UNIT - 2

COMBINATIONAL CIRCUITS & SEQUENTIAL CIRCUITS

Introduction :-

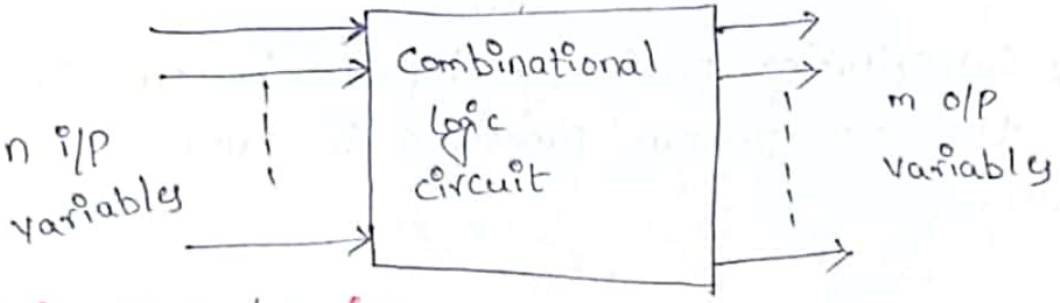
→ When logic gates are connected together to produce a specified o/p for certain specified combinations of i/p variably, with no storage involved, the resulting circuit is called Combinational circuit.

→ In Combinational logic, the o/p variably are at all times dependant on the combination of i/p variably.

→ A Combinational ckt consists of i/p variably, logic gate, and o/p variably.

→ The logic gates accept signals from the i/p variably and generate o/p signals.

→ The below figure shows the block diagram of a Combinational ckt.



Analysis Procedure :-

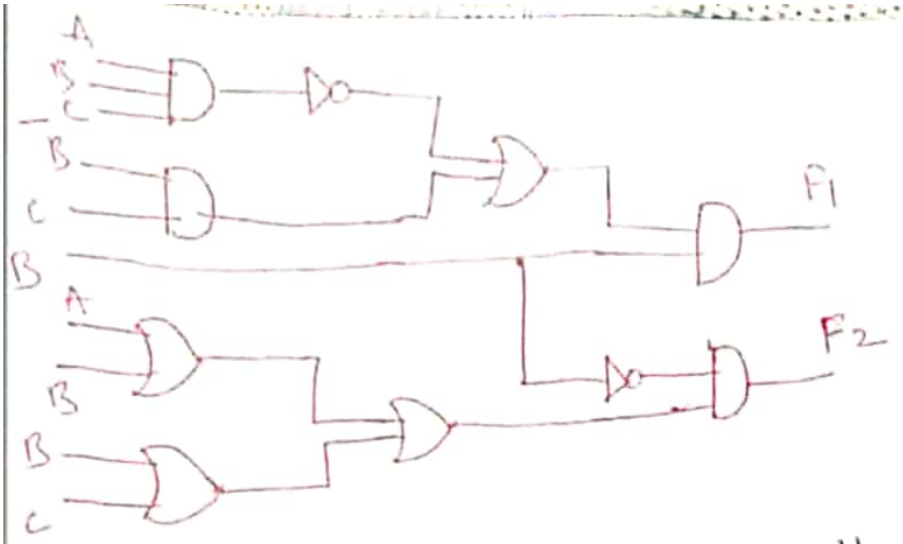
→ The analysis of a Combinational ckt is the procedure by which we can determine the function that the circuit implement.

→ In this procedure from the given ckt digram we have to obtain a set of Boolean functions for o/p's of circuit, a truth table (or) a possible explanations of the ckt operation.

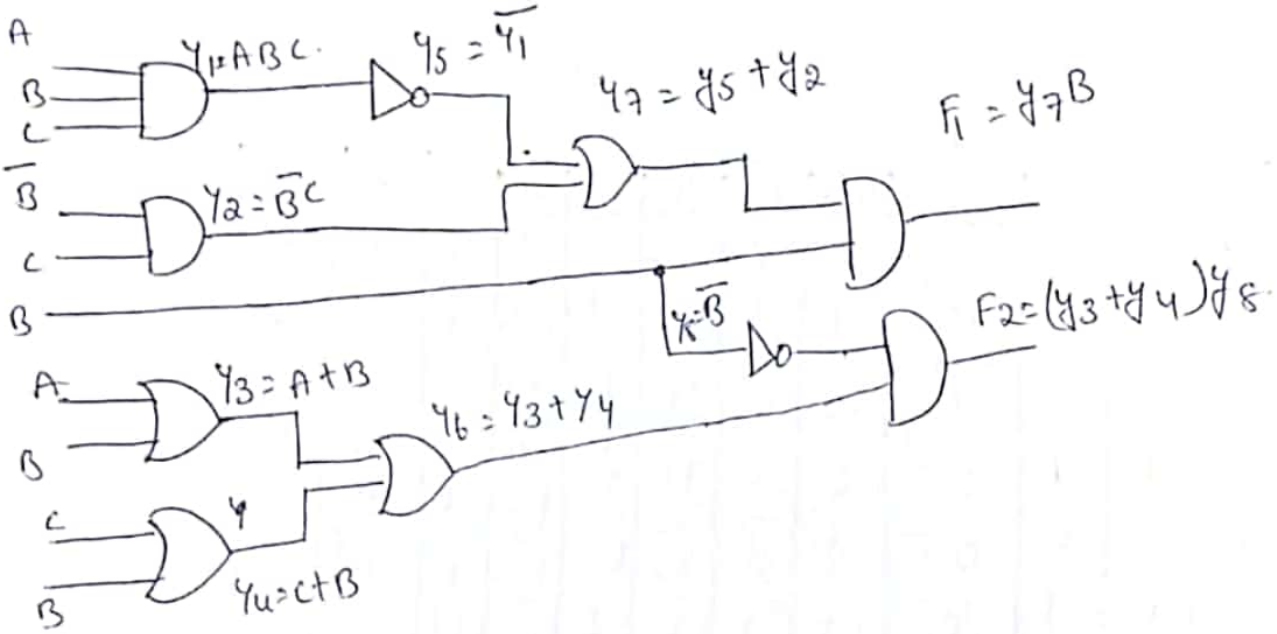
→ let see the procedure to determine the Boolean functions for o/p's of circuits from the given circuits.

1. first makes sure that given circuit is combinational circuit and not the sequential circuit.
2. The combinational circuit has logic gates with no feedback path (or) memory elements.
3. Label all gate o/ps that are a function of i/p variably with arbitary symbols, and determine the boolean functions for each gate o/p.
4. Label the gates that are a function of i/p variably and previously labeled gates and determine the boolean function for them.
5. Repeat the step 3 untill the Boolean function for o/ps of the circuit are obtained
6. finally, substituting previously defined boolean functions, obtain the o/p Boolean functions in terms of i/p variably.

Example :-1. obtain the boolean function for o/p's of the given circuit.



Sol:- After labeling the gate o/ps, the boolean functions at the o/ps of each gate as shown in figure



$$F_1 = y_7 B = (y_5 + y_2) B$$

$$= (\overline{y_1} + y_2) B$$

$\therefore$ where $\overline{y_1} = \overline{ABC}$, $y_2 = \overline{B}C$. then we get

$$F_1 = (\overline{ABC} + \overline{B}C) B$$

$$= \left[(\overline{A} + \overline{B} + \overline{C}) + \overline{B}C \right] B$$

$$= \overline{A}B + B\overline{B} + \overline{C}B + B\overline{B}C$$

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

$$F_1 = (\bar{A}+\bar{C})B$$

$F_2 = y_6 y_8 \Rightarrow (y_3 + y_4) y_8$

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

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

$$F_2 = (A+C)\bar{B}$$

∴ The truth table for given logic circuit

A	B	C	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	F_1	F_2
0	0	0	0	0	0	0	1	0	1	1	0	0
0	0	1	0	1	0	1	1	1	1	0	0	1
0	1	0	0	0	1	1	1	1	1	0	1	0
0	1	1	0	0	1	1	1	1	1	0	1	0
1	0	0	0	0	1	0	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1	1	1	0	1
1	1	0	0	0	1	1	1	1	1	0	1	0
1	1	1	1	0	1	1	0	1	0	0	0	0

Design using conventional logic gates :-

Example 2 :-

→ The design procedure of the Combinational circuit involves following steps

1. The problem definition.

2. The determination of number of available i/p variables and required o/p variables.
3. Assigning letter symbols to i/p and o/p variables.
4. The derivation of truth table indicating the relation ship bln i/p and o/p variables.
5. obtain simplified boolean expression for each o/p.
6. obtain the logic diagram.

Example 2:- Design a combination logic circuit with three i/p variables that will produce a logic 1 o/p when more than one i/p variables are logic 1.

Sol:- Given Problem specifies that three i/p Variables and one o/p Variable.

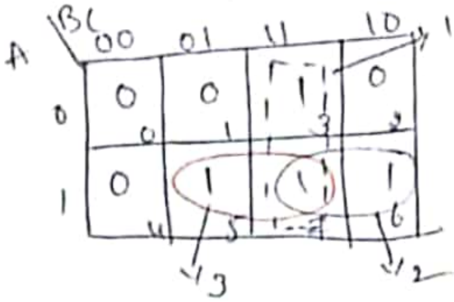
→ we assume A, B, and C are three i/p variables and assign "y" as a o/p variable.

→ The relationship bln i/p and o/p variable can be tabulated in below.

A	B	C	y
0	0	0	0
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	1

Now we obtain the simplified Boolean expression for o/p variable "Y" using K-map simplification.

Group F1 : $\bar{A}BC + ABC$



$F1 = BC(A+\bar{A})$

$\boxed{F_1 = BC}$

Group $F_2 = ABC + AB\bar{C}$

$= AB(C+\bar{C})$

$\boxed{F_2 = AB}$

Group $F_3 = A\bar{B}C + ABC$

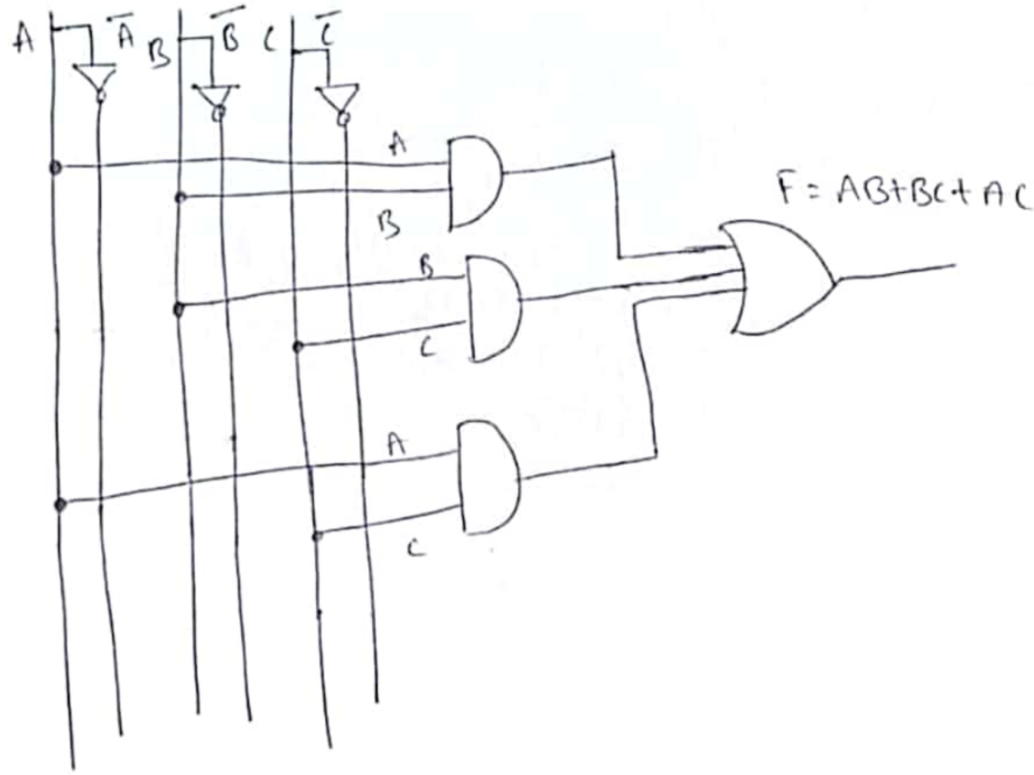
$= AC(B+\bar{B})$

$\boxed{F_3 = AC}$

$\therefore F = F_1 + F_2 + F_3$

$\boxed{F = AB + BC + AC}$

Logic diagram :-



Adders :—

→ Digital computers perform various arithmetic operations
→ The most basic operations no, doubt, is the addition of two binary digits.
→ This simple addition consists of four possible elementary operations, namely.

$0+0 = 0$
$0+1 = 1$
$1+0 = 1$
$1+1 = 10_2$

→ The first three operations produce a sum whose length is one digit, but when the last operation performed sum is two digits.
→ The higher significant bit of this result is called Carry, and lower significant bit called Sum.
→ The logic circuit which performs this operation is called a half-adder.
→ The circuit which performs addition of three bits is a full adder

Half adder :—

→ The half adder operation needs two binary i/ps and two binary o/ps.
→ The truth table gives the relationship bln i/p and o/p variably of half-adder operation.

Inputs		outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

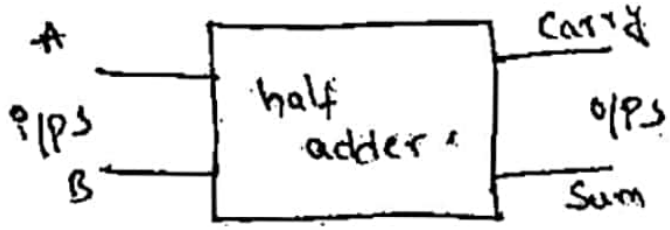
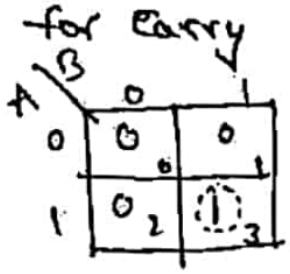
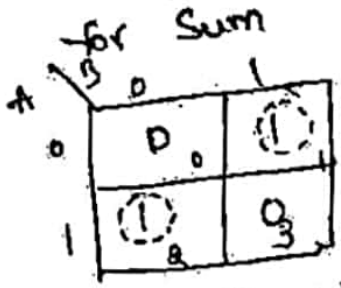


fig :- Block diagram of half adder.

K-map Simplification of Carry and Sum.



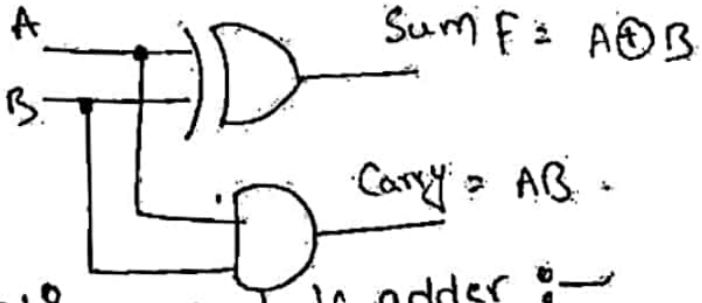
$F = AB$



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

$F = A \oplus B$

Logic diagram of half adder :—



Limitation of half adder :—

→ In multidigit addition we have to add two bits along with the carry of previous digit addition.

→ Effectively such addition requires addition of three bits.

→ This is not possible with half adder.

→ Hence half adder is not used in practise.

FULL-ADDER :-

→ A full-adder is a combinational circuit that performs the arithmetic sum of three i/p bits.

→ It consists of three i/ps and two o/ps.

→ The truth table for full adder is shown below.

inputs			outputs	
A	B	C	carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

fig :- Truth table of full adder.

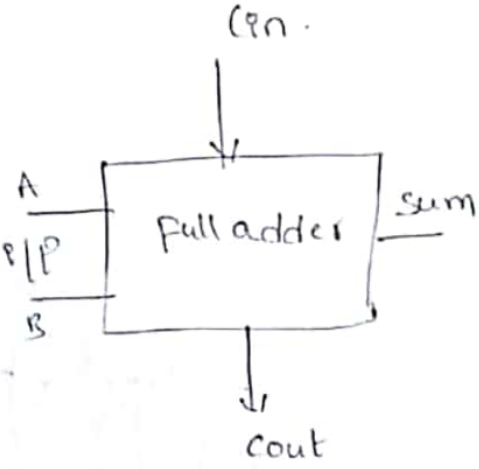
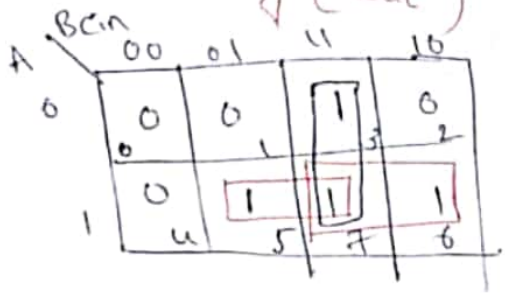


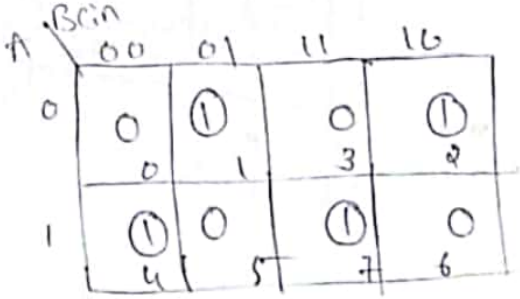
fig :- Block diagram of full adder

K-map simplification for Carry and Sum.

for Carry (Cout)



for Sum



$Cout = AB + ACin + BCin$

$Sum = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC.$

The Boolean function for Sum can be further simplified as follows

$$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$= C[\bar{A}\bar{B} + AB] + \bar{C}[A\bar{B} + \bar{A}B]$$

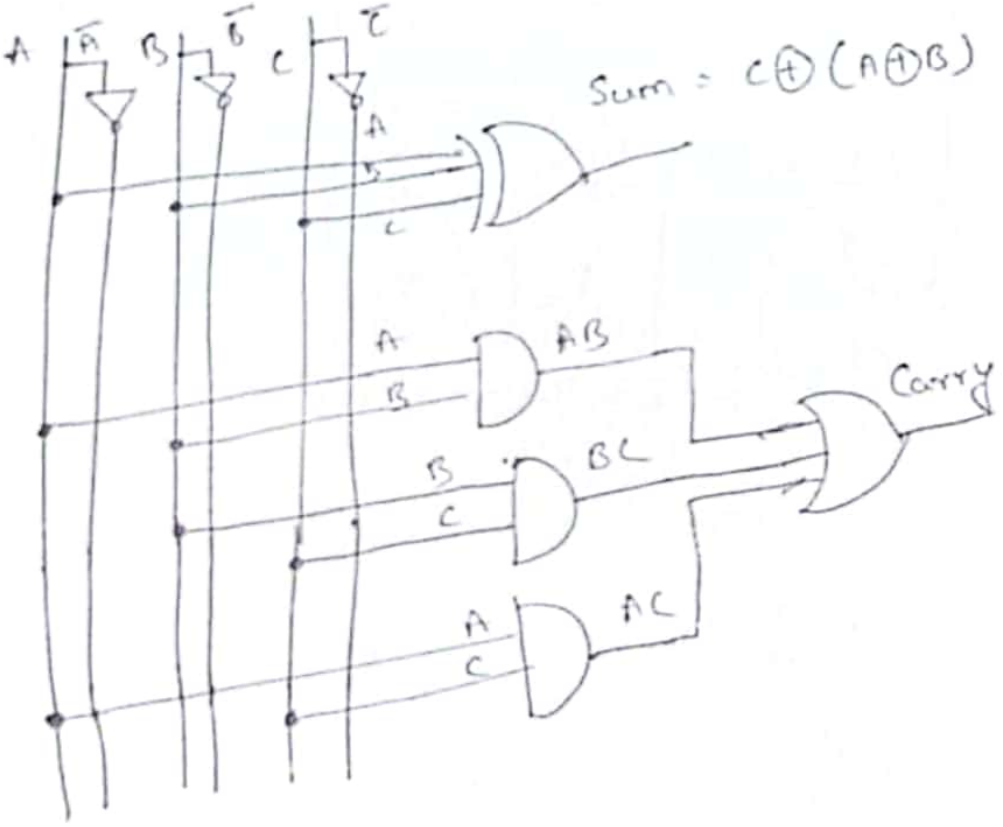
$$= C[A \odot B] + \bar{C}[A \oplus B]$$

$$= C[\overline{A \oplus B}] + \bar{C}[A + B] \quad \because C\bar{A} + AC = C \oplus A$$

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

$$Sum = C \oplus A \Rightarrow C \oplus (A \oplus B)$$

Logic diagram for full adder can be implemented in Boolean function.



Subtractors :-

→ A full adder can also be implemented with two half-adders and one OR gate.

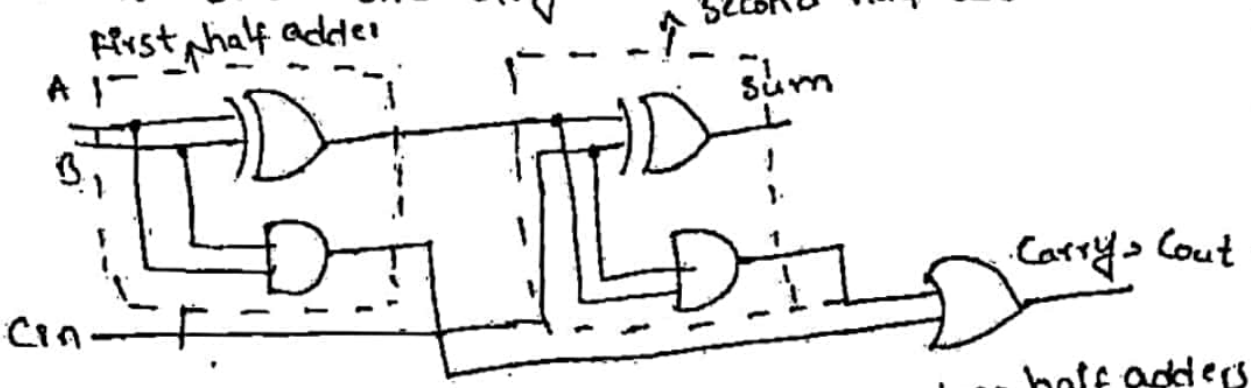


fig :- Implementation of full-adder with two half adders and an OR gate.

SUBTRACTORS :-

→ The subtraction consists of four possible elementary operations, namely.

0 - 0 = 0
0 - 1 = 1 with 1 borrow
1 - 0 = 1
1 - 1 = 0

HALF - SUBTRACTOR :-

→ A half subtractor is a combinational circuit that performs subtraction bln two bits and produces their difference.

→ It also has an o/p to specify if a 1 has been borrowed.

→ The truth table for half subtractor is given below.

i/ps		o/ps	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

k-map Simplification for half Subtractor :-

for difference

A \ B	$\bar{B}$ 0	B 1
$\bar{A}$ 0	0 0	1 1
A 1	1 2	0 3

for borrow

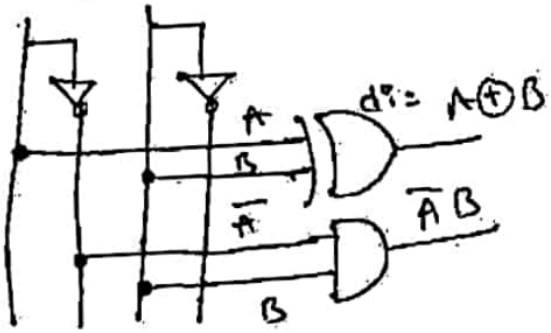
A \ B	$\bar{B}$ 0	B 1
$\bar{A}$ 0	0 0	1 1
A 1	0 2	0 3

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

Borrow = $\bar{A}B$

Di = $A \oplus B$

logic diagram :-



FULL-SUBTRACTOR :-

→ A Full-Subtractor is a Combinational circuit that performs a Subtraction b/n two bits, taking into account borrow of the least significant stage.

→ This ckt has three I/Ps and two O/Ps.

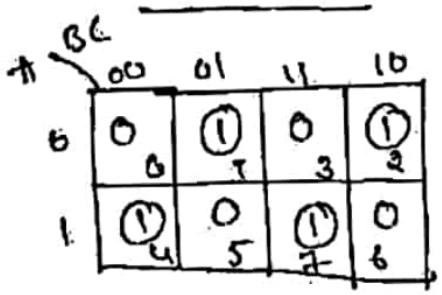
→ The two O/Ps D and B are denoted by difference and borrow.

→ The truth table for full Subtractor as shown in below.

Inputs			outputs	
A	B	C	D	Bout
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

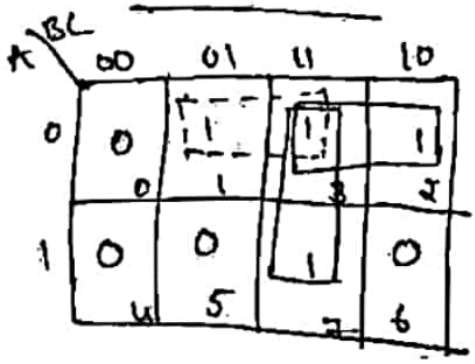
K-map simplification of for D and Bout.

for difference



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

for Borrow



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

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

→ The boolean function for D (difference) can be simplified

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

$= C[\bar{A}\bar{B} + AB] + \bar{C}[\bar{A}B + A\bar{B}]$

$= C[A \odot B] + \bar{C}[A \oplus B]$

$= C[\overline{A \oplus B}] + \bar{C}[A \oplus B]$

$$\boxed{D = C \oplus [A \oplus B]}$$

The borrow o/p for circuit shown below.

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

$= \bar{A}B + (AB + \bar{A}\bar{B})C$ $\quad \langle \because B + C = 1 \rangle$

$= \bar{A}B + ABC + \bar{A}\bar{B}C.$

$= \bar{A}B(1 + C) + ABC + \bar{A}\bar{B}C.$

$= \bar{A}B + \bar{A}BC + ABC + \bar{A}\bar{B}C.$ $\quad \langle \because A + \bar{A} = 1 \rangle$

$= \bar{A}B + BC[A + \bar{A}] + \bar{A}\bar{B}C$

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

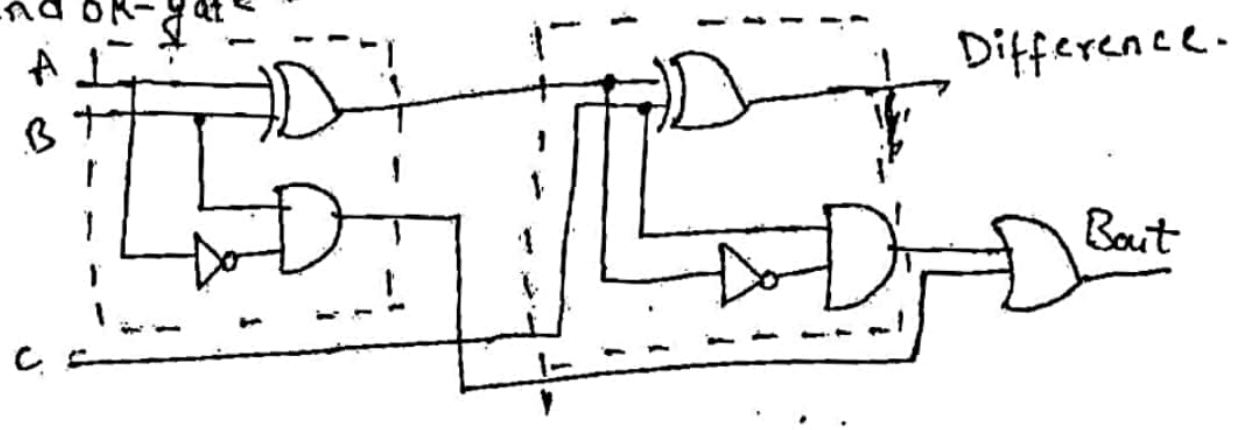
$= \bar{A}B[1 + C] + BC + \bar{A}\bar{B}C$

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

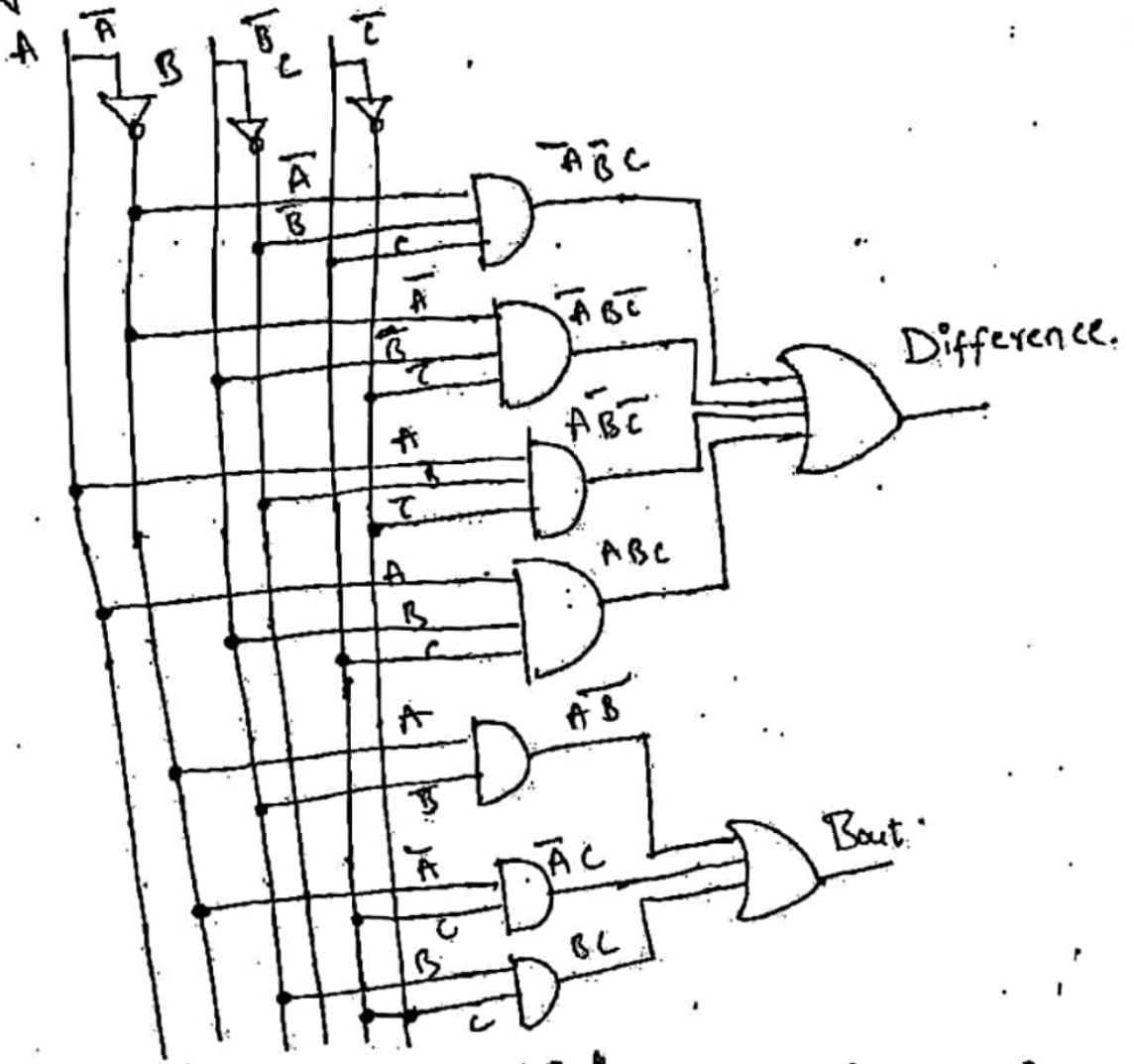
$= \bar{A}B + \bar{A}C[B + \bar{B}] + BC$

$$\boxed{B_{out} = \bar{A}B + \bar{A}C + BC} \quad //$$

Implementation full adders subtractors using half-subtractor and OR-gate



Logic diagram for full subtractor:-



BINARY ADDER/ PARALLEL ADDER :-

→ A single full-adder is capable of adding two one-bit numbers and an i/p carry.

→ In order to add binary numbers more than one-bit, additional full-adders must be employed.

→ A n-bit parallel adder can be constructed using no. of full-adder circuits connected in parallel.

→ The below block diagram shows of n-bit parallel adder using no-of full adder circuits connected in cascade.

→ is Carry o/p of each adder is connected to the Carry i/p of the next higher-order adder

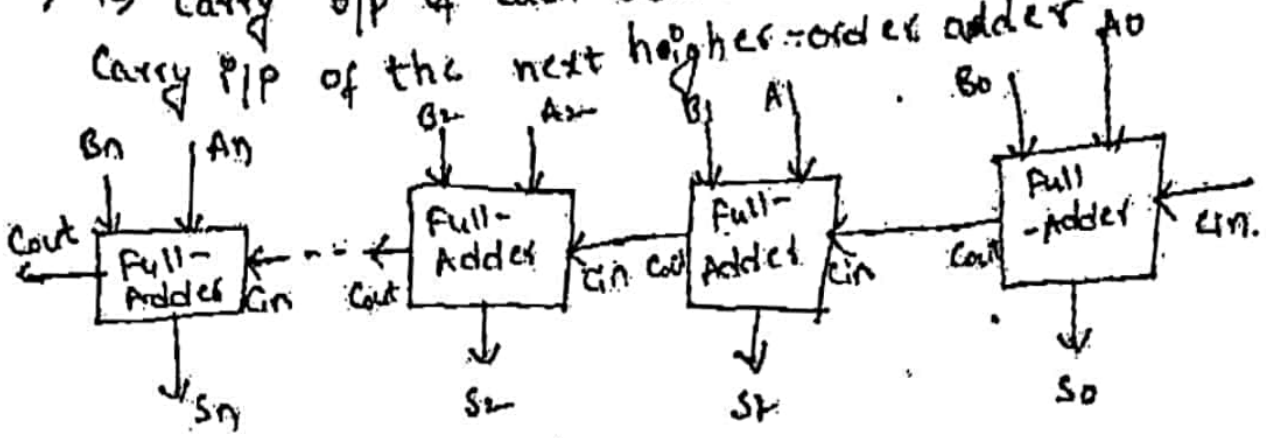


fig :- Block diagram of n-bit parallel adder

Example :- Design a 4-bit parallel adder using full Adders.

sol :- The below diggrams shows the block diagram and logic symbol for 4-bit parallel adder.

→ Here least significant position, Carry i/p of full-adder is made "0"

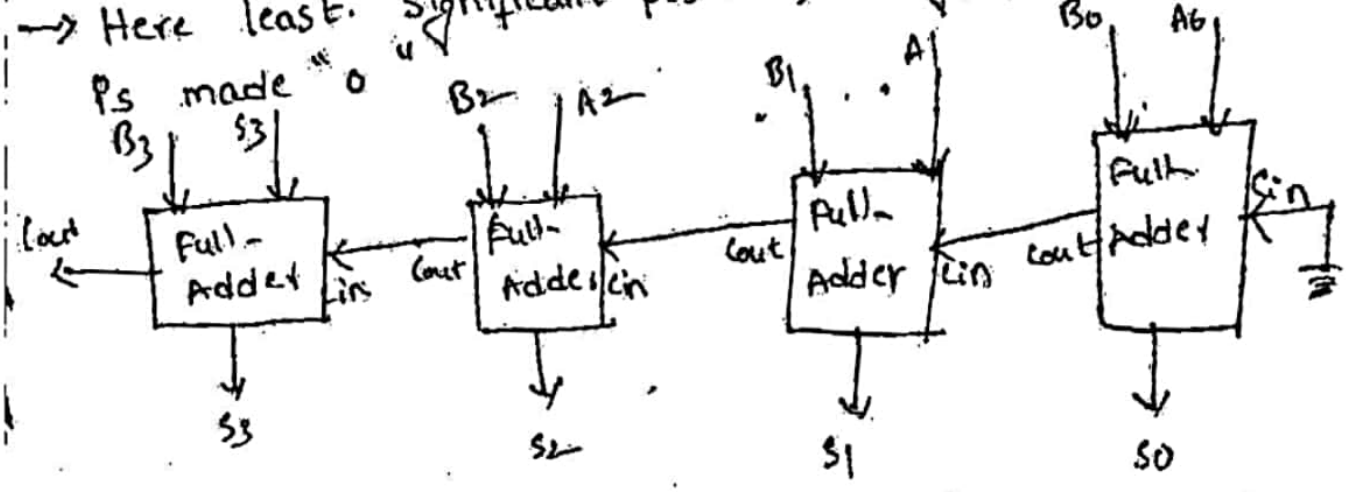


fig :- Block diagram for 4-bit parallel adder.

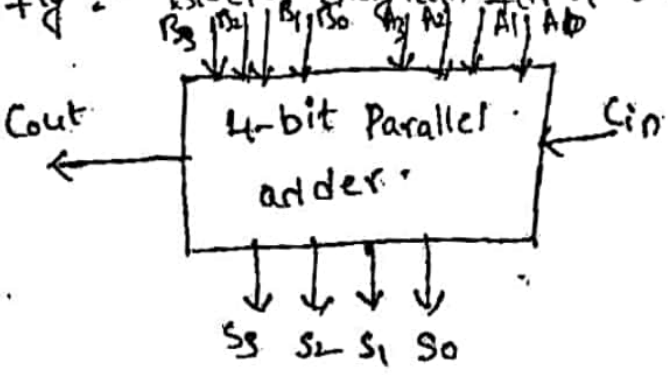
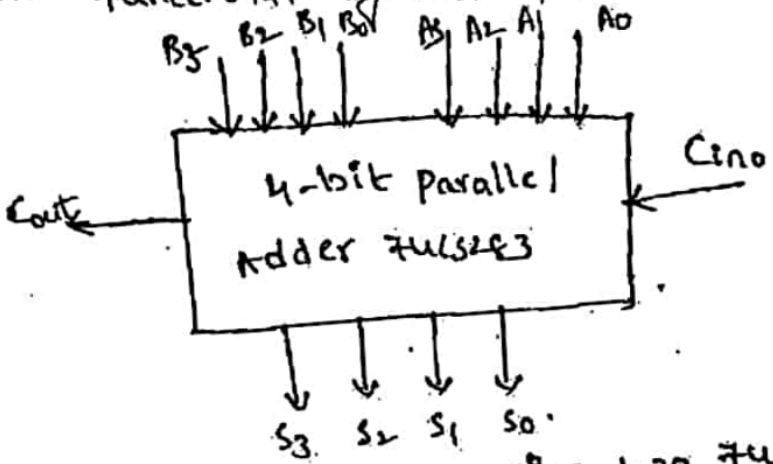


fig :- logic symbol for 4-bit Parallel adder.

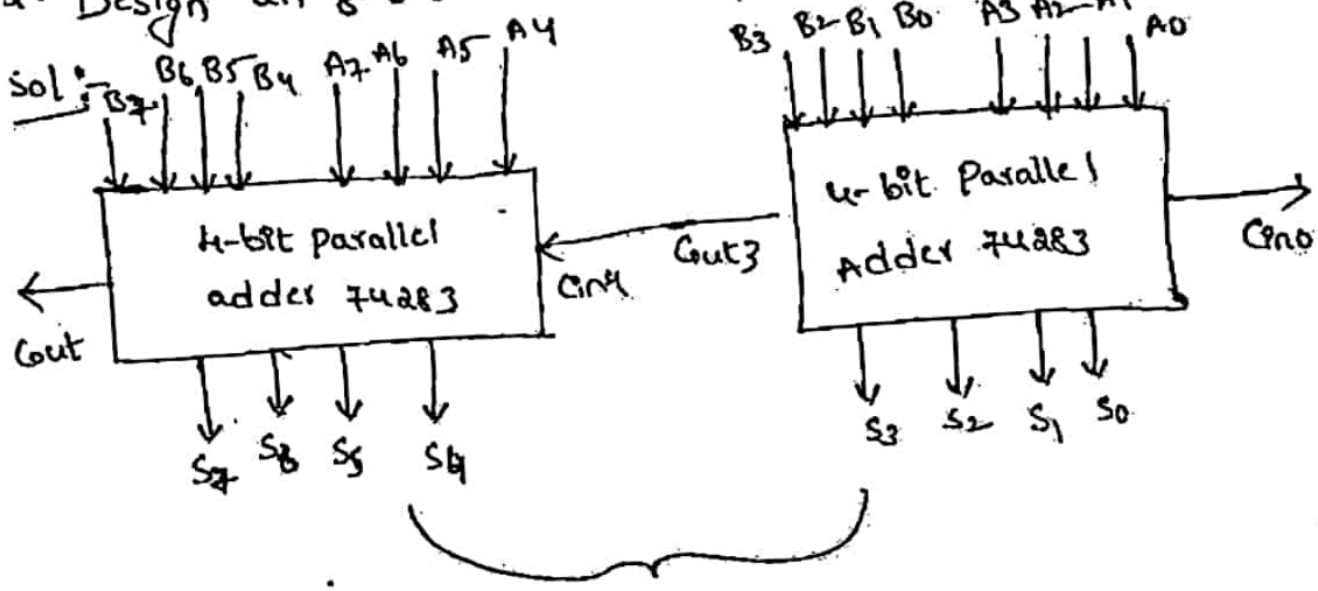
→ IC no. of the Binary parallel adder is 74LS 83/ 74LS 283

→ The functional symbol for the 74LS 283



2. Design an 8-bit adder using two 74283.

Sol:



BINARY SUBTRACTOR / PARALLEL SUBTRACTOR :-

→ The Subtraction of binary numbers can be done most Conveniently by means of Complements.

→ Remember that the Subtraction A-B can be done by taking 2's Complement of B and adding to it A.

→ The 2's Complement can be obtained by taking the 1's Complement and adding one to the least

Significant pair of bits.

→ The 1's Complement can be implemented with Inverters and a 1 can be added to the Sum through the i/p Carry.

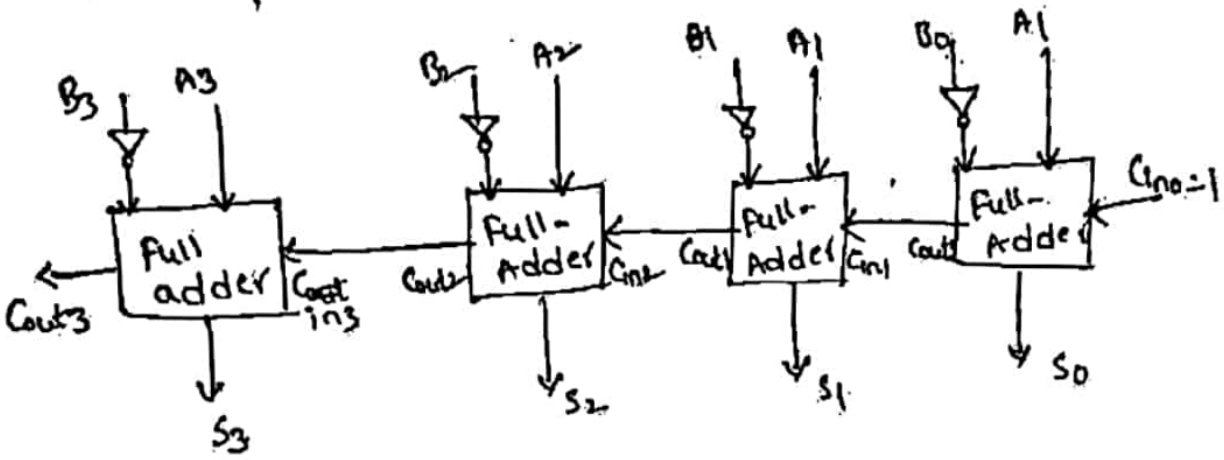


fig :- 4-bit parallel Subtractor.

PARALLEL ADDER- SUBTRACTOR :-

→ The addition and Subtraction operations can be Combined into one circuit with one common binary adder.

→ This is done by including exclusive -OR gate with each of full Adder as shown below figure.

→ The mode i/p M Controls the operation of the circuit.

→ When M=0, the circuit is an adder; and when M=1, the circuit becomes a Subtractor.

→ Each Exclusive -OR gate receive i/p m and one of the i/ps of B.

→ When M=0, we have B⊕0=B. The full adder receive the value of B, the i/p Carry is "0".

→ The B i/p s are all Complemented and a 1 is added through the i/p carry.

→ The Circuit performs the operation A plus the 2's complement of B. ie "A-B"

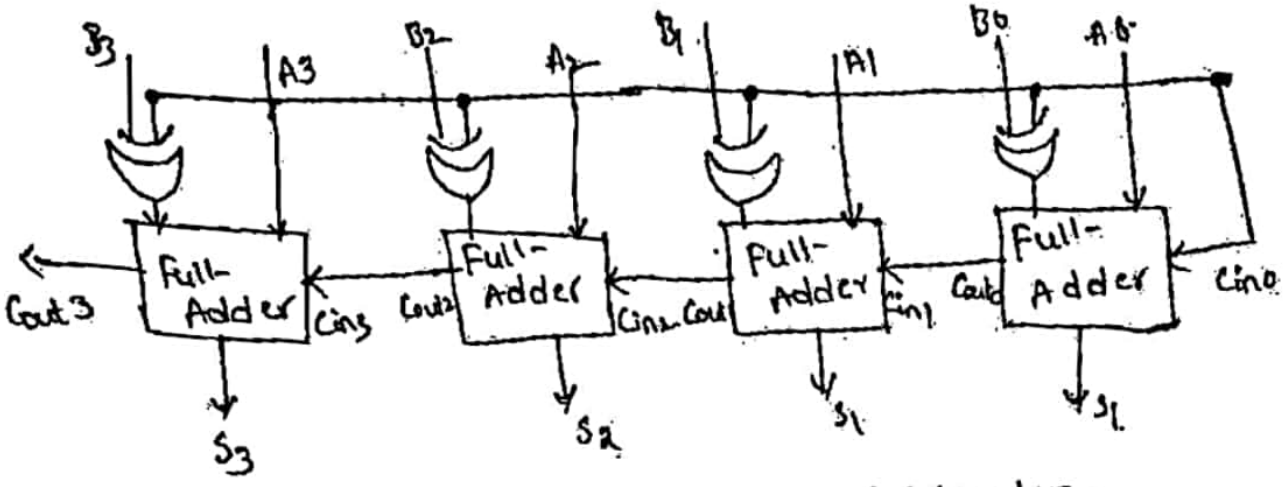


fig :- 4-bit adder-Subtractor.

Comparision b/n Serial and Parallel Adder :-

Serial Adder	Parallel Adder
1. Serial adder uses shift Register.	Parallel adder uses registers with Parallel load capacity.
2. The Serial adder requires only one full-adder ckt	The no-of full-adder ckts in the Parallel adder equal to the no-of bits in the binary numbers.
3. The Serial adder is a Sequential ckt	Excluding the registers, the Parallel adder is a purely Combinational Ckt.
4. Time required for addition depends on no. of bits.	Time required for addition does not depends on no-of bits
5. It is slower.	It is faster.

DECIMAL ADDER :-

→ The digital systems handle the decimal number in the form of binary coded decimal numbers (BCD).

→ A BCD adder is a circuit that adds two BCD and produces a Sum digit also in BCD.

→ BCD use 10 digits, 0 to 9 which are represented in the binary form 0000 to 1001.

→ when we write BCD number say 526, it can be represented as

5 2 6
↓ ↓ ↓
0101 0010 0110

→ The addition of two BCD numbers can be best understood by considering the three cases that occur when two BCD digits are added.

Sum equals 9 (or) less with carry 0

Let us Consider additions of 3 and 6 in BCD

6	0110	→ BCD for 6
+3	0011	→ BCD for 3
+9	1001	← BCD for 9.

Sum equal 9 (or) less with carry 1

Let us Consider 8

	1000	→ BCD for 8
+9	1001	→ BCD for 9
17	0001 0001	→ Incorrect BCD

```
 8        1000
+9        1001
--        ----
17      ------
        0001 0001
        0000 0110
        ---------
        0001 0111   ← BCD for 17.
        ---- ----
          1    7
```

Sum greater than 9 with carry 0 :-

Let us consider addition of 6 and 8

```
 +6        0110  → BCD for 6
  8        1000  → BCD for 8.
--        ----
14      ------
        1110  ← Invalid BCD number
        ----
```

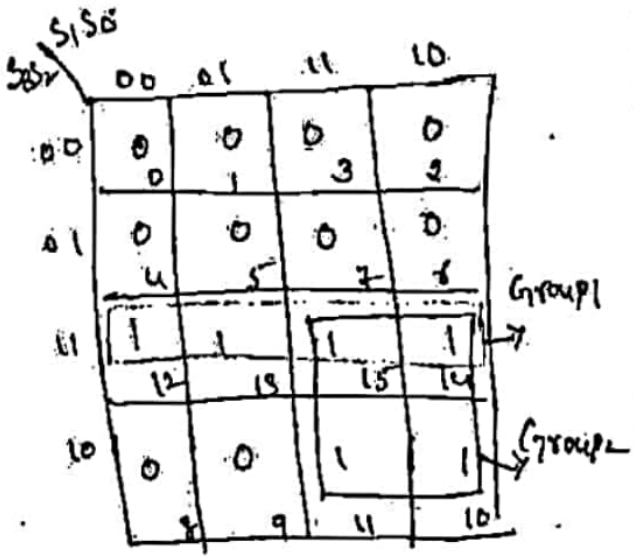
Add "6" ~~for~~ (0110) for an valid BCD number then we can get the valid BCD number.

```
        0110
        1000
        ----
        1110
        0110
        ----
      1 0100
      ------
0001 0100   ← valid BCD number.
---- ----
  1    4.
```

→ The logic circuit to detect sum greater than 9 can be determined by simplifying the boolean expression of the given table.

Inputs				o/p's
S_3	S_2	S_1	S_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

10 → 11 → 12 → 13 → 14 → 15



⊛ $Y = S_3S_2 + S_3S_1$.

The o/p Y=1 Indicates sum is greater than 9.

→ with this information we can draw the block diagram of BCD adder as shown in below.

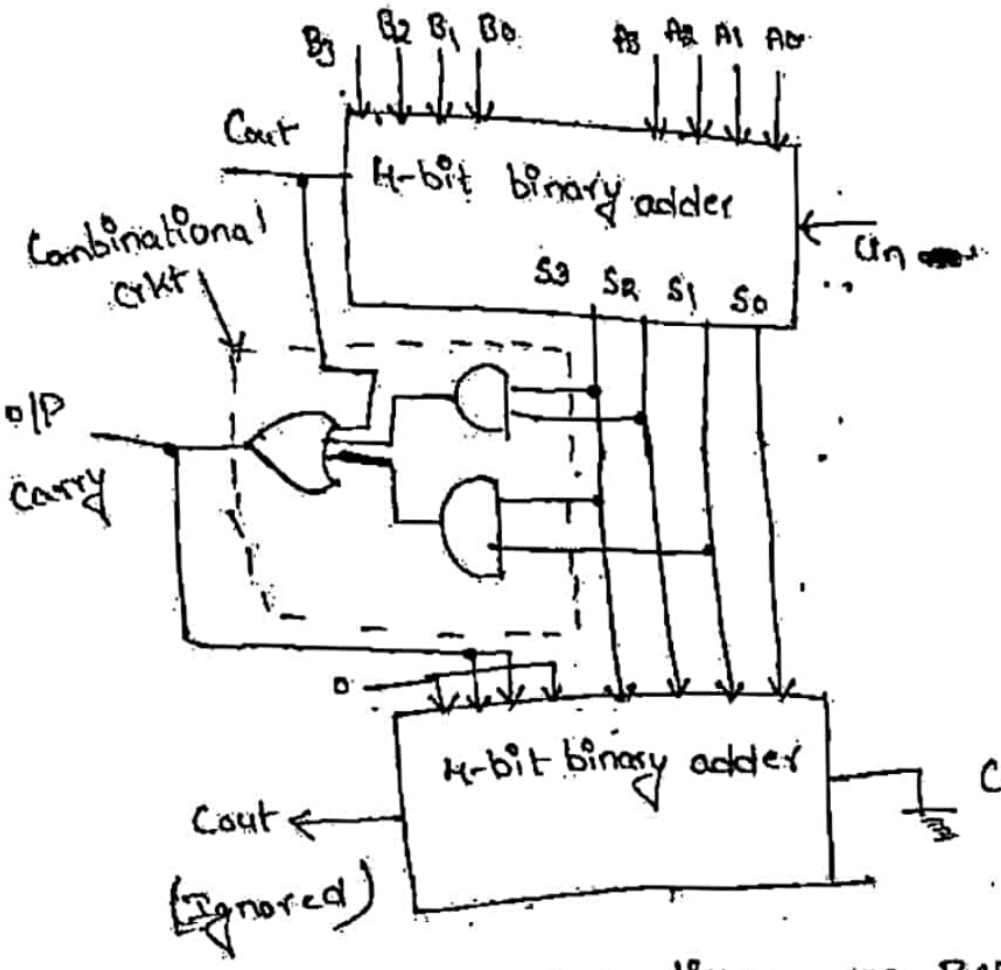


fig :- Block diagram of BCD Adder.

Example 1:-

1. Design 8-bit BCD adder using Ic 7483

Sol:- To implement 8-Bit BCD adder we have to cascade two 4-bit BCD adders.

→ In cascade connection carry o/p of the lower position (digit) is connected as a carry i/p of the higher position (digit).

→ The below figure shows the block diagram of 8-bit "bcd" adders.

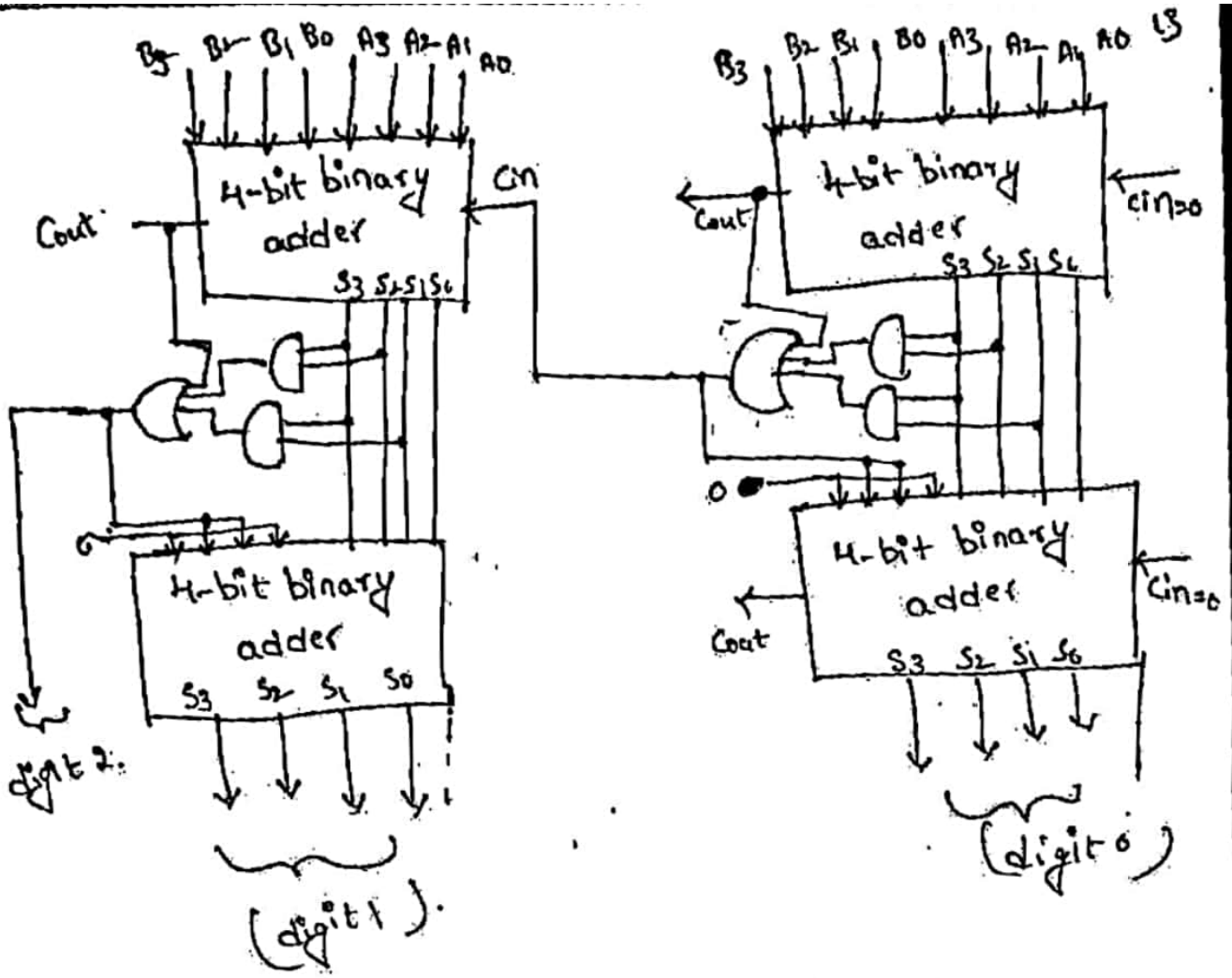


fig :- 8-bit BCD adder using

Binary MULTIPLIER :-

→ The Combination circuit implemented to perform multiplication is called Combinational multiplier.

→ The advance in VLSI technology have made it possible to build Combinational circuits that perform nxn bit multiplication for fairly large values of n.

→ Let us generalised the multiplication process for 4x4 multiplier for two assigned integers.

multiplicand A = $A_3 A_2 A_1 A_0$, and multiplier B = $B_3 B_2 B_1 B_0$.

→ The below shows the, each shifted multiplicand which is multiplied by either 0 (or) 1 depending on

Corresponding multiplier bit is called partial product.

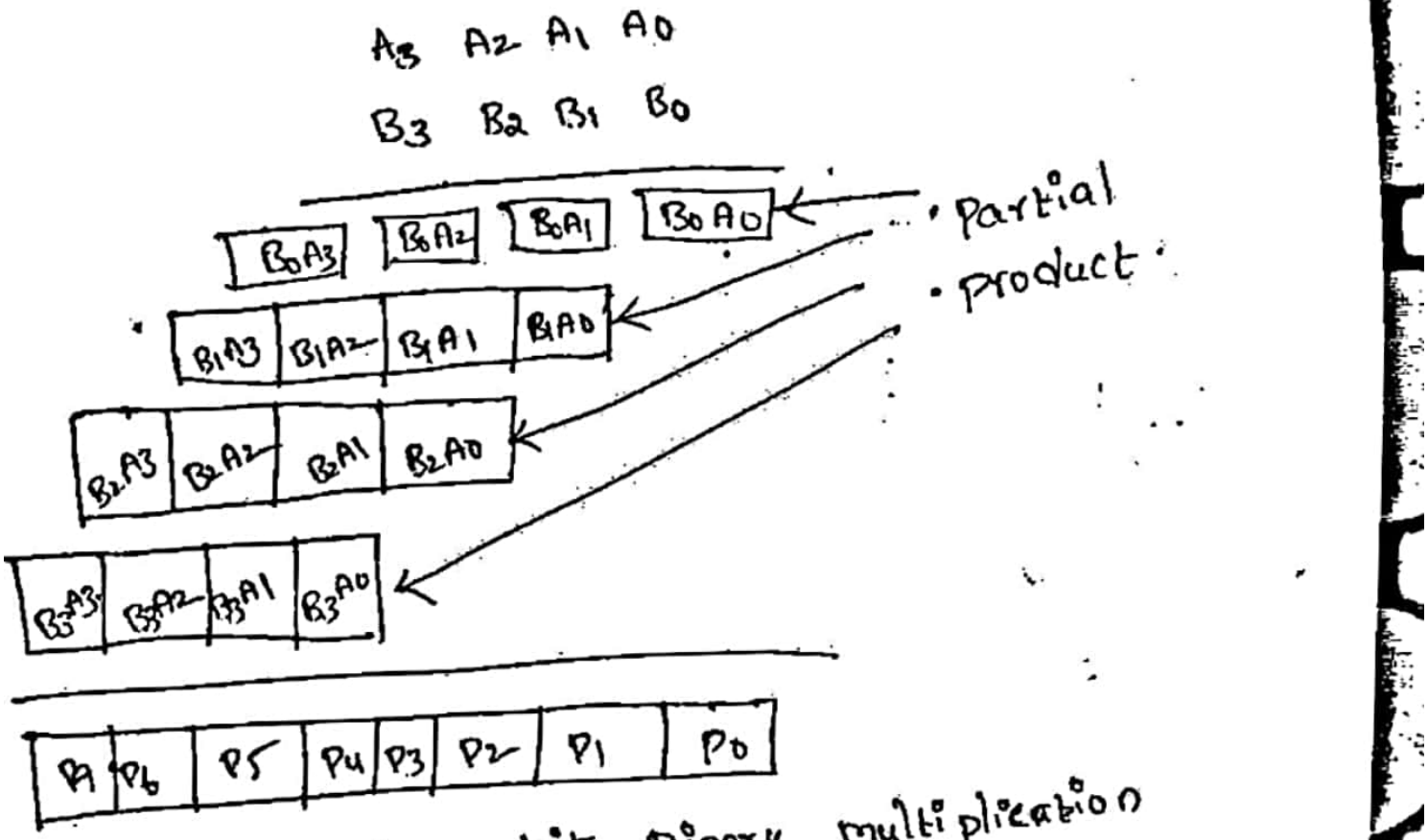


fig :- 4 by 4-bit Binary multiplication

→ The first Partial Product is formed by multiplying "B_0" by A3 A2 A1 A0

→ The 2nd Partial Product is formed by multiplying "B_1" by A3 A2 A1 A0

→ The 3rd Partial Product is formed by multiplying "B_2" by A3 A2 A1 A0

→ The 4th partial Product is formed by multiplying "B_3" By A3 A2 A1 A0.

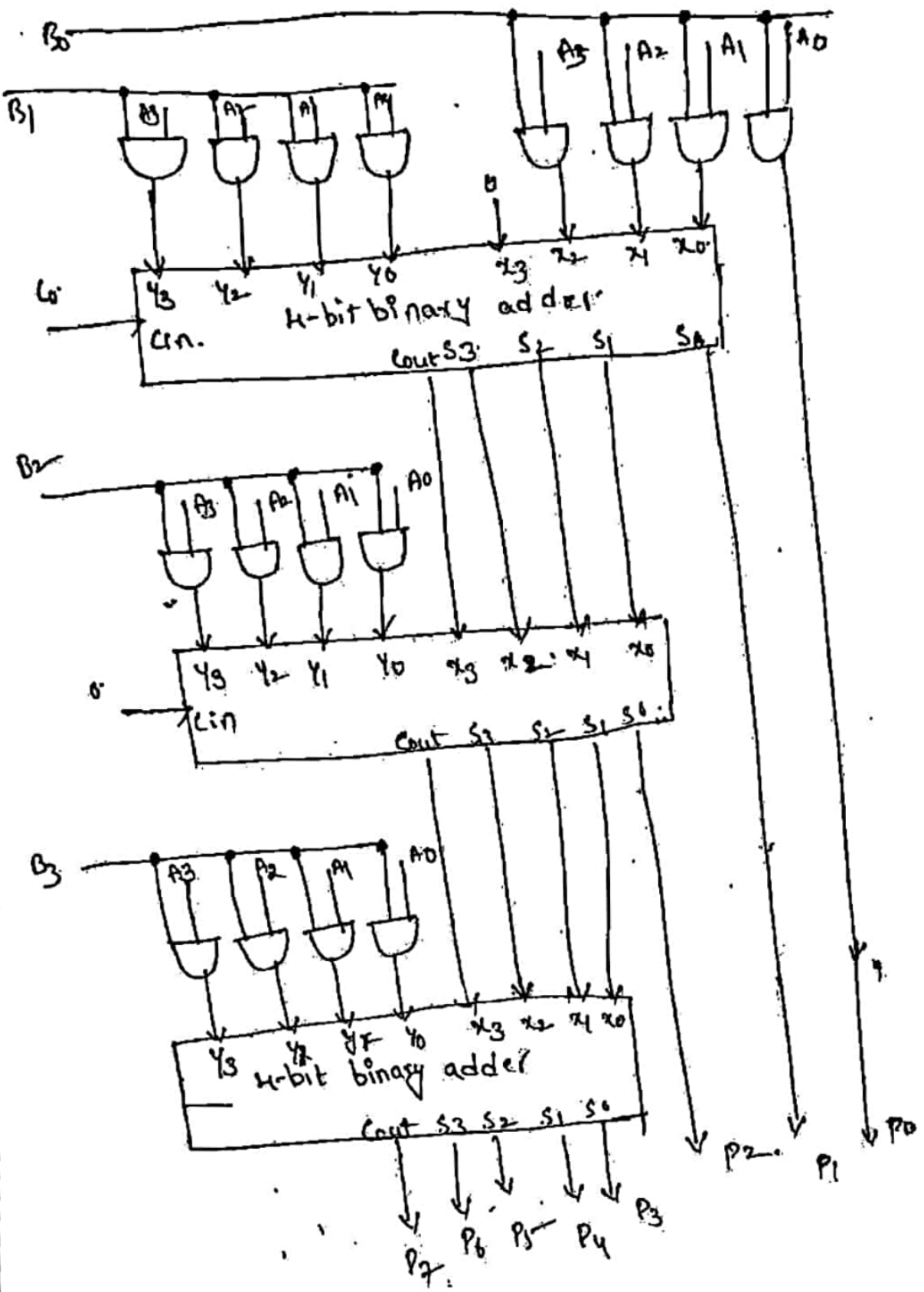
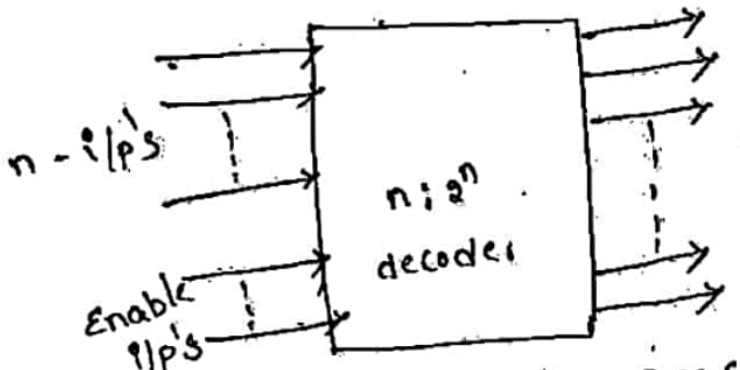


fig:- 4x4 bit binary multiplier.

DECODERS :-

→ A decoder is a multiple-i/p, multiple o/p logic circuit which converts coded inputs into coded outputs
→ where the i/p and o/p codes are different.
→ The i/p code generally has fewer bits than the output code.
→ Each i/p code word produces a different o/p code word
→ The general structure of the decoder circuit is shown in below.



→ The encoded information is presented as n-i/ps producing 2^n possible outputs.
→ The "2^n" output values from 0 to $2^n - 1$.

BINARY DECODER :-

→ A decoder which has an n-bit binary i/p code and a one activated output out-of-2^n o/p code is called binary decoder.
→ The below figure shows 2 to 4 decoder.
→ Here 2 input are decoded into four outputs, each output representing one of the minterms of the 2-i/p variables.

→ The truth table for 2-to-4 decoder is shown below. 15

→

inputs			outputs			
$\overline{En}$	A	B	Y_3	Y_2	Y_1	Y_0
1	x	x	0	0	0	0
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0

$Y_0 = \overline{A}\,\overline{B}$

$Y_1 = \overline{A}\,B$

$Y_2 = A\overline{B}$

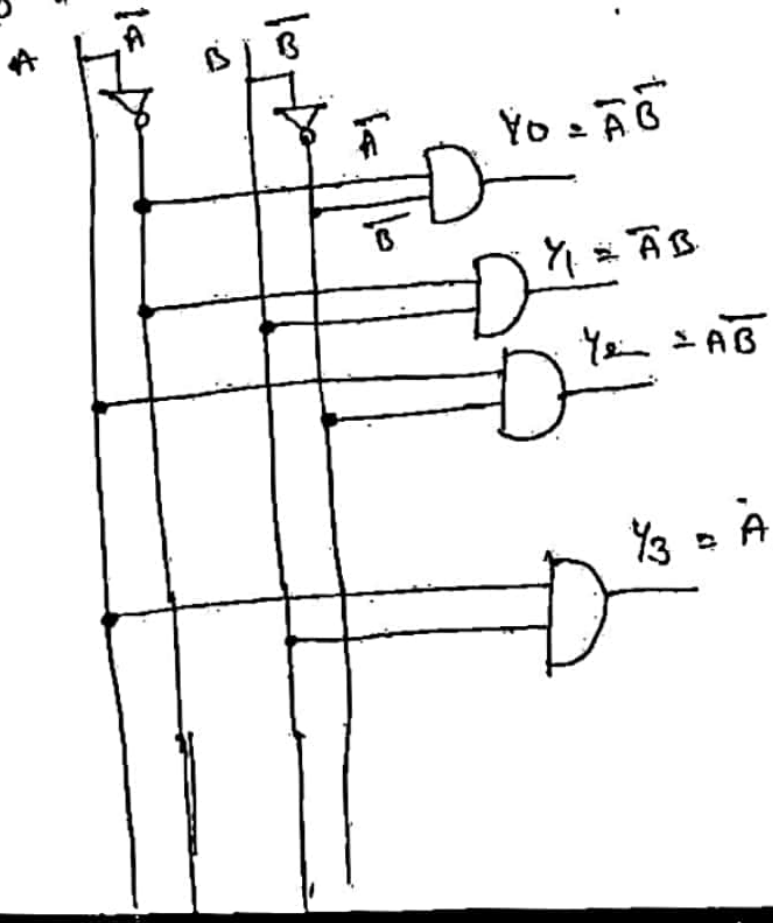
$Y_3 = AB$.

→ In the truth table, if enable i/p is 1 ($En=1$) one and only one of the o/p's Y_0 to Y_3 is active for given.

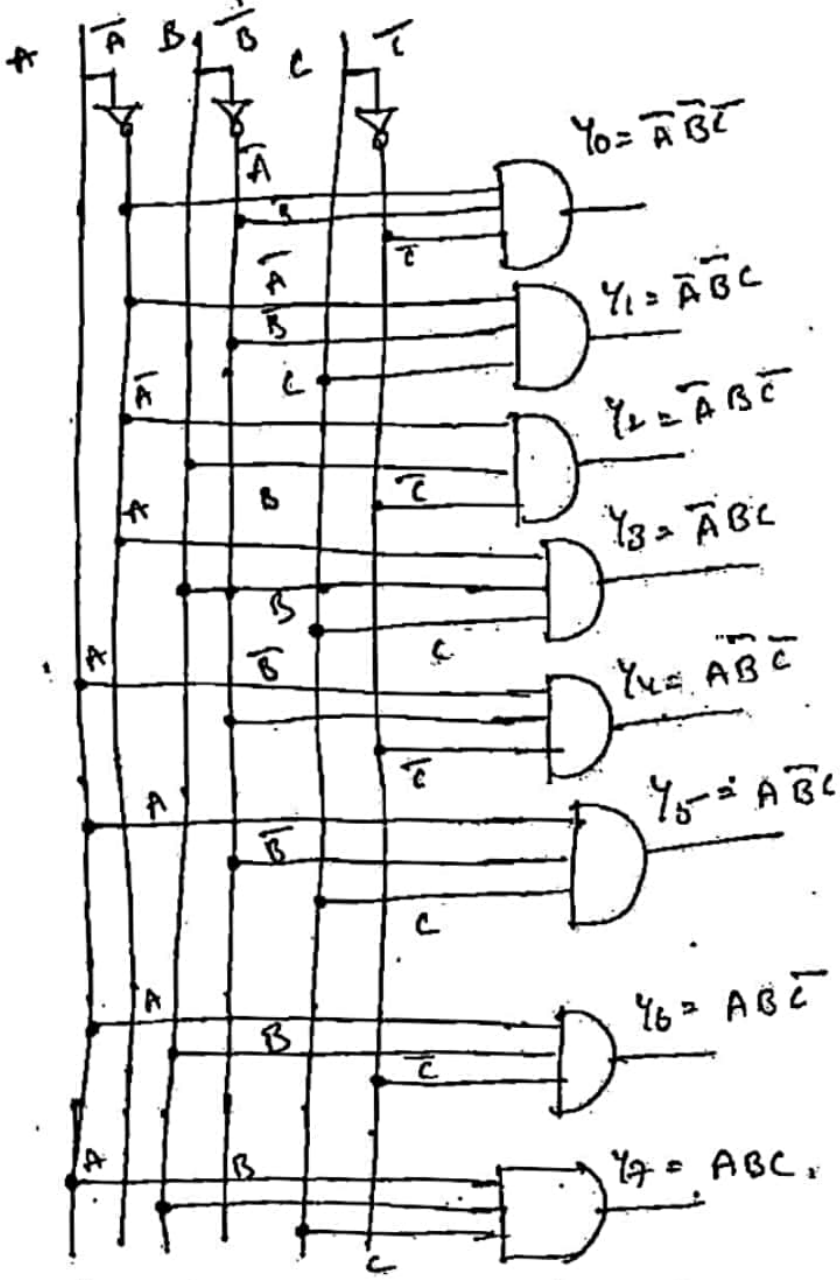
→ The o/p Y_0 is active, i.e $Y_0=1$, when $A=B=0$.

→ The " Y_1 " " " $Y_1=1$ " $A=0, B=1$.

→ If enable input is 0, i.e $EN=0$, then all the outputs are "0".



→ The logic diagram of 3-to 8 decoder.



74x138 3 to 8 decoder :—

→ The 74x138 IS commercially available 3-to 8 decoder.

→ It accepts three binary i/ps (A, B and C) and enabled, provides eight individual active low o/ps. ($Y_0 - Y_7$).

→ The device has three i/ps, two active low, and enable ($\overline{G_{2A}}, \overline{G_{2B}}$)

one active high (G1). The truth table and logic symbol shown in below.

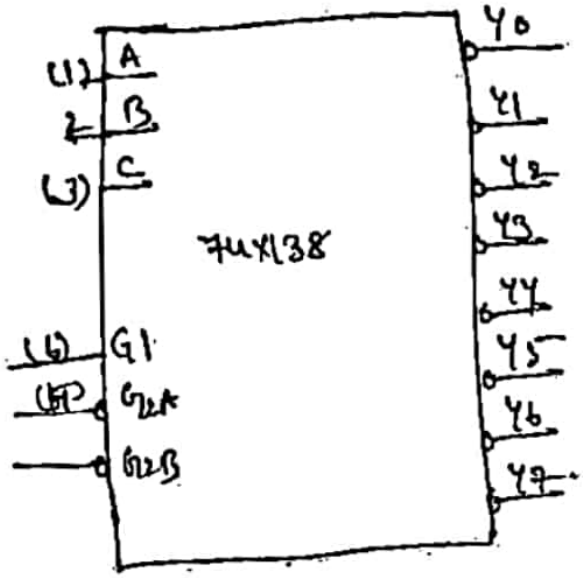


fig:- logic symbol of 74x138

inputs						outputs							
G_{2B}	G_{2A}	G_1	C	B	A	$\overline{Y_7}$	$\overline{Y_6}$	$\overline{Y_5}$	$\overline{Y_4}$	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

74x*139 Dual 2-to-4 decoder :-

→ The 74x*139 Consists of two independent and identical 2-to-4 decoders.

→ The enable i/p and o/ps of IC 74x*139 are active low.

→ The figure shows the logic symbol and truth table for 74x*139.

→ The truth table for one half of a 74x139 dual 2 to 4 decoder.

→ The truth table for other half is same as first half.

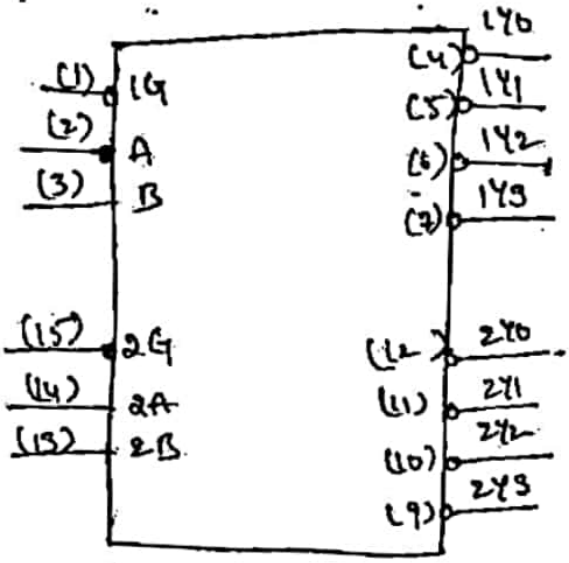


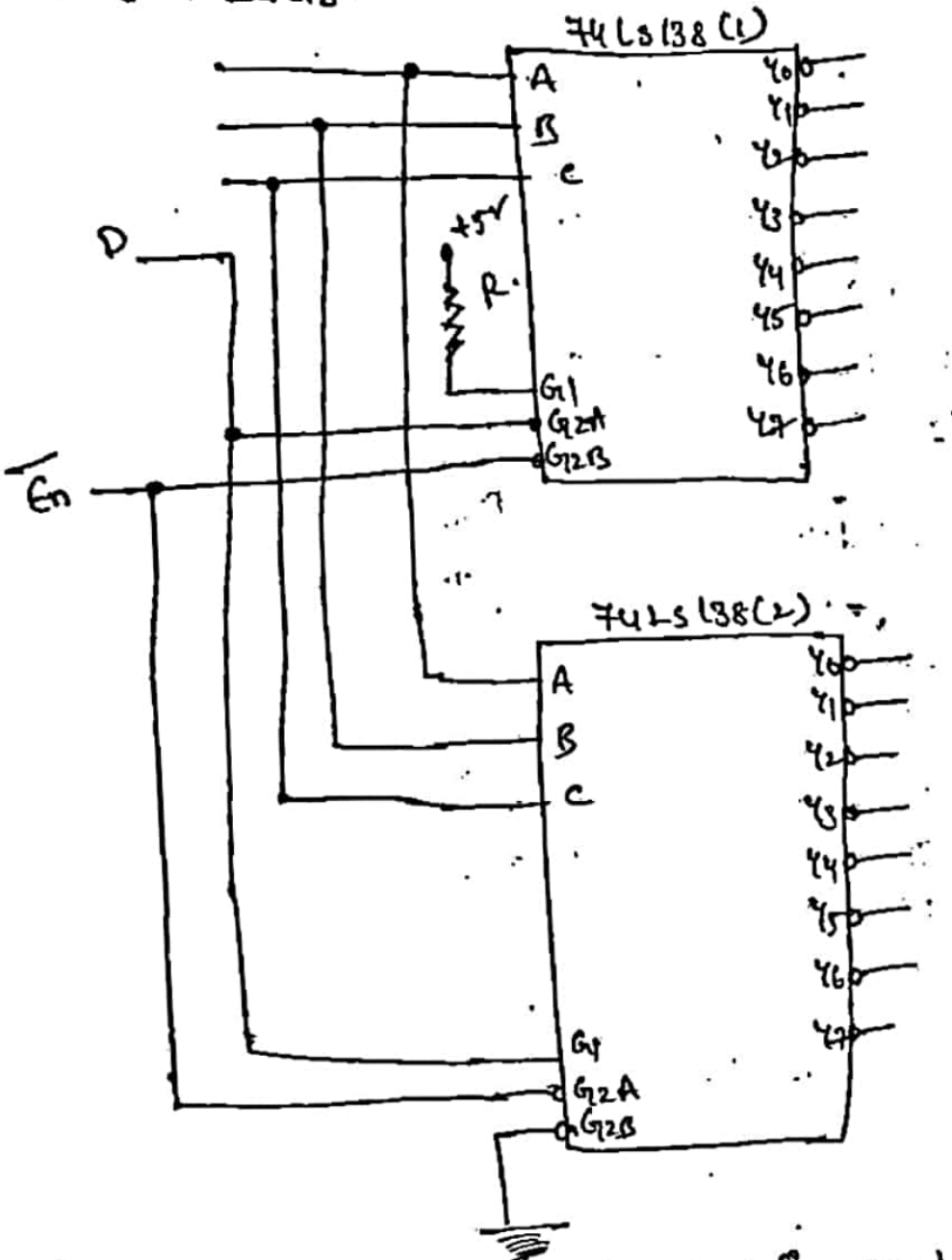
fig :- Logic symbol for 74x139.

i/ps			o/ps			
$\overline{G}$	A	B	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Cascading Binary DECODERS :-

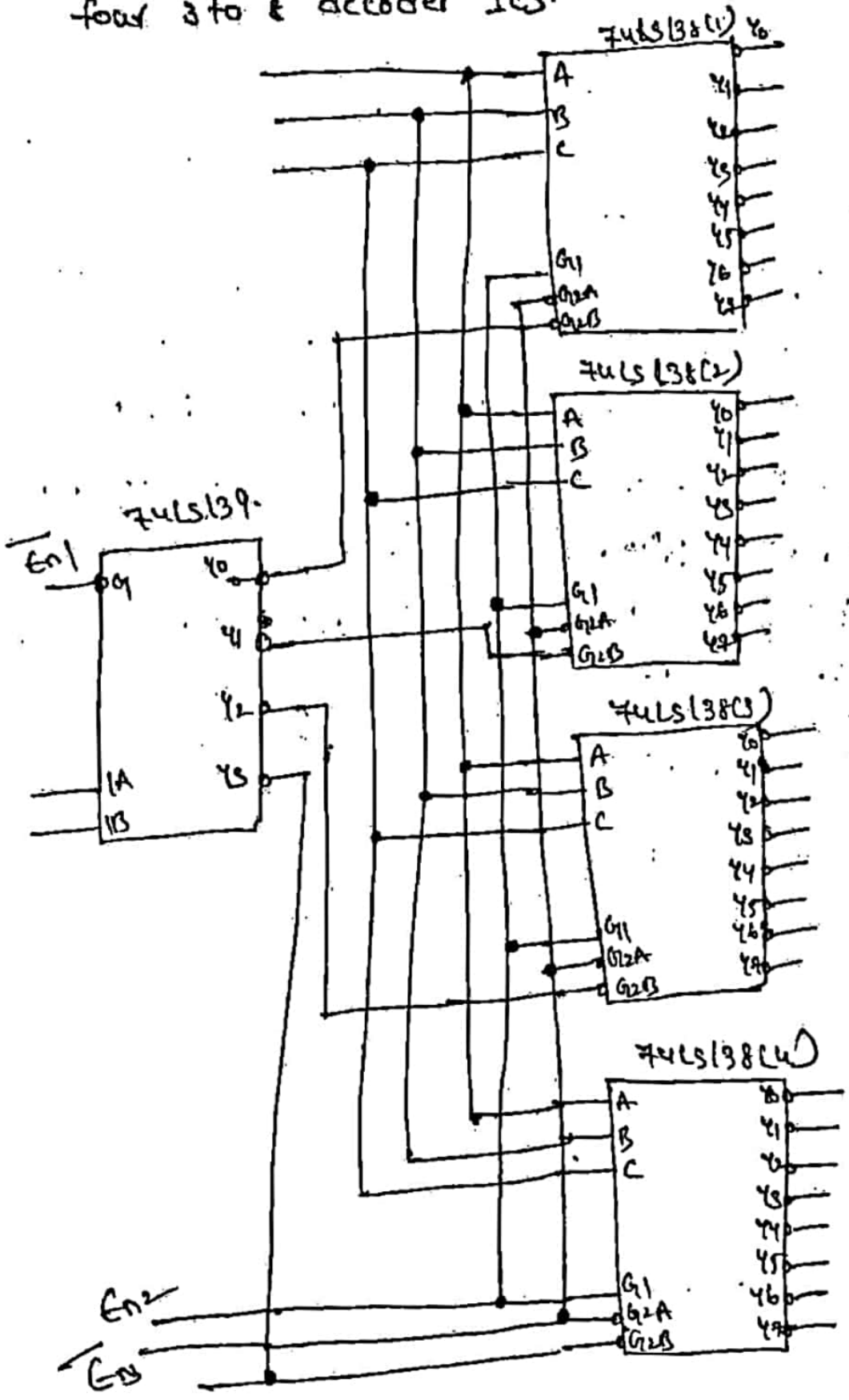
→ Binary decoder circuits can be connected together to form larger decoder circuit as shown in below.

→ The below figure show the 4x16 decoder using two 3x8 decoders.



→ when D=0 the top decoder is enabled and other is disabled. Thus the bottom decoder o/p's are all 1's and the top eight o/ps generate minterms 0000 to 0111. when D=1 The enable is reversed and generate o/ps from 1000 to 1111

1. Design 5-to-32 decoder using one 2to4 decoder and four 3 to 8 decoder ICs.



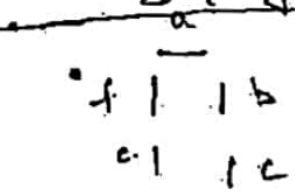
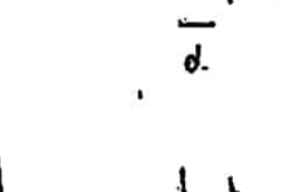
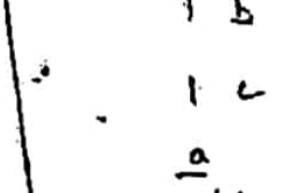
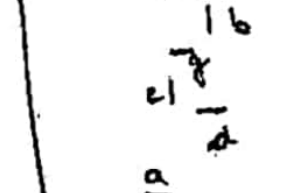
BCD to 7 Segment display decoder:—

→ In most pratical applications, seven segment displays are used to give a visual indicatPon of the o/p states of digital Ics such as decade latches, decade Counter etc.

→ The special BCD to seven segment decoder/driver Ics are used to Convert the BCD signal into a form suitable for driving these displays.

→ In this section, we are going to study LED and LCD decoders / drivers for seven segment displays.

→ The table for the segment activated during each digit display.

Digits	segment activated.	Display.
0	a, b, c, d, e, f	
1	b, c	
2	a, b, d, e, g	
3	a, b, c, d, g.	

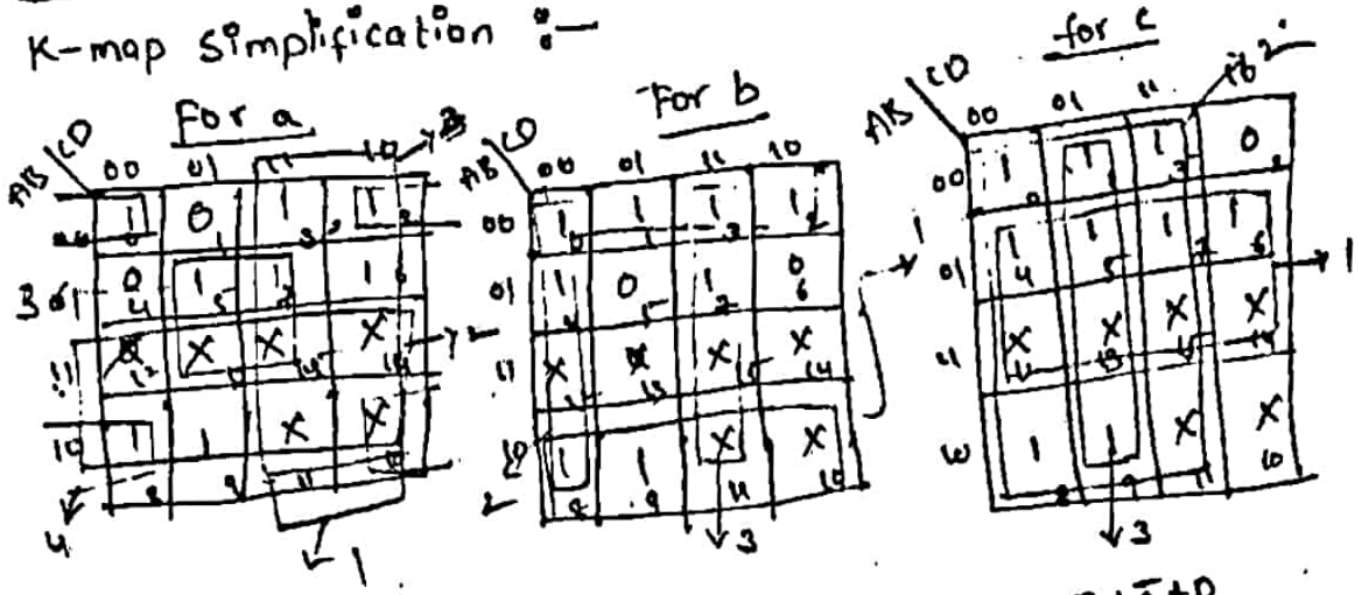
4	b, c, f, g	f | b g | c
5	a, c, d, f, g	f | a g | c d
6	a, c, d, e, f, g	f | a e | g | c d
7	a, b, c	a | b. | c
8	a, b, c, d, e, f, g	f | a | b e | g | c d
9	a, b, c, d, f, g.	f | a | b g | e d.

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

The above truth table BCD - common - cathode 7-segment decoder / driver.

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

K-map Simplification :-

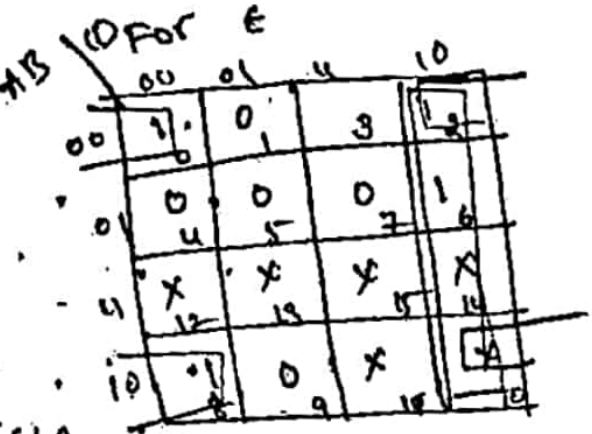


$a = A + C + BD + \bar{B}\bar{D}$ $\quad b = \bar{B} + \bar{C}\bar{D} + CD$ $\quad C = B + \bar{C} + D$

For d

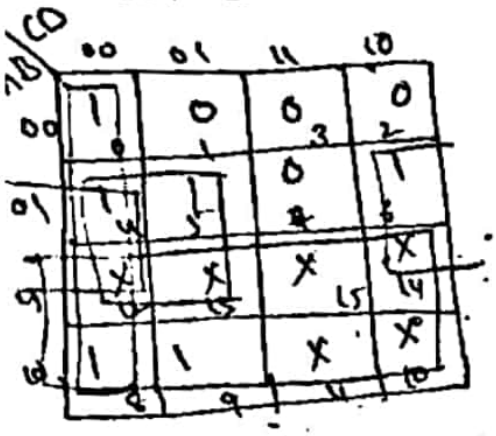
$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$

For E



$E = \bar{B}\bar{D} + C\bar{D}$

For F



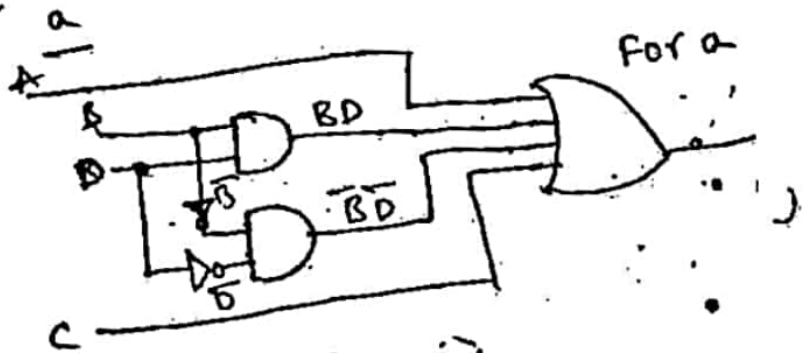
$f = A + C\bar{D} + B\bar{C} + B\bar{D}$

For g

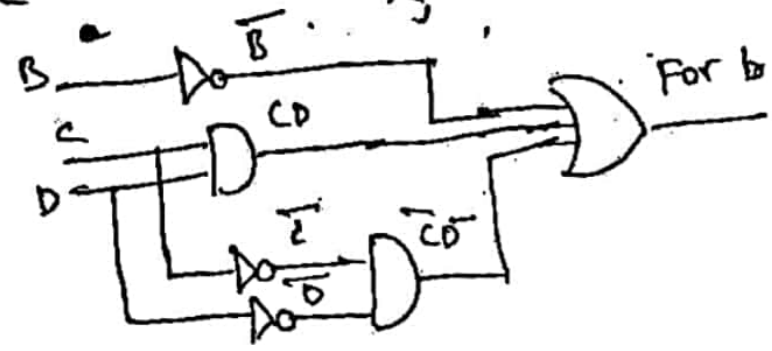
$g = A + B\bar{C} + \bar{B}C + C\bar{D}$

Logic diagram :-

for a

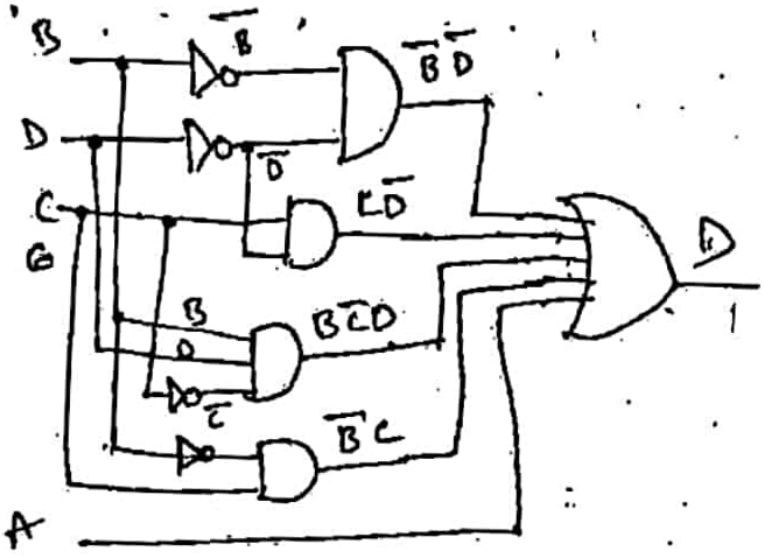


for b :-

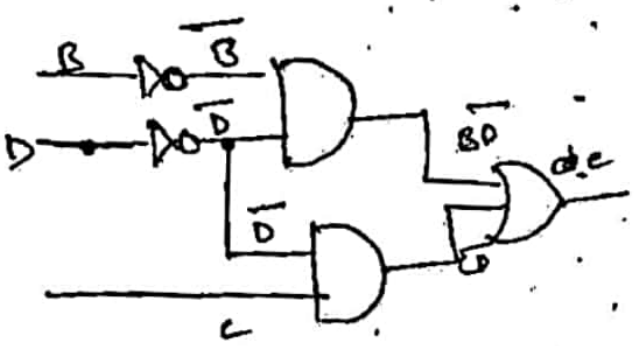


For C :-

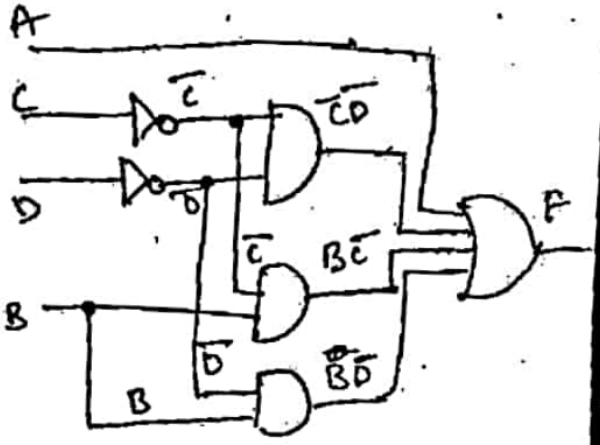
For D



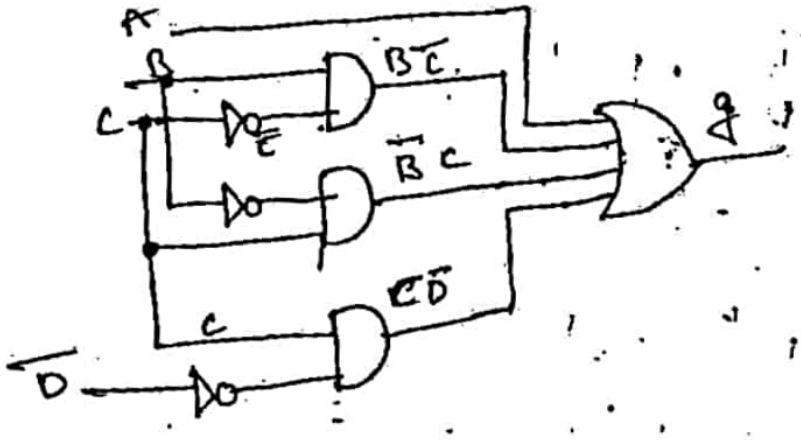
For E :-



For F



For G



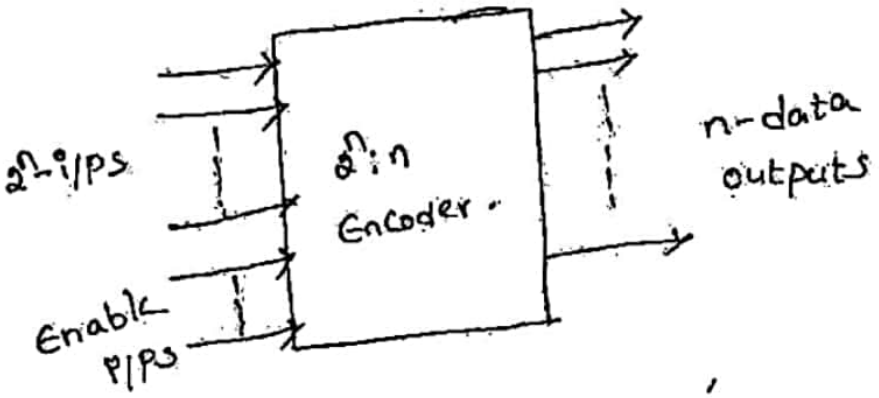
ENCODERS :—

→ An encoder is a digital circuit that performs the inverse operation of a decoder.

→ An encoder has 2^n input lines and n-output lines.

→ In encoder the output lines generate the binary code corresponding to the I/P value.

→ The below figure shows the general structure of the encoder circuit.



Decimal to BCD ENCODER :—

→ The decimal to BCD encoder, usually has ten I/P lines and four O/P lines.

→ The decoded decimal data act as an I/P for encoder and encoded BCD output is available on the four output lines.

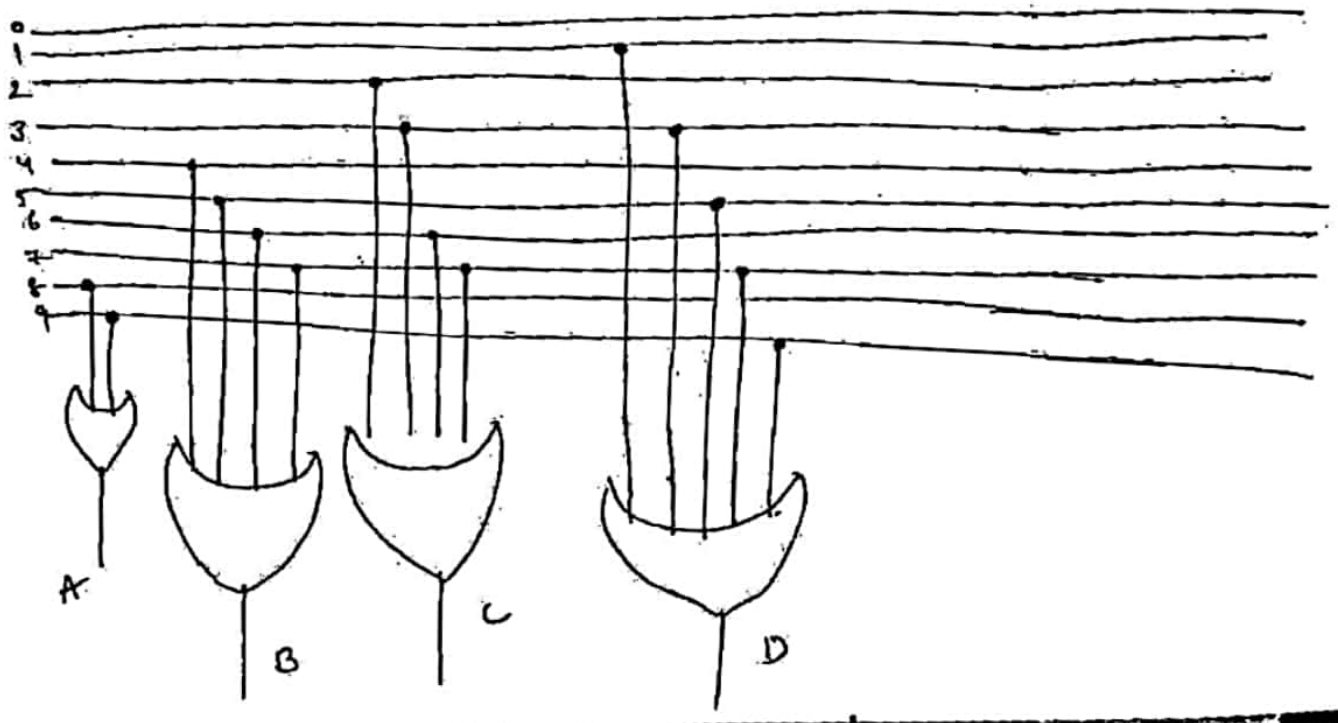
→ The logic symbol for decimal to BCD encoder is shown in below. IC 74xx147 :

→ Both input and output lines are asserted active low.

Decimal Inputs										BCD o/Ps			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

$A = 8+9$ | $B = 4+5+6+7$ | $C = 2+3+6+7$

$D = 1+3+5+7+9$



Octal to Binary Encoder :—

→ It is well-known that binary-to-octal decoder accepts 3 input code and activates one of eight output lines corresponding to that code.

→ An octal-to-binary encoder performs the opposite function; it accepts eight inputs and produces a 3-bit output code corresponding to the activated input.

→ The truth table for 8-to-3 encoder is shown in below.

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$Y_0 = D_1 + D_3 + D_5 + D_7$; $Y_1 = 2+3+6+7$,

$Y_2 = 4+5+6+7$.

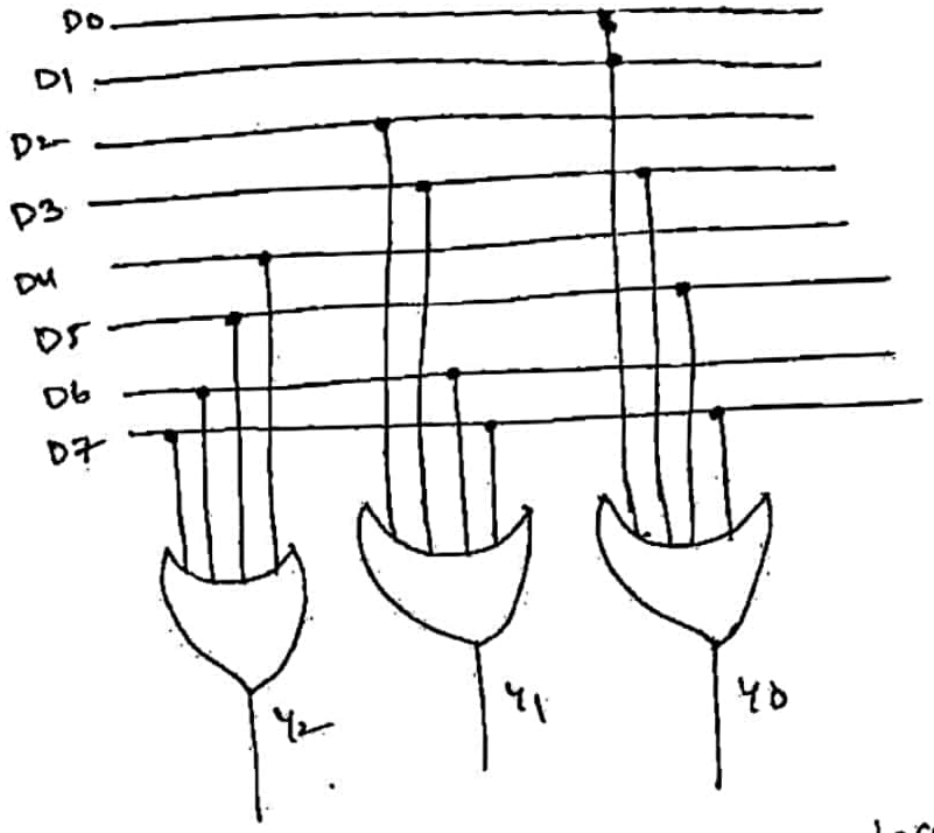


fig:- Logic symbol 8 to 3 decoder.

Decimal-to-BCD Encoder:-

→ A decimal-to-BCD encoder is one with ten i/ps corresponding to ten decimal digits (0 to 9) and four outputs (A, B, C, D) representing the BCD value of input decimal digit.

→ The truth table for decimal-to-BCD Encoder is shown in below.

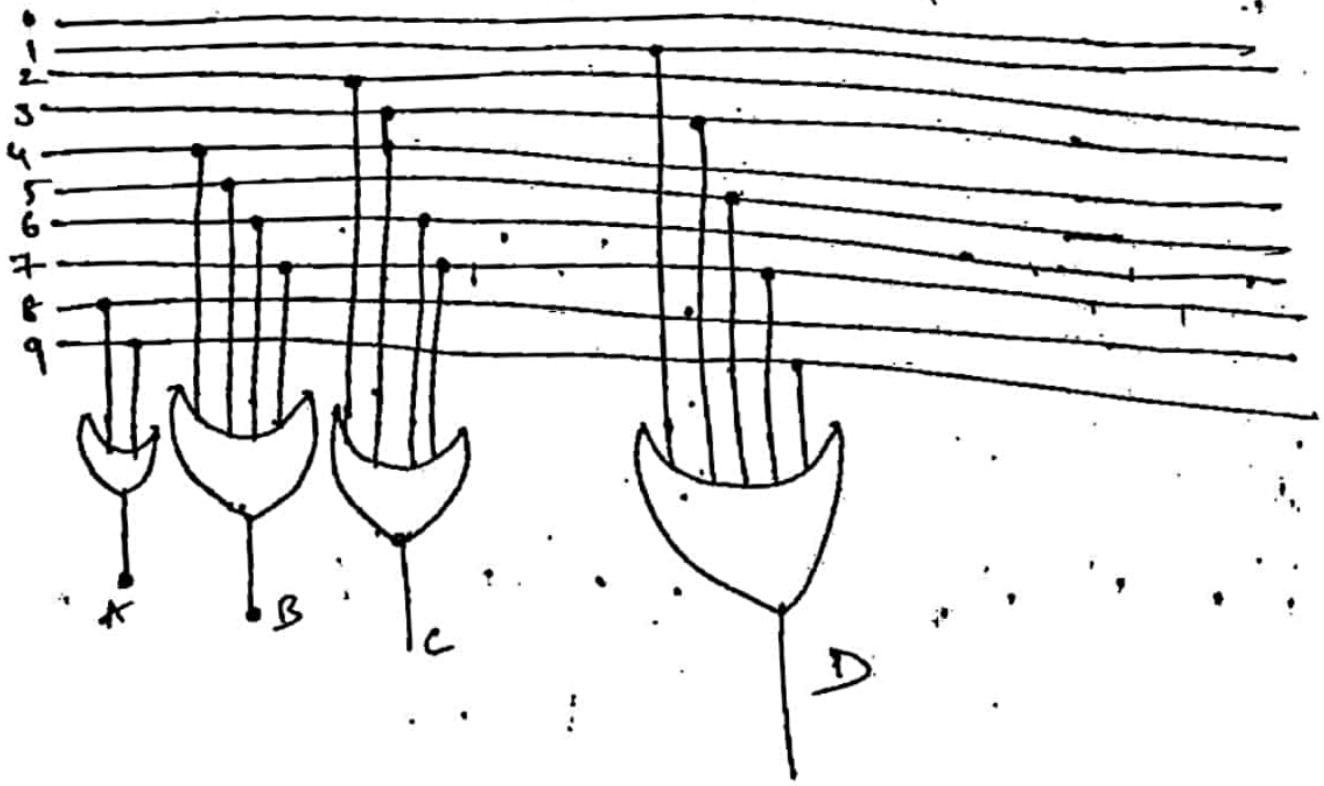
Decimal inputs										BCD outputs			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

→ Here the o/p A is high whenever 8(or) 9 is high, Therefore

$A = 8+9$

$B = 4+5+6+7$, $C = 2+4+6+7$

$D = 1+3+5+7+9$



Priority Encoder :-

→ A priority encoder is an encoder that includes the priority function.

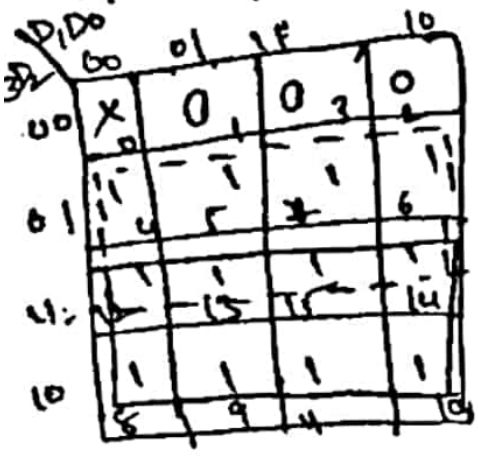
→ The operation of the priority encoder is such that if two (or) more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

→ The truth table of a four bit input priority encoder is given below table.

Inputs				Outputs	
D_0	D_1	D_2	D_3	Y_1	Y_0
0	0	0	0	x	x
1	0	0	0	0	0
x	1	0	0	0	1
x	x	1	0	1	0
x	x	x	1	1	1

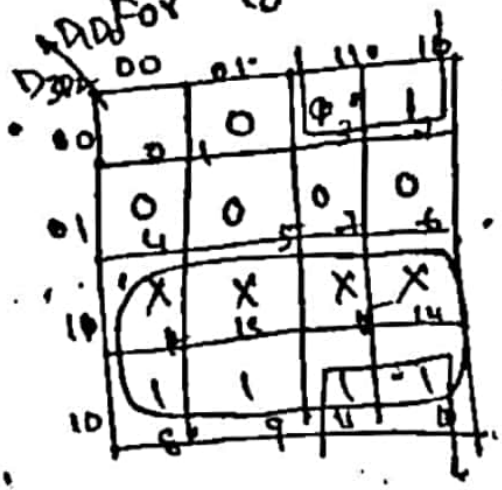
K-map simplification of Y_1 and Y_0 :

for Y_1



$Y_1 = D_2 + D_3$

for Y_0



$Y_0 = D_3 + D_1\bar{D}_2$

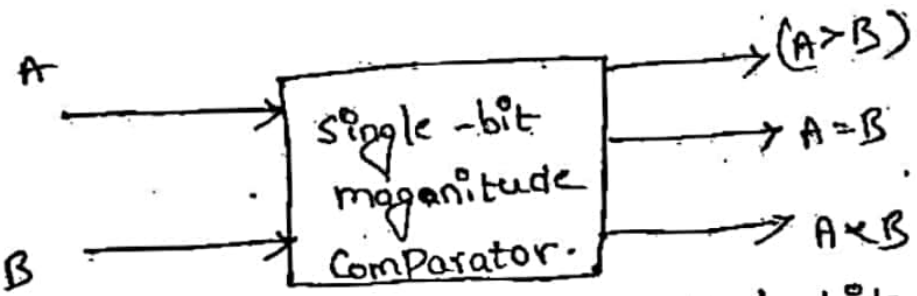
→ 8 to 3 priority Encoder is also having the IC value is same as the priority Encoder by using four input

→ IC 74147 is a Decimal to BCD priority Encoder.

COMPARATOR :—

→ A maganitude Comparator is a combinational circuit that Compares the maganitude of two numbers (A and B) and generaty one of the following outputs: A=B, A<B, and A>B.

→ The block diagram of a siggle-bit magnitude Comparator is shown in blow.



→ The truth table of the siggle bit magnitude Comparator is

inputs		outputs		
A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

→ Here the function of the A>B is having

$A>B = \bar{A}\bar{B}$

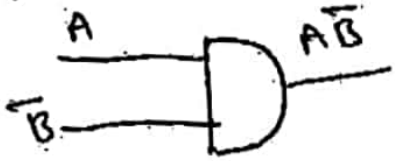
$A<B = \bar{A}B$ and $A=B = AB + \bar{A}\bar{B}$

Assume $x_i = A_i \odot B_i = A \odot B$

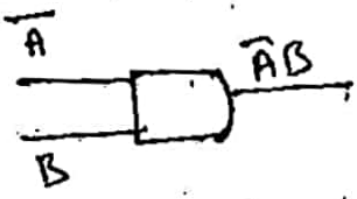
$x_i = A_i \odot B_i = \bar{A}B + A\bar{B}$

→ The logic diagram for A>B, A<B and A=B

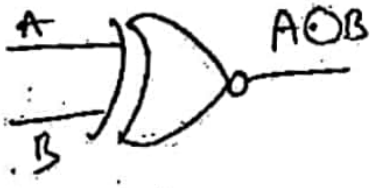
For A>B



For A<B



For A=B



4-Bit magnitude Comparator :-

→ A 4-bit magnitude Comparator Compares two 4-bit numbers A and B gives one of the following outputs : A=B, A<B, and A>B.

→ Let $A = A_3 A_2 A_1 A_0$, $B = B_3 B_2 B_1 B_0$ be the two 4-bit numbers are Compared.

→ The steps involved in Comparing two 4-bit numbers ~~are~~.

→ The following table shows the steps involved by Comparing the two 4-bit numbers are.

A_3B_3	A_2B_2	A_1B_1	A_0B_0	$A>B$	$A<B$	$A=B$
$A_3>B_3$	X	X	X	1	0	0
$A_3<B_3$				0	1	0
$A_3=B_3$	$A_2>B_2$	X	X	1	0	0
x_3	$A_2<B_2$			0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	1	0	0
x_3	x_2	$A_1<B_1$		0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	1	0	0
x_3	x_2	x_1	$A_0<B_0$	0	1	0
x_3	x_2	x_1	x_0	0	0	1

→ $A>B = A_3\bar{B_3} + x_3 A_2\bar{B_2} + x_3 x_2 A_1\bar{B_1} + x_3 x_2 x_1 A_0\bar{B_0}$

$A<B = \bar{A_3}B_3 + x_3 \bar{A_2}B_2 + x_3 x_2 \bar{A_1}B_1 + x_3 x_2 x_1 \bar{A_0}B_0$

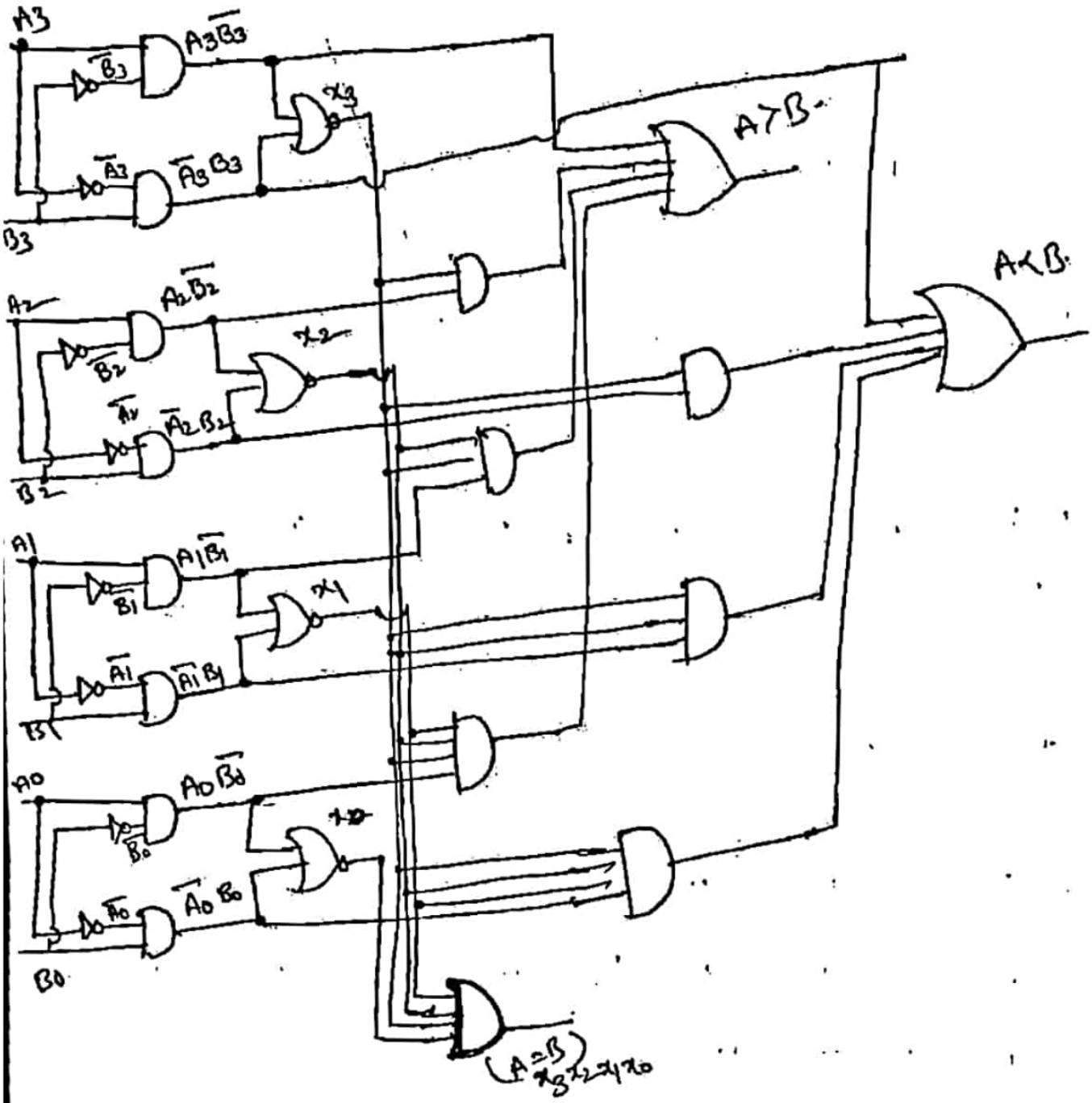
$A=B = x_3 x_2 x_1 x_0$

→ The logic diagram for $A>B$, $A<B$ and $A=B$ is shown in below.

where $x_i = A_i\bar{B}_i + \bar{A}_i B_i$ $\langle i = 3, 2, 1, 0 \rangle$

$x_3 = A_3\bar{B}_3 + \bar{A}_3 B_3$, $x_2 = A_2\bar{B}_2 + \bar{A}_2 B_2$

$x_1 = A_1\bar{B}_1 + \bar{A}_1 B_1$, $x_0 = A_0\bar{B}_0 + \bar{A}_0 B_0$



EXAMPLE 4.17 Implement the following logic function using an 8 × 1 MUX:

$$F(A, B, C, D) = \Sigma\, m(1, 3, 4, 11, 12, 13, 14, 15)$$

Solution

EXAMPLE 4.18 Implement the following function using 8 : 1 MUX

$$F(x, y, z) = \Sigma\, m(0, 2, 3, 5)$$

Solution

ıy.

5 Implement the following function with a MUX:

$F(a, b, c) = \Sigma\, m(1, 3, 5, 6)$

as select inputs

4.9 Realize the logic expression given below using a

(i) 8:1 MUX (ii) 16:1 MUX

$$F = \Sigma\, m(0, 1, 3, 5, 8, 11, 12, 14, 15).$$

4.10 Implement the logic expression given below using a

CODE CONVERTERS :-

→ There is a wide variety of binary codes used in digital systems.

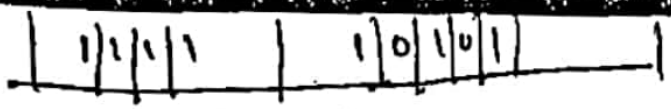
→ Some of these codes are binary-coded-decimal (BCD), excess-3, Gray code and so on.

→ Many times it is required to convert one code to another.

Binary to BCD Converter :-

→ Let us see the truth table for Binary to BCD Converter is shown below.

Binary Code				BCD Code				
A	B	C	D	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1
1	1	1	0	1	0	1	0	0

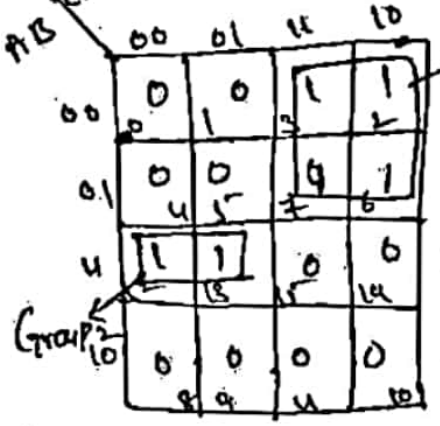


k-map simplification :-

For B_0

For ⊙ $B_0 = D$

For B_1



Group 1

$G_1 = \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + \bar{A}BCD + \bar{A}BC\bar{D}$

$\Rightarrow \bar{A}CD(B+\bar{B}) + \bar{A}C\bar{D}(B+\bar{B})$

$= \bar{A}CD + \bar{A}C\bar{D}$

$= \bar{A}C(D+\bar{D})$

$\boxed{G_1 = \bar{A}C}$

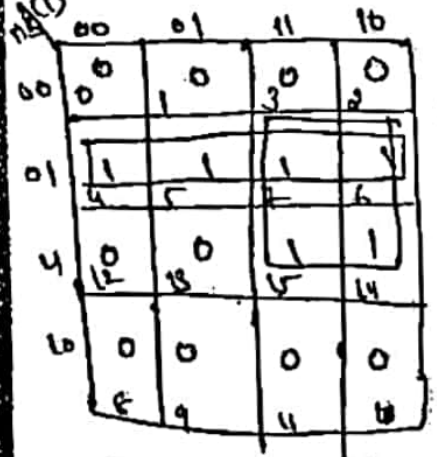
Group 2 :-

$G_2 = AB\bar{C}\bar{D} + AB\bar{C}D$

$= ABD$

$\therefore \boxed{B_1 = \bar{A}C + ABD}$

For B_2



Group 1 $= \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}BCD + \bar{A}BC\bar{D}$

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

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

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

$\boxed{G_1 = \bar{A}B}$

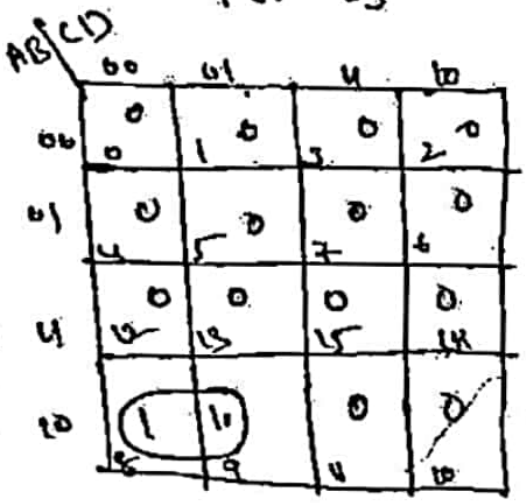
Group $G_2 = \bar{A}BCD + \bar{A}BC\bar{D} + ABCD + ABC\bar{D}$

$= \bar{A}BC(D+\bar{D}) + ABC(D+\bar{D})$

$= \bar{A}BC + ABC \Rightarrow \boxed{BC = G_2}$

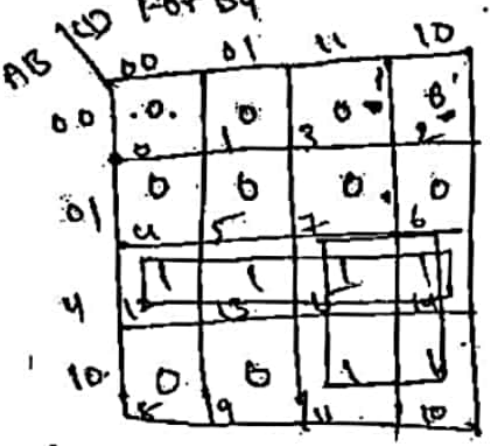
$\boxed{B_2 = \bar{A}B + BC}$ //

For B_3



$B_3 = D\bar{C}\bar{B}$

For B_4



Group 1 $\Rightarrow$ $AB\bar{C}\bar{D} + AB\bar{C}D + \bar{A}BCD + ABCD$

$= AB\bar{C}(D+\bar{D}) + ABC(D+\bar{D})$

$= AB(C+\bar{C})$

$G_{11} = AB$

Group 2 $\Rightarrow$

$\Rightarrow \bar{A}BCD + A\bar{B}CD + A\bar{B}C\bar{D} + ABC\bar{D}$

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

$= ABC + A\bar{B}C$

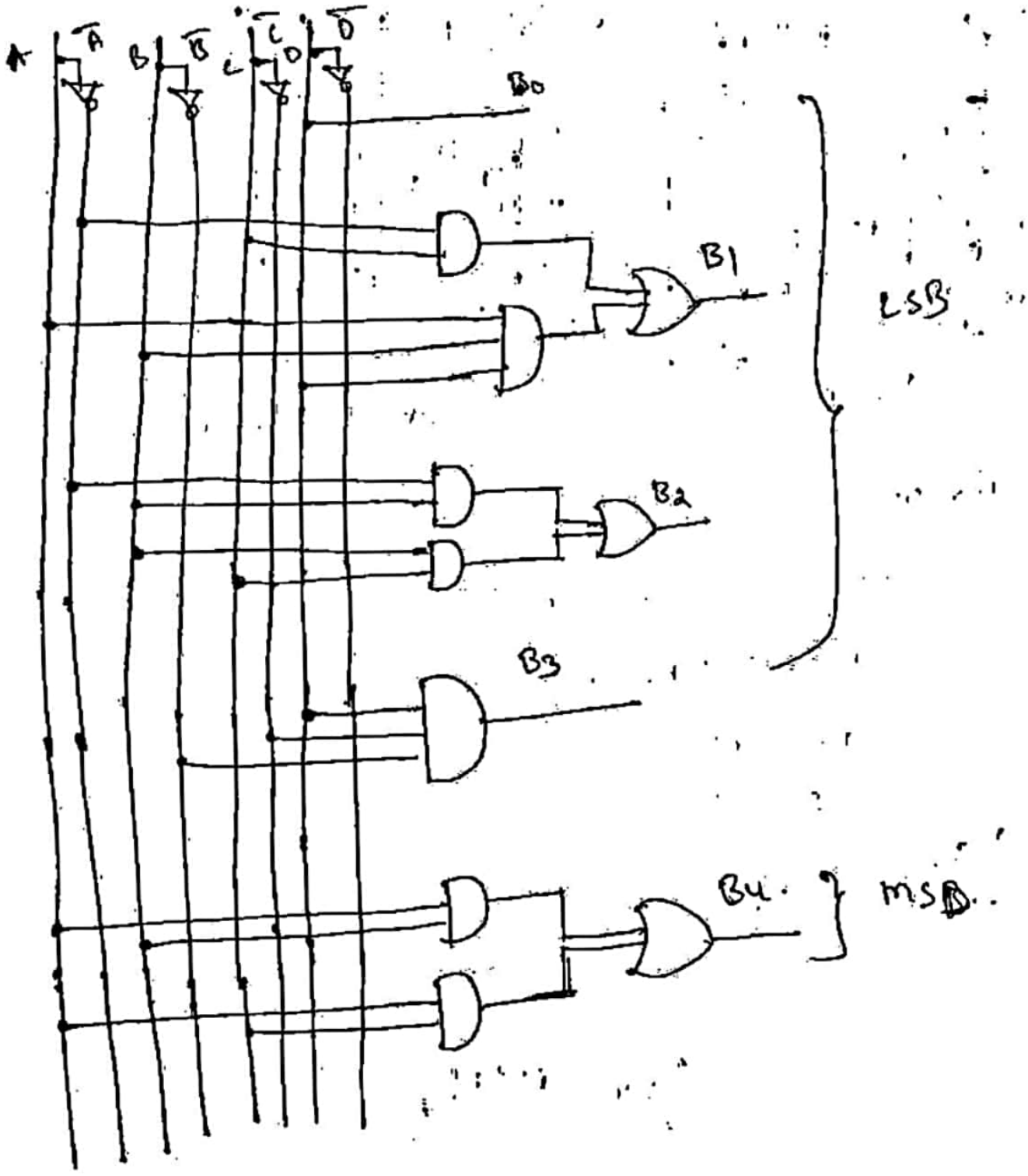
$= AC(B+\bar{B})$

$G_{12} = AC$

$\therefore B_4 = G_{11} + G_{12}$

$B_4 = AC + AB$ //

logic diagram :-



Binary-To-Gray Code Converter :-

→ The Input to the 4-bit binary to Gray code Converter Circuit is a 4-bit binary and output is a 4-bit Gray code.

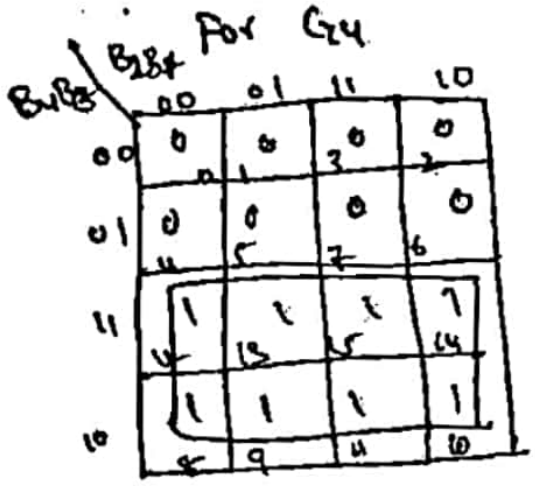
→ There are, 16 possible combinations of 4-bit binary Input and all of them are valid.

→ The Conversation of the Binary to Gray Code truth table is shown in below

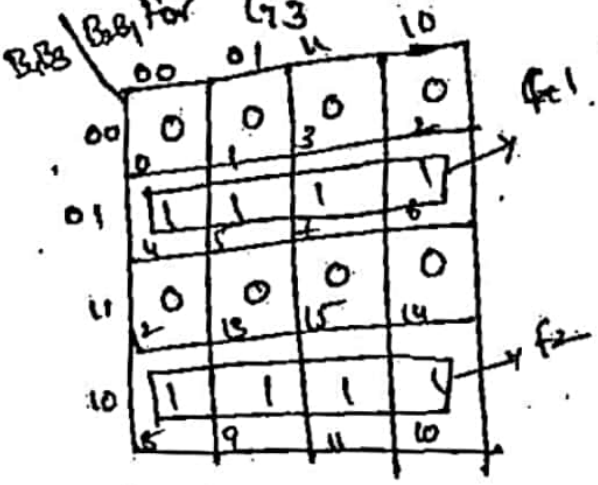
Input				output			
B_4	B_3	B_2	B_1	G_4	G_3	G_2	G_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0

1101	1011
1110	1001
1111	1000

K- simplification :-



$G_4 = B_4$



$F_1 = \bar{B_4}\bar{B_3}\bar{B_2}\bar{B_1} + \bar{B_4}B_3\bar{B_2}B_1 + \bar{B_4}B_3B_2B_1 + \bar{B_4}B_3B_2\bar{B_1}$

$= \bar{B_4}\bar{B_2}B_3(B_1+\bar{B_1}) + \bar{B_4}B_3B_2(B_1+\bar{B_1})$

$\Rightarrow \bar{B_4}\bar{B_2}B_3 + \bar{B_4}B_3B_2$

$\Rightarrow \boxed{\bar{B_4}B_3 = F_1}$

$F_2 = B_4\bar{B_3}\bar{B_2}\bar{B_1} + B_4\bar{B_3}\bar{B_2}B_1 + B_4\bar{B_3}B_2B_1 \; B_4\bar{B_3}B_2\bar{B_1}$

$= B_4\bar{B_3}\bar{B_2}(B_1+\bar{B_1}) + B_4\bar{B_3}B_2(B_1+\bar{B_1})$

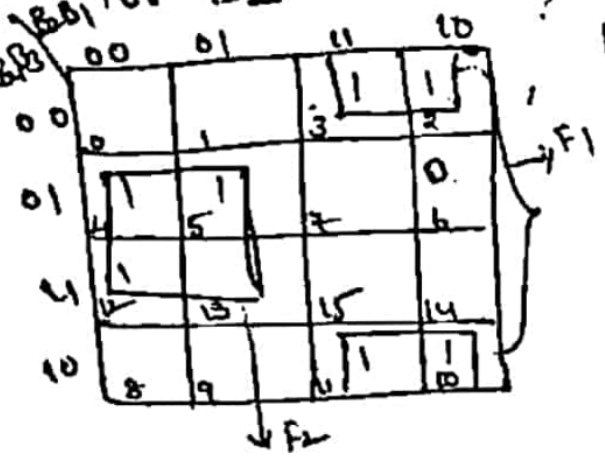
$= B_4\bar{B_3}B_2 + B_4\bar{B_3}\bar{B_2}$

$F_2 = B_4\bar{B_3}$

$F_{13} = F_1 + F_2 \Rightarrow \bar{B_4}B_3 + B_4\bar{B_3}$

$\boxed{F_{13} \Rightarrow = B_4 \oplus B_3}$

For B_2



$F_1 = \bar{B_4}B_3B_2B_1 + \bar{B_4}\bar{B_3}B_2\bar{B_1} + B_4\bar{B_3}B_2B_1 + B_4\bar{B_3}B_2\bar{B_1}$

$= \bar{B_4}\bar{B_3}B_2(B_1+\bar{B_1}) + B_4\bar{B_3}B_2(B_1+\bar{B_1})$

$= \bar{B_4}\bar{B_3}B_2 + B_4\bar{B_3}B_2$

$= \bar{B_3}B_2(B_4+\bar{B_4})$

$\boxed{F_1 = \bar{B_3}B_2}$

$F_2 = \bar{B_4}B_3\bar{B_2}\bar{B_1} + \bar{B_4}B_3\bar{B_2}B_1 + B_4B_3\bar{B_2}\bar{B_1} + B_4B_3\bar{B_2}B_1$

$= \bar{B_4}B_3\bar{B_2}(B_1+\bar{B_1}) + B_4B_3\bar{B_2}(B_1+\bar{B_1})$

$= \bar{B_4}B_3\bar{B_2} + B_4B_3\bar{B_2}$

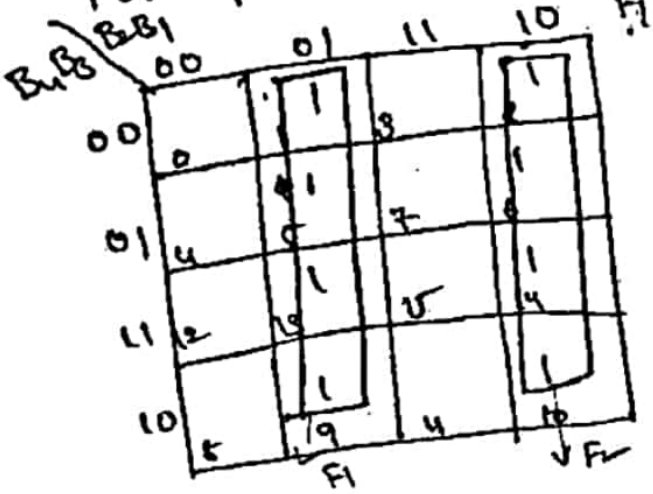
$= \bar{B_2}B_3(B_4+\bar{B_4})$

$F_2 = \bar{B_2}B_3$

$G_{12} = F_1 + F_2 \Rightarrow G_{12} = \bar{B_3}B_2 + B_3\bar{B_2}$

$\boxed{G_{12} = B_3 \oplus B_2}$

For B_1



$F_1 = \bar{B_4}\bar{B_3}\bar{B_2}B_1 + \bar{B_4}B_3\bar{B_2}B_1 + B_4B_3\bar{B_2}B_1 + B_4\bar{B_3}\bar{B_2}B_1$

$= \bar{B_4}\bar{B_2}B_1(B_3+\bar{B_3}) + B_4\bar{B_2}B_1(B_3+\bar{B_3})$

$= \bar{B_4}\bar{B_2}B_1 + B_4\bar{B_2}B_1$

$= \bar{B_2}B_1(B_4+\bar{B_4})$

$\boxed{F_1 = \bar{B_2}B_1}$

$F_2 = \bar{B}_4 \bar{B}_3 B_2 \bar{B}_1 + \bar{B}_4 B_3 B_2 \bar{B}_1 + B_4 B_3 B_2 \bar{B}_1 + B_4 \bar{B}_3 \bar{B}_2 \bar{B}_1$

$= \bar{B}_4 B_2 \bar{B}_1 (B_3 + \bar{B}_3) + B_4 B_2 \bar{B}_1 (B_3 + \bar{B}_3)$

$= B_2 \bar{B}_1 (\bar{B}_4 + B_4)$

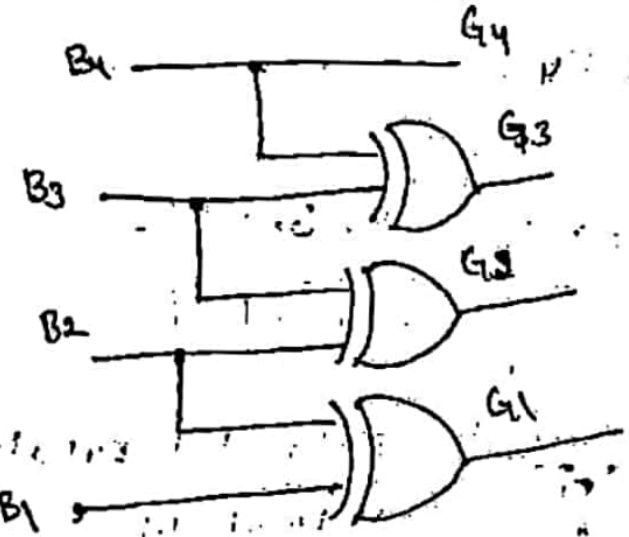
$\boxed{F_2 = B_2 \bar{B}_1}$ //

$G_1 = F_1 + F_2$

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

$\boxed{G_2 = B_2 \oplus B_1}$ //

Logic diagram :-



DESIGN 4-BIT GRAY TO BINARY CODE CONVERTER :- 20

→ The i/p to the 4-bit Gray to Binary code Converter Circuit is 4-bit Gray Code and output 4-bit Binary.

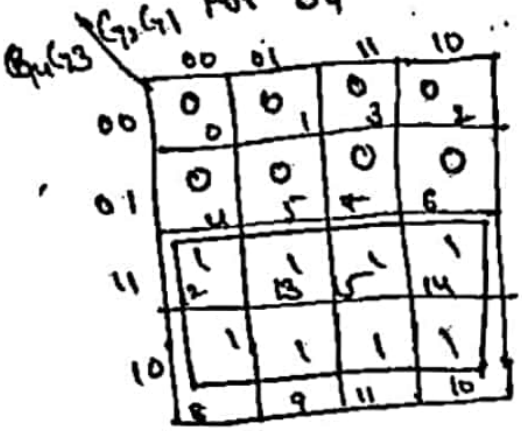
→ There are 16 combinations of 4-bit Gray code Input all of them are valid.

→ The truth table for Gray to Binary code Converter is shown below.

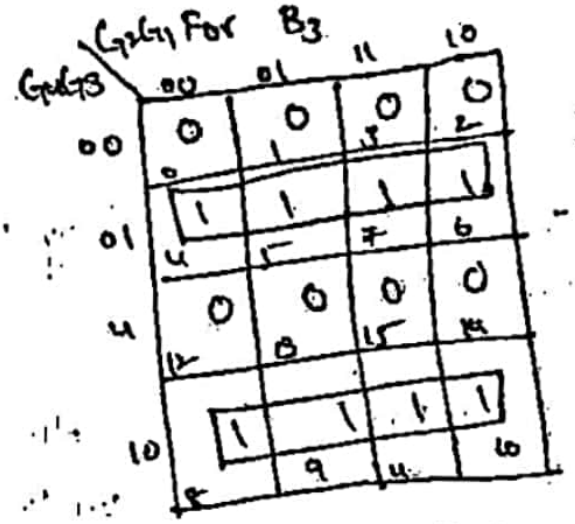
Input				o/p			
G_4	G_3	G_2	G_1	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1

G4 G3 G2 G1	B4 B3 B2 B1
1 0 0 0	1 1 1 1

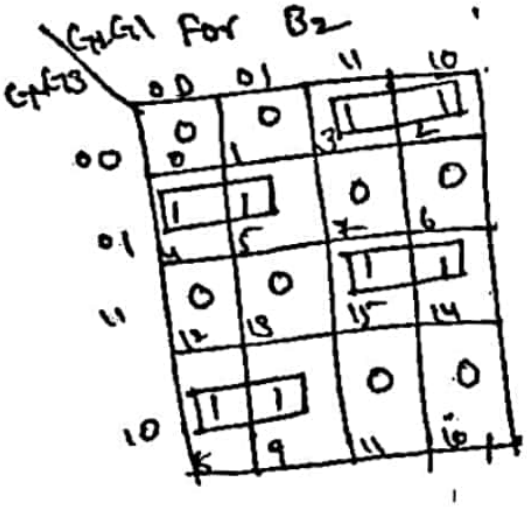
K-map Simplification :-



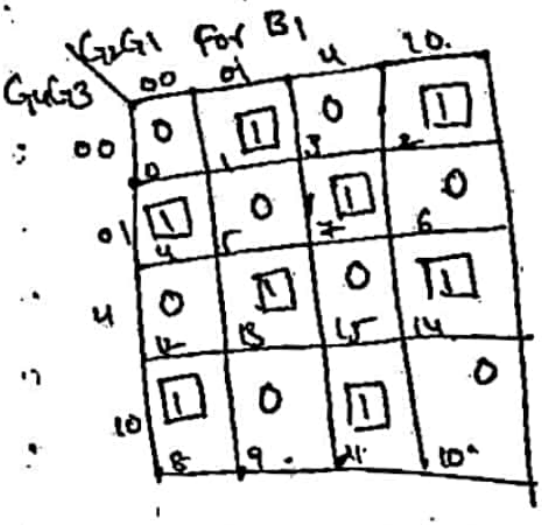
$B_4 = G_4$



$B_3 = G_4 \oplus G_3$

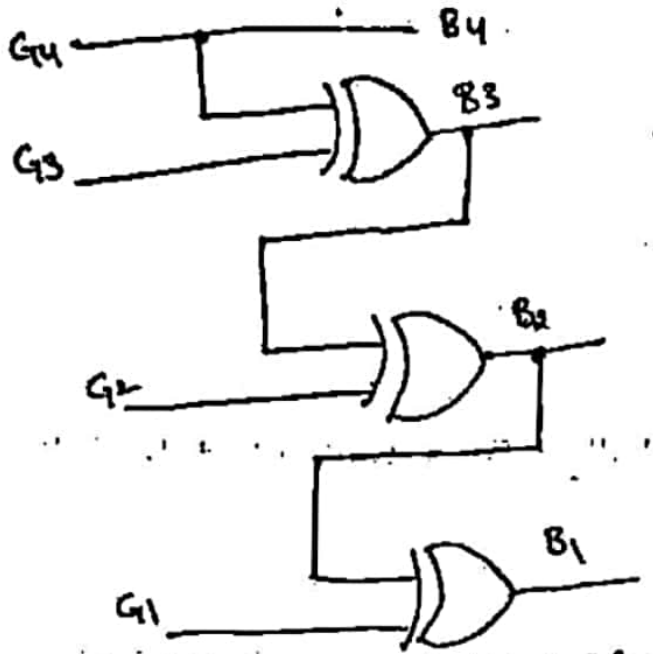


$B_2 = G_4 \oplus G_3 \oplus G_2$



$B_1 = G_4 \oplus G_3 \oplus G_2 \oplus G_1$

Logic diagram :-



DESIGN OF 4-BIT 'BCD-to-EX-3' CODE CONVERTER :-

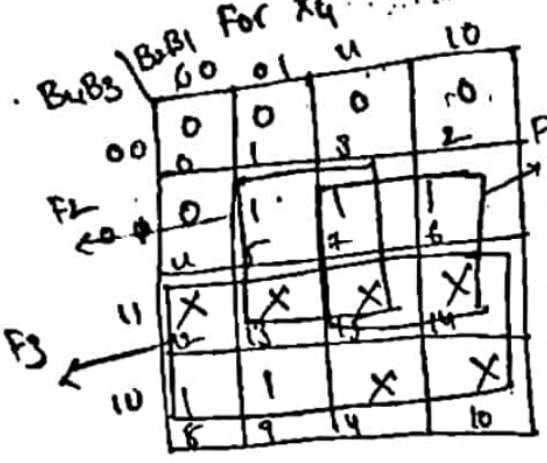
→ BCD means Dual BCD.

→ The 4-bit Input BCD code ($B_4 B_3 B_2 B_1$) and the Corresponding output XS-3 code ($x_4 x_3 x_2 x_1$) numbers are shown in the Conversation table.

→ The i/p combinations 1010, 1011, 1100, 1101, 1110, 1111 are invalid BCD.

→ ∴ So they treated as do not cares.

K-map simplification for x_4



$F_1 = \bar{B_4} B_3 B_2 B_1 + \bar{B_4} B_3 B_2 \bar{B_1} + B_4 B_3 B_2 B_1 + B_4 B_3 B_2 \bar{B_1}$

$= \bar{B_4} B_3 B_2 (B_1 + \bar{B_1}) + B_4 B_3 B_2 (B_1 + \bar{B_1})$

$= \bar{B_4} B_2 B_3 + B_4 B_3 B_2$

$= B_2 B_3 (B_4 + \bar{B_4})$

$\boxed{F_1 = B_2 B_3}$

$F_2 = \bar{B_4}B_3\bar{B_2}B_1 + \bar{B_4}B_3B_2B_1 + B_4B_3\bar{B_2}B_1 + B_4B_3B_2B_1$

$= \bar{B_4}B_3B_1(B_2+\bar{B_2}) + B_4B_3B_1(B_2+\bar{B_2})$

$= \bar{B_4}B_3B_1 + B_4B_3B_1$

$= B_3B_1(B_4+\bar{B_4})$

$\boxed{F_2 = B_3B_1}$

~~$F_3 = B_4B_3\bar{B_2}\bar{B_1} + B_4B_3\bar{B_2}B_1 + B_4B_3B_2B_1 + B_4B_3B_2\bar{B_1}$~~

$= B_4B_3\bar{B_2}$

$F_3 = \boxed{B_4B_3\bar{B_2}\bar{B_1} + B_4B_3\bar{B_2}B_1 + B_4B_3B_2B_1 + B_4B_3B_2\bar{B_1}} \rightarrow B_4B_3$

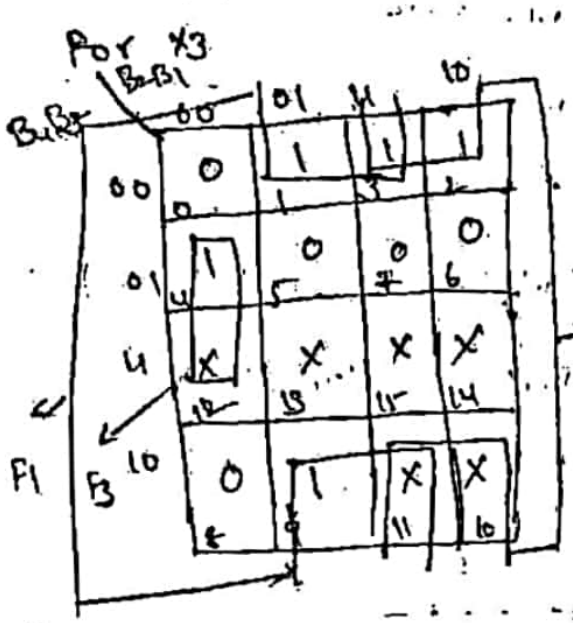
$+ \boxed{B_4\bar{B_3}\bar{B_2}\bar{B_1} + B_4\bar{B_3}\bar{B_2}B_1 + B_4\bar{B_3}B_2B_1 + B_4\bar{B_3}B_2\bar{B_1}} \rightarrow B_4\bar{B_3}$

$= B_4B_3 + B_4\bar{B_3}$

$= B_4(B_3+\bar{B_3})$

$F_3 = B_4$

$\therefore \boxed{X_4 = B_3B_2 + B_3B_1 + B_4}$



$F_1 = \bar{B_4}\bar{B_3}\bar{B_2}B_1 + \bar{B_4}\bar{B_3}B_2B_1 + B_4\bar{B_3}B_2B_1$

$+ B_4\bar{B_3}B_2\bar{B_1}$

$= \bar{B_4}\bar{B_3}B_1(B_2+\bar{B_2}) + B_4\bar{B_3}B_1(B_2+\bar{B_2})$

$= \bar{B_4}\bar{B_3}B_1 + B_4\bar{B_3}B_1$

$= \bar{B_3}B_1(B_4+\bar{B_4})$

$\boxed{F_1 = \bar{B_3}B_1}$

$F_2 = \bar{B}_4 \bar{B}_3 B_2 B_1 + \bar{B}_4 \bar{B}_3 B_2 \bar{B}_1 + \bar{B}_4 \bar{B}_3 B_2 B_1 + B_4 \bar{B}_3 B_2 \bar{B}_1$

$= \bar{B}_4 \bar{B}_3 B_2 (B_1 + \bar{B}_1) + B_4 \bar{B}_3 B_2 (B_1 + \bar{B}_1)$

$= \bar{B}_4 \bar{B}_3 B_2 + \bar{B}_3 B_2 B_4$

$\boxed{F_2 = \bar{B}_3 B_2}$

$F_3 = \bar{B}_4 B_3 \bar{B}_2 \bar{B}_1 + B_4 B_3 \bar{B}_2 \bar{B}_1$

$= B_3 \bar{B}_2 \bar{B}_1 (B_4 + \bar{B}_4)$

$\boxed{F_3 = B_3 \bar{B}_2 \bar{B}_1}$ //

$\therefore X_3 = \bar{B}_3 B_1 + \bar{B}_3 B_2 + B_3 \bar{B}_2 \bar{B}_1$

For X_2

$F_1 = \bar{B}_4 \bar{B}_3 \bar{B}_2 \bar{B}_1 + \bar{B}_4 B_3 \bar{B}_2 B_1 + B_4 B_3 \bar{B}_2 \bar{B}_1 + B_4 \bar{B}_3 \bar{B}_2 \bar{B}_1$

$= \bar{B}_4 \bar{B}_2 \bar{B}_1 (B_3 + \bar{B}_3) + B_4 \bar{B}_2 \bar{B}_1 (B_3 + \bar{B}_3)$

$\boxed{F_1 = \bar{B}_2 \bar{B}_1}$

$F_2 = \bar{B}_4 \bar{B}_3 B_2 B_1 + \bar{B}_4 \bar{B}_3 B_2 \bar{B}_1 + B_4 B_3 B_2 B_1 + B_4 \bar{B}_3 B_2 B_1$

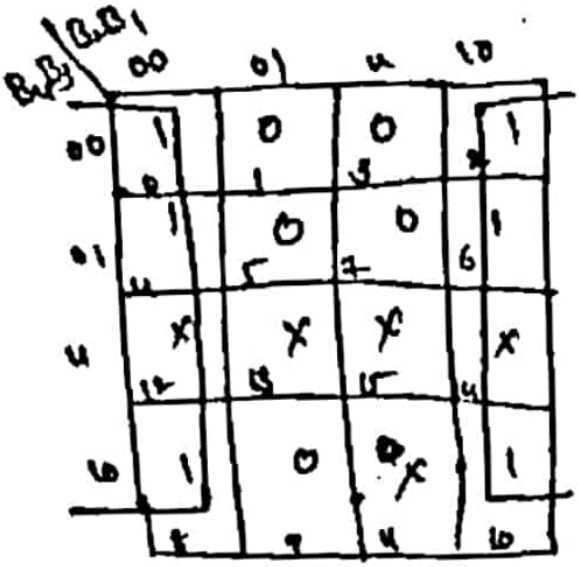
$= \bar{B}_4 B_2 B_1 (B_3 + \bar{B}_3) + B_4 B_2 B_1 (B_3 + \bar{B}_3)$

$\boxed{F_2 = B_2 B_1}$

$\therefore X_2 = \bar{B}_2 \bar{B}_1 + B_1 B_2$

$\therefore \boxed{X_2 = B_1 \odot B_2}$ //

For x_1 :-



$= x_1 = \bar{B}_4 \bar{B}_3 \bar{B}_2 \bar{B}_1 + \bar{B}_4 B_3 \bar{B}_2 \bar{B}_1 + B_4 B_3 \bar{B}_2 \bar{B}_1 + B_4 \bar{B}_3 \bar{B}_2 \bar{B}_1$

$+ \bar{B}_4 \bar{B}_3 B_2 \bar{B}_1 + \bar{B}_4 B_3 B_2 \bar{B}_1 + B_4 B_3 B_2 \bar{B}_1 + B_4 \bar{B}_3 B_2 \bar{B}_1$

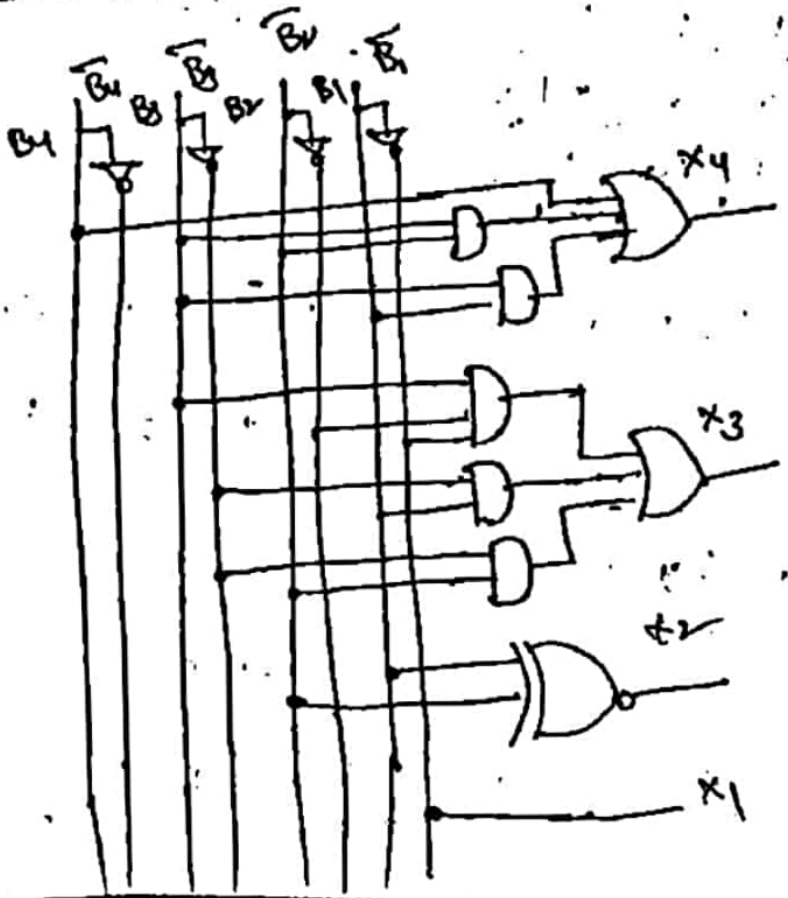
$= \bar{B}_4 \bar{B}_2 \bar{B}_1 (B_3 + \bar{B}_3) + B_4 \bar{B}_3 \bar{B}_1 (B_2 + \bar{B}_2) + \bar{B}_4 B_2 \bar{B}_1 (B_3 + \bar{B}_3)$

$+ B_4 B_2 \bar{B}_1 (B_3 + \bar{B}_3)$

$= \bar{B}_4 \bar{B}_1$

$\boxed{x_1 = B_1}$

Logic diagram :-



DESIGN of BCD-GRAY Code CONVERTER :-

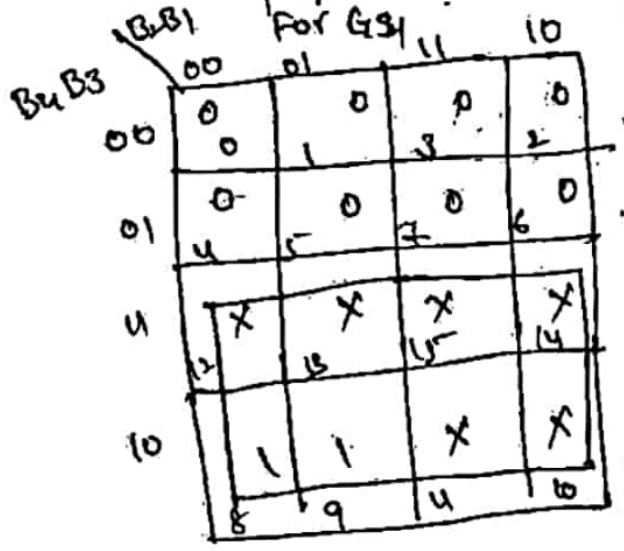
→ The BCD to gray code conversion table is shown in below.

→ For a 4-bit BCD code minterms 10, 11, 12, 13, 14 and 15 are donot cares.

BCD	Gray	XS-3 code
$B_4B_3B_2B_1$	$G_4G_3G_2G_1$	$x_4\ x_3\ x_2\ x_1$
0000	0000	0 0 1 1
0001	0001	0 1 0 0
0010	0011	0 1 0 1
0011	0010	0 1 1 0
0100	0110	0 1 1 1
0101	0111	1 0 0 0
0110	0101	1 0 0 1
0111	0100	1 0 1 0
1000	1100	1 0 1 1
1001	1101	1 1 0 0

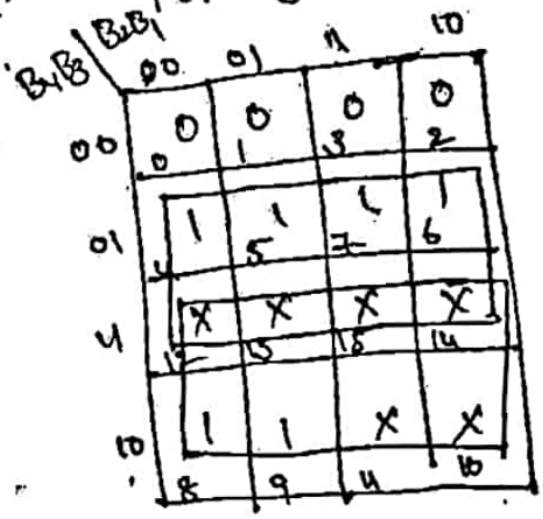
- K-map simplification :-

For G_4



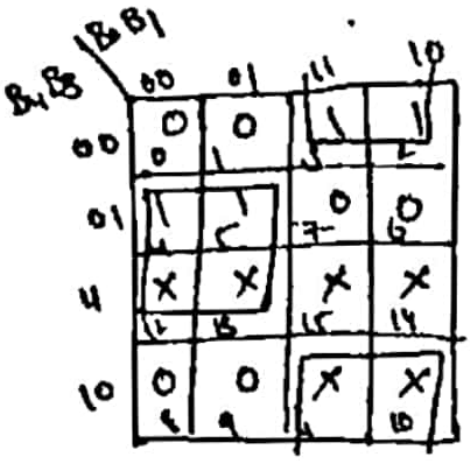
$G_4 = B_4$

For G_3



$G_3 = B_3 + B_4$

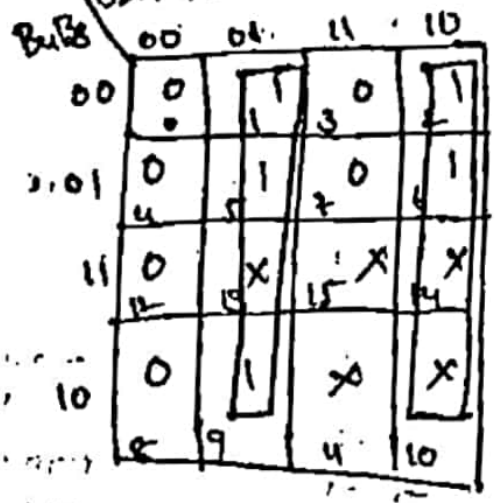
For G_2



$G_2 = B_3\bar{B}_2 + \bar{B}_3 B_2$

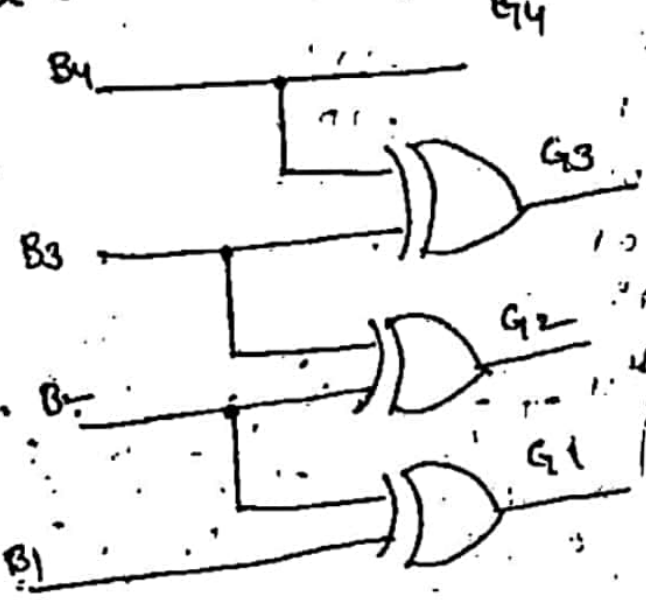
$= B_3 \oplus B_2$

B_2B_1 for G_1



$G_1 = B_2 \oplus B_0$

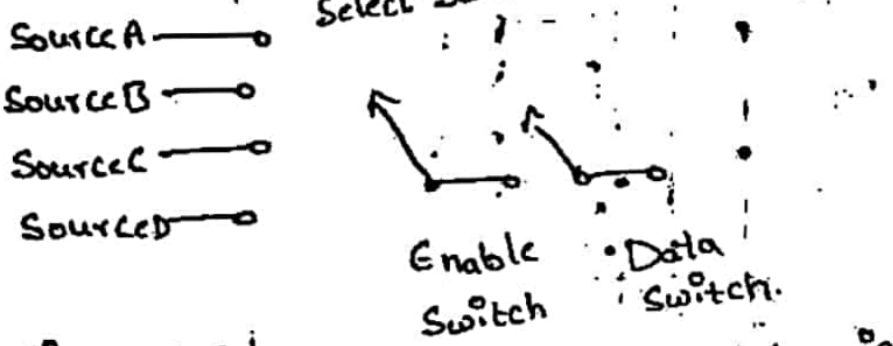
Logic Diagram :—



MULTIPLEXERS :-

→ Multiplexer is a digital switch.

→ It allows digital information from several sources to be routed onto a single output line as shown in below figure.



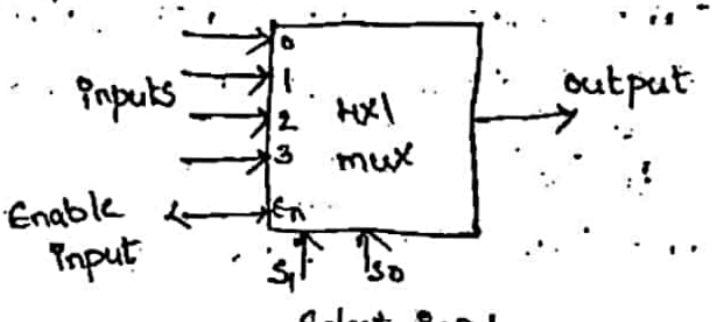
→ The Basic multiplexer has several data-input lines and a single output line.

→ The selection of a particular input line is controlled by a set of selection lines.

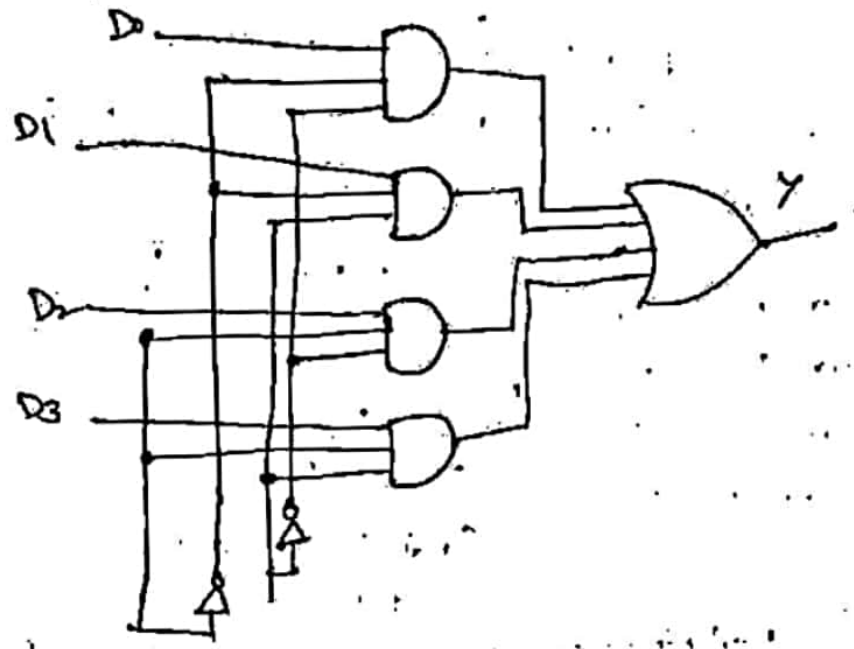
→ Normally, there are 2^n input lines and "n" selection lines whose bit combinations determine which input is selected.

→ Therefore, multiplexer is "many into one" and it provides the digital equivalent of an analog selector switch.

→ The logic symbol for multiplexer is shown in below figure



→ The below figure shows logic diagram of 4-to-1 line multiplexer.



S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

→ functional table for 4-to-1 multiplexer is shown above.

→ For example, when $S_1S_0 = 01$, the AND gate associated with data input "D_1" has two of its inputs equal to "1" and the third input connected to D_1.

→ The other three AND gates have atleast one input equal to 0, which makes their output equal to "0".

→ The "OR" gate o/p is equal to the value of D_1.

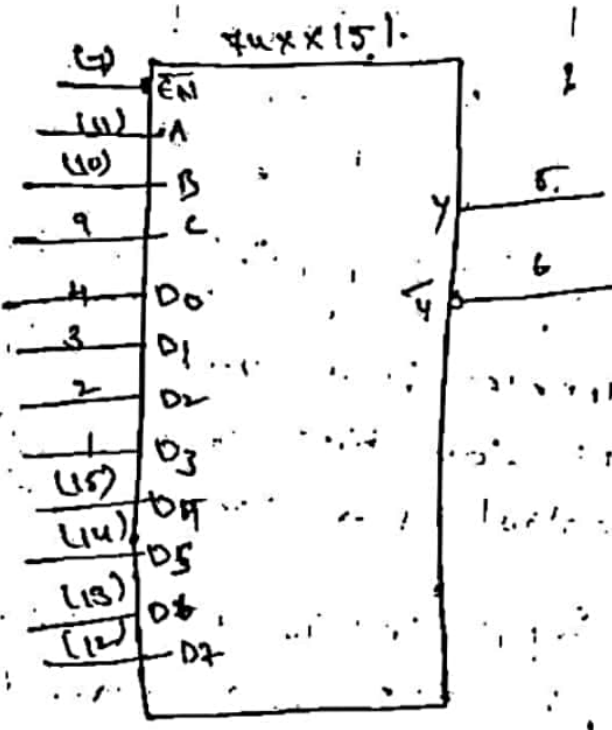
→ We can say data bit D_1 is routed to the output when $S_1 S_0 = 01$.

74xx151 8-to-1 Multiplexer :-

→ The 74xx151 is a 8 to 1 Multiplexer.

→ It has eight inputs. It provides two o/p's, one is active high, the other active low.

→ The below figure shows logic symbol for 74xx151.



→ In this truth table for each input combination output is not specified in 1s and 0s.

→ Because we know that multiplexer is a data switch, It is not generate any data of its own.

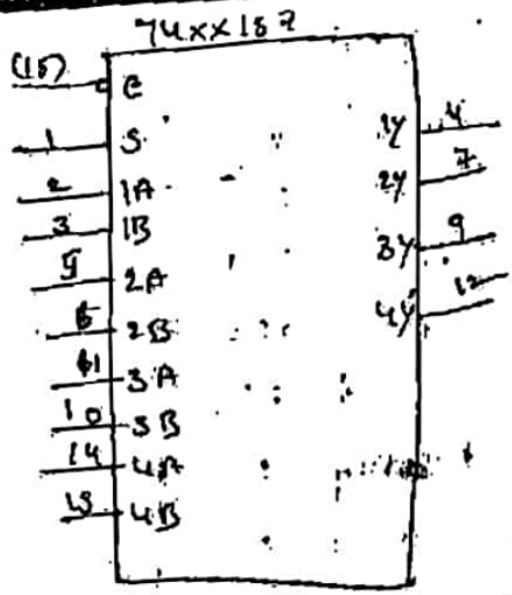
Input			output	
Select A B C		Enable $\overline{EN}$	Y	$\overline{Y}$
X X X		1	0	1
0 0 0		0	D_0	$\overline{D_0}$
0 0 1		0	D_1	$\overline{D_1}$
0 1 0		0	D_2	$\overline{D_2}$
0 1 1		0	D_3	$\overline{D_3}$
1 0 0		0	D_4	$\overline{D_4}$
1 0 1		0	D_5	$\overline{D_5}$
1 1 0		0	D_6	$\overline{D_6}$
1 1 1		0	D_7	$\overline{D_7}$

74HXX157 Quad 2-input multiplexer :-

→ The IC 74XX157 is a quad 2-input multiplexer which selects four bits of data from two sources under the control of a common select input (s)

→ The Enable input ($\overline{En}$) is active low. when $\overline{E}$ is high all of the o/ps are forced low regardless of all other input conditions.

→ The logic symbol and truth table of 74XX157 is shown in below.



truth table of 74xx157 :-

input		output			
$\overline{E}$	S	1Y	2Y	3Y	4Y
1	X	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

74xx153 Dual 4 to 1 multiplexer :-

→ The 74xx153 is a dual 4 to 1 multiplexer.

→ The below figure shows the logic symbol for 74xx153.

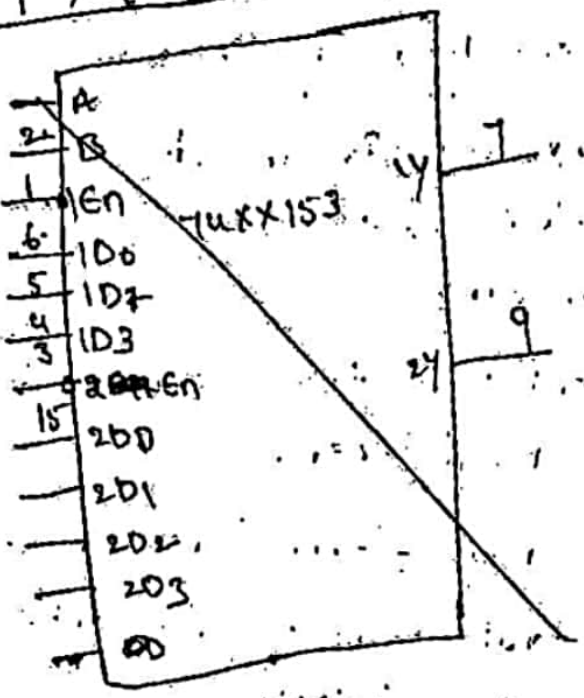
→ It contians two identical and independant 4 to 1 multiplexer.

→ Each multiplexer has a separate enable inputs.

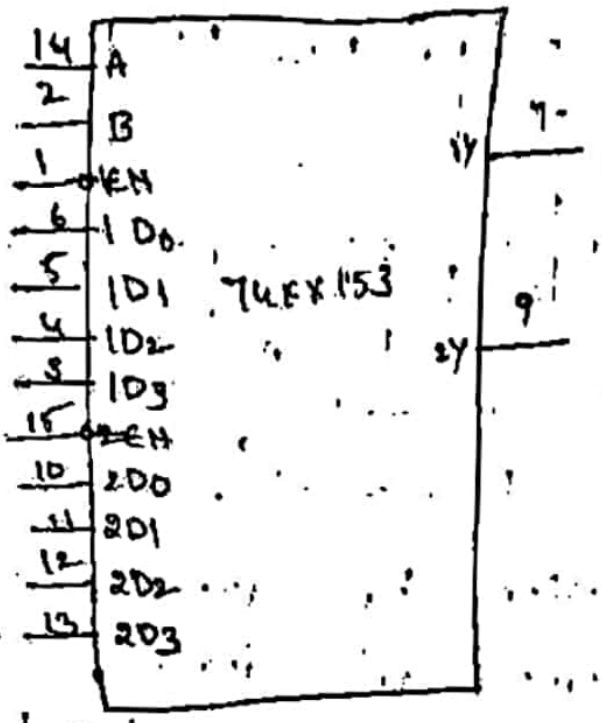
→ The truth table for 4 to 1 multiplexer is shown in below.

Input				outputs	
1En	2En	B	A	1Y	2Y
0	0	0	0	1D0	2D0
0	0	0	1	1D1	2D1
0	0	1	0	1D2	2D2
0	0	1	1	1D3	2D3
0	1	0	0	1D0	0
0	1	0	1	1D1	0
0	1	1	0	1D2	0
0	1	1	1	1D3	0
1	0	0	0	0	2D0
1	0	0	1	0	2D1
1	0	1	0	0	2D2
1	0	1	1	0	2D3
1	1	X	X	0	0

Logic Symbol:-



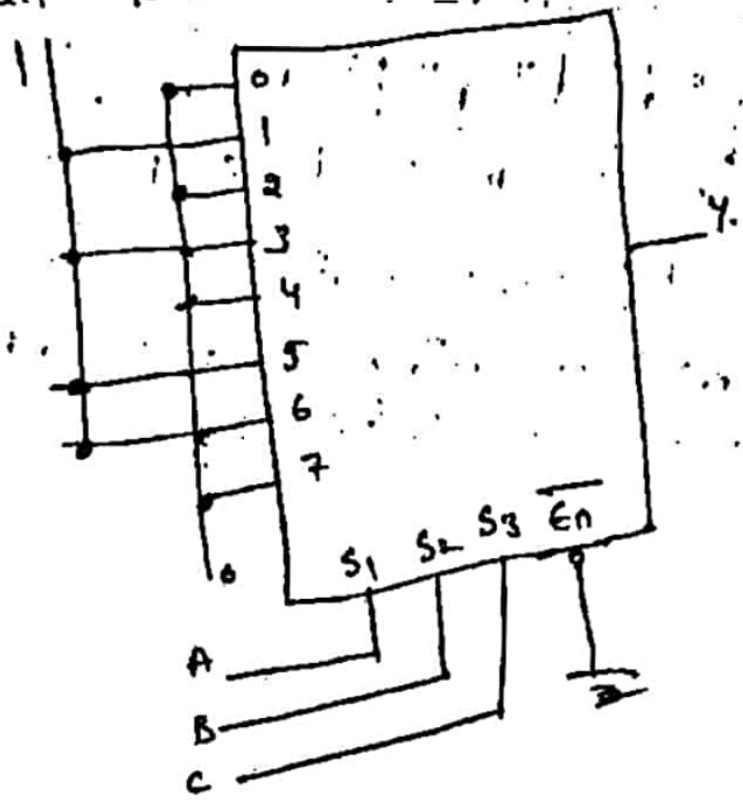
IC symbol for 74XX153 :—



Problem 1 :—

1. Implement the following Boolean function using 8:1 multiplexer : $f(A,B,C) = \sum m(1,3,5,6)$

Sol :— Boolean function implementation using "mux"



Implementation table :-

	D_0	D_1	D_2	D_3	
$\bar{A}$	0	(1)	2	(3)	row1
A	4	(5)	(6)	7	row2
	0	1	A	$\bar{A}$	

2. Implement the following Boolean function using 4:1 mux $F(A,B,C,D) = \Sigma m(0,1,8,4,6,9,12,14)$

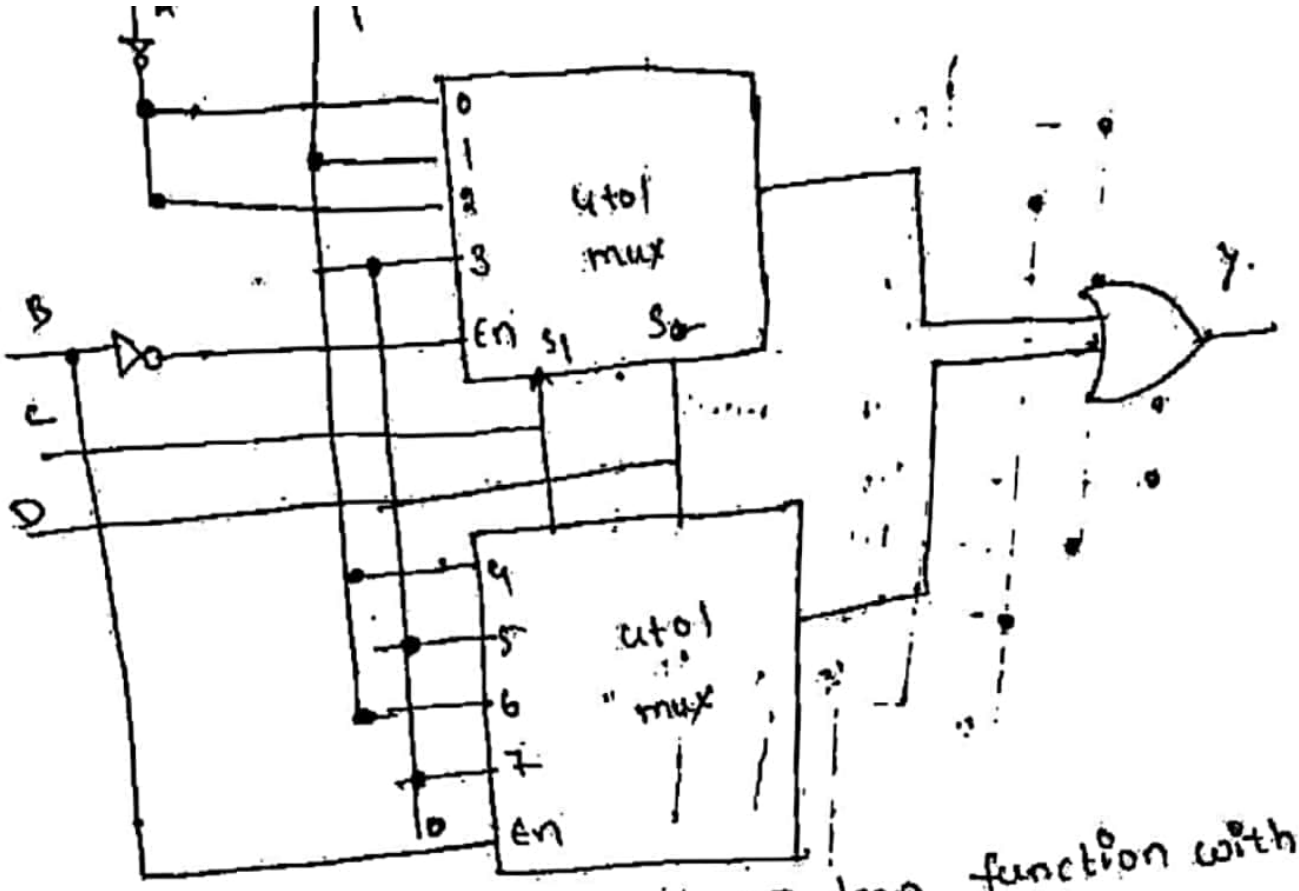
Sol:- The function has four variables.

→ To implement this function we require 8:1 multiplexer i.e, two 4 to 1 multiplexer.

	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	
	$\bar{A}$	1	$\bar{A}$	0	1	0	1	0	

fig Implementation of table.

→ Implementation using two 4 to 1 multiplexer. is shown in below figure.

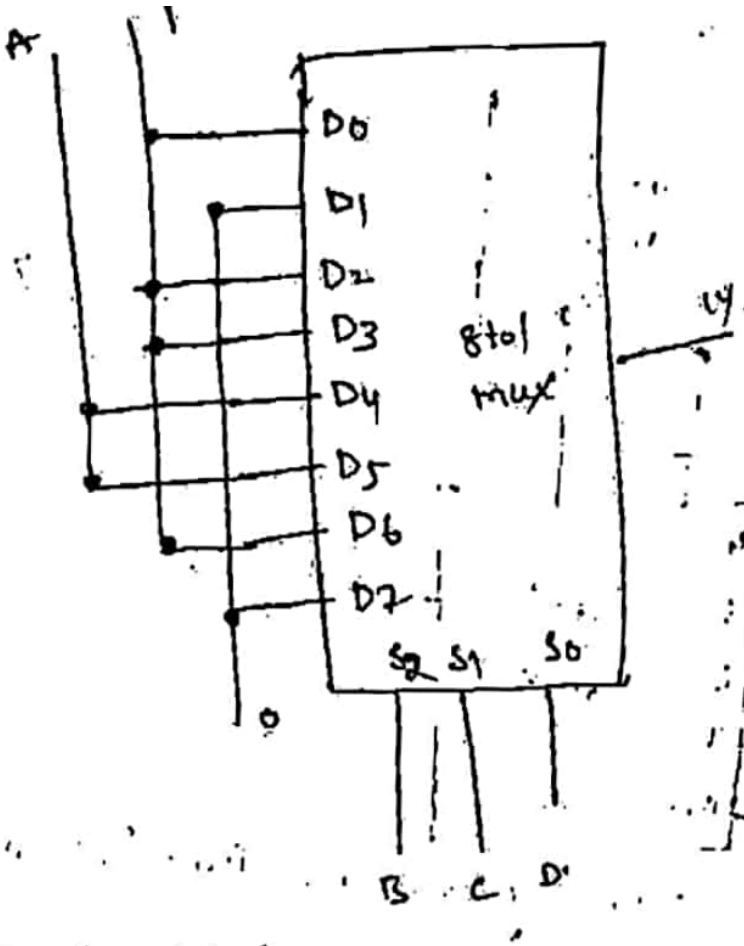


3. Implement the following Boolean function with 8:1 multiplexer.

$F(A,B,C,D) = \Sigma m(0,2,6,10,11,12,13) + d(3,8,14)$

Sol:

	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
	1	0	1	1	A	A	1	0



DEMULTIPLEXER :-

→ A demultiplexer is a circuit that recives information on a single line and transmits this information on one of 2^n possible o/p lines.

→ The selection of specific o/p lines are controlled by the value of 'n' selection lines.

En	S_1	S_0	Din	Y_0	Y_1	Y_2	Y_3
0	X	X	X	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1

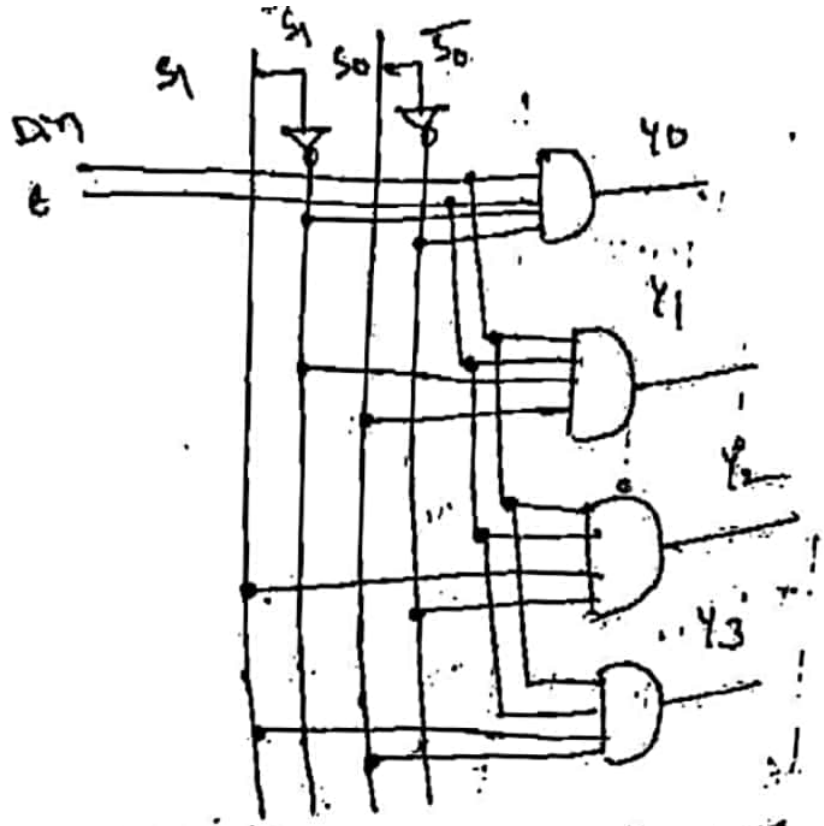


fig:- logic diagram.

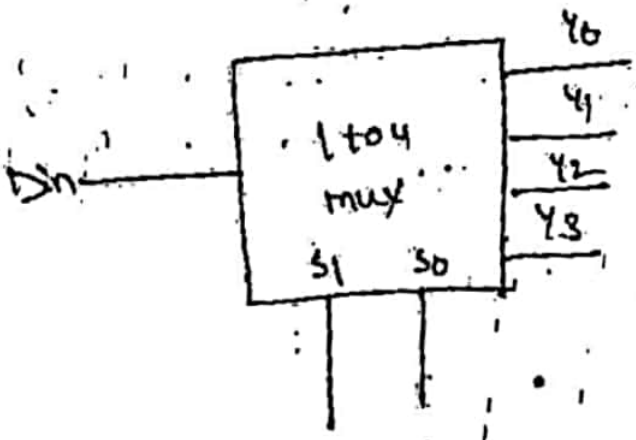


fig:- logic symbol.

Problem 2:-

Design 1 to 8 demultiplexer using two 1 to 4 demuxs

sol:- The cascading of demultiplexers is similar to the cascading of decoder.

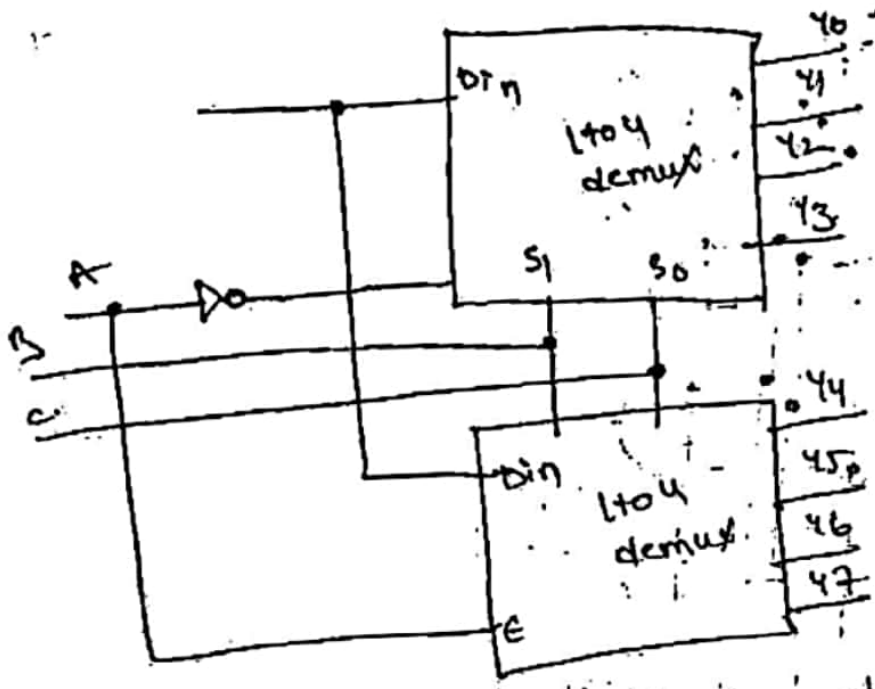


fig:- Cascading of demultiplexer.

2. Implement the full subtractor using demultiplexer

Sol:- Let us see the truth table of full subtractor.

A	B	B	D	B_{out}
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

$Din = f(ABC) = \Sigma m(1,2,4,7)$, $B_{out} = \Sigma m(1,2,3,7)$

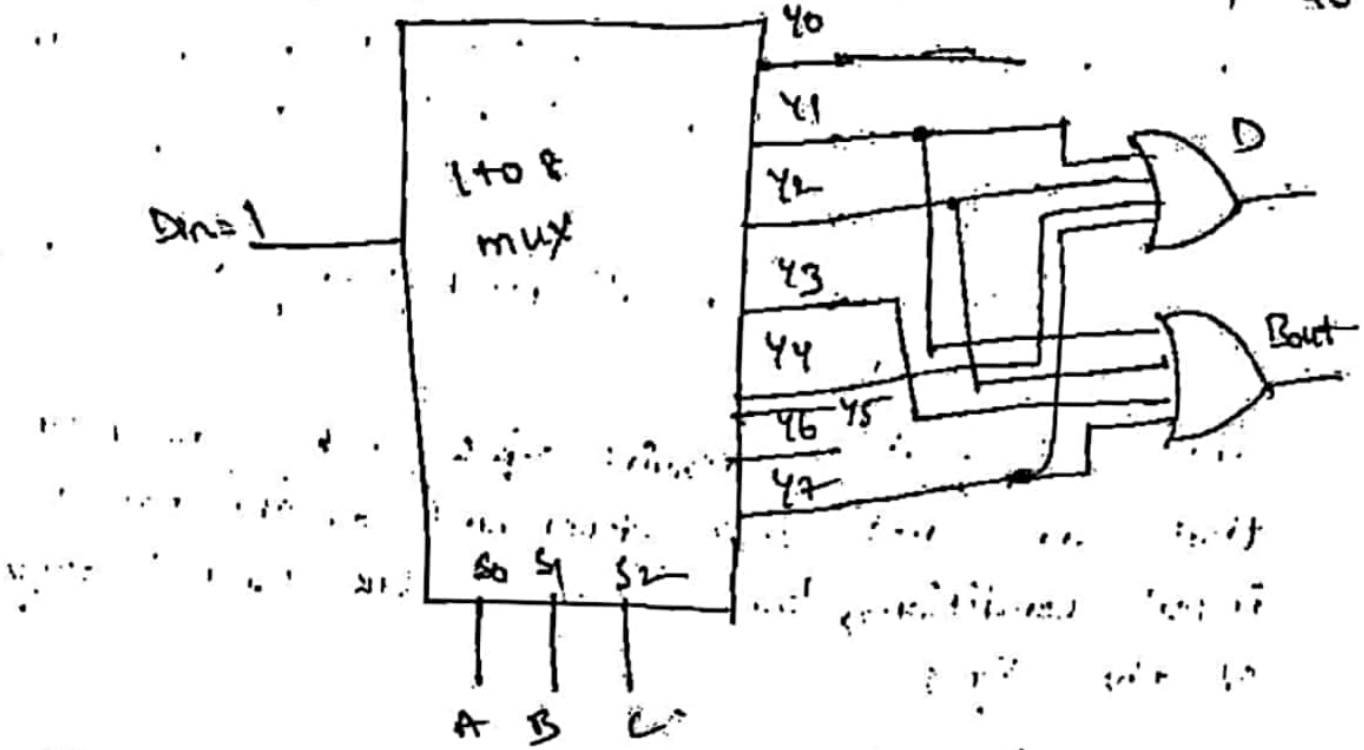


fig :- full Subtractor using 1 to 8 demux

SEQUENTIAL CIRCUITS :-

→ Sequential switching circuits are those whose output levels at any instant of time are dependant not only on the levels present at the input at that time, but also on the its past input.

→ The past history is provided by the feedback from the o/p back to the input.

→ It means that sequential switching circuits have memory.

→ Sequential circuits are, thus, made of Combinational circuits and memory elements.

→ The past history is provided by feedback from the o/p back to the i/p

→ There are many applications in which digital o/ps are required to be generated in accordance with, the sequence in which the i/p signals are received.

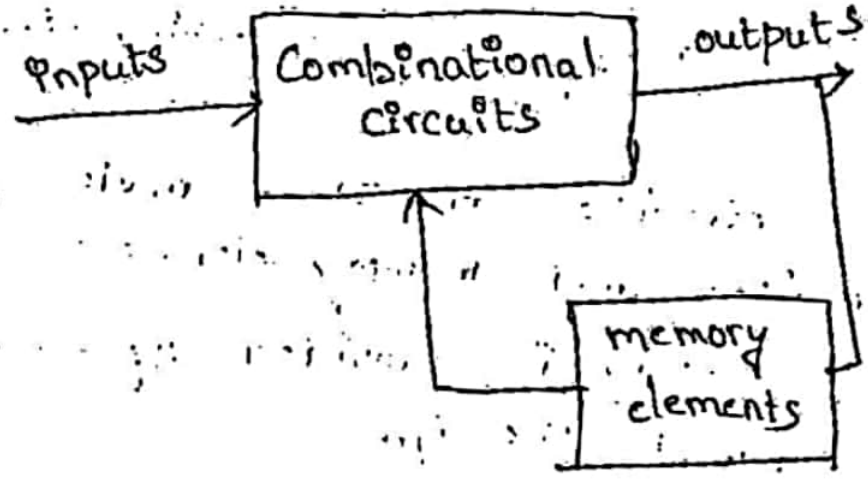
→ This requirement is not satisfied using a combinational logic system.

→ These applications require o/ps to be generated that are not only dependant on the present input conditions, but also upon the past history of the i/ps.

→ Parallel adder, decoder, encoder, subtractors, Code Converters are the example of the Combinational circuits.

→ Counters, shift registers, serial adders, sequence generators, logic function generators are the example of the sequential circuits.

→ The block diagram of sequential circuits is shown in below.



CLASSIFICATION OF SEQUENTIAL CIRCUITS :-

→ The sequential circuit can be classified as

1. Synchronous sequential circuits
2. Asynchronous sequential circuits.

→ The sequential circuits which are controlled by a clock are called "synchronous sequential circuits".

→ These circuits will be active only when clock signal present.

→ The sequential circuits which are not controlled by a clock, are called "asynchronous sequential circuits".

→ The sequential circuits in which events can take place any time the inputs are called "asynchronous sequential circuits".

LATCHES AND FLIP-FLOPS :-

→ The most important memory element is the flip-flop,

→ The main difference bln latchs and flip-flops is in the method used for changing their state.

→ flip-flop for sequential ckts that normally sample its input and change its outputs only at time determined by clocking signal.

SR- Latch :-

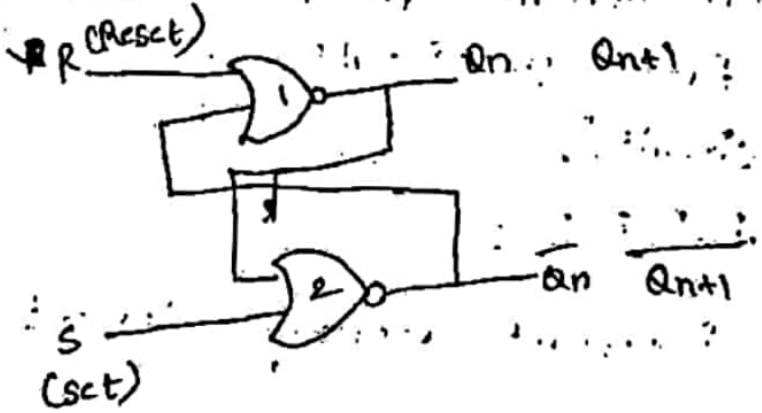
→ The simplest type of latch is the set-reset (SR) latch,

→ It can be constructed from either two NAND gates (or) two NOR gates.

SR-Latch using NOR gate :-

→ The SR Latch using two NOR gates is shown in below figure.

→ The latch has two outputs Q and $\bar{Q}$, and two inputs set and reset.



Truth table for SR latch :-

→ The SR latch inputs of latch is S, R, and Qn

→ when Qn is present state and Qn+1 is next state

S R	PS Qn	N.S Qn+1	
0 0	0	0	(No change)
0 0	1	1	
0 1	0	0	(Reset)
0 1	1	0	
1 0	0	1	(set)
1 0	1	1	
1 1	0	X	(In determined)
1 1	1	X	

Case (i) : if S=0, R=0, then the next state Q_{n+1} is

$$Q_{n+1} = \overline{0 + \bar{Q}_n} = \bar{\bar{Q}}_n = Q_n$$

→ The Q_{n+1} is No change condition

Case :-2 if S=0, R=1 then the next state Q_{n+1} is

where : $Q_{n+1} = \overline{1 + \bar{Q}_n} = \overline{1} = 0$

→ The value of Q_{n+1} is 0 that means reset condition.

→ No need to check the Q_{n+1} condition

Case :- 3

if S=1, R=0, then N.S Q_{n+1} is

$$Q_{n+1} = \overline{0 + \bar{Q}_n} = \bar{\bar{Q}}_n = Q_n$$

→ we need to check $\overline{Q_{n+1}}$ case also then

$\overline{Q_{n+1}} = \overline{1 + Q_n} = \overline{1} = 0$

∴ where $\overline{Q_{n+1}}$ is "0" then Q_{n+1} is value "1" then the condition will be set.

Case :-4

if S=1, R=1 then the Q_{n+1} is

$Q_{n+1} = \overline{1 + \bar{Q}_n} = \overline{1} = 0$

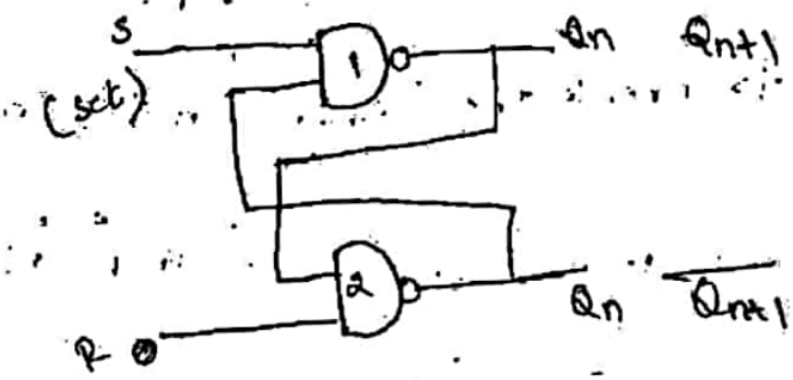
$\overline{Q_{n+1}} = \overline{1 + Q_n} = \overline{1} = 0$

→ This case is pratically not possible, so the condition will be "Indetermined (Id)"

SR-latch using NAND gates :-

→ The SR latch using two NAND gates as shown in below.

→ The latch has two o/p's Q_n and $\overline{Q}_n$ and two inputs S and R. (Set and Reset).



→ The truth table for SR Latch using, NAND gate is shown below.

→ In every Sequential ckt we donot the present state of the circuit, we can consider present inputs.

S	R	P-S Q_n	N-S Q_{n+1}
0	0	X	Id
0	1	X	1
1	0	X	0
1	1	X	Q_n

Here ~~NAND Gate~~ SR Latch using NAND gate is having four cases there are:

Case (i) :- S=0, R=0 then, Q_{n+1} is

$$Q_{n+1} = \overline{0 \cdot \overline{Q_n}} = 1$$
$$\overline{Q_{n+1}} = \overline{0 \cdot Q_n} = 1 \quad \Big\} \text{ ID}$$

→ when, S=0, R=0, the next state is indetermined.

→ Q_{n+1}, $\overline{Q_{n+1}}$ are having the same state it is not possible in pratically.

Case (ii) :- S=0, R=1

$$Q_{n+1} = \overline{0 \cdot \overline{Q_n}} = 1 \quad \Big\} \text{ set}$$
$$\overline{Q_{n+1}} = \overline{1 \cdot Q_n} = 0$$

→ where, 'S=0, R=1, the next state 'Qn+1' is "1" that means it is set condition.

→ Case(iii) :-

when S=1, R=0, Qn+1 = ?

$$Q_{n+1} = \overline{1 \cdot \bar{Q}_n} = 0$$

$$\overline{Q_{n+1}} = \overline{0 \cdot Q_n} ; \bar{0} = 1$$

→ 'Qn+1' (next state) having the '0' then the condition will be Reset condition.

Case(iv) :-

when S=1, R=1, then Qn+1

$$Q_{n+1} = \overline{1 \cdot \bar{Q}_n} = Q_n$$

$$\overline{Q_{n+1}} = \overline{1 \cdot Q_n} = \bar{Q}_n$$

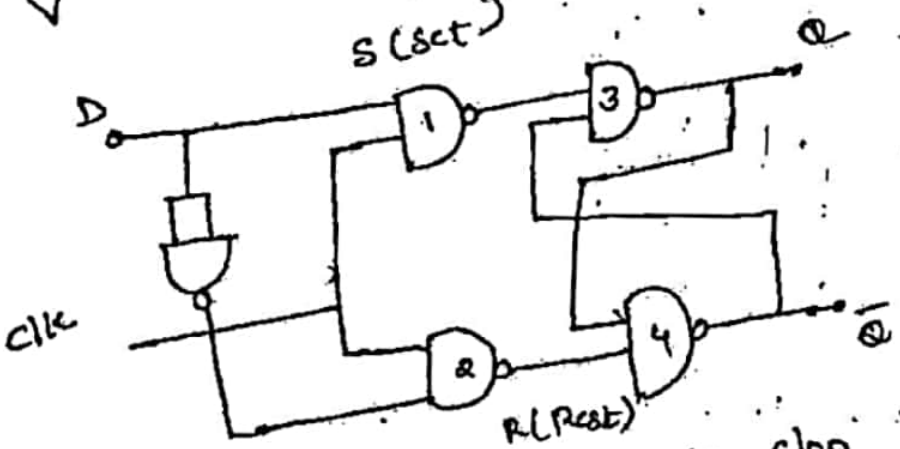
→ when the 'next' state is having same present state then the condition will be "no change" condition.

D-flip-flop :-

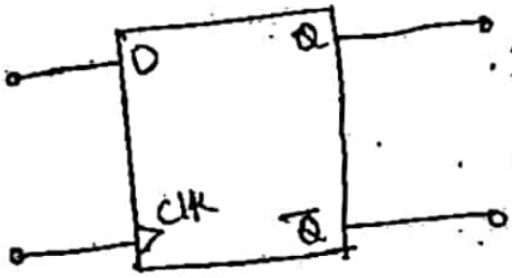
→ The delay (D) flip-flop has only one input called the delay (D) input and two o/p's Q and $\bar{Q}$.

→ It is constructed from an SR-flip-flop by inserting an inverter b/n S and R and assigning the symbol "D" to the S input.

→ The structure of D flip-flop is shown in below figure.



→ The logic symbol for D-flip flop



→ A simple way of building a delay D-flip-flop is shown in below figure.

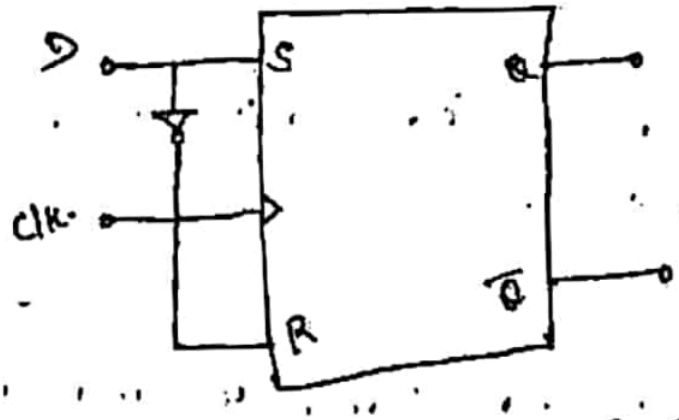


fig :- using SR-flip-flop

→ The truth table of D-flip-flop

Clk	input D	Q_n	Q_{n+1}	
↑	0	x	0	→ Reset
↑	1	x	1	→ set

Case (i) D=0, S=0, R=1

$Q_{n+1} = \overline{1 \cdot \bar{Q}_n} = 0$ (Reset)

$\overline{Q_{n+1}} = \overline{0 \cdot Q_n} = 1$

Case (ii) D=1, R=0, S=1

$Q_{n+1} = \overline{Q \cdot \bar{Q}_n} = 1$ [set]

$\overline{Q_{n+1}} = \overline{1 \cdot Q_n} = 0$

K-map for D-flip flop :-

D \ Q_n	0	1
0	0	0
1	1	1

$Q_{n+1} = D$

→ The next state is having the same input of the "D". Then D-ff is also called transparent ff.

Characteristics & Excitation table for JK

Truth table

clk	J	K	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q_n}$ → toggle

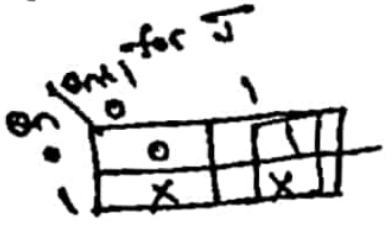
Characteristic table :-

Q_n	J	K	Q_{n+1}
0	0	0	Q_n
0	0	1	0
0	1	0	1
0	1	1	1 → $(\overline{Q_n})$
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0 → $(\overline{Q_n})$

Excitation table :-

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

k-map



$J = \overline{Q_n} Q_{n+1} + Q_n \overline{Q_{n+1}}$

$\boxed{J = Q_{n+1}}$

$K = \overline{Q_n} \overline{Q_{n+1}} + Q_n \overline{Q_{n+1}}$

$= \overline{Q_{n+1}} (Q_n + \overline{Q_n})$

$\boxed{K = \overline{Q_{n+1}}}$

characteristic & Excitation table for D-ff.

Truth table

Clk	D	Qn+1
0	X	Qn
1	0	0
1	1	1

characteric table

Clk	Qn	D	Qn+1
↑	0	0	0
↑	0	1	1
↑	1	0	0
↑	1	1	1

Excitation table

Qn	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

Qn+1 = D

characteristic & Excitation table for T-ff. :-

Truth table :-

Clk	T	Qn+1
0	X	Qn
1	0	Qn
1	1	$\overline{Qn}$

characteristic table

Qn	T	Qn+1	
0	0	Qn	0
0	1	$\overline{Qn}$	1
1	0	Qn	1
1	1	$\overline{Qn}$	0

Excitation table

Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

SR- flip-flop :-

Truth table

Clk	S	R	Qn+1
0	X	X	Qn
1	0	0	Qn
1	0	1	0
1	1	0	1
1	1	1	x Id

Characteristic table.

clk	Qn	S	R	Qn+1
↑	0	0	0	Qn 0
↑	0	0	1	0
↑	1	1	0	1
↑	0	1	1	x
↑	1	0	0	1
↑	1	0	1	0
↑	1	1	0	1
↑	1	1	1	x

Excitation table :-

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

flip flop Conversation :-

Step 1 :-
Identify the avail flip flop, required flip-flop.

Step 2 :-
make the characteristic for required ff

step 3 :-
make the ~~characteris~~ excitation table for avail ff

step 4 :-
make the k-map avail ff

step 5 :-
Implement the expression

SR- to JK- flip flop conversation

Step 1 :-

∴ avail ff - SR

∴ required ff - JK.

Step 2 :-

characteristic table JK :- 3 : Excitation table.

clk	Q_n	J	K	Q_{n+1}
↑	0	0	0	$Q_n = 0$
↑	0	0	1	0
↑	0	1	0	1
↑	0	1	1	$\overline{Q_n} = 1$
↑	1	0	0	$Q_n = 1$
↑	1	0	1	0
↑	1	1	0	1
↑	1	1	1	$\overline{Q_n} = 0$

Q_n	JK	Q_{n+1}	S	R
0	0 0	0	0	X
0	0 1	0	0	X
0	1 0	1	1	0
1	1 1	1	1	0
1	0 0	1	X	0
1	0 1	0	0	1
1	1 0	1	X	0
1	1 1	0	0	1

table for SR ff

SR ff Excitable

S	R	Qn	Qn+1
0	X	0	0
1	0	0	1
0	1	1	0
X	0	1	1

4. k-map for ~~SR~~ JK ff

for R

Qn \ JK	00	01	11	10
0	X (0)	X (1)	0 (3)	0 (2)
1	0 (4)	1 (5)	1 (7)	0 (6)

$Q_{n+1} = Q_n\bar{J}K + Q_n\bar{J}\bar{K}$

$\boxed{Q_{n+1} = Q_nK}$

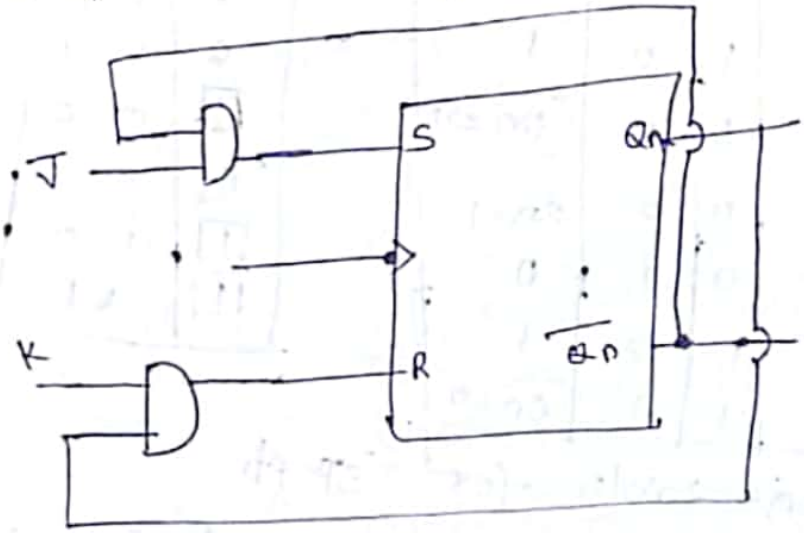
JK For S

Qn \ JK	00	01	11	10
0	0	0	1	1
1	X	0	X	0

$Q_{n+1} = \bar{Q}_n\bar{J}K + \bar{Q}_n\bar{J}\bar{K}$

$\boxed{Q_{n+1} = \bar{Q}_n\bar{J}}$

5. Implement of ~~SR~~ FF By using ~~func~~ Boolean function.



JK ff - SR FF

1. avai - JK

req - SR.

2. Characteristic table for "SR"

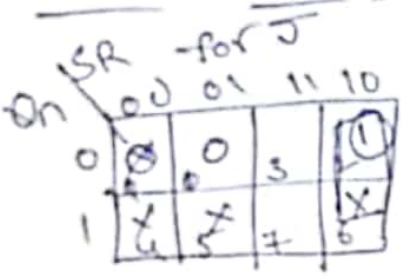
2. characteristic table

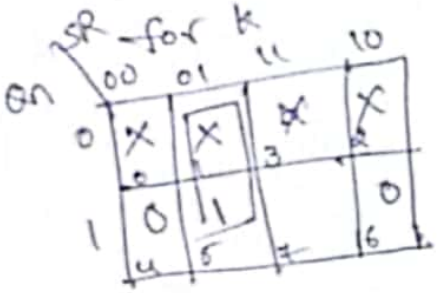
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Id
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Id

3. Excitation table

Q_n	S	R	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	X
0	1	1	Id		
1	0	0	1	X	0
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	X2d		

4. k-map for JK :-





$Q_{n+1} = \bar{Q}_n S\bar{R} + Q_n S\bar{R}$

$\bar{S}_{n+1} = S\bar{R}\,(Q_{n+1} + \bar{Q}_n)$

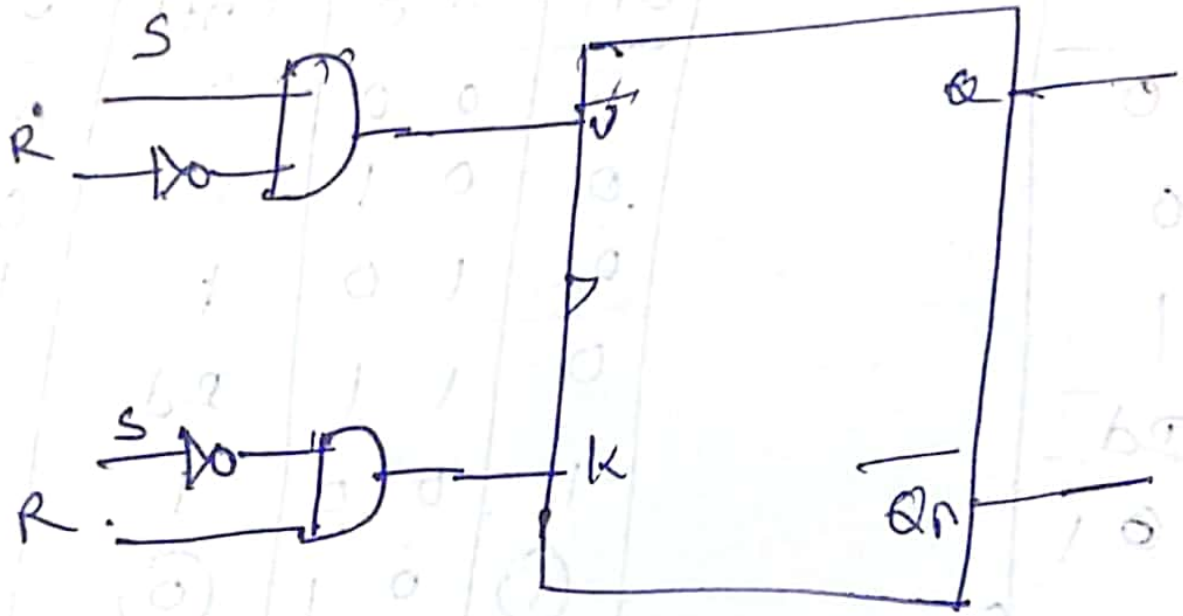
$\bar{J}_{Q_{n+1}} = S\bar{R}$

for K

$Q_{n+1} = \bar{Q}_n \bar{S} R + Q_n \bar{S} R$

$K \Rightarrow Q_n \Rightarrow \bar{S} R$

logic diagram :-



SR flip flop - D - flip flop :-

1. avail flip flop - SR
   required flip flop - D

2. characteristic table for D-flip flop.

clk	Qn	D	Qn+1
↑	0	0	0
↑	0	1	1
↑	1	0	0
↑	1	1	1
↑			

3. Excitation table for SR :-

~~S R~~

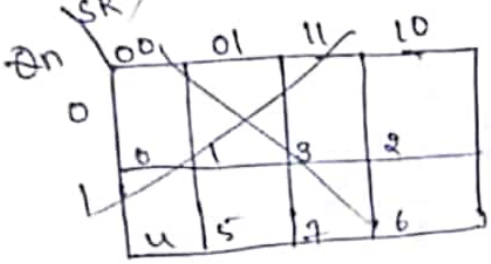
Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

By considering the characteristic table

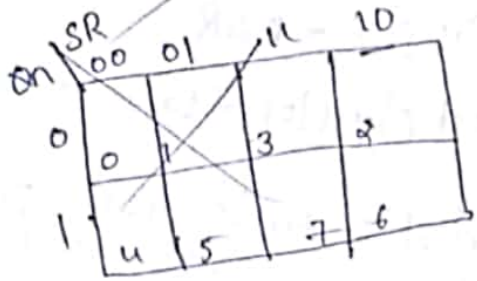
Qn	D	Qn+1	S	R
0	0	0	0	X
0	1	1	1	0
1	0	0	0	1
1	1	1	X	0

i. K-map for above table

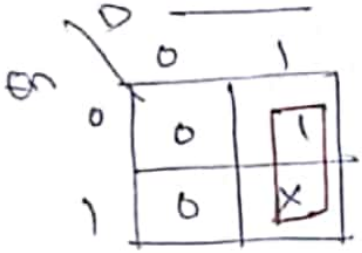
~~for~~ S



for R

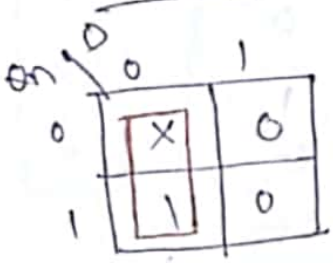


for S



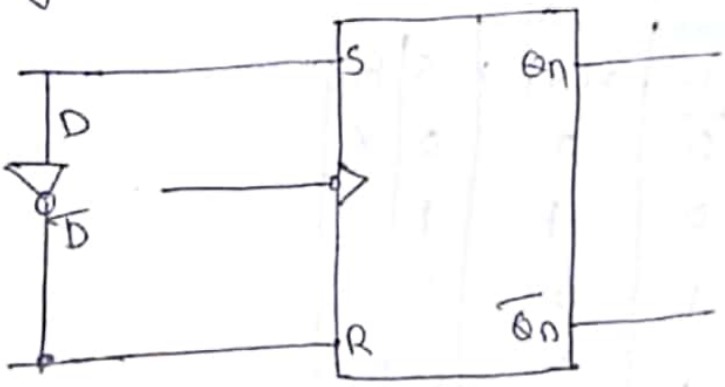
$Q = S = D$

for R



$R = \overline{D}$

Logic diagram :-



SR flipflop - T flipflop

1. available flipflop - SR

   required flipflop - T

2. Characteristic table for T-flipflop.

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

3. Excitation table :-

Qn	T	Qn+1	S	R
0	0	0	0	X
0	1	1	1	0
1	0	1	X	0
1	1	0	0	1

4. K-map simplification.

for S

Qn \ T	0	1
0	0	(1)
1	X	0

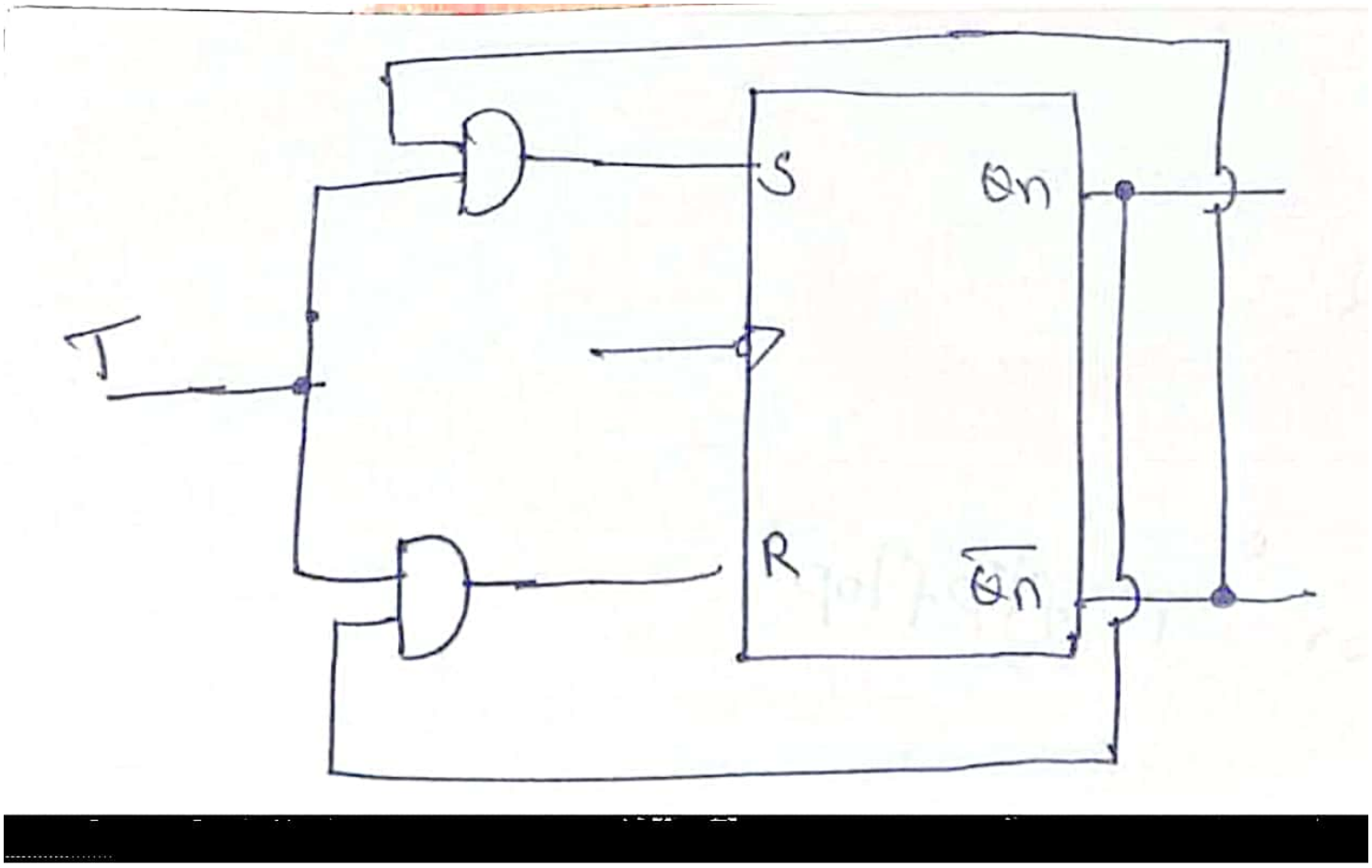
$S = \overline{Qn}T$

for R

Qn \ T	0	1
0	X	0
1	0	(1)

$R = Qn\overline{T}$

5. Logic diagram.



JK flip flop - T flip flop :-

1. avail flip flop - JK

   required flip flop - T

2. characteristic table for T-flip flop

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

JK - Excitation table

Qn	J	K	Qn+1
0	0	X	0
0	X	0	1
1	1	X	0
1	X	1	1

3. Excitation table for JK.

Qn	T	Qn+1	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

4. K-map simplification.

for J

Qn \ T	0	1
1	0	X
0	(1)	X

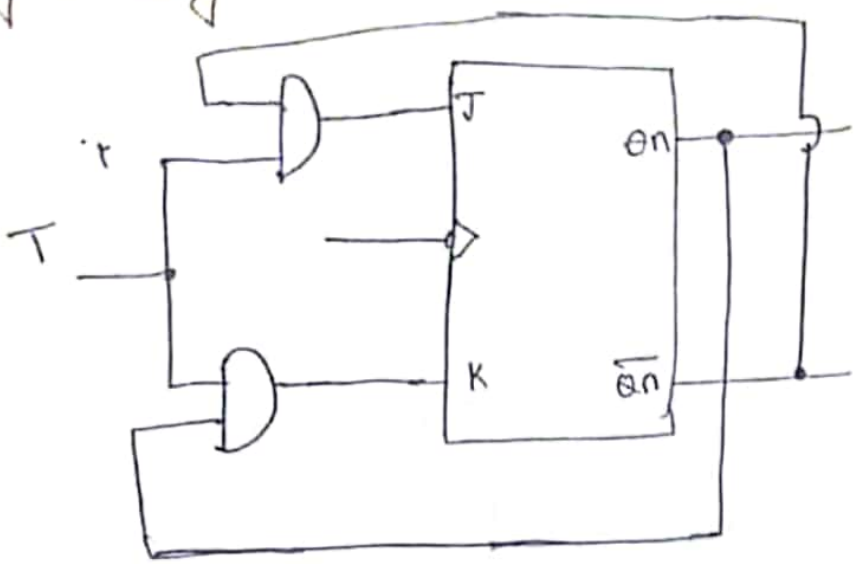
for J = $\overline{Q_n}T$

for K

Qn \ T	0	1
0	X	0
1	X	(1)

K = $Q_n T$

Logic diagram :-



D flip-flop - T-flip flop :-

1. available flipflop - D

required flipflop - T

Truth table T

T	Qn	Qn+1
0	x	Qn
1	x	$\overline{Qn}$

2. Characteristic table for T - flip flop

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

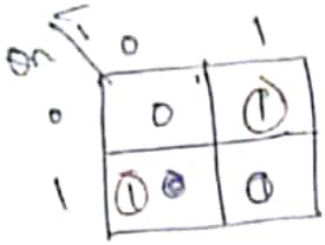
Excitation table D

Qn	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

3. Excitation table for T

Qn	T	Qn+1	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

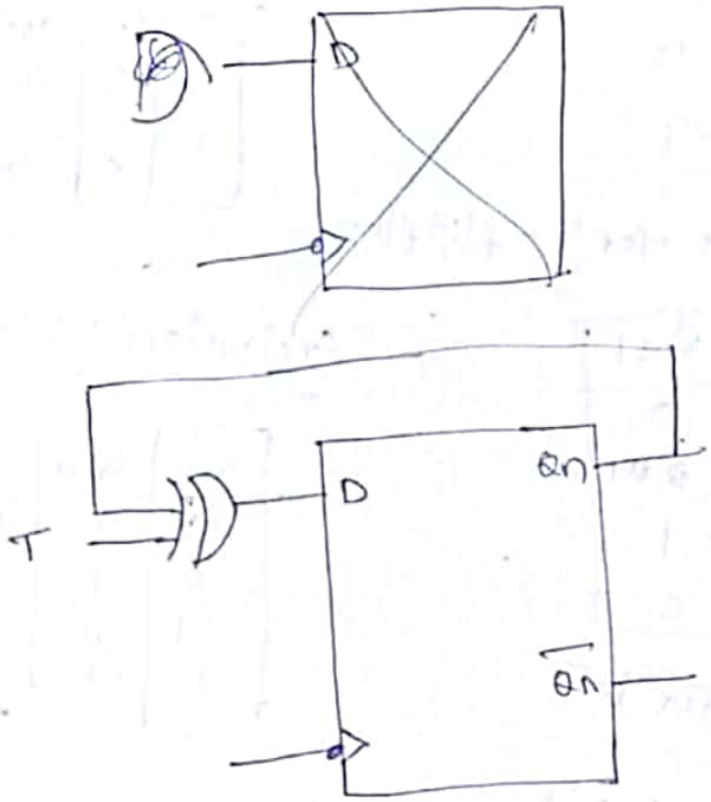
4. Logic k-map for D



$D = Q_n \cdot \overline{T} + \overline{Q_n}\, T$.

$$\boxed{D = Q_n \oplus T}$$

5. Logic diagram :-



JK flip flop - D flip flop :-

1. avail flipflop - JK
   required flip flop - D

2. characteristic table for D

Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

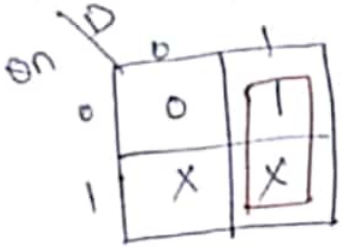
Excitation table for JK

Qn	Qn+1	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

3. Excitation table for D :-

Qn	Qn+1	D	J	K
0	0	0	0	x
0	1	1	1	x
1	0	0	x	1
1	1	1	x	0

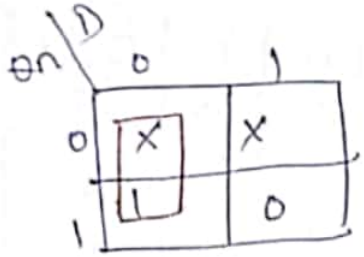
4. K-map for D for J



for J = $\bar{Q}_n D + D\bar{Q}_n$

$\boxed{J = D}$

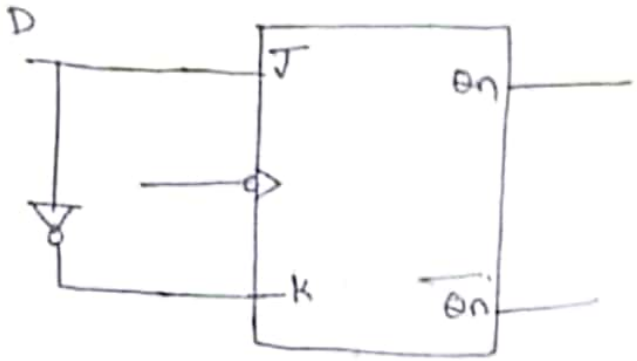
for K



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

$\boxed{K = \bar{D}}$

5. Logic diagram.



T flip flop - D flip flop :-

1. Source - T flipflop
   destination - D flipflop

2. Characteristic table for D

Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

Excitation for T

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

3. Excitation table for D

Qn	D	Qn+1	T
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

4. K-map simplification for D

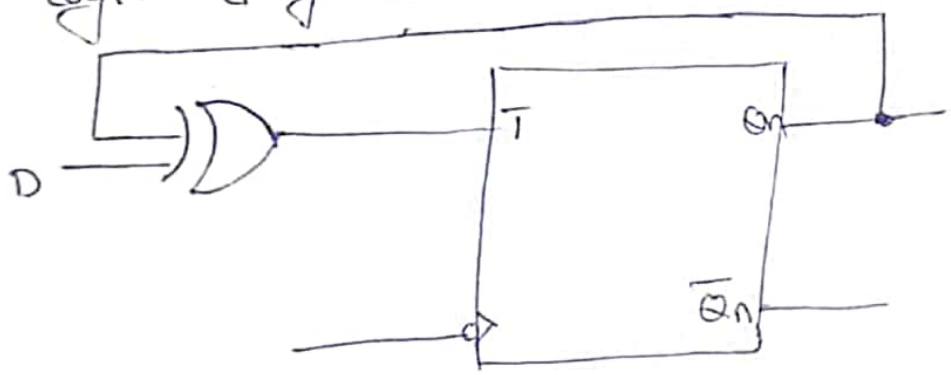
for $\overline{T}$

Q_n \ D	0	1
0	0	1
1	1	0

for $\overline{T}$ $= \overline{Q_n}\, D + Q_n \overline{D}$

$T = Q_n \oplus D$

5. Logic diagram



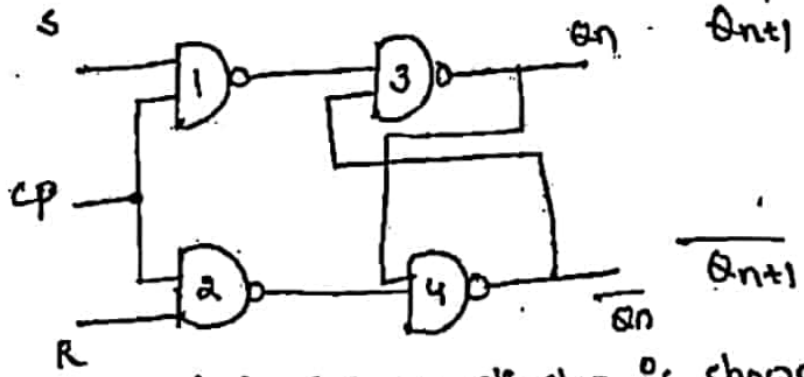
SR flip flop :-

→ The below figure shows the clocked SR-flipflop.

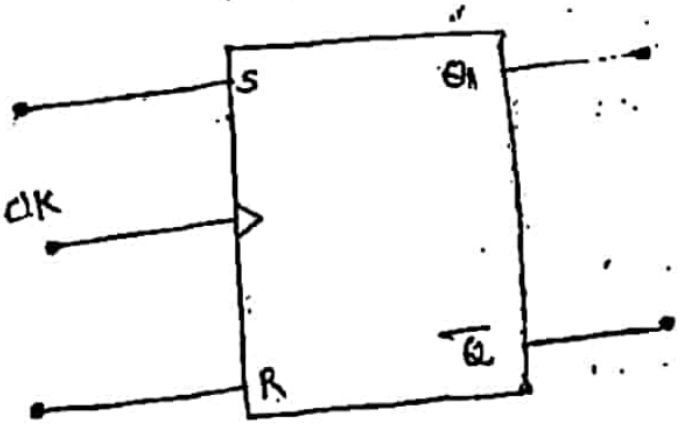
→ The SR flipflop "S" is known as set and "R" knowns as Reset.

→ On the positive Edge of the clock pulse.

→ The ckt responds to the S and R Inputs



→ The logic symbol for SR flipflop is shown in below



→ The characteristic table SR flipflop is shown in below.

Clk	S	R	Qn	Qn+1	
0	0	0	0	0	↔ no change
↑	0	0	0		↔ no change
↑	0	1	0	0	↔ Reset
↑	0	1	1		
↑	1	0	0	1	↔ Set
↑	1	0	1		
↑	1	1	0	Ind	↔ Indetermined
↑	1	1	1		

Case (i) S=0, R=0 Clk=1.

$$Q_{n+1} = \overline{Q_n \cdot \bar{Q}_n} = Q_n$$

$$\overline{Q_{n+1}} = \overline{1 \cdot Q_n} = \bar{Q}_n$$

The Q_{n+1} is in no change condition.

Case (ii) S=0, R=1, Clk=1.

$$Q_{n+1} = \overline{1 \cdot \bar{Q}_n} = Q_n$$

$$\overline{Q_{n+1}} = \overline{0 \cdot Q_n} = \bar{0} = 1$$

when $\overline{Q_{n+1}}$ is "1" then Q_{n+1} value is 0,

∴ the Q_{n+1} is in Reset condition.

Case(iii) S=1, R=0, Clk=1

$Q_{n+1} = 0 \cdot \overline{Q_n} = \overline{0} = 1$

→ The "Q_{n+1}" is in 1 means set condition.

Case(iv) S=1, R=1, Clk=1

$Q_{n+1} = \overline{0 \cdot \overline{Q_n}} = 1$

$\overline{Q_{n+1}} = \overline{0 \cdot Q_n} = 1$

→ Both Q_{n+1} and $\overline{Q_{n+1}}$ value are equal.
→ It is not possible in pratically. so we can say the "Q_{n+1}" is a "Indetermined" condition.

JK flip-flop :-

→ The behaviour of inputs and J and K is same as the S and R inputs of the S-R flip-flop.
→ The letter J stands for set and "K" stands for clear.
→ when the both inputs J and K have a high state, the flip-flop switchy to the complement state. so far a value of Q=1, it switchy to Q=0, and for a value of Q=0, and it switchy to Q=1.

characteristic table for JK.

clk	J	K	Q_{n+1}	
0	X	X	memory	
1	0	0	(0) memory	↔ no change
1	0	1	0	↔ Reset
1	1	0	1	↔ set
1	1	1	Toggle.	

→ Excitation table for JK :-

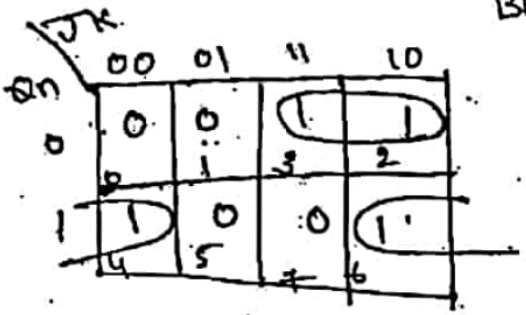
Qn	Qn+1	J	K
0	0	0	0
0	0	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	1	1	0
1	0	1	1

wrong

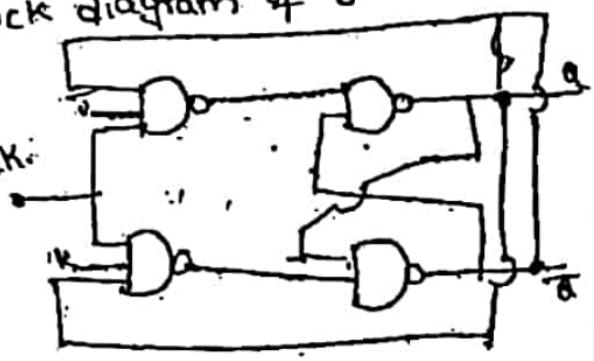
k-map

Qn	J	K	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

K-map for Qn+1.

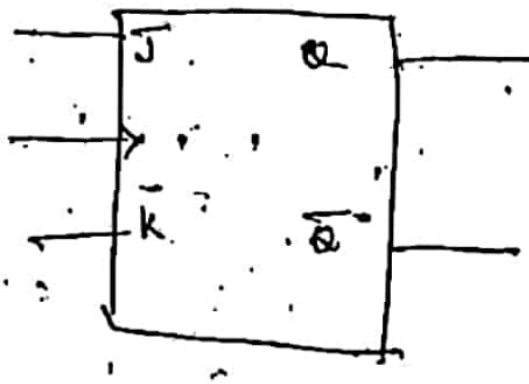


Block diagram of JK :-



$Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$

logic diagram :-



(i) J=0, K=0 (clk=1)

$Q_{n+1} = 1 \cdot \overline{Q_n} = Q_n$

$\overline{Q_{n+1}} = \overline{1 \cdot \overline{Q_n}} = \overline{Q_n}$

no change

(ii) J=0 K=1 (clk=1)

$Q_{n+1} = 1 \cdot \overline{Q_n} = \overline{Q_n}$

$\overline{Q_{n+1}} = 0 \cdot \overline{Q_n} = 1$

$Q_{n+1} = 0$ reset

(iii) J=1 K=0 (clk=1)

$Q_{n+1} = 1$

set

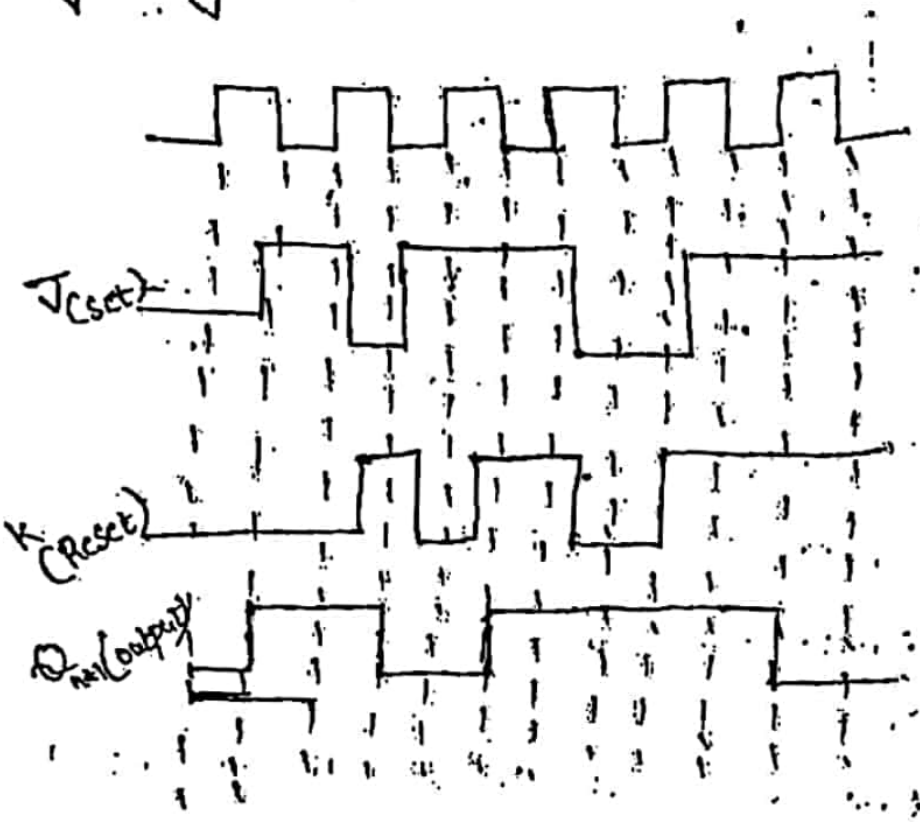
(iv) J=1 K=1

Toggle Condition

$Q_{n+1} = \overline{Q_n}$

$Q_{n+1} = 1$

Timing Diagram :-

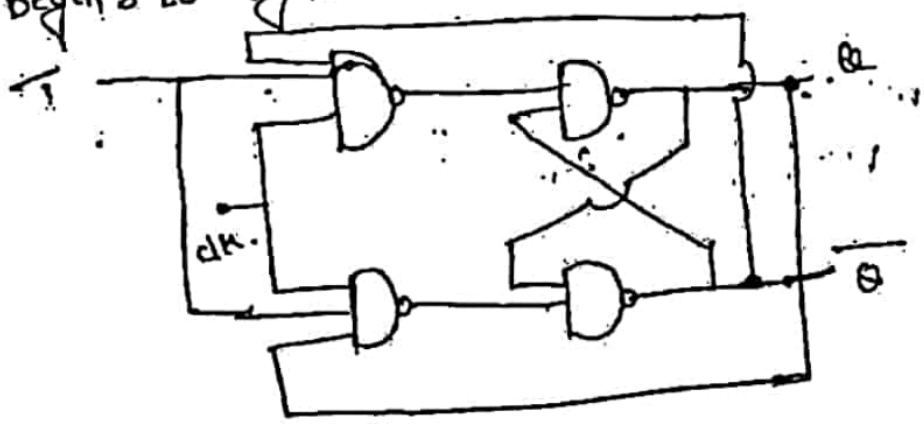


T-flip flop :-

→ This is a much simpler version of the J-K flipflop.

→ Both J and K inputs are connected together and thus are also called a single input J-K flip-flop.

→ when clock pulse is given to the flip flop, output begins to logic.



Excitation table for T-flipflop :-

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

K-map for T :-

Q_n \ T	0	1
0	0	1
1	1	0

$Q_{n+1} = T\cdot\overline{Q_n} + \overline{T}\cdot Q_n$

Case(i) $\overline{T}=0$ Clk=1 (Q_{n+1} is ?,

Ans where $T=0, \overline{T}=0, K=0$

$Q_{n+1} = 1\cdot\overline{Q_n} = Q_n$

$\overline{Q_{n+1}} = \overline{1\cdot Q_n} = \overline{Q_n}$

∴ → where Q_{n+1} is equal to Q_n that means it is a no change Condition.

Case(ii) $\overline{T}=1$, Clk=1, Q_{n+1} = ?

where $T=1, \overline{T}=1, K=1$.

$Q_{n+1} = 0\cdot\overline{Q_n} = 1$, $\overline{Q_{n+1}} = \overline{0\cdot Q_n} = 1$

→ It is not possible in pratically. take the toggle Condition.

$\overline{Q_{n+1}} = \overline{0\cdot Q_n} = 1$

then the next state Q_{n+1} is equal to 0.

MASTER-SLAVE SR FLIP FLOP

→ A master-slave flip-flop is constructed from two flip-flops.

→ One ckt serves as a master and other ckt as a slave. and the overall ckt is referred to as a master-slave circuit.

→ The SR master-slave flip-flop is shown in below figure.

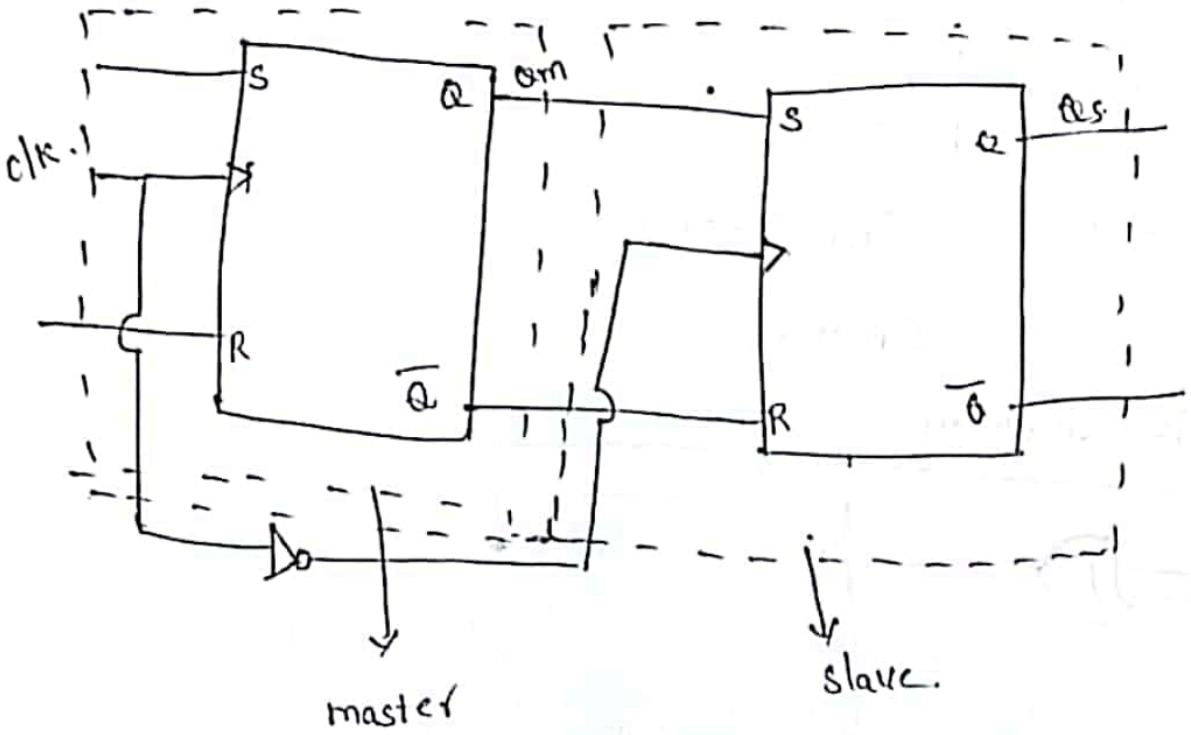
→ It consists of a master flip-flop, a slave flipflop. and an inverter.

→ Both flip-flops are "+"ve level triggered, but.

inverter connected at the clock i/p of the slave flip flop forces it to trigger at the "-"ve level.

(or)

→



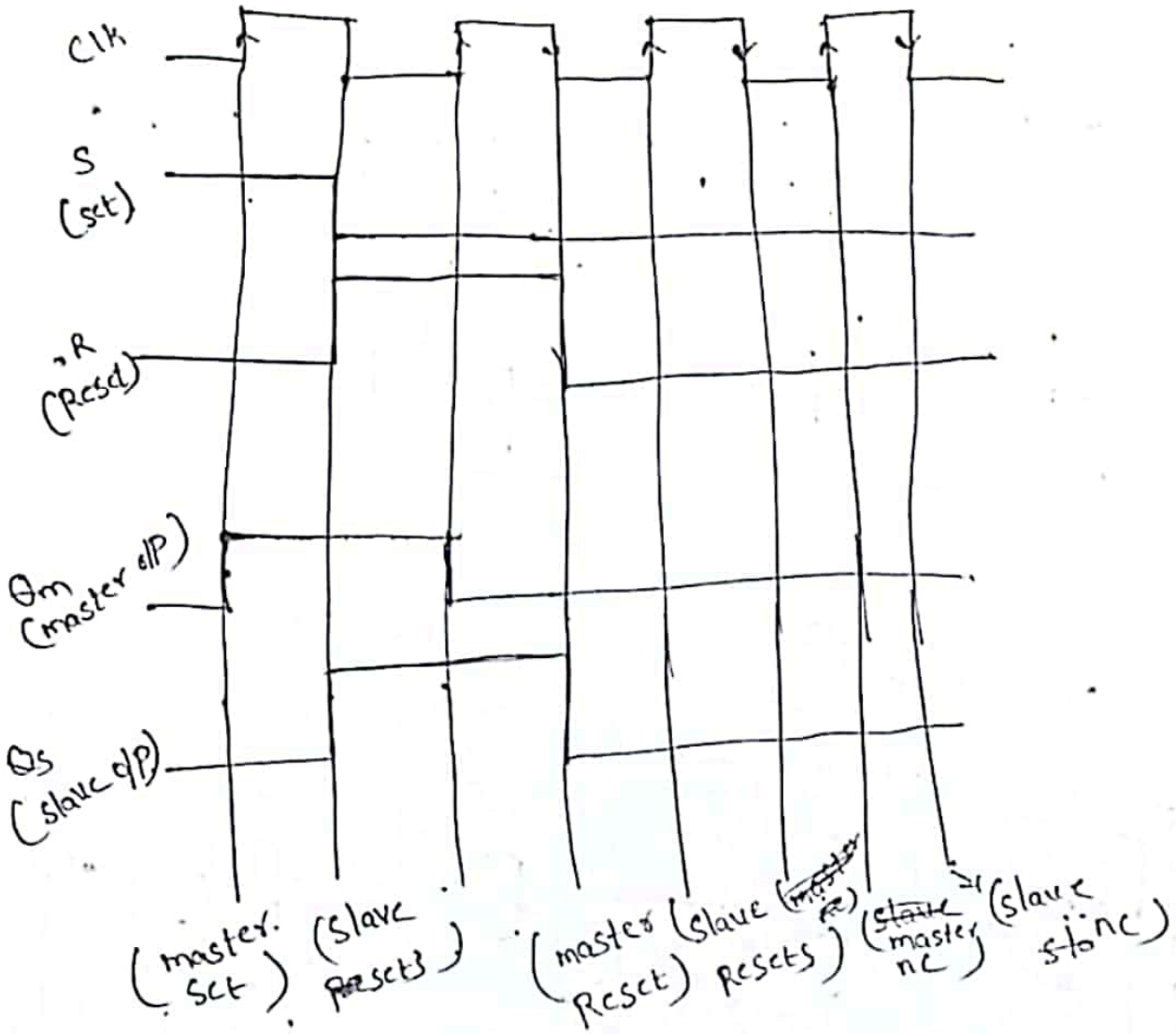
→ The truth table for SR flip flop

Qn	S	R	Qn+1
X	0	0	Qn
X	0	1	0
X	1	0	1
X	1	1	Id.

→ The input and output wave-forms for master-slave flip-flop is shown below.

→ The o/p state of the master flipflop is determined by the S and R inputs at the "+"ve clockpulse.

→ The output state of the master is transferred to the input ~~state~~ of the slave flipflop.



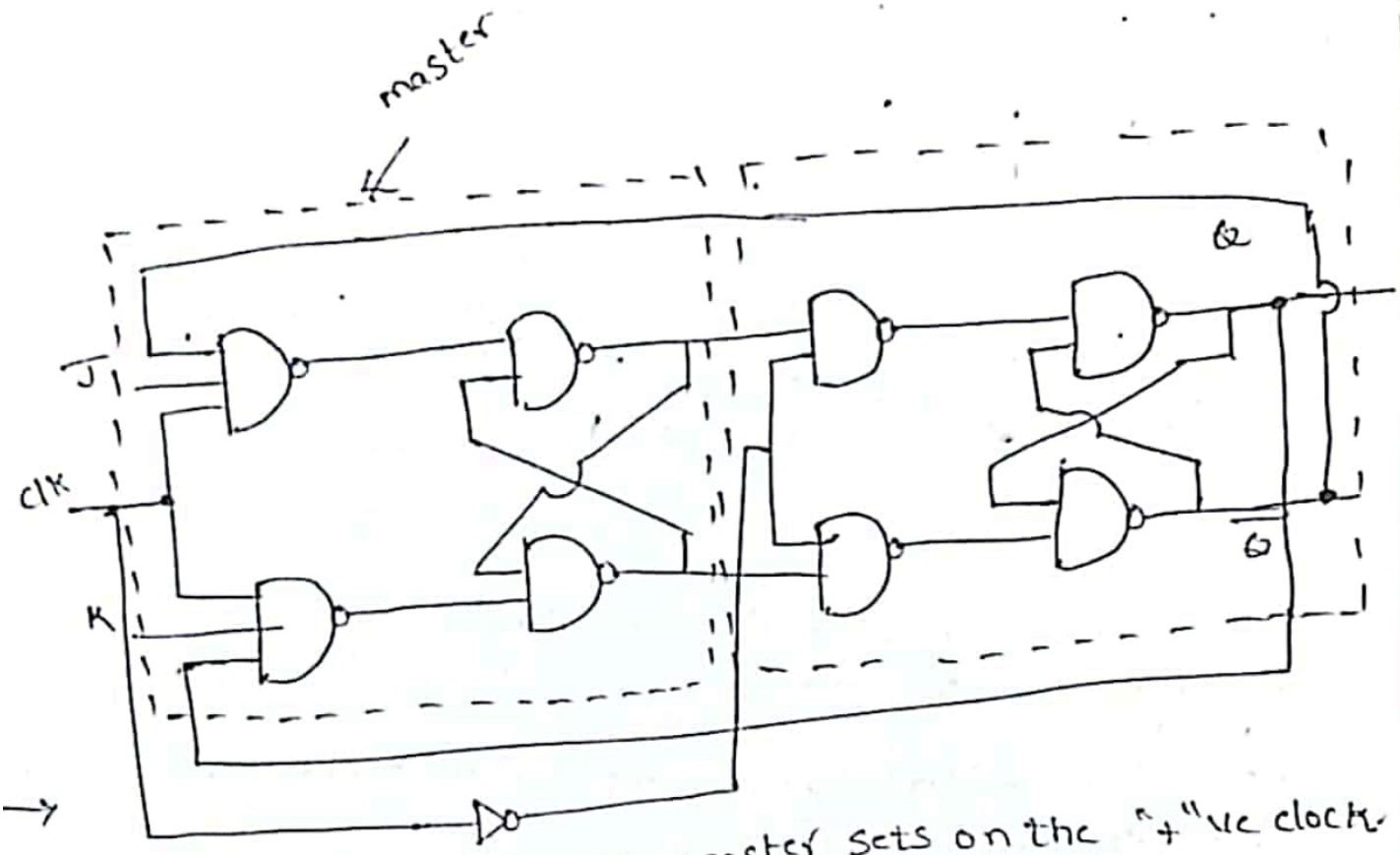
Master Slave JK flipflop :—

→ The master-slave combination can be constructed for any type of flip-flop.

→ The below figure shows one way to build a JK-master slave flip flop.

→ It consists of clocked JK-flip-flop as a master-~~slave~~ and clocked SR-flip-flop as a slave.

→ Clock signal is connected directly to the master flip-flop, but it is connected through inverter to the slave flip-flop.



→ when J=1, and K=0, the master sets on the "+"ve clock.

→ The master o/p is high drives the S input of the slave, so at "-"ve clock, slave sets, copying the action of the master.

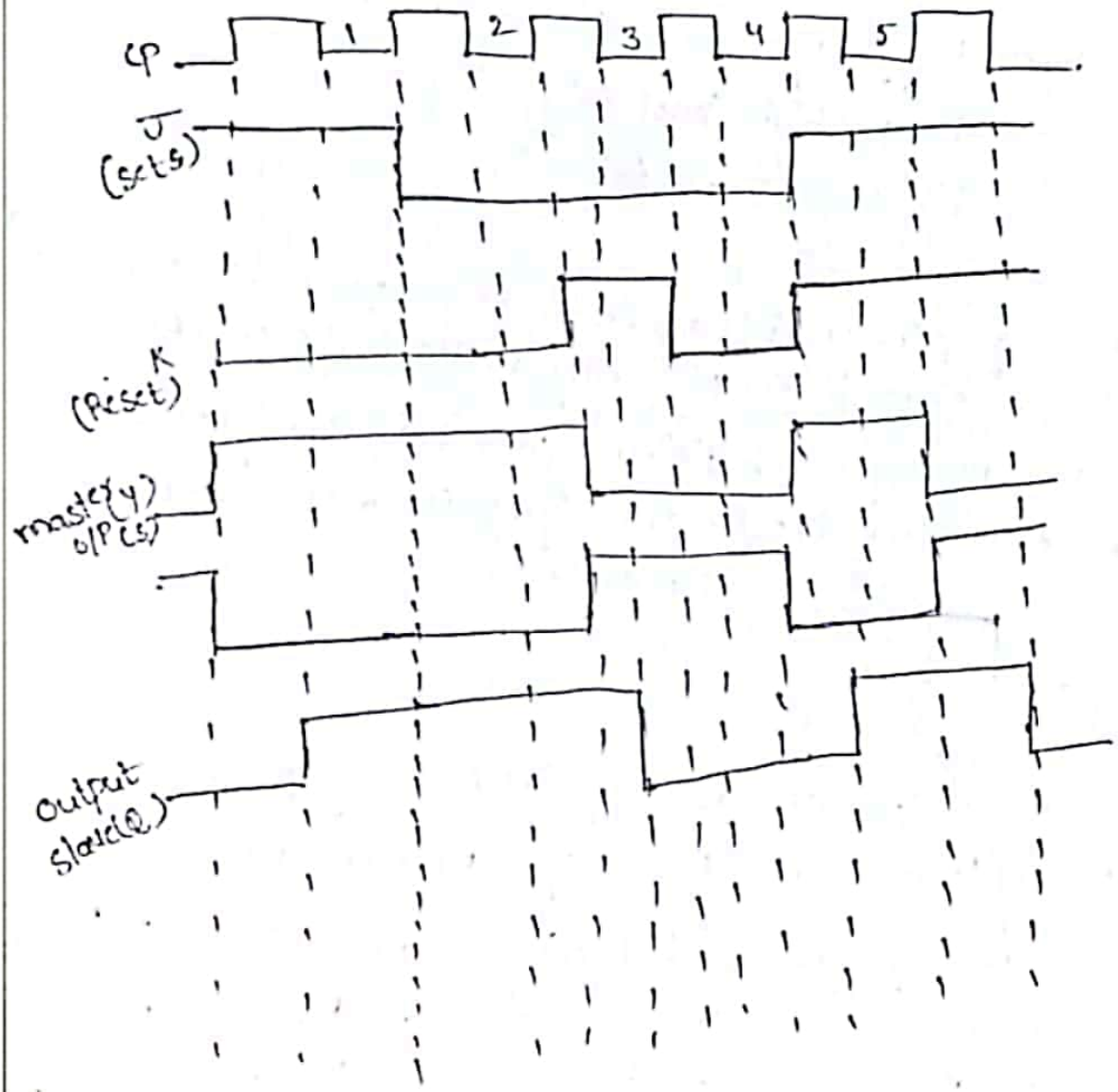
→ When J=0, K=1, the master resets on the +ve clock.

→ The high $\bar{Y}$ o/p of the master goes to the R i/p of the slave.

→ Therefore, at the negative clock slave resets, again copying the action of the master

→ when J=1, K=1, master toggle on the positive clock and slave then copy the master flipflop are complemented, but as it is negative half of the clock pulse master flipflop is inactive.

→ The i/p and o/p waveforms of master-slave JK-ff.



EDGE TRIGGERING IN FLIP-FLOPS :-

→ flip-flops are synchronous bistable devices. The term synchronous means that the change in the o/p occur at specified point on a triggering i/p called the clock.

→ Based on the specific interval (or) point in the clock during (or) at which triggering of flipflop take place, it can be classified into two different typy.
  (i) Level triggering.
  (ii) Edge triggering

level triggering :-

→ The level triggering can classified into two typy
  (i) pe. high level triggering.
  (ii) Low level triggering.

high level triggering :-

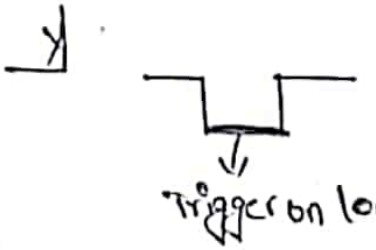
→ when a flip-flop is required to respond at its high state, a "high level triggering method" is used.

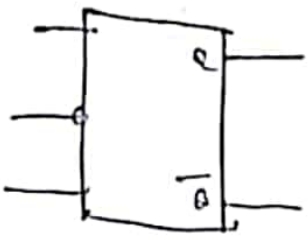
→ It is mainly identified from the straight lead from the clock input. the symbolic representation

low level triggering :-

→ when a flip-flop is required to respond at its low state, "low level triggering method" is used.

→ It is mainly identified from the clock i/p lead along with a low state indicator bubble. The symbolic representation is shown below.





Edge triggering D flip-flop :-

1. Positive Edge triggering D flip-flop
2. negative edge triggering D flip-flop

Positive Edge triggering D-flipflop :-

→ Like an D Latch, D-flipflop the basic SR flipflop is used with complemented inputs.

→ The below figure shows logic symbol and truth table for D-flipflop and input and output waveforms.

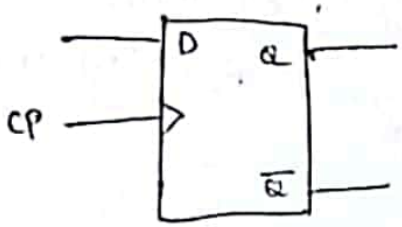
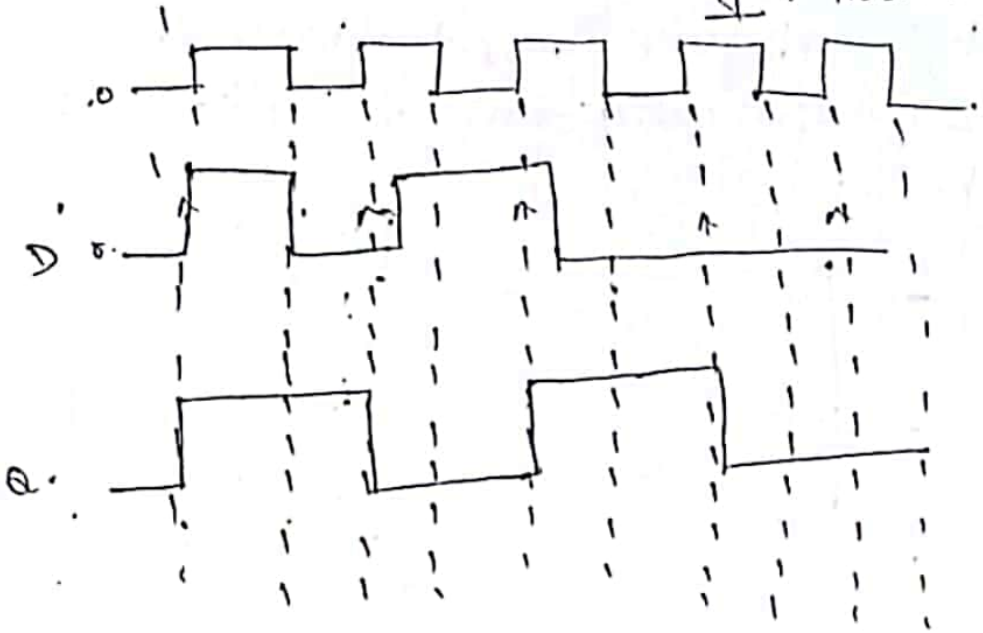


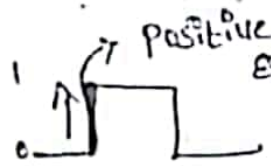
fig :- logic symbol

CP	D	Qn+1
↑	0	0
↑	1	1
0	x	Qn

fig b :- Truth table of D-flipflop



→ The positive edge triggering is to a transistion from low state to high state (0 to 1), Here using the positive edge triggering method.



→ The Characteristic equantion for D flipflop is

$Q_{n+1} = D$ //.

Negative edge triggered D-flipflop :-

→ when the flipflop is respond the transistion from high state to low state, (1 to 0), a negative edge triggering method is used.

→ In case of negative edge triggering, the o/p is sensitive at the negative edge of the clock input.

→ we see the negative edge triggered D-flipflop as an example.

→ ~~The~~ In negative edge triggering D flipflop clock identicate the bubble ~~for the negative~~

→ The logic diagram, truthtable and input and output waveforms of D flipflop is shown below.

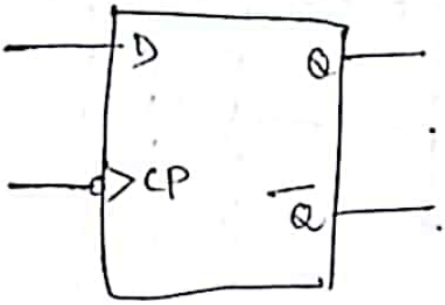
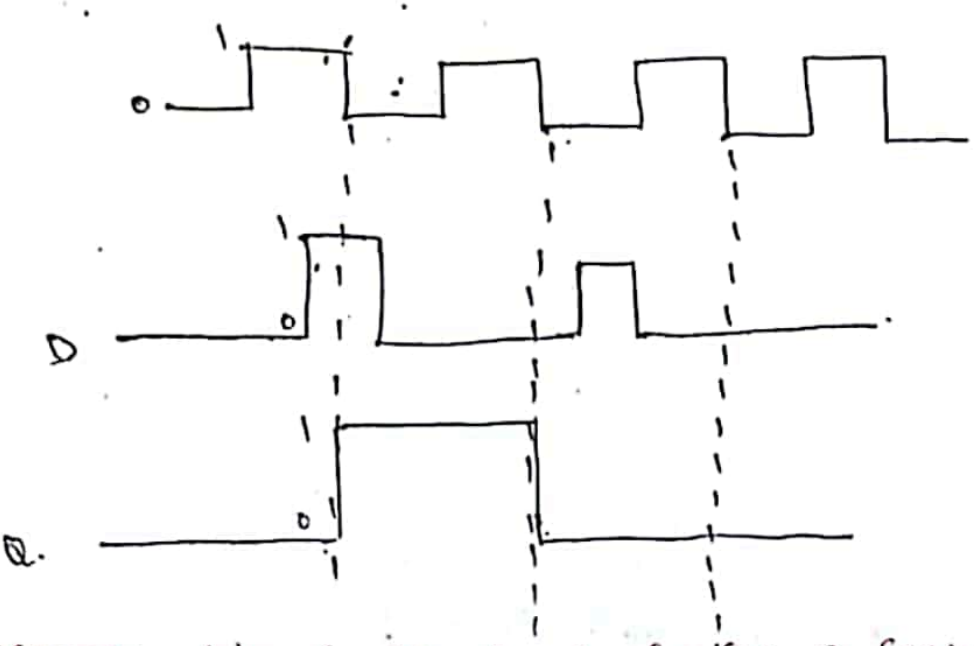


fig:- logic symbol

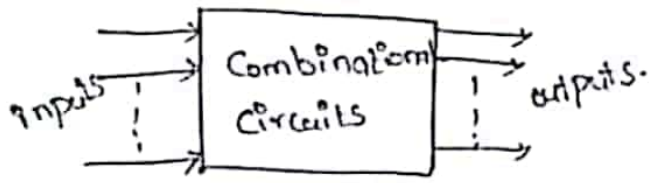
CP	D	Q_{n+1}
↓	0	0
↓	1	1
0	x	Q_n

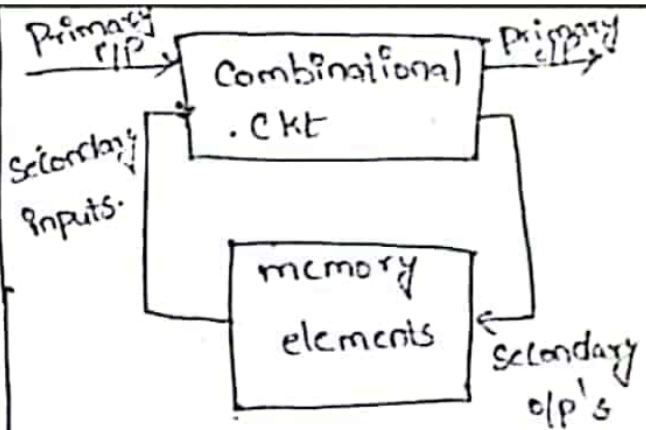
fig:- truth table D flip flop



Difference b/n Combinational circuits & Sequential circuits:-

Combinational circuits	Sequential CKts
1. The ckt whose o/p at any instant depends only on the input present at that instant only is known as combinational circuits.	1. The ckt whose p/p at any instant depends not only on the p/p. present but also on the past o/p a ps known by sequential circuit.
2. This type of ckt has no memory unit.	This type of ckt has memory unit for store past output.
3. Example of Combinational circuits are half adder, full adder, comparator, mux, demux etc.	Example of sequential CKts are flip flops, register, Counters etc....
4. faster in speed	slower compared to Combinational circuits.





Applications of flip-flops :-

→ flip flops find wide applications in Counter circuits, frequency dividers, shift and storage registers.

→ A wide variety of serial decoding, comparision and timing functions can be accomplished using flip-flops.

→ An operation that occurs very frequently in digital system is the transfer of information from one flipflop (or) group of bi flipflops to another flipflop.

—— :. End :. ——

👉 UNIT 2 - THE END 👈