Boolean Algebra.

Set: A set is a collection of well-defined objects.
↓
Property should be defined properly.

⇒ Coversion of Grey code to Binary code;-

Gray code → $g_3\ g_2\ g_1\ g_0$.
Binary code → $b_3\ b_2\ b_1\ b_0$.

$b_3 = g_3$
$b_2 = b_3 \oplus g_2$
$b_1 = b_2 \oplus g_1$
$b_0 = b_1 \oplus g_0$

ex→ 1001

$b_3 = 1$
$b_2 = 1 \oplus 0 = 1$
$b_1 = 1 \oplus 0 = 1$
$b_0 = 1 \oplus 1 = 0$.

~~Grey code to binary~~.
Binary to Gray code:-

1 1 0 1 1 1 (Discarding carries)
↓
1 0 1 1 0 0

(a) 10111011
11000100 → corresponding binary.
↳
$2^7 + 2^6 + 2^2 = 128 + 64 + 4 = \underline{196}$

⇒ Weighted code: This obey position weighting principle, which states the the position of each number represent a specific weight.
In these codes each decimal digit is represented by group of bits.
ex- BCD, 2421, 5211
Non-weighted code: In this, there are no specific weights assigned to the bit position. Gray code is a non-weighted code & it is not arithmetic codes.
ex- excess 3 code.

Scanned by CamScanner

⇒ Minimization of Boolean exp:- (1) Boolean Algebra
(2) K-Map (can handle don't care condition)

⇒ De-Morgan's Law :- (To find complement of a complicated exp)

a) $\overline{A_1 + A_2 + \cdots + A_n} = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdots \bar{A}_n$

b) $\overline{A_1 \cdot A_2 \cdot A_3 \cdots A_n} = \bar{A}_1 + \bar{A}_2 + \cdots + \bar{A}_n$

⇒ i) $A + BC = (A+B)(A+C)$

ii) $(A+B)(A+C) = A + BC$

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

iv) $\bar{A} + AB = (\bar{A} + A)(\bar{A} + B) = \bar{A} + B$

v) $XY + \bar{X}Y\,WZ = (XY + \bar{X}Y)(XY + WZ) = XY + WZ$

Q: $V + \bar{V}W + \bar{V}\bar{W}X + \bar{V}\bar{W}\bar{X}Y + \bar{V}\bar{W}\bar{X}\bar{Y}Z$

$V + \bar{V}(W + \bar{W}X + \bar{W}\bar{X}Y + \bar{W}\bar{X}\bar{Y}Z)$ $\quad [A + \bar{A}B = A + B]$

$V + W + \bar{W}X + \bar{W}\bar{X}Y + \bar{W}\bar{X}\bar{Y}Z$

$= V + W + \bar{W}(X + \bar{X}Y + \bar{X}\bar{Y}Z)$

$= V + W + X + \bar{X}Y + \bar{X}\bar{Y}Z$

$= V + W + X + \bar{X}(Y + \bar{Y}Z)$

$= V + W + X + Y + \bar{Y}Z$

$= V + W + X + Y + Z$

Scanned by CamScanner

Boolean Algebra.

Set: A set is a collection of well-defined objects.
↓
Property should be defined properly.

⇒ Coversion of Gray code to Binary code;-

Gray code → $g_3\ g_2\ g_1\ g_0$.
Binary code → $b_3\ b_2\ b_1\ b_0$.

$b_3 = g_3$
$b_2 = b_3 \oplus g_2$
$b_1 = b_2 \oplus g_1$
$b_0 = b_1 \oplus g_0$

ex→ 1001

$b_3 = 1$
$b_2 = 1 \oplus 0 = 1$
$b_1 = 1 \oplus 0 = 1$
$b_0 = 1 \oplus 1 = 0$

~~Gray code to binary~~.
Binary to Gray code:-

110111 (Discarding carries)
↓
101100

(a) 10111011
11000100 → corresponding binary.
↳
$2^7 + 2^6 + 2^2 = 128 + 64 + 4 = \underline{196}$

⇒ Weighted code: This obey position weighting principle, which states the the position of each number represent a specific weight.
In these codes each decimal digit is represented by group of 4 bits.
ex- BCD, 84-2,-1
Non-weighted code: In this, there are no specific weights assigned to the bit position. Gray code is a non-weighted code & it is not arithmetic codes.
ex- excess 3 code.

Scanned by CamScanner

⇒ Minimization of Boolean exp:- (1) Boolean Algebra
(2) K-Map (can handle don't care condition)

⇒ De-Morgan's Law :- (To find complement of a complicated exp)

a) $\overline{A_1 + A_2 + \cdots + A_n} = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdots \bar{A}_n$

b) $\overline{A_1 \cdot A_2 \cdot A_3 \cdots A_n} = \bar{A}_1 + \bar{A}_2 + \cdots + \bar{A}_n$

⇒ i) $A + BC = (A+B)(A+C)$

ii) $(A+B)(A+C) = A + BC$

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

iv) $\bar{A} + AB = (\bar{A} + A)(\bar{A} + B) = \bar{A} + B$

v) $XY + \bar{X}\bar{Y}WZ = (XY + \bar{X}\bar{Y})(XY + WZ) = XY + WZ$

Q: $V + \bar{V}W + \bar{V}\bar{W}X + \bar{V}\bar{W}\bar{X}Y + \bar{V}\bar{W}\bar{X}\bar{Y}Z$

$V + \bar{V}(W + \bar{W}X + \bar{W}\bar{X}Y + \bar{W}\bar{X}\bar{Y}Z)$ $\quad [A + \bar{A}B = A + B]$

$V + W + \bar{W}X + \bar{W}\bar{X}Y + \bar{W}\bar{X}\bar{Y}Z$

$= V + W + \bar{W}(X + \bar{X}Y + \bar{X}\bar{Y}Z)$

$= V + W + X + \bar{X}Y + \bar{X}\bar{Y}Z$

$= V + W + X + \bar{X}(Y + \bar{Y}Z)$

$= V + W + X + Y + \bar{Y}Z$

$= V + W + X + Y + Z$

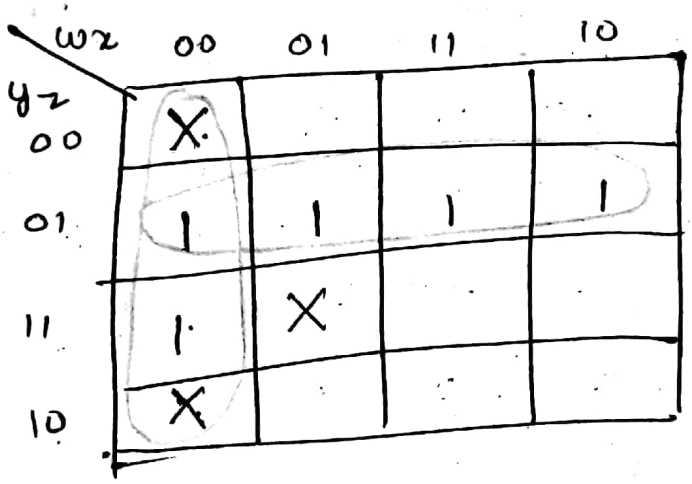
Scanned by CamScanner

⇒ K-Map with Don't Care condition:-

Don't care condition arises when function is not-specified for certain combination of variables.

A don't care minterm is a combination of variables whose logical value is not-specified. Ex.

Ex. $F(w, x, y, z) = \sum(4, 5, 6, 7, 12)$ with don't care fun^{n}
c $d(w, x, y, z) = \sum(0, 8, 13)$



$F = \bar{y}z + \bar{w}\bar{x}$

* We can treat- X as 0 or 1 for sake of simplicity of Boolean function.

Scanned by CamScanner

⇒ Postulates of Boolean Algebra :-

i) $x + x' = 1$

ii) $x \cdot x' = 0$

Principle of Duality :- Every algebraic expression deducible from the postulates of boolean algebra remains true if the operators & identity elements are interchanged.

Theoram : $x \cdot x = x$

$$\Rightarrow x \cdot x = x \cdot x + 0$$
$$= x \cdot x + x \cdot x'$$
$$= x(x + x')$$
$$= x \cdot 1$$
$$= x$$

Theoram : $x + 1 = 1$

$$\Rightarrow x + 1 = 1 \cdot (x + 1)$$
$$= (x + x')(x + 1)$$
$$= x + x' \cdot 1$$
$$= x + x'$$
$$= 1$$

By principal of duality

$$\boxed{x \cdot 0 = 0}$$

Theoram : $x + xy = x$

& $x \cdot (x + y) = x$.

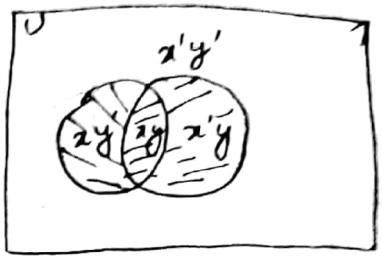
Scanned by CamScanner

⇒ Operator Precedence :-

1) Parenthesis
2) NOT (complement)
3) AND
4) OR

Decreasing Precedence

⇒ Venn Diagrams :-



⇒ Boolean function :-

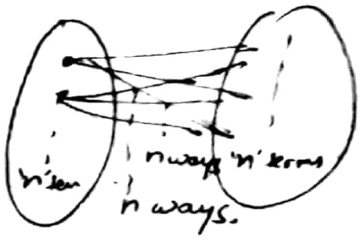
n^n → function ; 2^{n^2} → Relation.

2^n → functions with n variables.

Relation : Any subset of A×B is relation

Function : A relation satisfying property of fun^n.

x	y	z	$F_2 = x + y'z$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



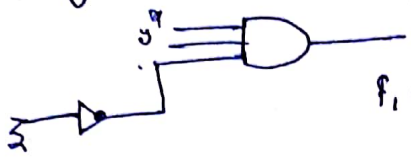
n×n×n - - - n times.
hence n^n functions.

Scanned by CamScanner

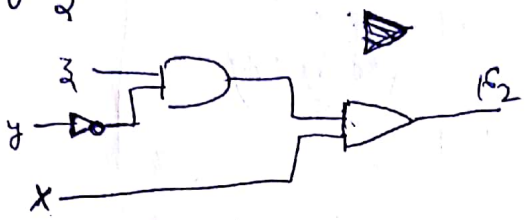
⇒ Implementation of Boolean Function :—

"Hamsab Bhartiya Hain!"

1. With gates



$f_2 = x + y'z$



⇒ **Literal** : A literal is a primed or unprimed variable.

(i) $x(x'+y) = xx' + xy = 0 + xy = xy$

(ii) $x + x'y = (x+x')(x+y) = x + y$.

(iii) $xy + x'y + yz = y(x + x' + z)$

$= y(1+z)$

$= yz$

(iv) $xy + x'z + yz = xy + x'z + yz(x+x')$

$= xy + x'z + xyz + x'yz$

$= xy(1+z) + x'z(1+y)$

$= xy + x'z$.

Scanned by CamScanner

⇒ Complement of a function : $(A'BC' + A'B'C)' = (A'BC')'(A'B'C)'$

$= (A+B'+C).(A+B+C') = A + AB + AC' + B'A + B'C' + CA + CB + CC'$

$= A + A(B+B'+C+C') + B'C' + BC = A + BC + B'C'$

⇒ Minterms & Maxterms (P.O.S)

↓ (Sum of Products).

For two variables → $x'y, xy', x'y', xy$.

For three variables → $x'y'z', x'y'z, x'y'z, x'yz,$

$xy'x', xyz', xyz, xy'z$.

For 'n' variables → 2^n minterms.

→ Complement of minterm is maxterm.

here, $x' \rightarrow 1$

$x \rightarrow 0$

→ Any boolean exp can be represented as standard form.

→ Standard Exp.

i) Sum of Minterms

ii) Product of maxterm.

→ $F(A,B,C) = \Sigma(1,4,5,6,7)$ → minterms.

1 →	001	$x'y'z$
4 →	100	$xy'z'$
5 →	101	$xy'z$
6 →	110	xyz'
7 →	111	xyz

→ $F = \Sigma(1,3,6,7)$

$F = \Pi(0,2,4,5)$

} Conversion from one form to another.

Scanned by CamScanner

→ Subtraction with complement–

two n-digit unsigned number M & N base 'r'

for M − N

① Add M to r's complement of N

r's complement of $N = r^n - N$

$\therefore (M + r^n - N)$

② If $M \geq N$, the sum will produce an end carry r^n which is discarded.

③ If $M < N$, sum does not produce an end carry, is equal to

$$r^n - (N - M)$$

which is r's complement of N − M. To obtain the answer in the familiar form, take the r's complement of the sum and place −ve sign.

Q: Find 72532 − 3250, using 10's complement

M = 72532

N = 03250

10's complement of N = $10^5 - 03250$

9's complement = 99999 − 03250 = 96749

10's complement = 96749 + 1 = 96750

Now, 72532 + 96750

= 169282

$= 169282 - 10^5$

$= \underline{69282}$.

Ans

Q2 Find $3250 - 72532$

10's complement $= 99999 - 72532 + 1$

Step 1:- $M = 3250$

$N = 72532 \rightarrow 27467$

$+1$

10's comp $\rightarrow 27468$

Now, $3250 + 27468$

$= 30718$

Step 2; Now, 10's complement of $\begin{array}{r} 99999 \\ 30718 \\ \hline 69281 \\ +1 \\ \hline 69282 \end{array}$

Step 3: Putting -ve sign $\rightarrow -69282$.

Q. $X = 1010100$ $\quad 64+4+2 = 84$

$Y = 1000011 \quad 64+3 = 67$

$84 - 67 = 17$

$67 - 84 = -17$

Perform the subtraction using two's complement-

(i) $X - Y$ (ii) $Y - X$.

(i) 2's complement of $Y = 0111100$

$+1$

0111101

2's comp $\rightarrow 0111101$

$$\begin{array}{r} 1010100 \\ +0111101 \\ \hline 10010001 \end{array}$$

$= (0010.001)_2 = (17)_{10}$

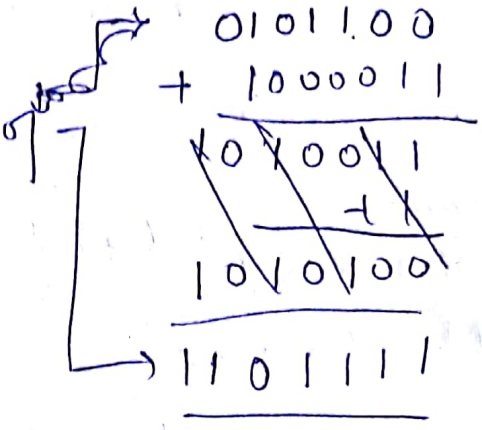
Scanned by CamScanner

(11) Y - X

2's complement of X = 0101011

+ 1

0101100



2's complement of

1's complement - 0010000

+1

$-(00100\ 01) =$

⇒ Signed Binary No's :-

Signed magnitude Notation → 1st bit represents rest represent- magnitude.

1. signed 1's complement :

+9 → 01001

-9 → 10110

2. Signed . 2's complement :

+9 → 01001

10110

+1

-9 → 10111

⇒ In 8 bits signed 1's complement- representation of -9

+9 : 00001001

1's complent, -9 : 11110110

246

Scanned by CamScanner

Signed two's complement of -9

```
  1111 0110
      +1
  ---------
  1111 0111
```

⇒ Binary Codes :

With 'n' bits of 0 & 1, how many diff. patterns are possible -

2^n patterns are possible

① → BCD : Binary coded decimal : Coding of decimal no. in binary

12 → 0001 0010

→ Total no. of bits = (no. of digits * 4)

② → Excess 3 code :- adding 3 to binary equivalent.

```
0 —  0011
1 —  0100
2 —  0101
:     :
:     :
6    1100
```

③ 84 - 2 = 1

2 → 0110 → 4×1 - 2×1 = 2

1 → 0011 → 4×1 - 2×1 - 1×1 = 1

④ 2421 →

```
0000
0001 → 2
0010
1000 → 2
```

→ When internal arithmetic in computer is done in decimal form, then BCD is used.

```
  12   00010010
+13    00010011
```

Scanned by CamScanner

⇒ Advantages of excess 3, 2421 & 84-2-1

The excess3, 2421 and 84-2-1 are self complementing codes..

9's complement of a number is obtained by changing 1's to 0's & 0's to 1's.

→ ex-

1001 → 9 (Binary) , -1 (signed magnitude representation)

9 (BCD) 6 (Excess 3) 7 (84-2-1) 9 (2421)

0110 → 6 ⇒ -6 (signed 1's complement method)

+1

0111 → 7 ⇒ -7 (Signed 2's complement method).

→ 93

-18 (M-N)

N → 18 81 +1

10's complement → 82 ~~90~~ 10010

23

82

105

Q. 84-2-1:

5 1011

-4 0100

9's complement → 0100 → 0101 → 10's complement

1011 +100

10001

→ discarding carry ⇒ 0001 = 1.

Scanned by CamScanner

⇒ Error Detection code :-

Parity scheme

Parity → no. of ones.

Odd parity	Parity
0000	1
0101	1

even parity	Parity
0000	0
0001	1

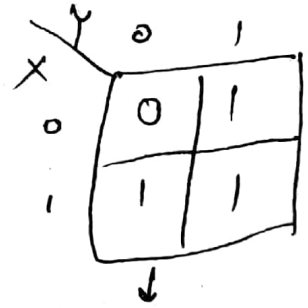
⇒ Gray Code :- Only one bit varies b/w successive numbers.

Gray Code	Decimal Equivalence
0000	0
0001	1
0010	2
0011	3
0110	4
0111	5
0101	6

⇒ Two & Three variable Maps :-

$F_1 = x + y$

x	y	x+1
0	0	0
0	1	1
1	0	1
1	1	1



K map rep of x+y.

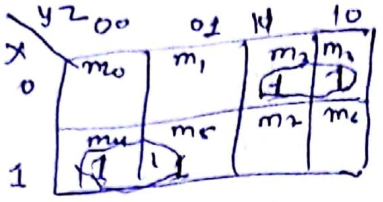
Scanned by CamScanner

→ combinations are made in '2' powers → 2, 4, 8 ...

⇒ 3-variable :-

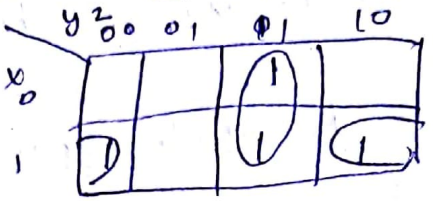
Q1. $f(x, y, z) = \Sigma(2, 3, 4, 5)$

$= m_2 + m_3 + m_4 + m_5$



$x'y + y'x = x \oplus y$

Q2. $f(x, y, z) = \Sigma(3, 4, 6, 7)$



$\Rightarrow yz + x\bar{z}$

20/08/18.

⇒ Prime Implicant : A prime implicant is a product-term obtained by combining the maximum possible no. of adjacent squares in the map. A single 1 on a map represents a prime implicant if it is not adjacent to any 1.

If a minterm in a square is covered by only one prime implicant in that implicant is said to be essential prime implicant.

$f(A, B, C, D) = \{0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15\}$

Prime Implicants: CD, $B'C$, AD, AB', BD, $B'D'$.

$B'D'$ → Essential.

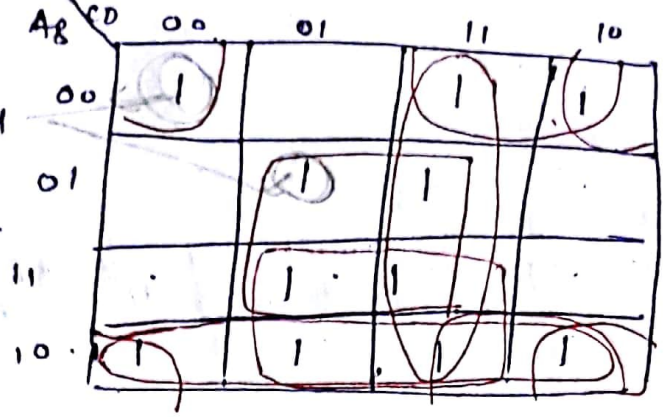
$f = BD + B'D' + CD + AD$

$= BD + B'D' + CD + AB'$

$= BD + B'D' + B'C + AD$

$= BD + B'D' + B'C + AB'$.

Scanned by CamScanner



covered by only one prime implicant

⇒ Five variable Maps :-

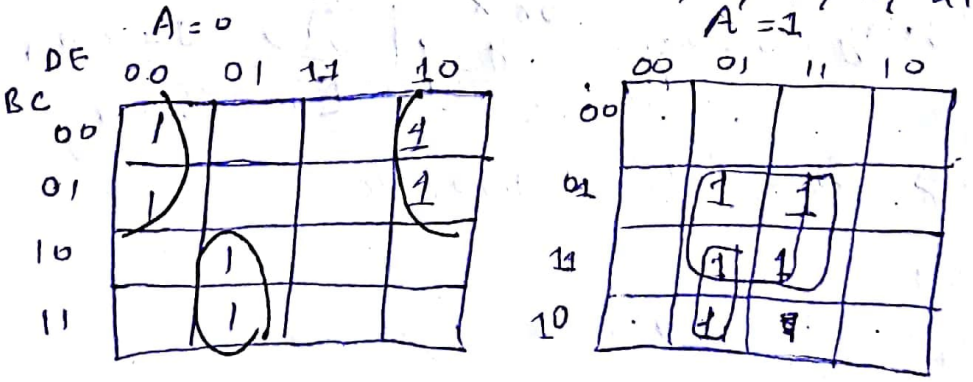
A = 0

0	1	3	2
4	5	7	6
12	13	15	14
8	9	11	10

A = 1

16	17	19	18
20	21	23	22
28	29	31	30
24	25	27	26

Ques:- $F(A, B, C, D, E) = \{0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31\}$



$A'BE' + A'BD'E + ACE$

there are six minterms from 0 to 15 belonging to A = 0 part of the map. Other five terms belongs to A = 1. The two squares in column 01 & the last two rows are common to both parts of the map.

Scanned by CamScanner

31/08/18

CIRCUIT

Combinational

A combinational circuit perform an operation that can be specified logically by a set of Boolean functions.

Sequential

In contrast, sequential circuit employ storage elements in addition to logic gates.

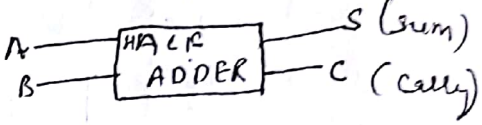
⇒ Adder design:-

HALF ADDER

bit 1	bit 2	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

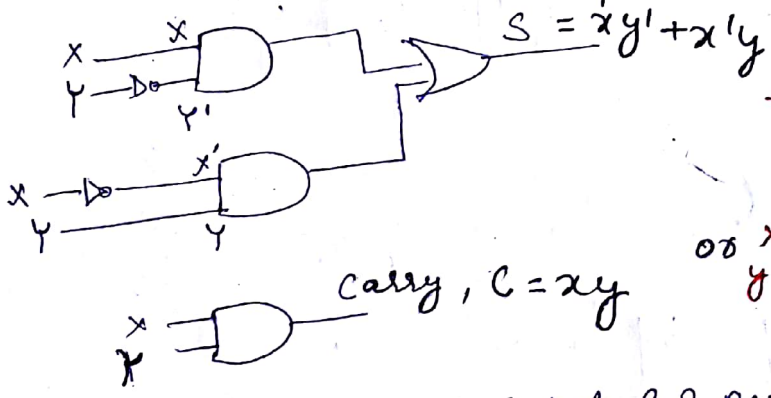
A combinational circuit that performs addition of two bits is called half adder.

Half adder ~~that~~ add two single bit no.



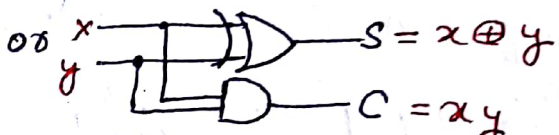
S (Sum of Product) = $x'y + xy'$
$= x \oplus y$

C (carry) = xy

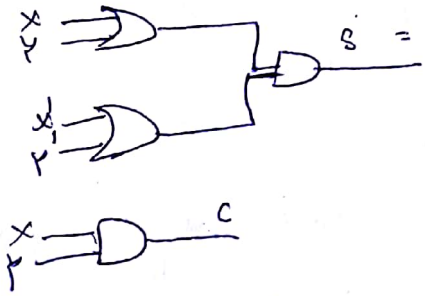


$S = xy' + x'y$

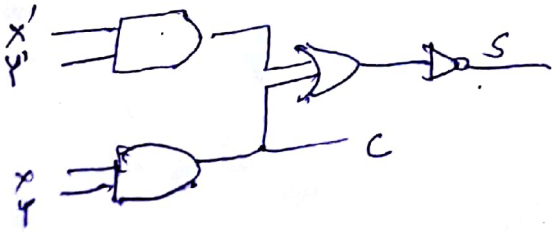
→ Implementation of half adder.



To implement- using **2 OR** gate & 2 AND gate



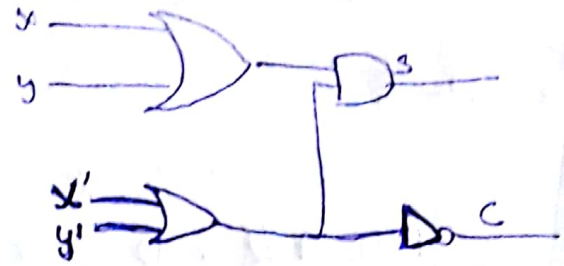
To implement using 2 AND, 1 OR & 1 inverter.



$S = (C + x'y)'$
$= (C') \cdot (x'y')'$
$= (xy)' \cdot (x+y)$
$= (x'+y') \cdot (x+y)$
$= x'y + xy'$

Scanned by CamScanner

⇒ To implement using 2 OR gate, 1 AND gate, 1 Inverter.

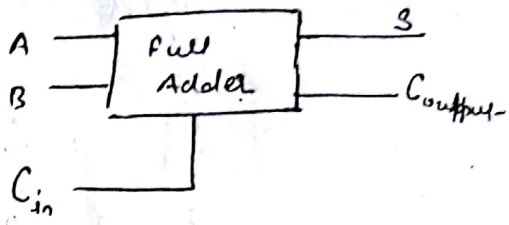


$S = (x+y)(x'+y') = xy' + y'x$

$C = (x'+y)' = xy$

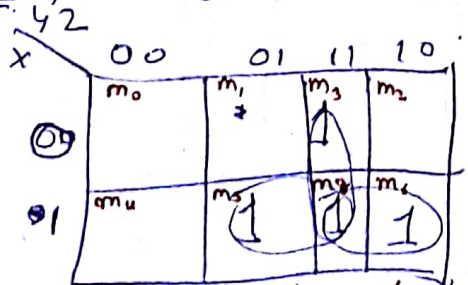
⇒ FULL ADDER :- It performs arithmetic sum of three bits.

x	y	z	C	S
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



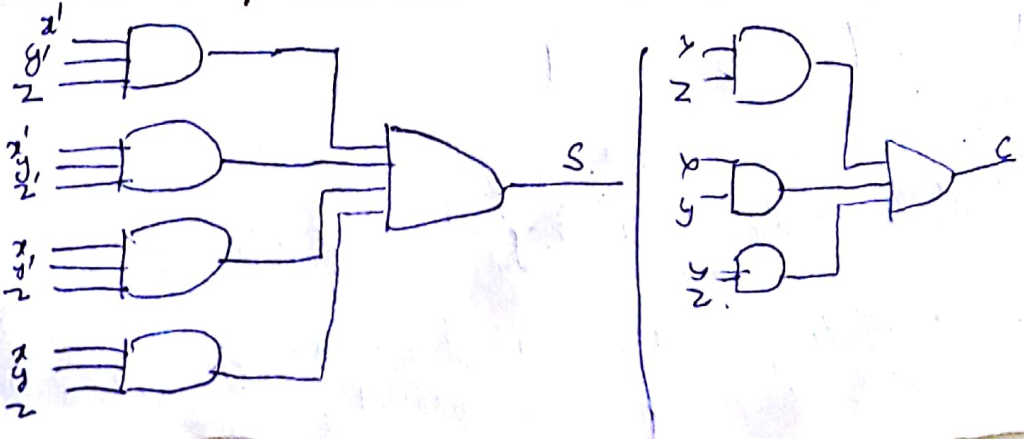
$$S = x'y'z + x'yz' + xy'z' + xyz$$

Carry $C = x'yz + xy'z + xyz' + xyz$.



$C = xz + xy + yz$

⇒ Implementation of full adder in S-O-P form.



Scanned by CamScanner

⇒ HALF SUBTRACTOR :- (Borrow) : To subtract two binary bits

X	Y	B	D (difference)
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

for X - Y

~~D = A⊕B~~ $D = X \oplus Y$

~~B = $\bar{A}$B~~ $B = \bar{X}Y$

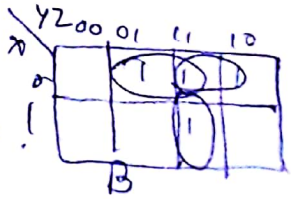
⇒ FULL SUBTRACTOR :-

X	Y	Z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$D = x'y'z + x'yz' + xy'z' + xyz$

$B = x'y'z + x'yz' + x'yz + xyz$

$B = x'z + x'y + yz$ (Simplified)



Scanned by CamScanner

Code Conversion:

Input- BCD				Output- excess-3 code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0

Sum of products (w) = 0101 → 5

0110 → 6

0111 → 7

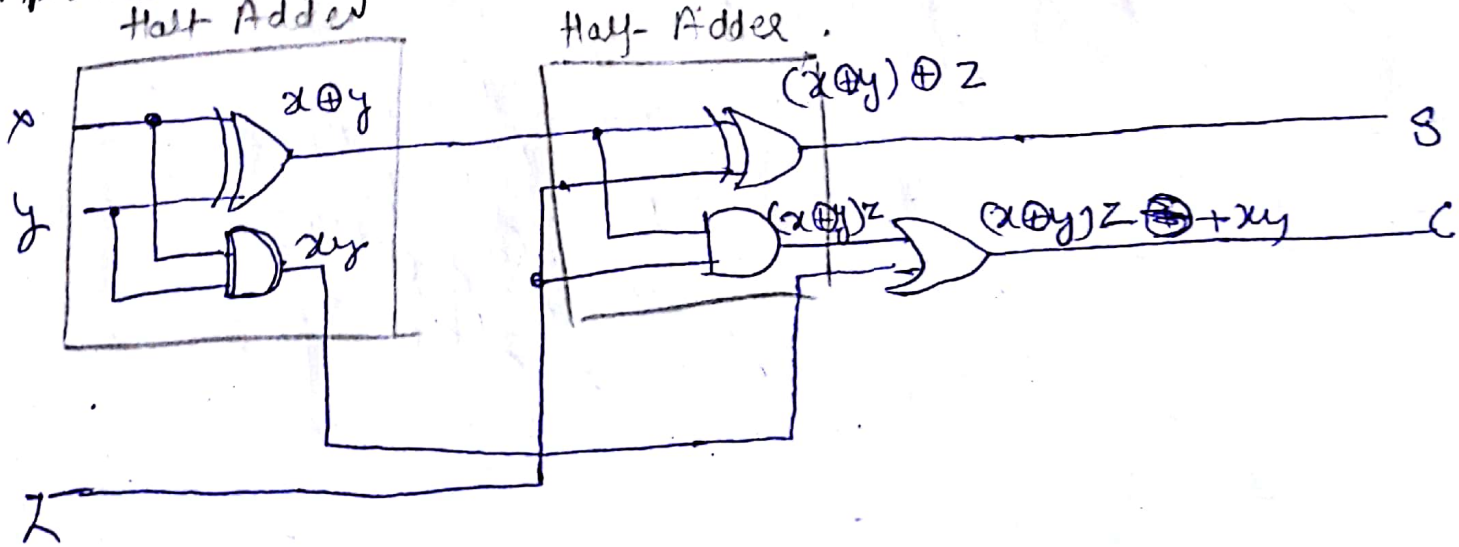
1000 → 8

1001 → 9

CD / AB	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10	1	1	X	X

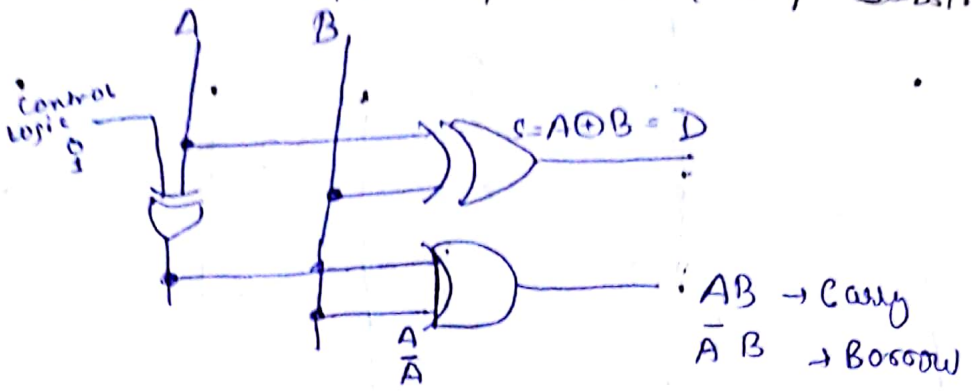
$$W = A + AB + BC$$

⇒ Implementation of full Adder with two half adder & OR gate.



Scanned by CamScanner

⇒ Common circuit for Half Adder & Half Subtractor



Note: we need min^m 5/9 → for Full Adder/Subtractor no. of NAND gate to design F/Half Adder

" " " 5/9 " " NAND " " " F/Half Subtractor

" " " 5/9 " " NOR " " " F/H.A.

" " " 5/9 " NOR " " F/H.S.

Scanned by CamScanner

Exclusive OR → commutative & Associative

$a \oplus b = b \oplus a$

$a \oplus (b \oplus c) = (a \oplus b) \oplus c$

1111 ⟶ 0111 I

Sender Receiver

Single error detection code : message + parity.

message

1111

↓

1011

Parity Bit-

↓

even matching Parity

$x \oplus 0 = x$

$x \oplus 1 = x'$

$x \oplus x = 0$

$x \oplus x' = 1$

$x \oplus y' = (x \oplus y)'$

$x' \oplus y = (x \oplus y)'$

$A \oplus B \oplus C$

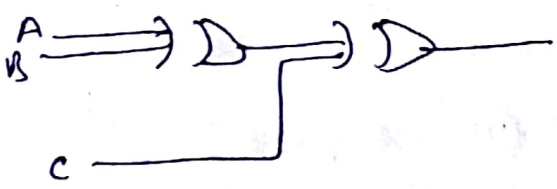
$x \oplus y = x'y + xy'$

$\Rightarrow A \oplus (B'C + BC')$

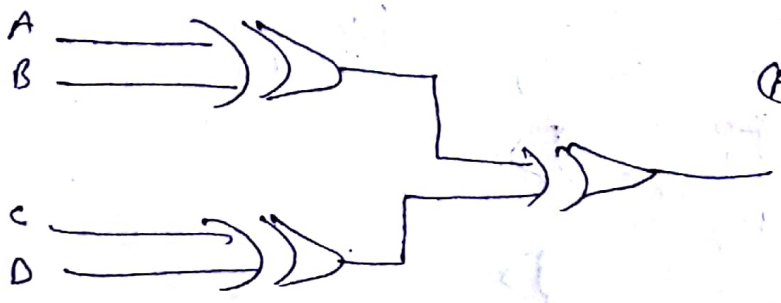
$\Rightarrow AB'C' + A'BC' + ABC + A'B'C$

100 010 111 001

⇒ Parity generator



$(A \oplus B) \oplus C$



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

Scanned by CamScanner

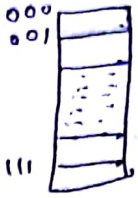
14/sept/18:-

Hindi Dipsas.
(DC) - 1st part

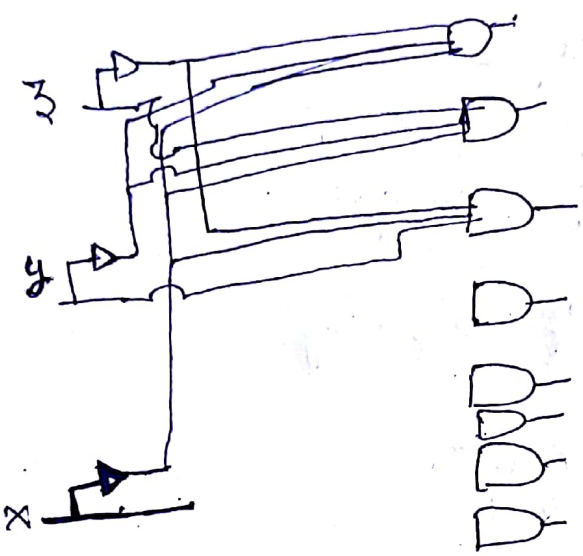
⇒ Decoders & Encoders :- (~~OR~~ AND Gate)

Standard Combination circuits

1. DECODER:-
A decoder is a combinational circuit that converts binary information from n input lines to a max^m of 2^n output lines.



001
3 bits
⇒ $2^3 = 8$ can be accessed.



$D_0 : x'y'z'$ (000)
$D_1 : x'y'z$ (001)
$D_2 : x'yz'$ (010)
.
.
.
D_7

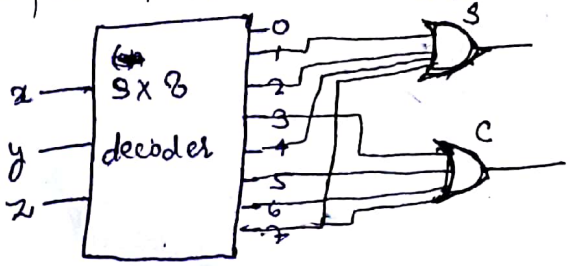
Only one output will be activated at a time rest will be zero.

⇒ Full adder with a decoder and two OR gates.

$S(x, y, z) = \sum 1, 2, 4, 7$ (Sum)

$C(x, y, z) = \sum 3, 5, 6, 7$. (Carry).

* Block diagram for Decoder ($n \times 2^n$)



Scanned by CamScanner

2. ENCODER:- (OR Gate)

2^n → input-

n → output-

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

Truth Table for Octal to binary encoder :-

Input-s								Output-		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
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			
0	0	0	0	0	1	0	0			
0	0	0	0	0	0	1	0			
0	0	0	0	0	0	0	1			

$D_4 \leftarrow 1$ 0 0

$D_5 \leftarrow 1$ 0 1

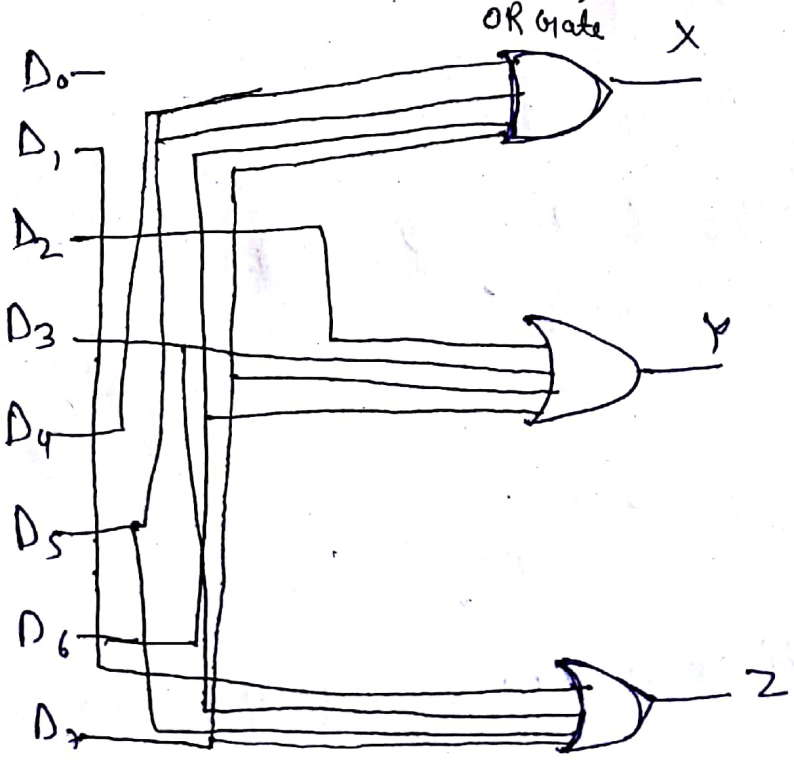
$D_6 \leftarrow 1$ 1 0

$D_7 \leftarrow 1$ 1 1

$X = D_4 + D_5 + D_6 + D_7$

$Y = D_2 + D_3 + D_6 + D_7$

$Z = D_1 + D_3 + D_5 + D_7$



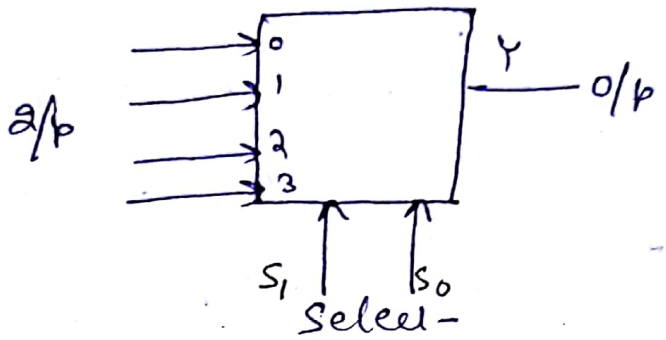
Note: D_0 never used

Scanned by CamScanner

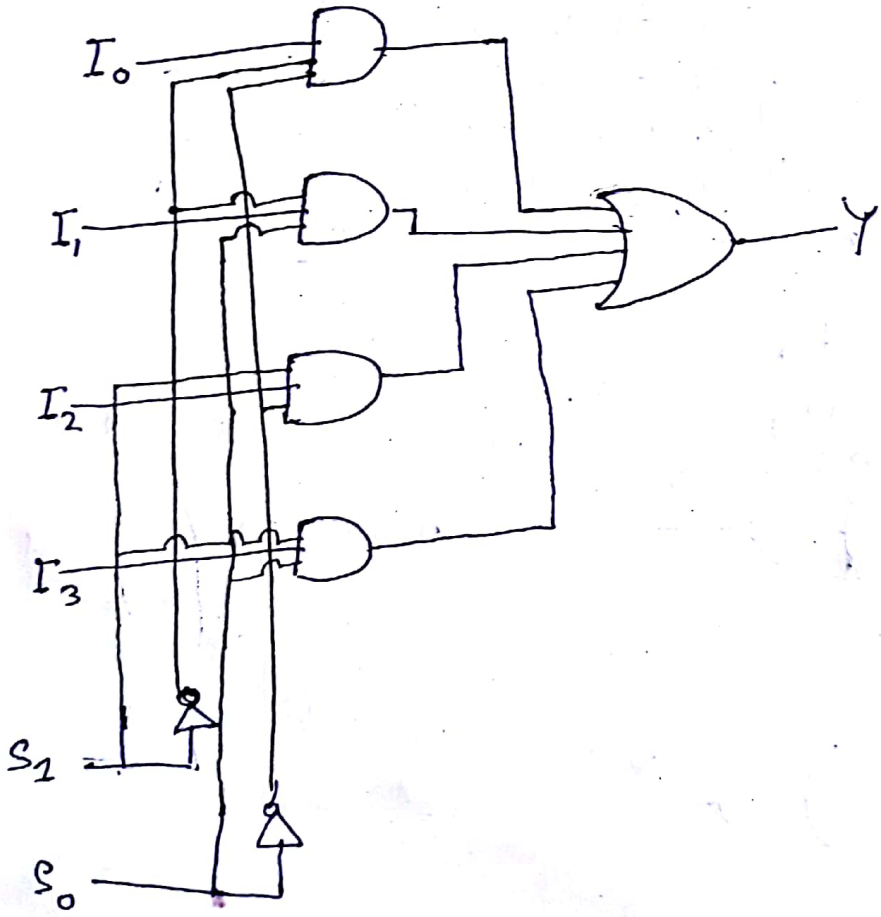
⇒ Multiplexer :-

: Transmitting a large information over a small no. of lines.

Block Diagram :-



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



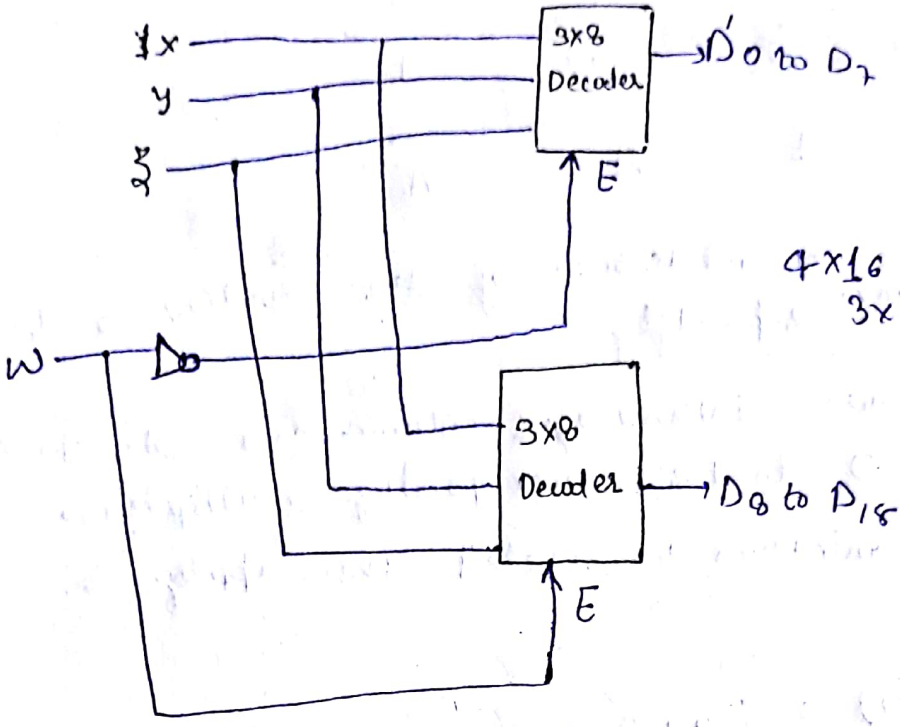
⇒ At any moment, almost one of I_0, I_1, I_2 & I_3 will be activated rest will be zero.

no. of selection lines be m & no. of inputs be n, then

$$2^m = n$$

Scanned by CamScanner

15/08/18.



4x16 decoder from 3x8 decoders.

⇒ Using multiplexer for implementing any boolean function

- Any boolean fun of n+1 variable can be implemented with 2^n to 1 multiplexer.
  $(2^n \times 1)$.

Ex→ $F(A,B,C) = \sum 1,3,5,6$.

	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

We will use two variables for selection lines & one variable for input.

Scanned by CamScanner

	I_0	I_1	I_2	I_3
A'	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	A'

Circle all the minterms of the function and inspect each column seperately.

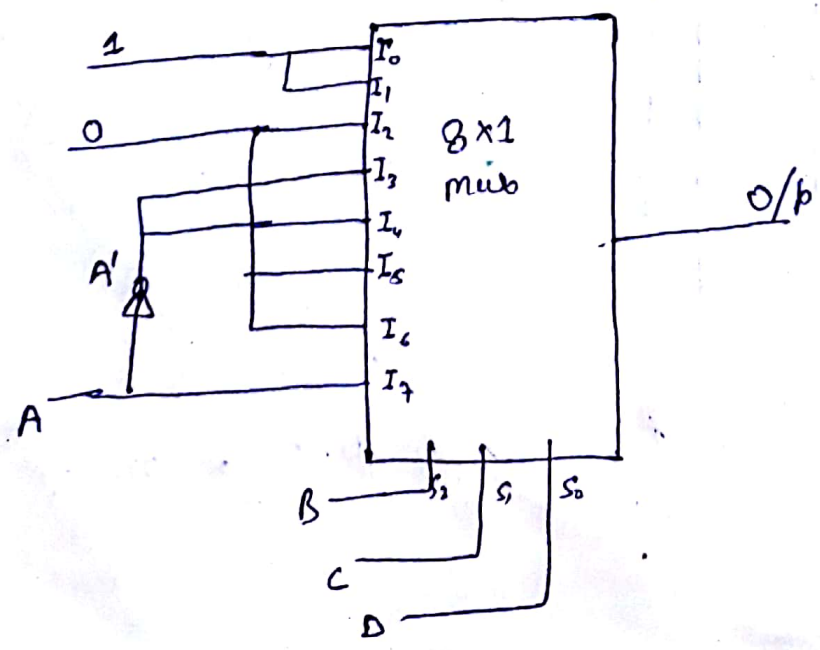
(i) If the two minterms of a column are not circled apply 0 to the corresponding multiplexer input.

(ii) If both minterms are circled then apply 1.

Q: $F(A,B,C,D) = \sum 0,1,3,4,8,9,15$.

Sol: No. of selection lines = 3.
No. of variables = 3
No of Input = 16.

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	(0)	(1)	2	(3)	(4)	5	6	7
A^3	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	A'	A'	0	0	A

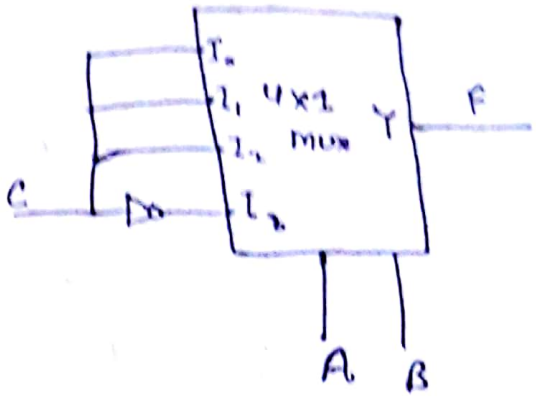


Scanned by CamScanner

Q $F(A, B, C) = (1, 3, 5, 6)$

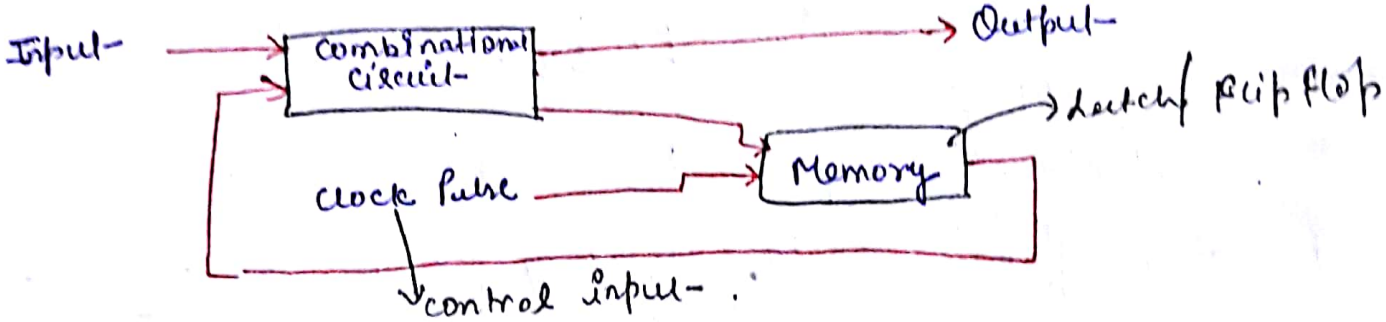
No. of selection lines = 2

	I_0	I_1	I_2	I_3
C'	0	2	4	(6)
C	(1)	(3)	(5)	7
	C	C	C	C'



SEQUENTIAL CIRCUIT

Sequential circuit is a combinational circuit with memory.



⇒ Types of latches :-

i) S-R (Set--Reset type)
ii) D (Delay) Type
iii) J-K type
iv) T (Toggle) type.

Scanned by CamScanner

5/10/18

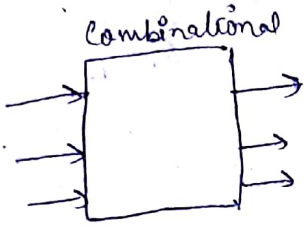
Digital Circuits

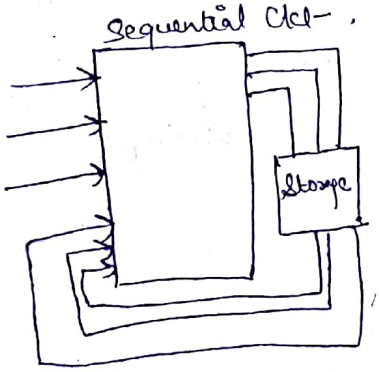
- Combinational
  - Output depends only in the present inputs
- Sequential
  - O/p depends on the present I/p as well as previous o/p. {we have memory or storage element to store past o/p.}

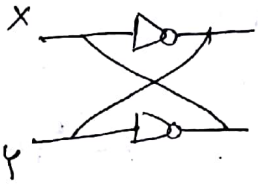
⇒ why sequential circuits are necessary.

Circuit for incrementing a number by 1.

ex- Traffic light controller







{Either 1 or 0, value will be hold}

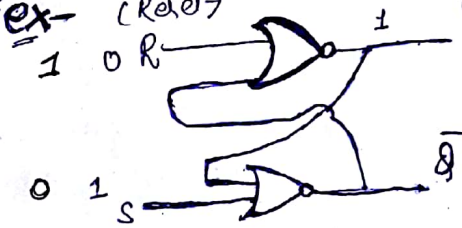
Latch circuit
(to lock or hold something)
(data)

⇒ S-R latch → two cross coupled NOR gates.

⇒ Realisation with NOR Gates:- ⇒ State table

1.

ex- (Reset)



(set)

S = 1	R = 0,	Q = 1,	$\bar{Q}$ = 0
case 1 ← S = 0	R = 0,	Q = 1,	$\bar{Q}$ = 0 (Q next)
case 2 ← S = 0,	R = 1,	Q = 0,	$\bar{Q}$ = 1 (Q present state)
S = 0,	R = 0,	Q = 0,	$\bar{Q}$ = 1
case 3 ← undesirable condition. S = 1,	R = 1,	Q = 0,	$\bar{Q}$ = 0 (forbidden)
S = 0,	R = 0 → unpredictable		

As we can see, after removing inputs we get same o/p, hence its R/S memory.

Faster gate will be 1 & slower will be ~~zero~~ Zero

⇒ Reset → Q = 0

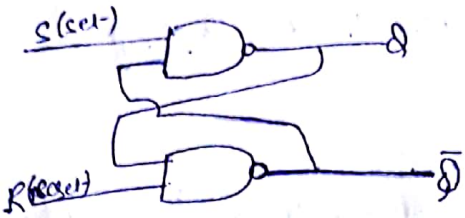
set → Q = 1

⇒ when Reset = 1, o/p i.e. Q = 0 & when Set = 1 ; Q = 1.

Scanned by CamScanner

⇒ Realisation with NAND Gate :-

Note the diff. in circuits of rep. with NOR & NAND gates



⇒ State table

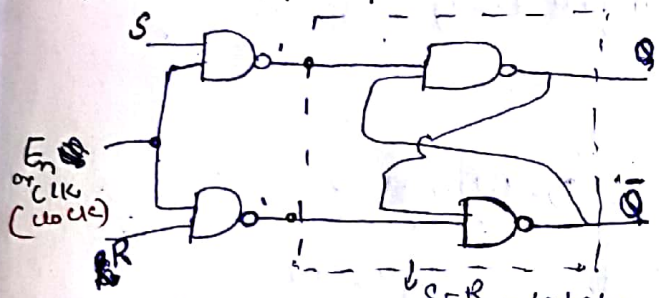
S	R	Q_n	$\bar{Q}$	
1	0	0	1	
1	1	0	1	(after S=1, R=0)
0	1	1	0	
1	1	1	0	(after S=0, R=1)
0	0	1	1	(forbidden)
1	1			unpredictable (Race condition)

memory state ← Present state

Not used state.

X	Y	O/p
0	0	1
0	1	1
1	0	1
1	1	0

⇒ control input :-



when we have En, then above ckt- is SR latch
when we have clk, then it is flip-flop.
⇒ Above ~~other~~ circuit can act- as both latch & flip-flop.

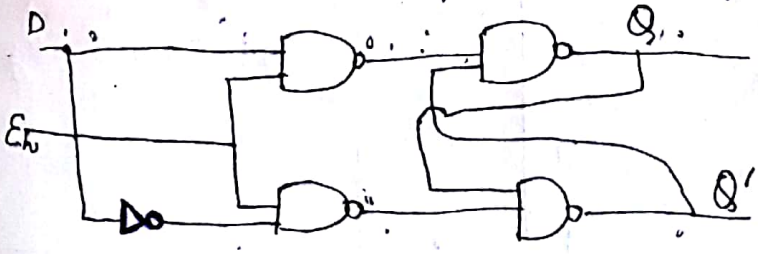
En	S	R	Next stage of Q
0	x	x	No change
1	0	0	No change
1	0	1	Q = 0, reset stage
1	1	0	Q = 1, set stage
1	1	1	Indeterminate

S.R. latch with control input.

6/10/18.

⇒ There is problem of undesirable input- with both realisation hence we have D latch.

+ D latch :- (transparent- Latch) → hold data



⇒ State table

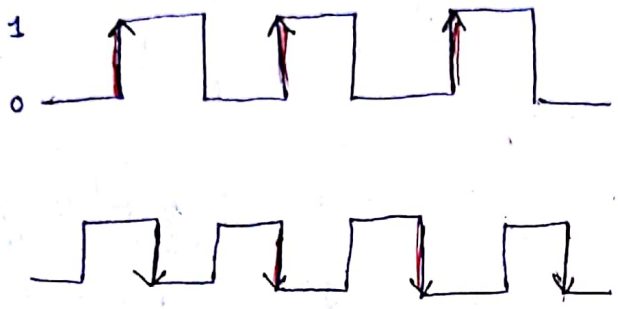
En	D	next- state of Q
0	X	No change
1	0	Q = 0, reset-state
1	1	Q = 1, set- state

Scanned by CamScanner

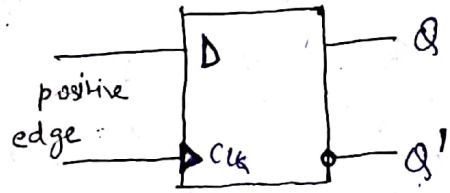
5 ⇒ Difference b/w Latch & Flip-flop :-

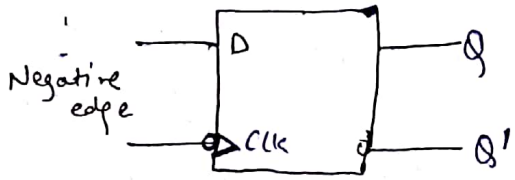
Both Latches and Flip-flops are storage elements.
Latches operate with signal levels & flip flops operate with signal transitions.

→ Response to positive level. → Latch.



⇒ Flip-flop



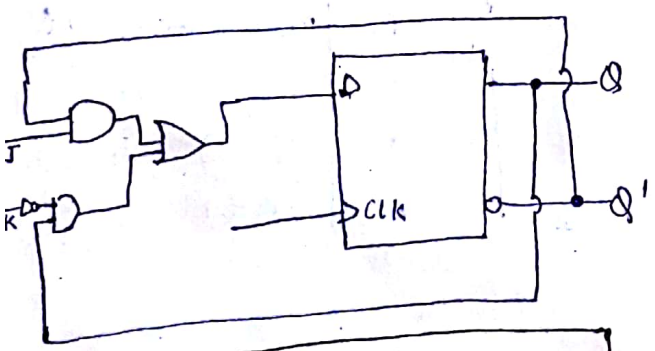


D- Flip flop

D	Q (t+1)
0	0
1	1

holds the input.

⇒ JK flip flop :-



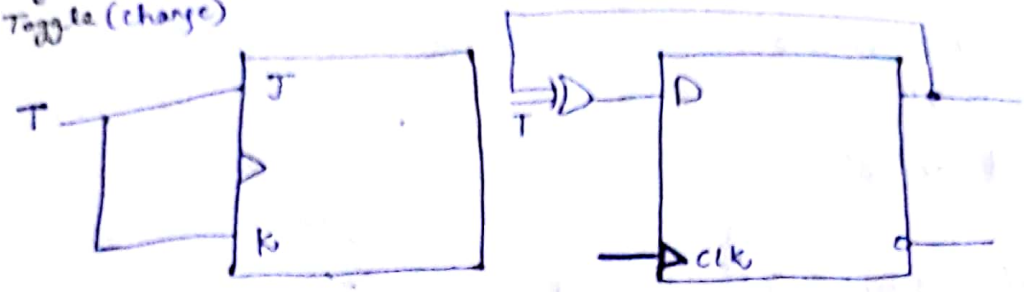
$D = JQ' + K'Q$

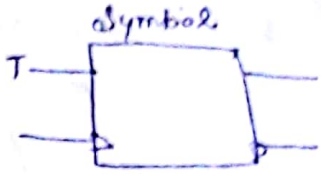
J	K	Q (t+1)
0	0	Q (t) (memory)
0	1	0
1	0	1
1	1	Q'(t)

Scanned by CamScanner

⇒ T - flip flop

Toggle (change)





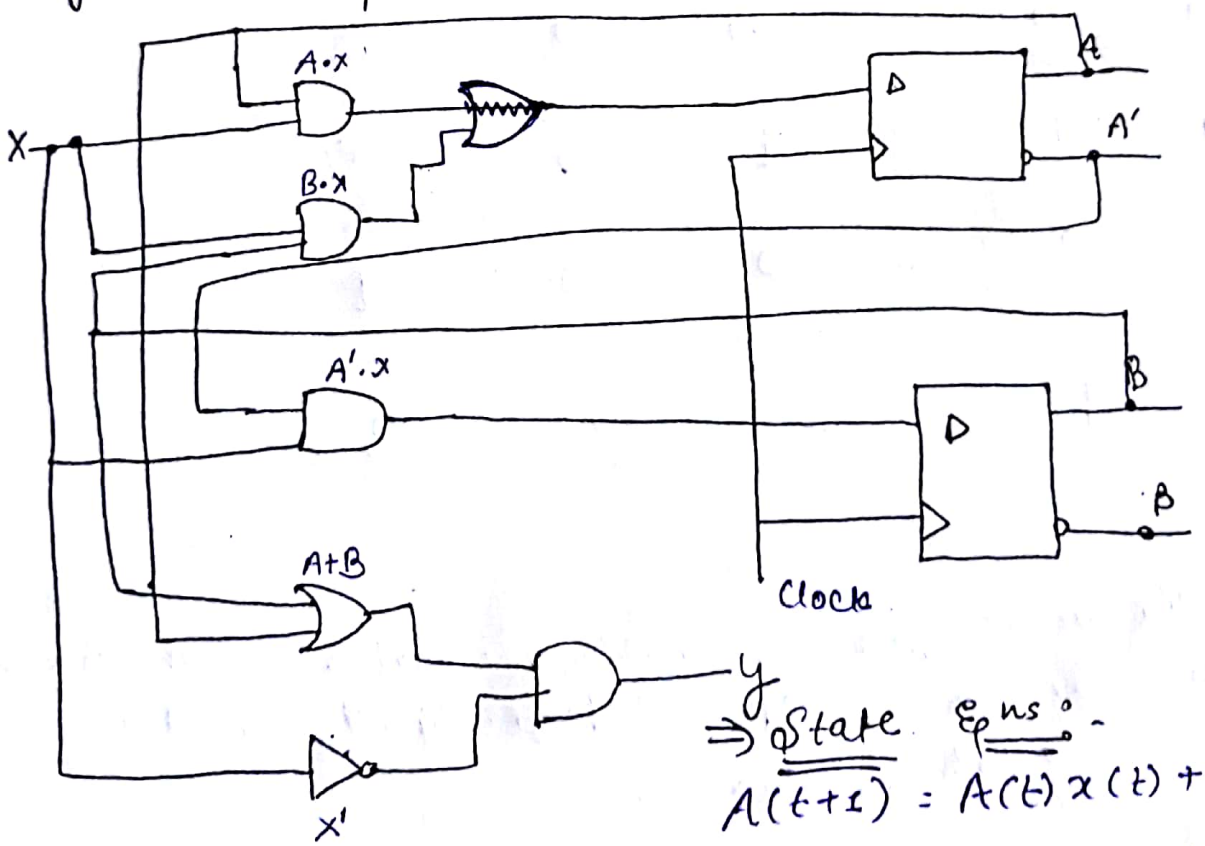
T flip flop

T	Q(t+1)
0	Q(t) no change
1	Q'(t) complement.

T flip-flop toggle the input, when T=1, o/p ~~output~~ changed.

8/10/2018 : Indian Airforce Day.

⇒ Analysis of Sequential Circuit :



⇒ State Eqns :-

$A(t+1) = A(t)\,x(t) + B(t)\,x(t)$

$B(t+1) = A'(t)\,x(t)$

$y(t) = [A(t)+B(t)]\,x'(t)$

Scanned by CamScanner

5 OR

$$A(t+1) = Ax + Bx$$
$$B(t+1) = A'x$$
$$y = (A+B)x'$$

⇒ Analysis of Sequential Circuit-

State Table.

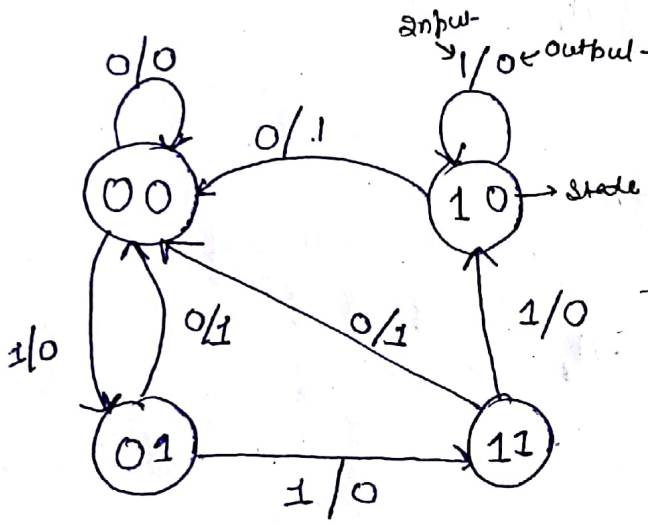
Present-state A	Present-state B	input- x	Next state A	Next state B	Output- y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

⇒ State Diagram:- The information available in the state table can be represented graphically in the form of State Diagram.

Scanned by CamScanner

⇒ Second Form of State Table.

Present- State		Next- State				Output-	
		$x=0$		$x=1$		$x=0$	$x=1$
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0



→ State Diagram.

Circle → states.

⇒ Steps for Analysis of Sequential Circuit :-

① Circuit- Diagram → ② State eqns → ③ State table → ④ State Diagram

Scanned by CamScanner

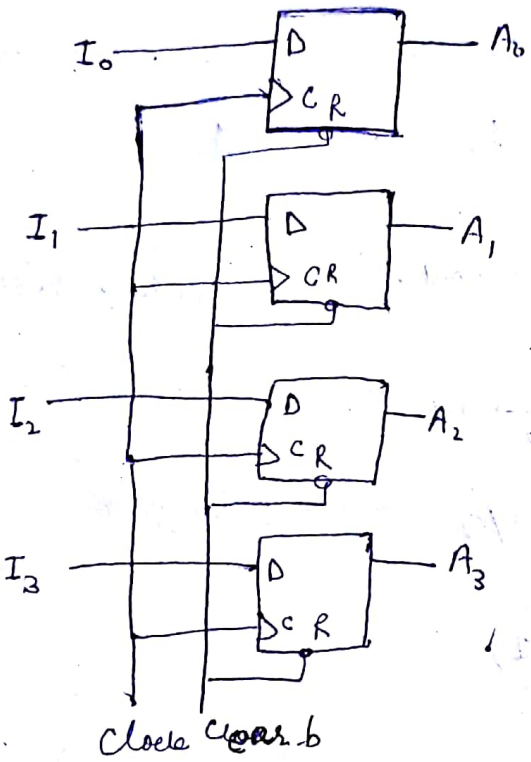
5

26/10/18 :-

Registers and Counters :-

Register : A group of flip-flops, each one of which shares a common clock & is capable of storing 1 bit of information. A n bit register has n flip-flops.

Counters :- It is basically a register that goes through a predefined sequence of states.



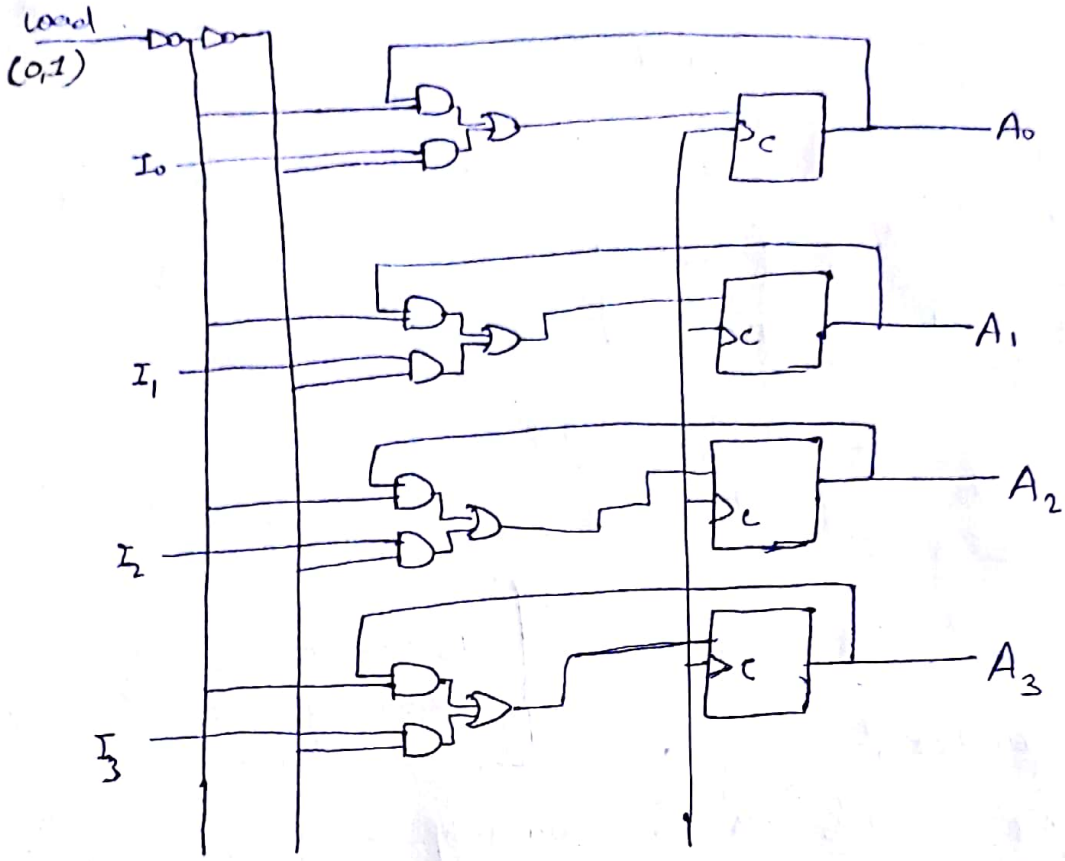
The common clock input triggers all flip flops on the positive edge and the binary data available at the four inputs is transferred into the registers.

clear_b input is useful for closing the register to all 0's be free the input is present.

⇒ Register with parallel load :- All the inputs should appear at o/p at same time.

Using clock input for creating the register with parallel load creates problems in Synchronisation.

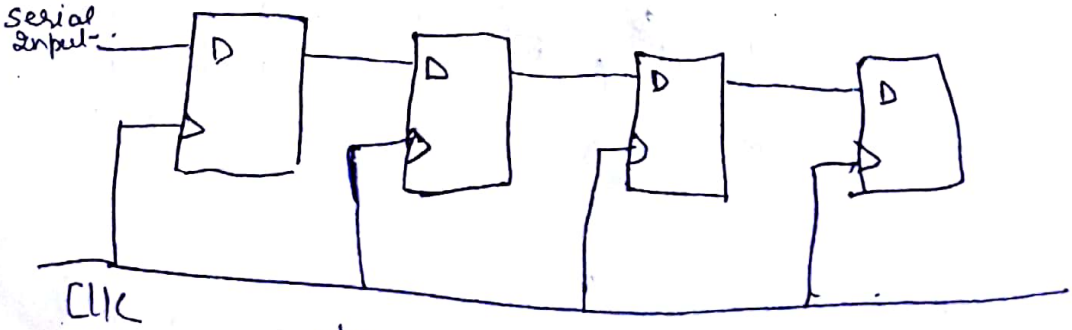
Scanned by CamScanner



When load input is 1, the data in four inputs is transferred into the register with next positive edge of the clock.

When load input is 0, the output of the flip-flops are connected to their respective inputs.
The feedback connection from o/p to i/p is necessary because D flipflop do not have a no change condition.

⇒ Shift Register :- If register capable of shifting the binary information is called Shift Register.

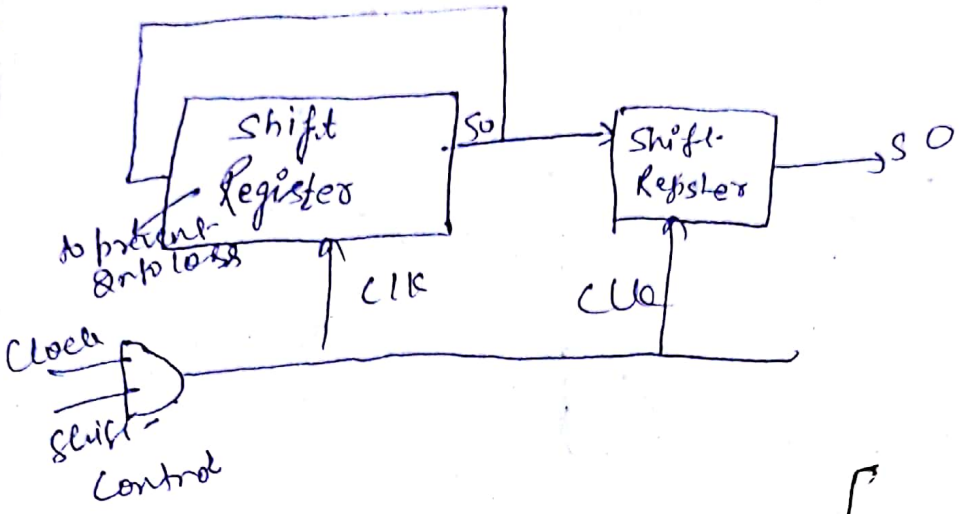


4 bit Shift Register.

Scanned by CamScanner

5 ⇒ Serial Transfer:-

A digital system is said to operate

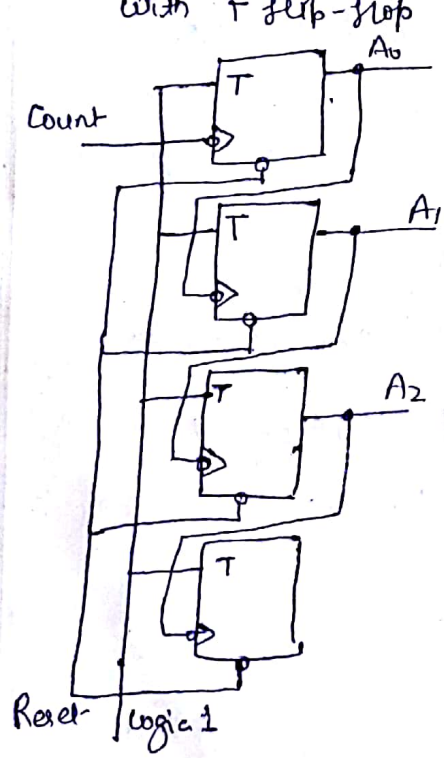


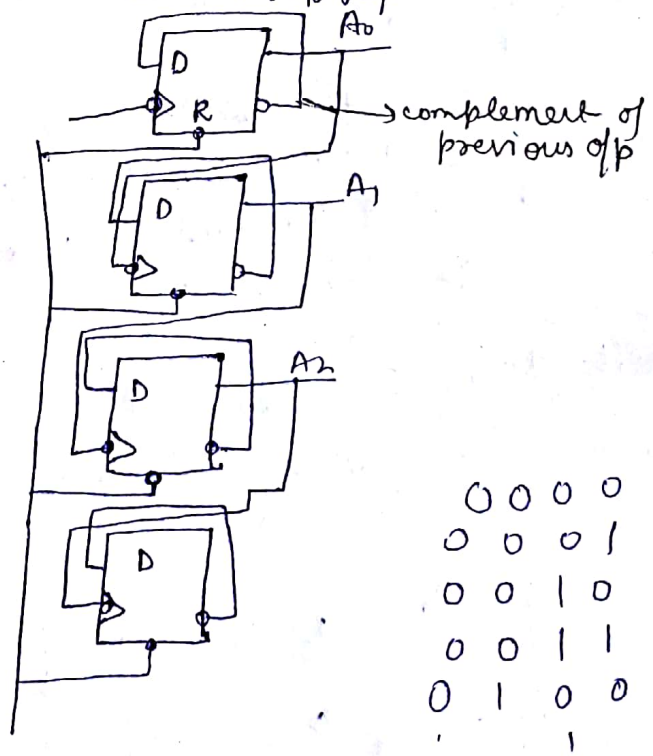
[D → ape trigger (when signal given from o/p 1 activates)
D → negative ape (when trigger 1 to 0 then activates)

⇒ 30/10/2018 :-

Binary Ripple Counter :-

A binary ripple counter is a series of complementing flip-flops with the output of each flip-flop connected to the C input of next higher cycles flip-flop.

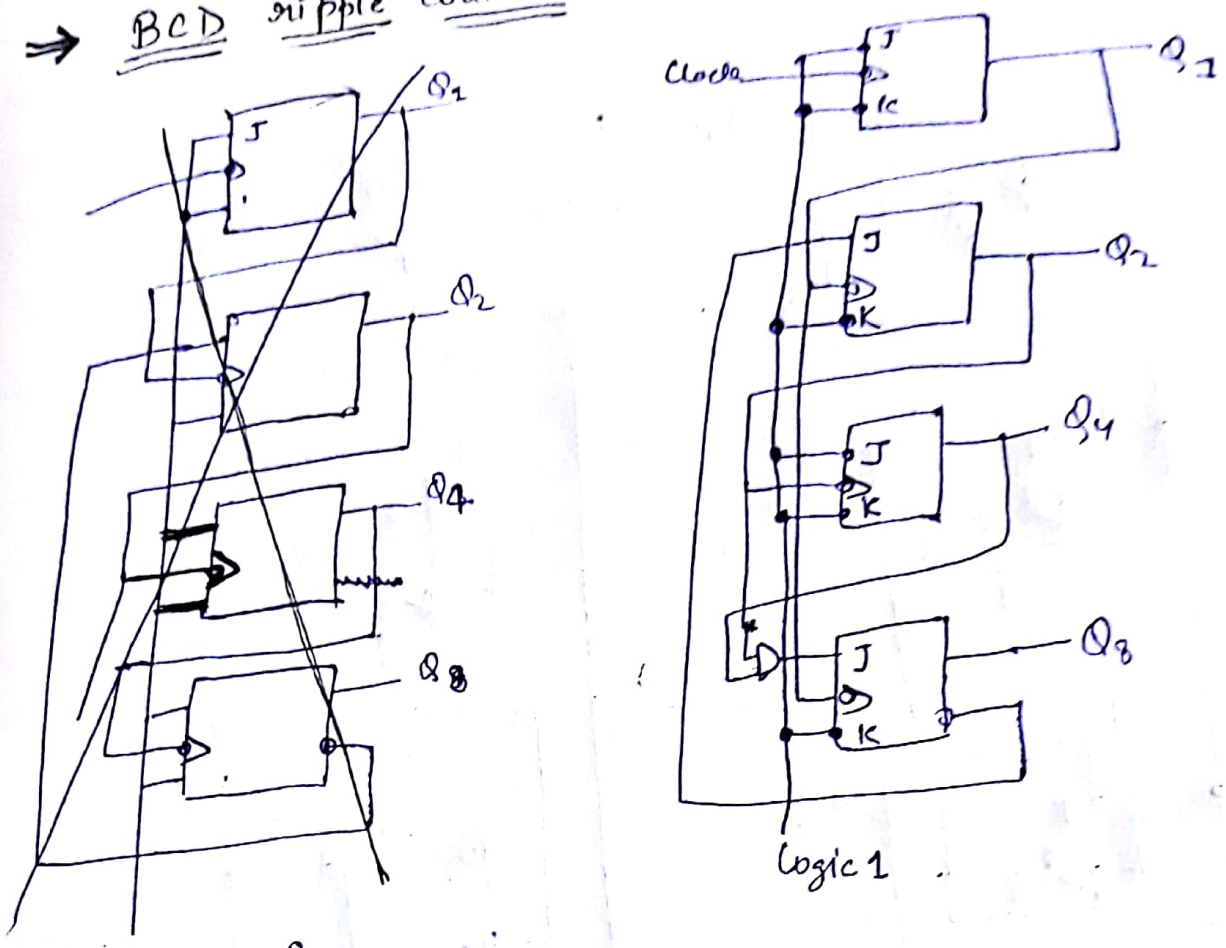




0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
.
.
.
1 1 1 1
(Reset) 0 0 0 0

Scanned by CamScanner

⇒ BCD ripple counter :-



⇒ For counting 0 to 9.

Q_8	Q_4	Q_2	Q_1
0	0	0	0

1) Q_1 changes state after each clockwise

– – 0 1

2) Q_2 complements every time Q_1 goes from 1 to 0, as long as $Q_8 = 0$.

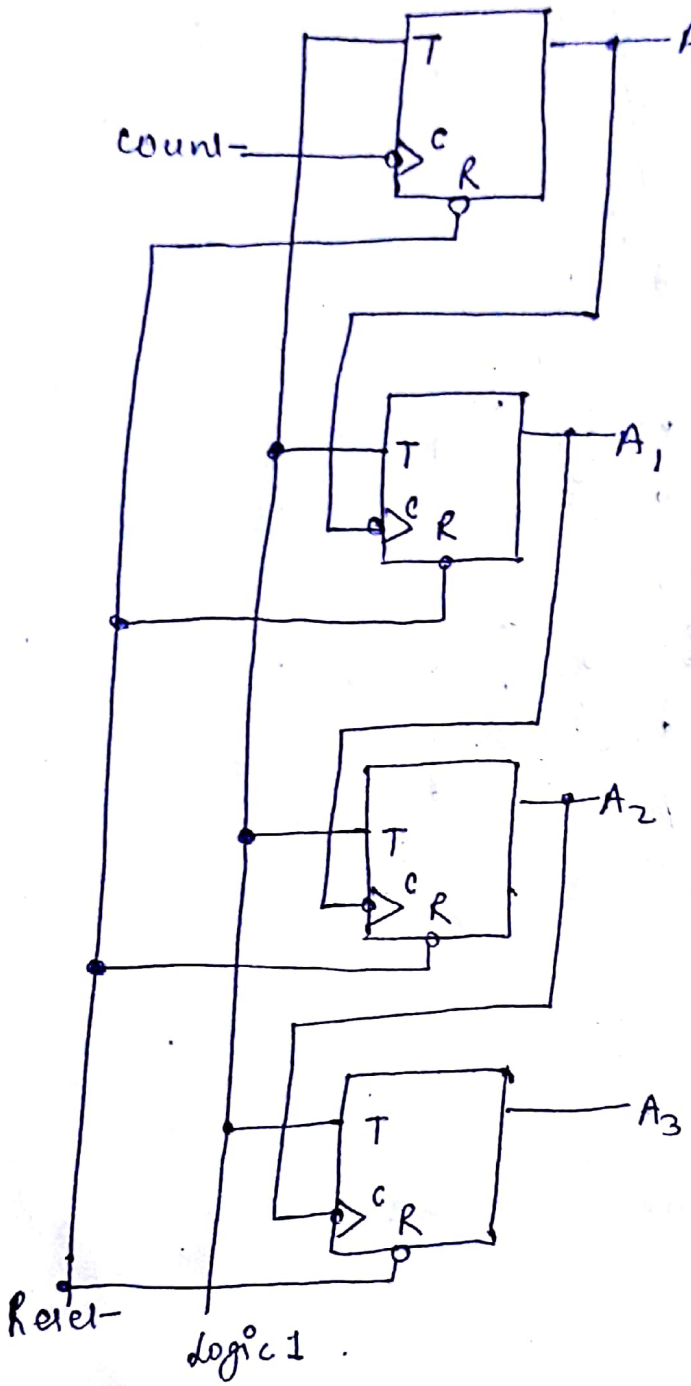
3) When Q_8 becomes 1 Q_2 remains at 0.

4) Q_4 complements each time when Q_2 goes from 1 to 0.

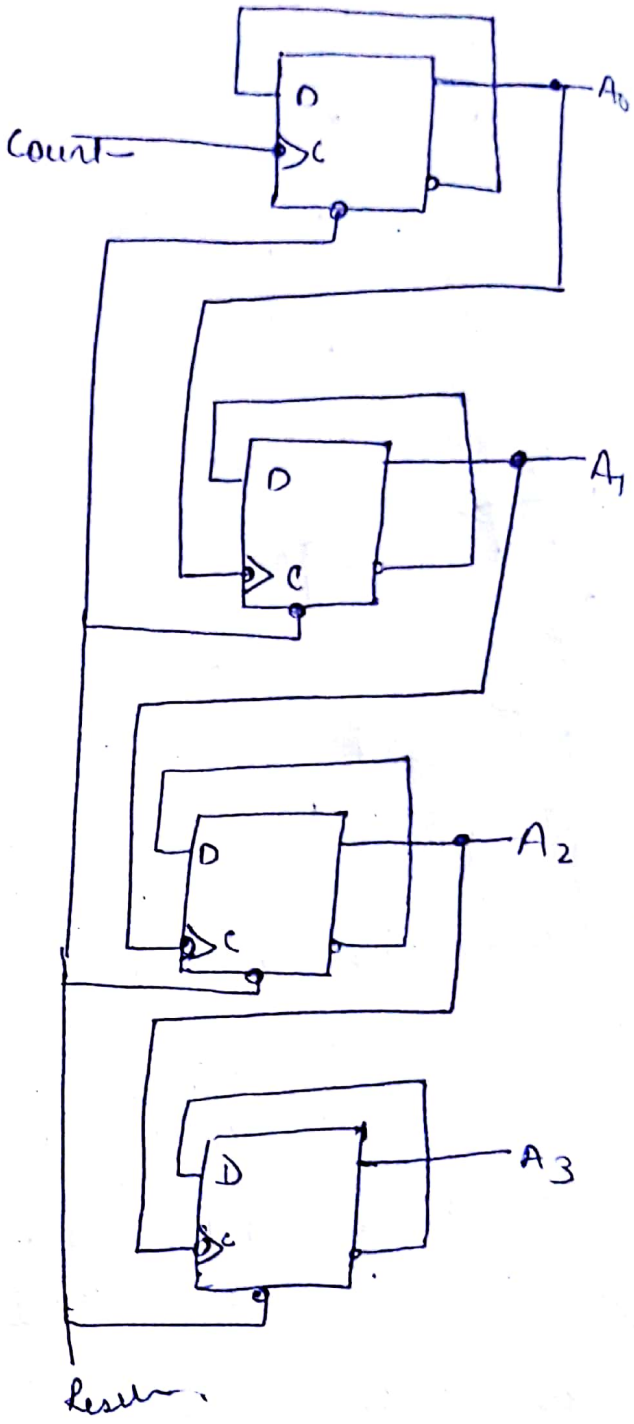
5) Q_8 remains at 0 as long as Q_2 or Q_4 is 0.

Scanned by CamScanner

Four bit binary ripple counter :-



with T flip-flop.



With D flip-flop.

Scanned by CamScanner