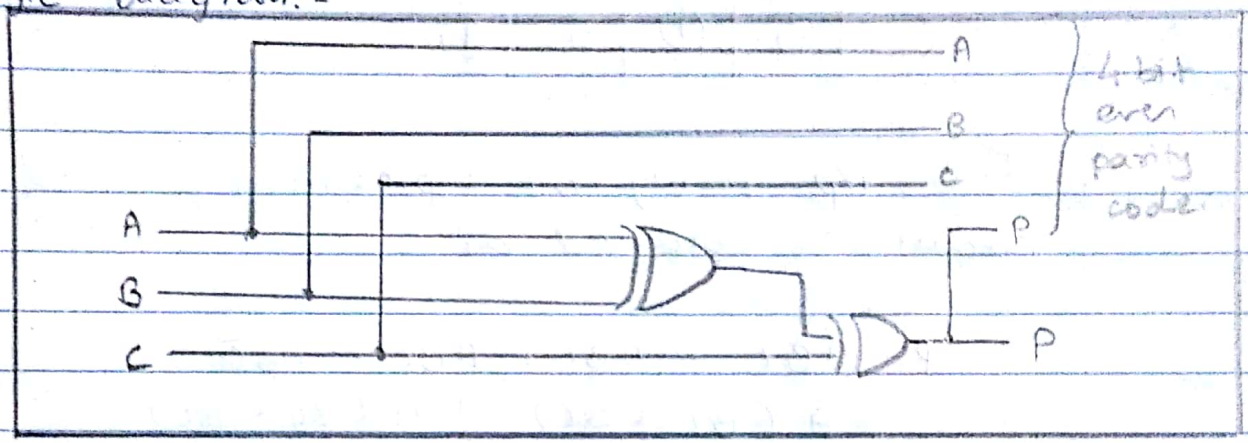
From above truth table, the simplified expression of the parity bit can be written as,

$$P = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC.$$
$$= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC)$$
$$= \bar{A}(B \oplus C) + A(\overline{B \oplus C})$$
$$P = A \oplus B \oplus C.$$

The above expression can be implemented by using 2 EX-OR gates. The logic diagram of even parity generator with two EX-OR gates is shown below.

To generate the even parity bit for a 4-bit data, 3 Ex-OR gates are required to add the 4bit & their sum will be parity bit.

Logic diagram:-



* Odd Parity Generator :-

-Let us consider that the 3 bit data is to be transmitted with an odd parity bit. The 3 i/p are A, B, C and P is the o/p parity bit. The total no of bits must be odd in order to generate the odd parity bit.

-In the given truth table, 1 is placed in the parity bit in order to make the total no of bits odd when the total no of 1's in the truth table is even.

3 bit message			Odd parity 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	1	0

The truth table of odd parity generator can be simplified by using K-map as

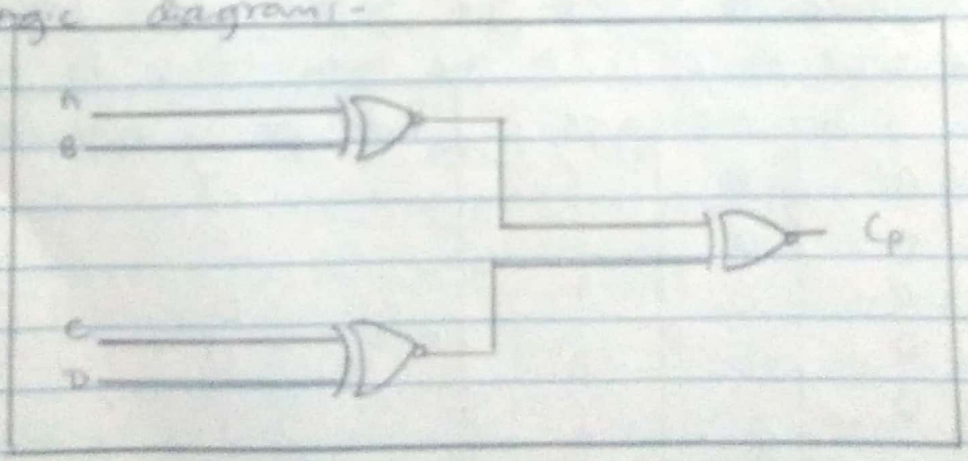
A \ BC	00	01	11	10
0	(01)	0	(1)	0
1	0	(1)	0	(1)

The op parity bit expression for this generator circuit is obtained as

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

The above Boolean expression can be implemented by using one Ex-OR gate & one Ex-NOR gate in order to design a 3-bit odd parity generator.

Logic diagram:-



* Conclusion:-

In this way we studied the design & implementation of parity generator using EX-OR.

3+3+4+4 = 14/15

Experiment No. 7.

Aim: Flip Flop conversion: Design and realization.

Theory:-

- The conversion of one type of flip-flop to the other needs a systematic approach using the excitation tables and K-map simplifications.
- fig 3.16.1 shows a generalized model for conversion from one flip flop to the other.
- As shown in fig, the req'd flip flop is actually a combination of the given ff and a combinational logic circuit using gates.
- The inputs to ff conversion logic are, the ff data inputs and the outputs of given f f. i.e. the given ff and the desired ff.
- The conversion logic is designed by combining the exitation tables of both the ff.

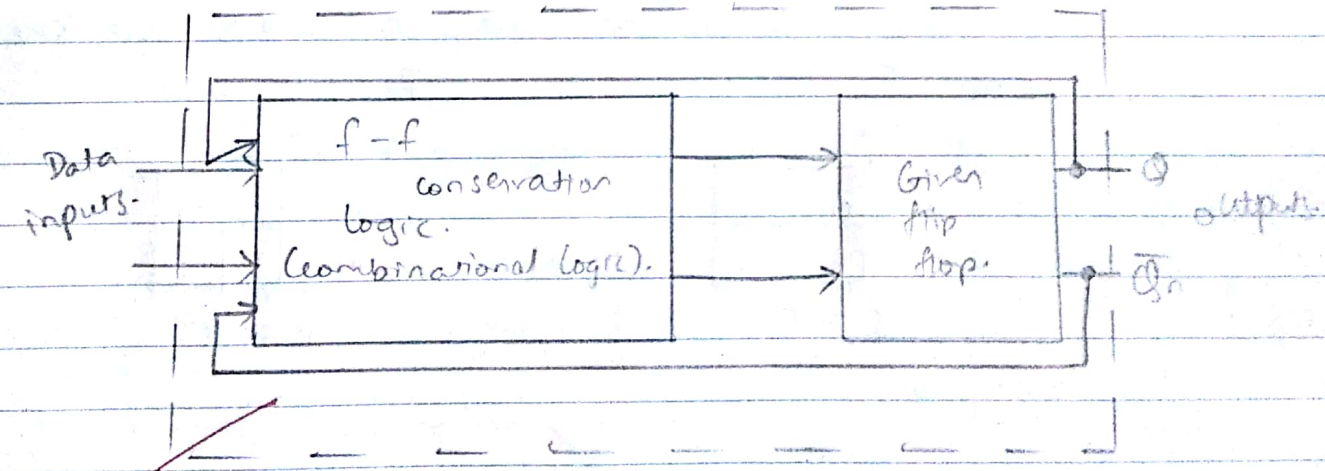


fig. Gen. model used to convert 1 type of ff to other.

- The truth table of a conversion logic as data inputs and Q & $\bar{Q}$ o/p of the given ff as

inputs where as the input of the given FF are the o/p of the truth table.
- Then we draw K-map for each o/p and obtain the simplified expression, the conversion logic is then implemented using gates.

ij.

Conversion of J-K to T FF :-

Step 1: Write the truth table for a conversion. The required truth table is obtained from the excitation tables of JK and T FF as follows.

T	i/p. Present state Q_n	Next state Q_{n+1}	o/p. J	K
0	0	0	0	X
1	0	1	1	X
1	1	0	X	1
0	1	1	X	0

Step 2: K-maps and simplification.
The K-maps for outputs J & K are shown in fig.

- for J o/p.

T \ Q_n	0	1
0	0	X
1	1	X

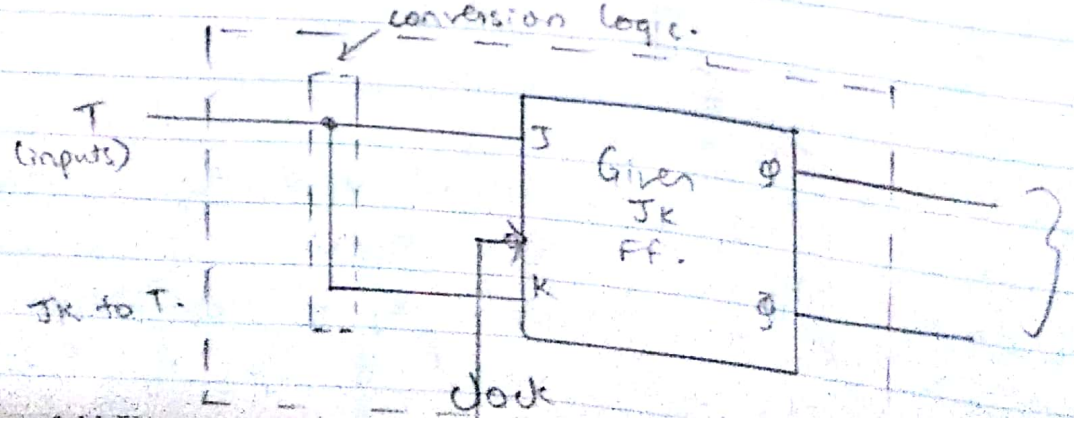
$J = T$.

- for K o/p.

T \ Q_n	0	1
0	X	0
1	X	1

$K = T$.

Step 3:-



Conversion of JK to D:-

Step 1: Truth Table.

D	I/P.		O/P.	
	Present State Q_n	Next State Q_{n+1}	J	K
0	0	0	0	X
1	0	1	1	X
0	1	0	X	1
1	1	1	X	0

Step 2: k-maps and simplification.

for J output:-

D \ Q_n	0	1
0	0	X
1	1	X

$J = D$.

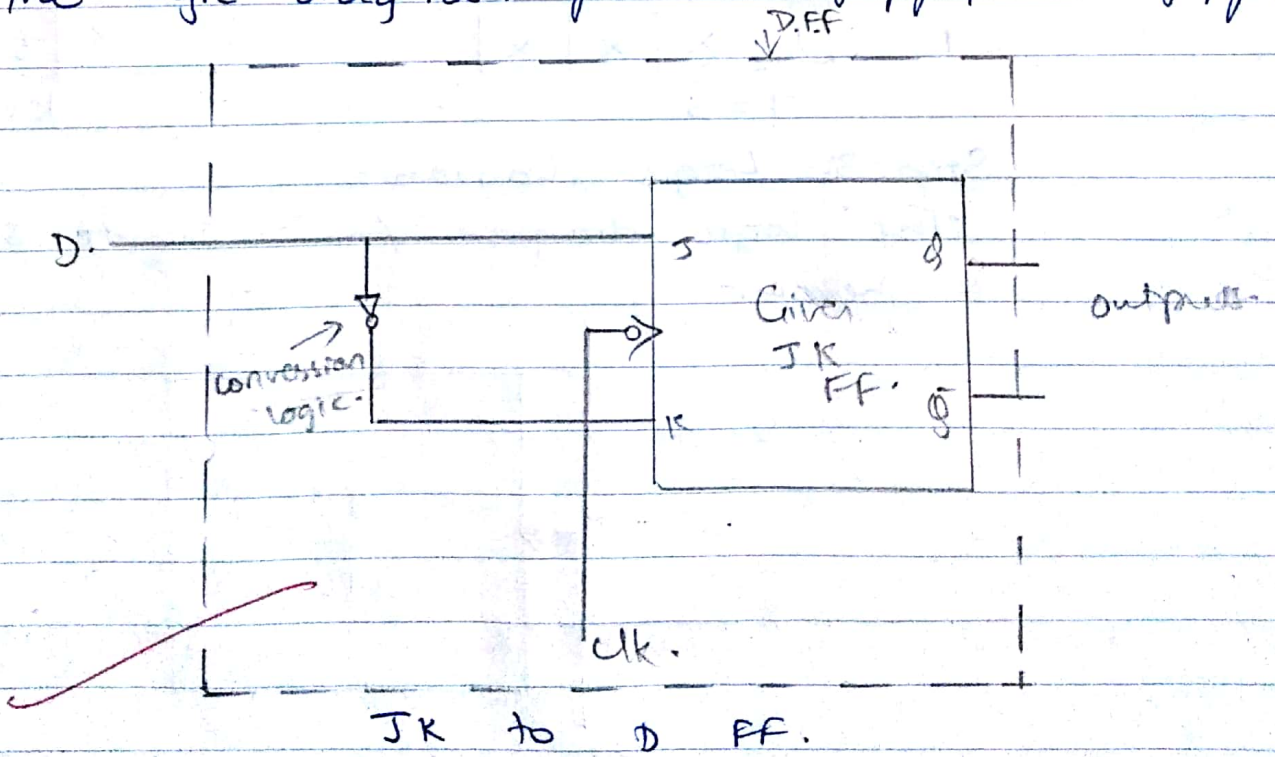
for K output.

D \ Q_n	0	1
0	X	1
1	X	0

$K = \overline{D}$.

Step 3:

The logic diagram for JK flipflop to D flipflop



JK to D FF.

Conversion of J-K to S-R :-

Step 1: write the truth table for S-R.

S	R	I/P.		O/P.	
		Present state Q_n	Next State Q_{n+1}	J	K.
0	0	0	0	0	X
0	1	0	0	0	X
1	0	0	1	1	X
1	0	0	1	1	X
0	1	1	0	X	1
0	1	1	0	X	1
0	0	1	1	X	0
1	0	1	1	X	0

Step 2:-

for J o/p:-

S \ RQn	00	01	11	10
0	0	X	X	X
1	1	X	X	X

$J = S$

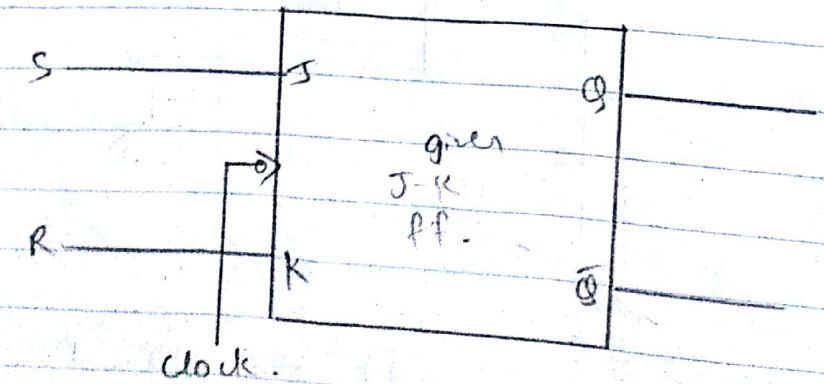
for K o/p:-

S \ RQn	00	01	11	10
0	X	0	1	X
1	X	0	X	X

$K = R$.

Step 3:- Logic diagram:-

The logic diagram for JK to SR FF shown below:-



JK to SR.

2] Conversion of D to J-K:-

Step 1: truth table:-

i/p.				o/p.
J	K	Present state Q_n	Next state Q_{n+1}	D
0	X	0	0	0
1	X	0	1	1
X	1	1	0	0
X	0	1	1	1

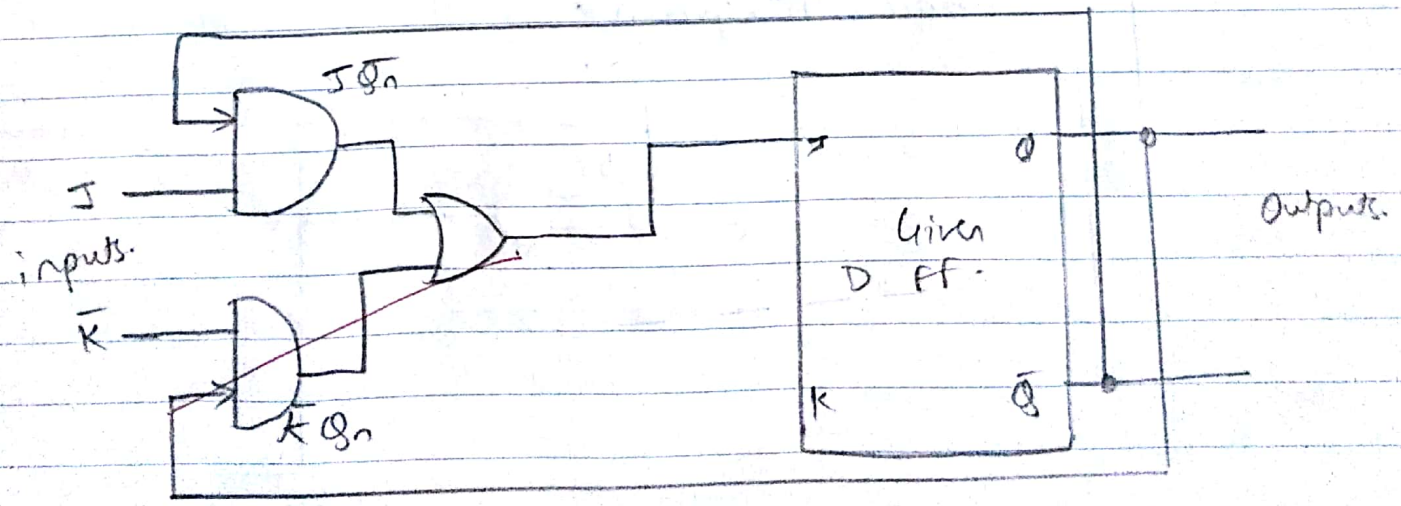
Step 2: K-maps and simplifications:-

for o/p D:-

	00	01	11	10
0	0	1	0	0
1	1	1	0	1

↑ $\bar{K}Q_n$ ↖ $J\bar{Q}_n$

$$D = \bar{K}Q_n + J\bar{Q}_n$$



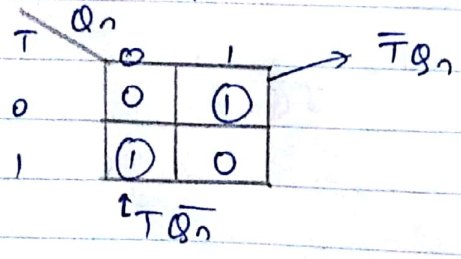
Conversion of D to T:-

Step 1:
Write the truth table for D to T FF conversion.

i/p.			o/p.
T	Present State Q_n	Next State Q_{n+1}	D
0	0	0	0
1	0	1	1
1	1	0	0
0	1	1	1

Step 2:
K-map and simplification and logic diagram.

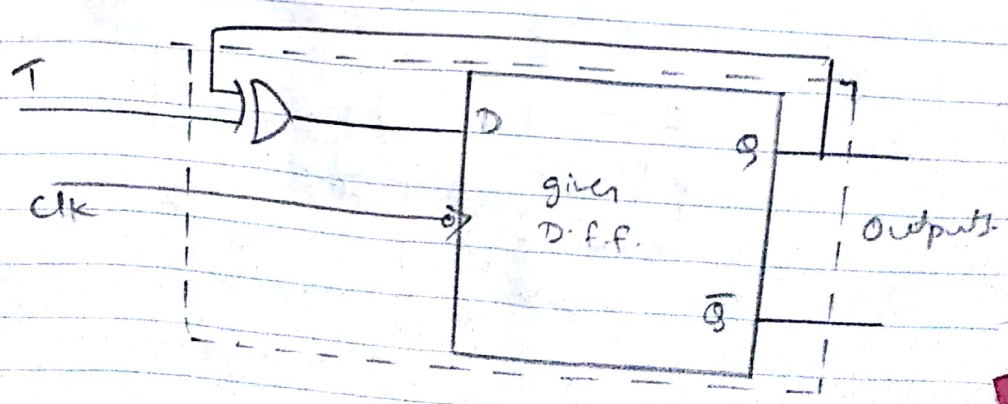
for output D:-



$D = T\bar{Q}_n + \bar{T}Q_n$.

$D = T \oplus Q_n$.

Step 3:-
Logic Diagram:-



Conversion of D to T Flipflop.

Conversion of D FF to SR FF:-

Step 1: Write the truth table.

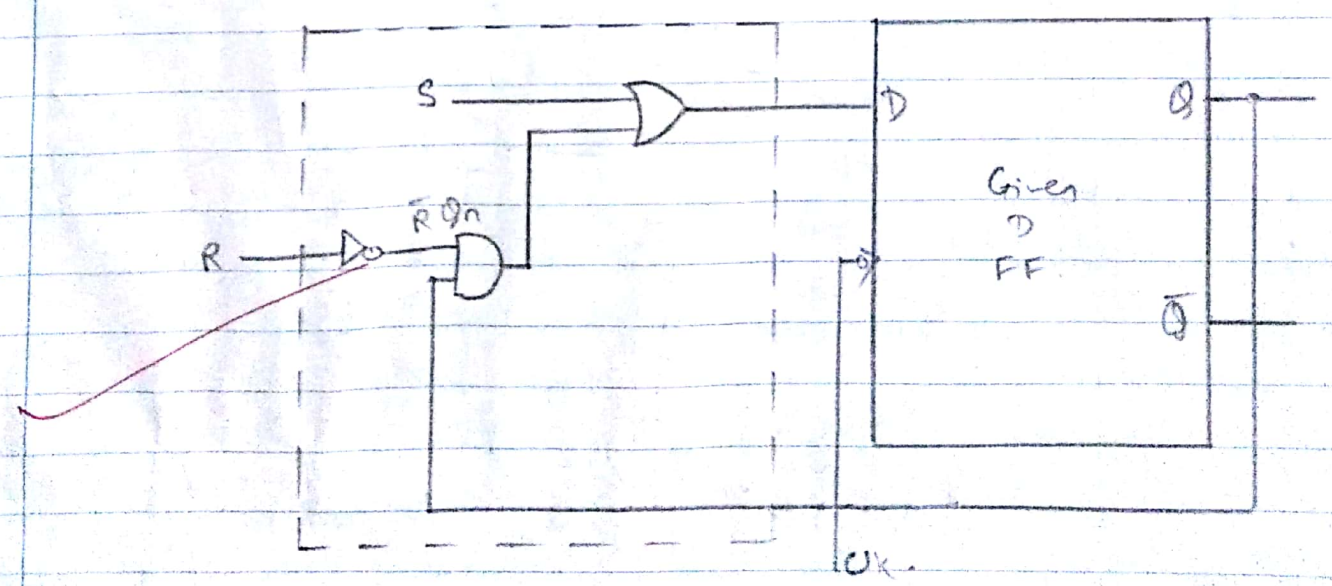
i/p.				o/p.
S	R	Present State Q_n	Next State Q_{n+1}	D.
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
0	1	1	0	0
0	0	1	1	1
1	0	1	1	1

Step 2: K-maps and simplifications-

S \ RQ_n	00	01	11	10
0	0	1	0	0
1	1	1	X	X

$$D = S + \bar{R} Q_n$$

Step 3: Logic Diagram:



Logic Diagram for D to SR Flipflop.

Conclusion:-

Hence, we have studied flip-flop conversion: Design and realization.

Experiment No - 9(A).

Aim: Realization of 3-bit up/down counter using MS JK flip flop.

Theory: A counter is a circuit that produces a set of unique output combinations in relation to the number of unique outputs of a counter is known as modulus and the counter having N no. of unique output is called mod N counter.

But in up/down counters it provides mode control i.e. M to select either up count or down count mode of operation.

Combinational circuit is required to be designed & used between each pair of flipflops in order to achieve the up/down operations.

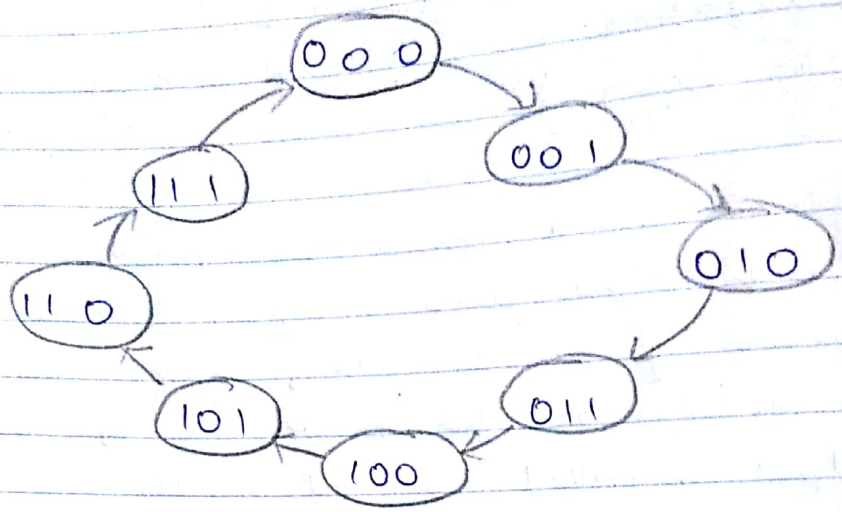
Step 1: Decide no of flip flops, as it is 3-bit counter, therefore no of flip flops is 3.

Step 2: We are going to design 3 bit up/down counter using JK flipflops. So we require excitation table of JK flipflop.

Excitation Table of JK flip flop:-

Present State	Next State	FF Inp.	
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3: State diagram:-



Circuit Excitation Table.

M	Present State			Next State			Flip Flop Inputs					
	Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	0	1	0	X	0	X	1	X
0	0	0	1	0	1	0	0	X	1	X	X	1
0	0	1	0	0	1	1	0	X	X	0	1	X
0	0	1	1	1	0	0	1	X	X	1	X	1
0	1	0	0	1	0	1	X	0	0	X	1	X
0	1	0	1	1	1	0	X	0	1	X	X	1
0	1	1	0	1	1	1	X	0	X	0	1	X
0	1	1	1	0	0	0	X	1	X	1	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

Step 4: K maps -

for JC

Q_C \ $Q_B Q_A$	00	01	11	10
00	0	0	1	0
01	X	X	X	X
11	X	X	X	X
10	0	0	1	0

$\therefore J_C = Q_B . Q_A .$

for KC,

Q_C \ $Q_B Q_A$	00	01	11	10
00	X	X	X	X
01	0	0	1	0
11	0	0	1	0
10	X	X	X	X

$\therefore K_C = Q_B Q_A .$

for JB.

Q_C \ $Q_B Q_A$	00	01	11	10
00	0	1	X	X
01	0	1	X	X
11	0	1	X	X
10	0	1	X	X

$\therefore J_B = \overline{Q}_A .$

- for K_B

Q_C \ $Q_B Q_A$	00	01	11	10
00	X	X	1	0
01	X	X	1	0
11	X	X	1	0
10	X	X	1	0

$\therefore K_B = Q_A$.

For J_A

Q_C \ $Q_B Q_A$	00	01	11	10
00	1	X	X	1
01	1	X	X	1
11	1	X	X	1
10	1	X	X	1

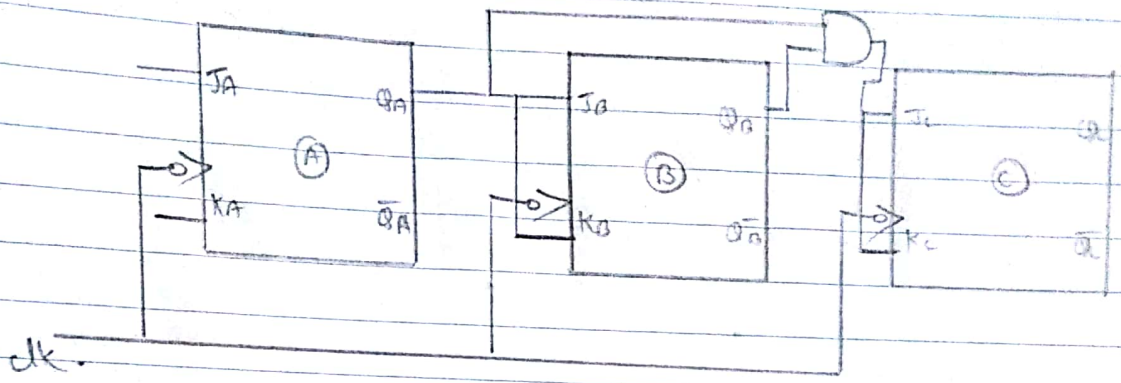
$\therefore J_A = 1$.

for K_A

Q_C \ $Q_B Q_A$	00	01	11	10
00	X	1	1	X
01	X	1	1	X
11	X	1	1	X
10	X	1	1	X

$\therefore K_A = 1$.

Step 5: Logic Diagram:-



Conclusion:- Hence we studied & realized 3-bit up/down counter using MS JK flip flop.

C

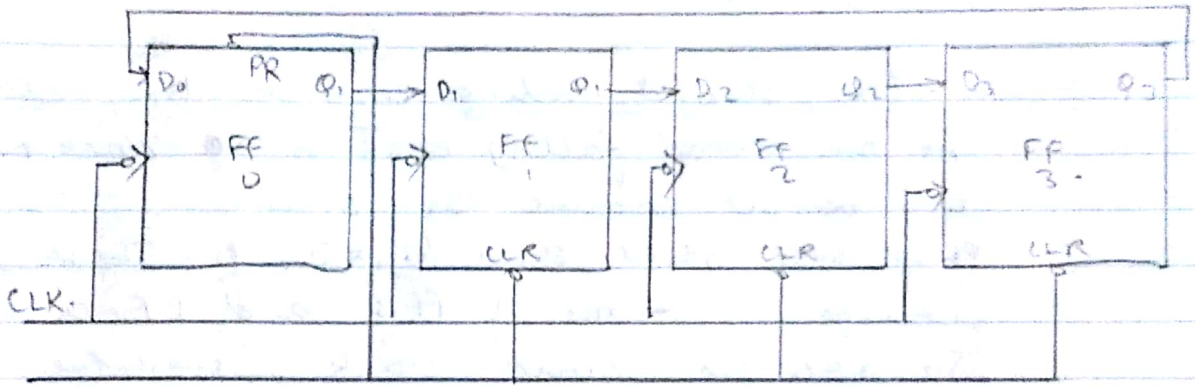
S1617037 -

Experiment No. 10

Aim: Design and realization of ring counter and Johnson's ring counter.

Theory:-

Ring counter:-



The above figure shows a typical application of shift registers called ring counters.

Operation:-

- Initially a low clear (CLR) pulse is applied to all the flip flops.
- Hence FF-3, FF-2, and FF-1 will reset but FF-0 will be present. So the outputs of shift register are

$$Q_3\ Q_2\ Q_1\ Q_0 = 0001.$$

- Now clear terminal is made inactive by applying a high level to it..
- The clock signal is then applied to all the flip flops simultaneously.

- On the first negative going clock edge:-
- As soon as the first falling edge of the clock hits, only FF-1 will set because $Q_0 = D_1 = 1$.
- The FF-0 will reset because $D_0 = Q_3 = 0$ and there is no change in the status of FF-2 and FF-3.
- Hence after the 1st clock pulse the output are $Q_3 Q_2 Q_1 Q_0 = 0010$.

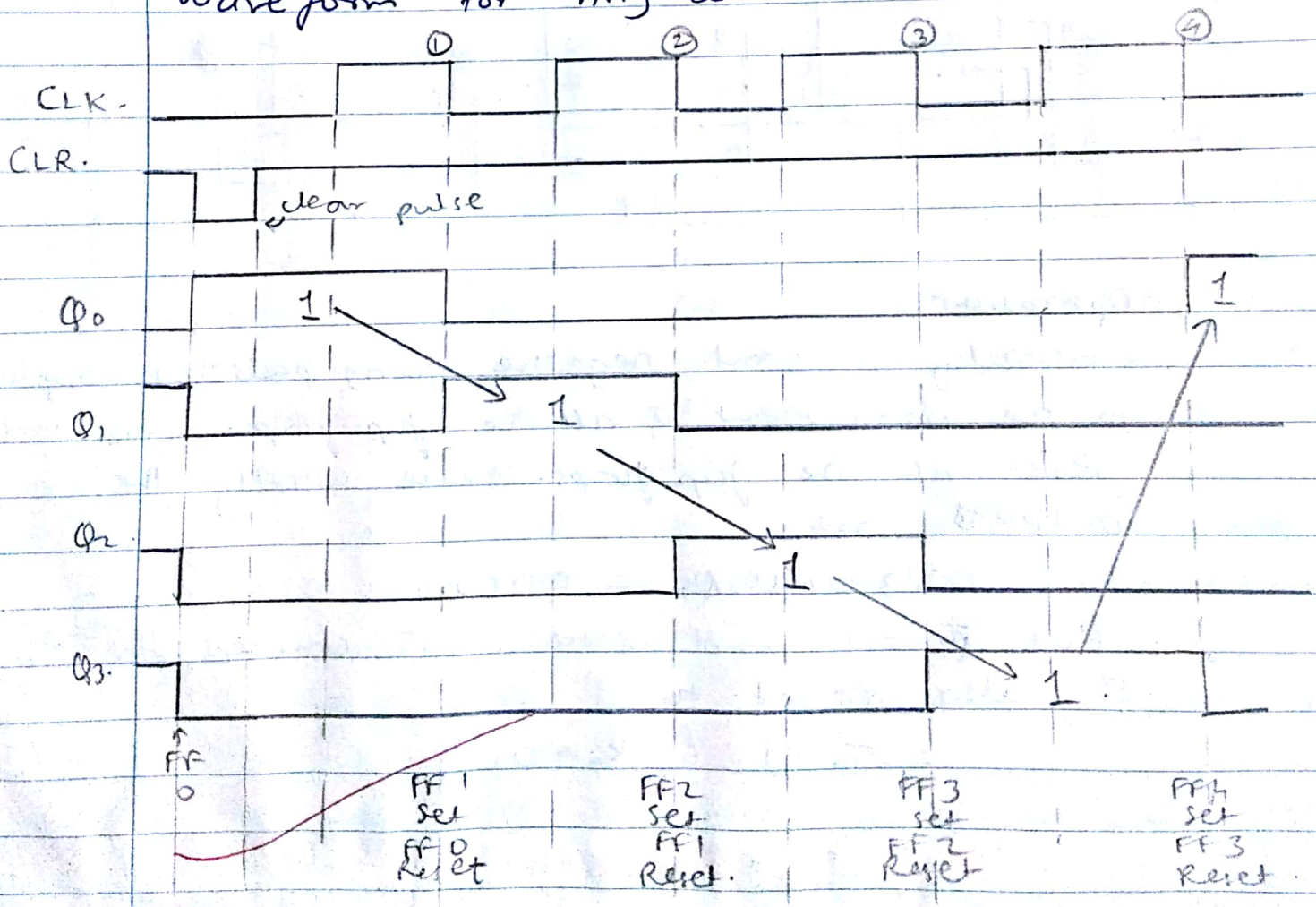
- On the second falling edge of the clock:-
- At the second falling edge of the clock, only FF-2 will be set because $D_2 = Q_1 = 1$.
- FF-1 will reset since $D_2 = D_0 = 0$. There is no change in status of FF3 and FF-0.
- So after the second clock pulse, the outputs are $Q_3 Q_2 Q_1 Q_0 = 0100$.
- Similarly after 3rd clock cycle, the o/p is $Q_3 Q_2 Q_1 Q_0 = 1000$.
- And after the 4th one the o/p are $Q_3 Q_2 Q_1 Q_0 = 0001$.

Hence the operation repeats from this point onwards.

* Summary of operation of ring counter.

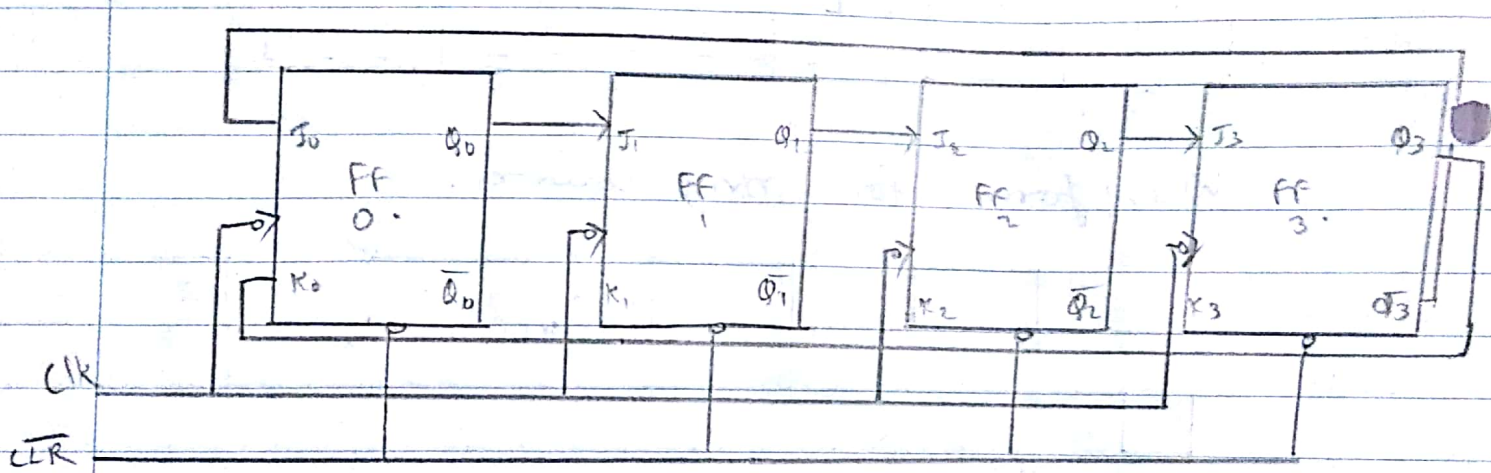
CLR	CLK	Q_0	Q_1	Q_2	Q_3
⊓	x	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	0	0	0	1
1	↓	1	0	0	0

Waveform for ring counter.



Jhonson's counter:-

In the ring counter the outputs of FF 3 were connected directly to the inputs of FF-0 i.e. Q_3 to J_0 $\overline{Q_3}$ to K_0. Instead if the outputs are cross coupled to the inputs i.e. if Q_3 is connected to the K_0 and $\overline{Q_3}$ is connected to J_0 then circuit is called as twisted ring counter or Johnson's counter.



Operation:—

- Initially a short negative going pulse is applied to the clear input of all the flip flops. This will reset all the flip flops. Hence initially the outputs are

$$Q_3\,Q_2\,Q_1\,Q_0 = 0000.$$

- But $\overline{Q_3} = 1$ and since it is coupled to J_0, it is also equal to 1.

$$\therefore J_0 = 1 \;\&\; K_0 = 0.$$

- On the 1st falling edge of clock pulse,

- As soon as the 1st negative edge of clock arrives, FF-0 will be set. Hence Q_0 will become 1.
- But there is no change in the status of any other flip-flop.
- Hence after the first negative going edge of the clock, the flip flop outputs are-

$$Q_3 Q_2 Q_1 Q_0 = 0001.$$

- On the second negative going clock edge -

- Before the second negative going clock edge, $Q_3 = 0$ & $\bar{Q}_3 = 1$. Hence, $J_0 = 1$ & $K_0 = 1$. Also $Q_0 = 1$ Hence $J_1 = 1$.
- Hence as soon as the second falling edge of clock arrives, FF0 continues to be in the set mode & FF1 will now set. Hence Q_1 will become 1 and $\bar{Q}_1 = 0$.
- There is no change in the ~~out~~ status of any other FF.
- Hence after the second clock edge the outputs are

$$Q_3 Q_2 Q_1 Q_0 = 0011.$$

- Similarly after the 3rd clock pulse, outputs are

$$Q_3 Q_2 Q_1 Q_0 = 0111.$$

- And after the 4th clock pulse, outputs are

$$Q_3 Q_2 Q_1 Q_0 = 1111.$$

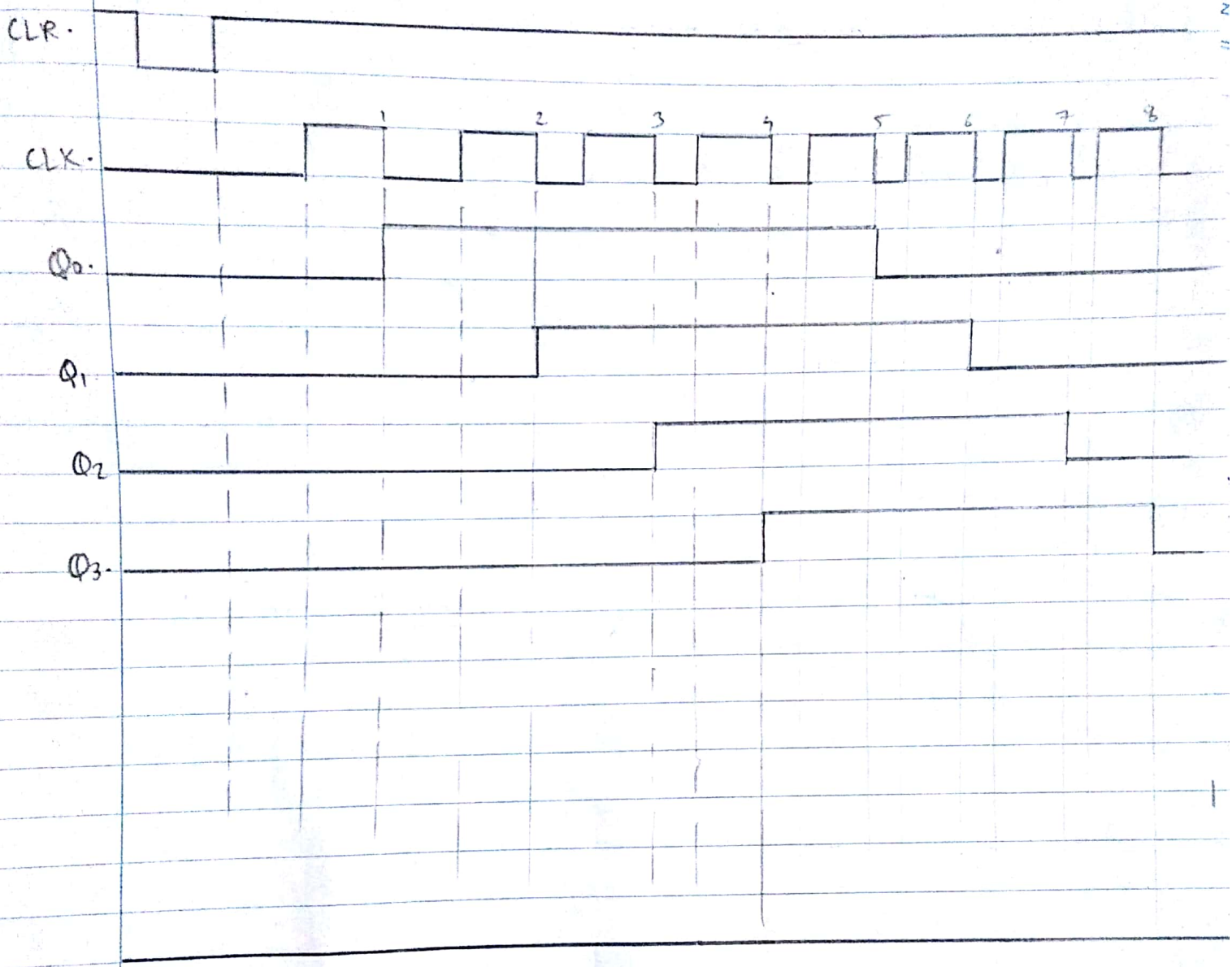
- As soon as fifth negative going clock pulse strikes, FF-0 will reset. But the outputs of other flip flops will remain unchanged so after 5th clock pulse, outputs are
  $Q_3 Q_2 Q_1 Q_0 = 1110$.

- Operation will continue till we reach all the zero output.

Summary of operation :-

$\overline{CLEAR}$	CLK	Q_3	Q_2	Q_1	Q_0	State no.	Decimal eq.
0	initial	0	0	0	0	1	0
1	↓	0	0	0	1	2	1
1	↓	0	0	1	1	3	3
1	↓	0	1	1	1	4	7
1	↓	1	1	1	1	5	15
1	↓	1	1	1	0	6	14
1	↓	1	1	0	0	7	10
1	↓	1	0	0	0	8	8
1	↓	0	0	0	0	1	0

Waveform for Jhonson's counter.



Conclusion:- Hence we studied design and realization of ring counter & Johnson's ring counter.

Experiment No. 11

Aim: Design & implement sequence generator using JK flip flop.

Theory:- Sequence Generator:-

- A sequence generator is a sequential ckt which generates the prescribed sequence at its output. The output sequence is produced in synchronization with the clock i/p.
- It is possible to design a sequence generator using either counters or using the shift registers.

Sequence generator using JK flip flop :- [100100].

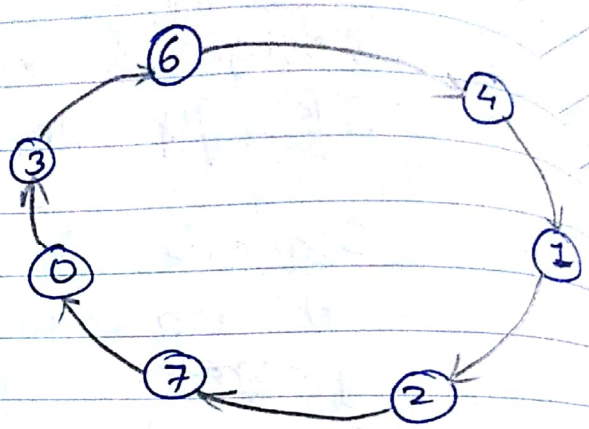
Step 1: Decide the no of flip-flop:-

- Number of 1's = 3 and number of 0's = 4.
- So select higher one of them i.e. 4. So N = 4.
- Hence $N = 2^{n-1}$, so $4 << 2^{n-1}$.
- Therefore no of flip flops n = 3.

Step 2: Write the state table-

- Assume that the required sequence is obtained at the output of flip-flop C. The state assignment for the others outputs is show in table.

Flip flop output			State
C	B	A	
1	0	0	4
0	0	1	1
0	1	0	2
1	1	1	7
0	0	0	0
0	1	1	3
1	1	0	6



State diagram.

Step 3: Draw the state diagram:

Step 4: Write the excitation table:-
The excitation table of JK FF is given in table (a) & the ckt excitation table is shown in table (b).

Present State Q_n	N.S. Q_{n+1}	FF inputs. J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation table of JK FF.

P. S.			N. S.			Flip Flop inputs.					
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	1	0	X	1	X	X1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	1	1	1	X	X	0	1	X
0	1	1	1	1	0	1	X	X	0	X	1
1	0	0	0	0	1	X	1	0	X	1	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	1	0	0	X	0	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

Step 5:- K-maps & simplified Boolean expression -

For J_C

C \ BA	00	01	11	10
0	0	0	1	1
1	X	X	X	X

$J_C = B$.

For K_C

C \ BA	00	01	11	10
0	X	X	X	X
1	1	X	1	0

$K_C = A + \bar{B}$.

for J_B.

C \ BA	00	01	11	10
0	1	1	X	X
1	0	X	X	X

$J_B = \bar{C}$

for K_B.

CBA	00	01	11	10
0	X	X	0	0
1	X	X	1	1

$K_B = C$.

for J_A

CBA	00	01	11	10
0	1	X	X	1
1	1	X	X	0

$J_A = \bar{B} + \bar{C}$

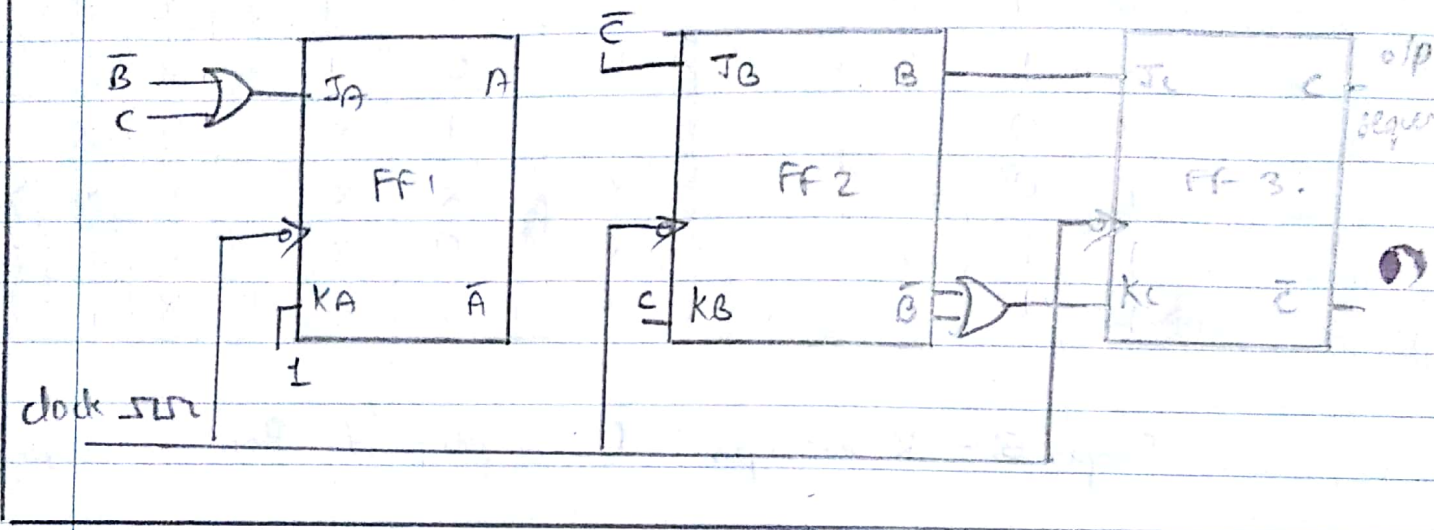
for K_A

CBA	00	01	11	10
0	X	1	1	X
1	X	X	1	X

$K_A = 1$.

Step 6:- Draw the logic diagram:-

The logic diagram is shown in fig:-



Logic diagram of required sequence generator.

Conclusion:-

From the above experiment we studied, how to design & implement sequence generator using JK flip-flop.