Unit 1 Design Concepts

Digital system

Combination circuits

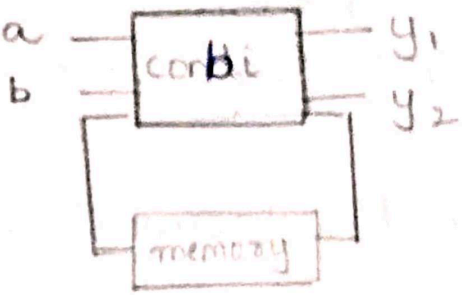
Output depends on only input

Eg: adder (Switch) sub

Sequential circuits

Output depends on past inputs which is stored in memory

Eg: lift elevator, traffic, light converter, etc



SSI - small scale integration circuit (10 transistors) - 10^2

MSI - Medium scale integration circuit (100) - 10^3

LSI - Large (1000) - 10^4

VLSI - very large (10,000) - 1K

ULSI - ultra large (1K - 10K)

SOC - system on chip (1 million)

NOC - Network on chip (1 billion)

Number system

Decimal (0-9)	Binary (0,1)		
0	00	8	1000
1	01	9	1001
2	10	10	1010
3	11	11	1011
4	100	12	1100
5	101	13	1101
6	110	14	1110
7	111	15	1111

* The radix / base of binary no system is 2 and for decimal is 10

Q. Convert decimal no system to binary

a) 48

$(48)_{10} = (110000)_2$

2 | 48 - 0
2 | 24 - 0
2 | 12 - 0
2 | 6 - 0
2 | 3 - 1
1

b $(148)_{10}$

$(148)_{10} =$

$(10010100)_2$

2 | 148 - 0
2 | 74 - 0
2 | 37 - 0
2 | 18 - 1
2 | 9 - 0
2 | 4 - 1

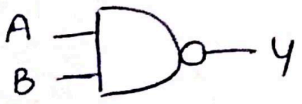
a. $(1010101)_2$

2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1

$64 + 16 + 4 + 1 = (85)_{10}$

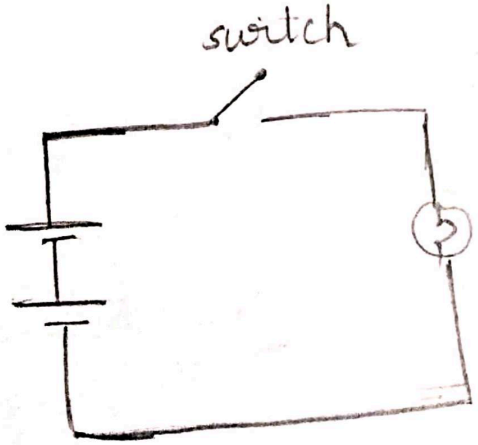
HDL - Hardware describe language

NAND GATE



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Binary quantities and variables.



S	L
open	off
closed	on

S	L
0	0
1	1

~~Digital Hardware~~ →

Basic building blocks in a digital system are called logic gates

Logic gates are classified into basic gates and universal gates

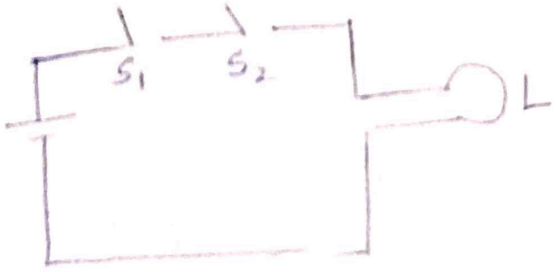
The working of logic gates can be analysed using logic equation and truth table or behavioural table

Simple gates

AND
OR
NOT

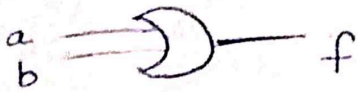
Functionality can be expressed by a truth table

a	b	F
0	0	0
0	1	0
1	0	0
1	1	1

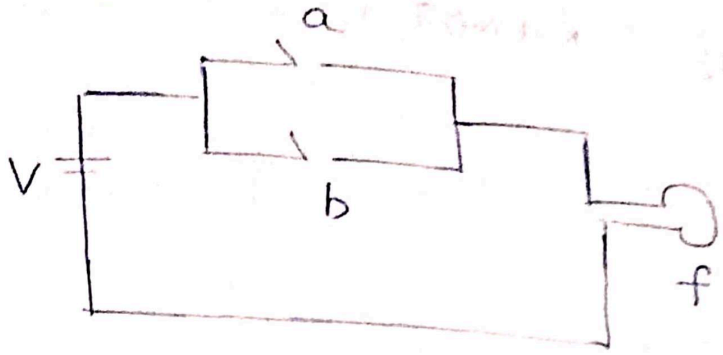


logic

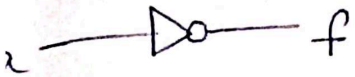
$F = a \cdot b$



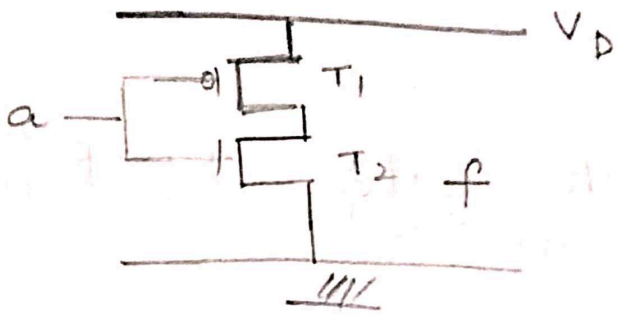
a	b	f
0	0	0
0	1	1
1	0	1
1	1	1



logic f = a + b



a	f
0	1
1	0



T - 0 - open ckt

T = 1 -

* A truth table lists output for each possible input combination

F = A * B – and

f = A + B – or

f = A' – not

Universal gates
Universal gates are NAND and NOR

NAND

Logic = $\overline{a \cdot b}$

NOR

a	b	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR (OR+NOT)

a	b	F
0	0	1
0	1	0
1	0	0
1	1	0

Logic : $\overline{a+b}$

XOR gate : All inputs are same and output is 0 and the diff in inputs leads to O/P

Logic expression

$f = a+b = \bar{a}b + a\bar{b}$

$\bar{a}b + a\bar{b}$

$1 \cdot 0 + 0\,1$

a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

XNOR (XOR+NOT)

$f = a \odot b = \bar{a}\bar{b} + ab$

$1 \cdot 1 + 0 \cdot 0 = 1$

$1 \cdot 0 + 0 \cdot 1 = 0$

$0 \cdot 0 + 1 \cdot 0 = 0$

$0 \cdot 0 + 1 \cdot 1 = 1$

a	b	f
0	0	1
0	1	0
1	0	0
1	1	1

Decimal	Binary
0	0 0
1	0 1
2	1 0
3	1 1

4 2 1

0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

8 4 2 1

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

2^3 2^2 2^1 2^0

8 4 2 1

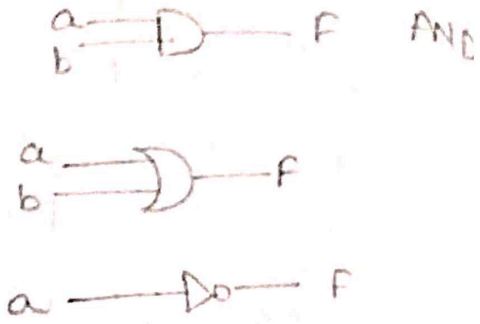
Realization of digital circuits using logic gates

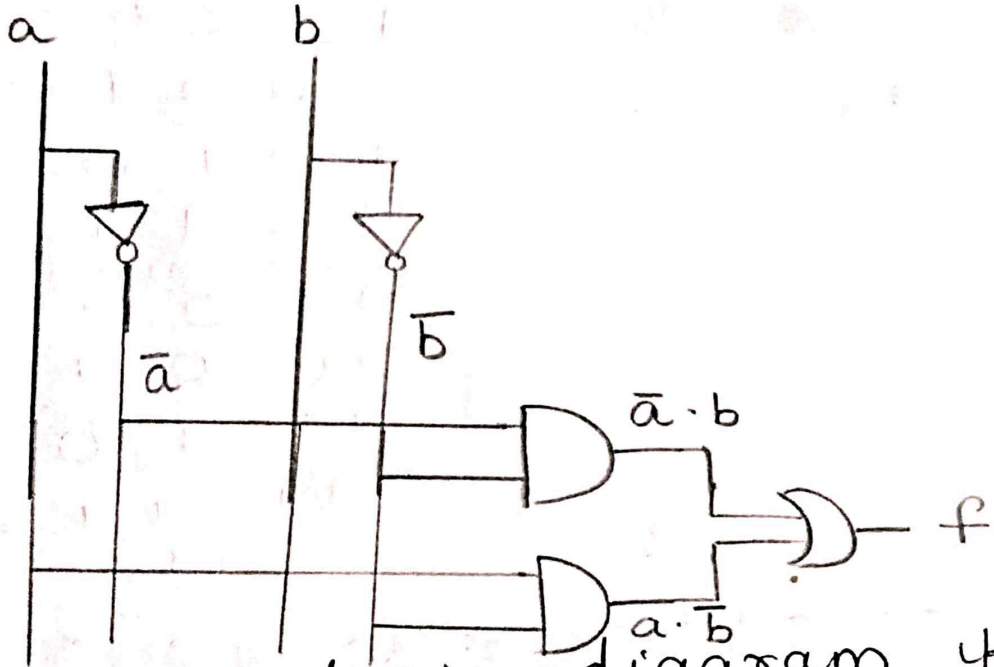
Draw XOR logic using basic gates

Sol:-

XOR gate

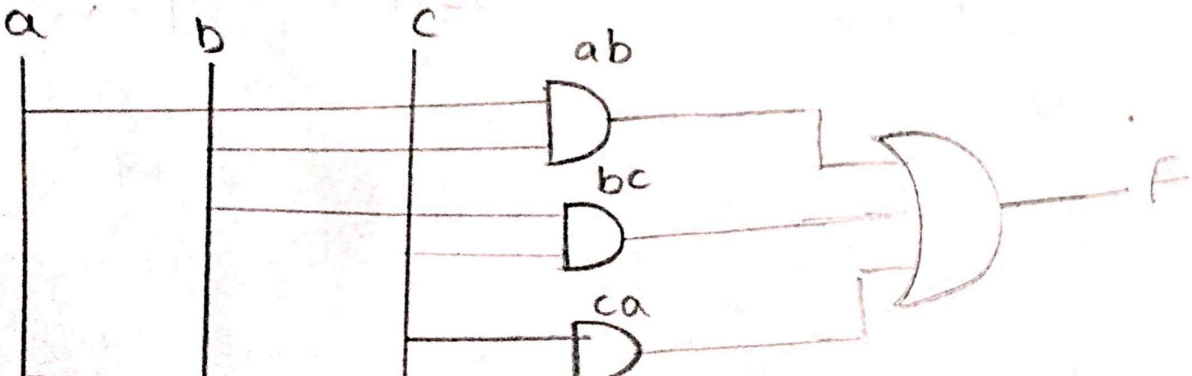
$f(a,b) = \bar{a} \cdot b + a \cdot \bar{b}$





Draw a logic diagram for the flg

$f(a,b,c) = ab + bc + ca$



...inciples

SOP expression

$f(a,b,c) = a \cdot b + b \cdot c + c \cdot a$

product term

variables

literals

called as sum of products (SOP)

The above SOP expression is non-standard or non-canonical

POS expression

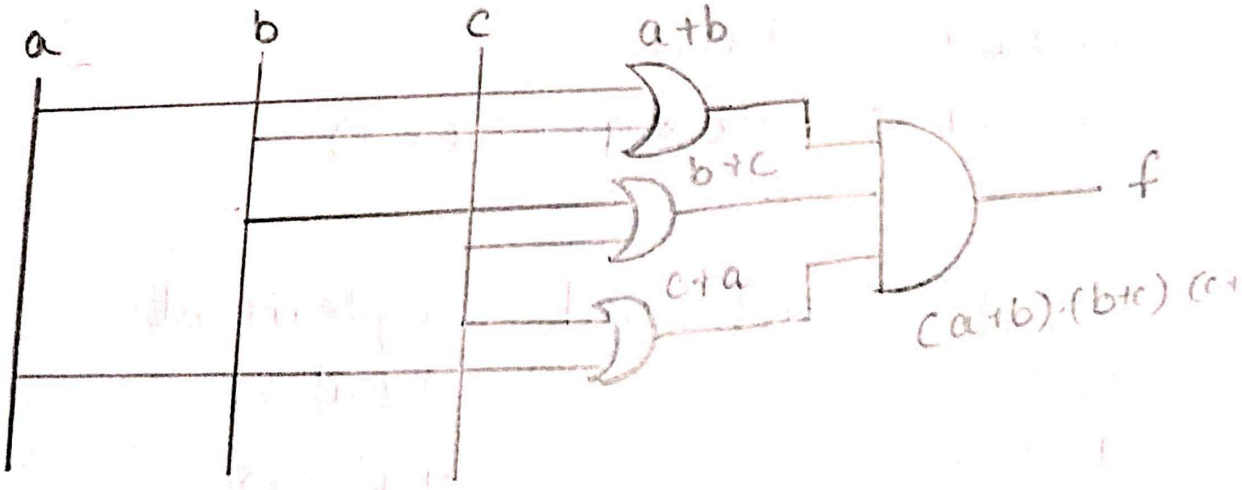
Product of sum

Consider, an expression $f(a,b,c)$

$f(a,b,c) = (a+b) \cdot (b+c) \cdot (c+a)$

sum terms

Draw a logic diagram for above POS expression



Boolean Algebra

On 1854, George Boole developed an algebraic system now called Boolean algebra

It is used to reduce the

Boolean laws and postulates

1. Associative law
2. Communicative law
3. Distributive law

1. Associative law: A binary operator * on a set S is said to be associative.

$$(x * y) * z = x * (y * z)$$

2. Communicative law :-

$$x * y = y * x$$

3. Distributive law :-

$$x * (y + z) = (x * y) + (x * z)$$

Identity element and complement

$a + 0 = a$ $\qquad$ $a + a' = 1$

$a \cdot 1 = a$ $\qquad$ $aa' = 0$.

Basic Theorems

Postulate 2	(a)	$x + 0 = x$	(b)	$x + 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x + 0 = 0$
Theorem 3: involution	(a)	$(x')' = x$	(b)	$\overline{xy} = yx$
Postulate 3 commutative	(a)	$x + y = y + x$	(b)	$x(yz) = (xy)$
Theorem 4: associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x + yz = (x + y)(x + z$
Postulate 4 distributive	(a)	$x(y + z) = xy + xz$	(b)	$(xy)' = x' + y'$
Theorem 5 DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$x(x + y) = x.$
Theorem 6 absorption	(a)	$x + xy = x.$	(b)	

Any logic function can be reduced

1) using boolean theorems and laws
2) using K map method
3) using Q and M method or tabular method.

Theorem 1(a) $x + x = x$

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

Theorem $x+1 = 1$

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

Theorem $x(x+y) = x$ by duality

$$= x(x+y)$$
$$= x \cdot x + x \cdot y$$
$$= x + x \cdot y$$
$$= x(1+y)$$
$$= x(1)$$
$$= x$$

Demorgan's law

$$\overline{x+y} = \overline{x} \cdot \overline{y}$$

				R.H.S		L.H.S
x	y	$\overline{x}$	$\overline{y}$	$\overline{x} \cdot \overline{y}$	$x+y$	$\overline{x+y}$
0	0	1	1	1	0	1
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	0	1	0

for a given function minimize function by using boolean algebra

$$f(a,b,c) = \bar{a}b + ab + \bar{a}\bar{c}$$

$$= b(\bar{a} + a) + \bar{a}\bar{c}$$

$$b + \bar{a}\bar{c}$$

Karnaugh Map (K Map)

Boolean expression can be graphically depicted and simplified with the use of Karnaugh maps. In a Karnaugh map 2^n possible min terms of an n-variable Boolean functions are represented by means of separate squares or cells on the map.

a b	Product term or min term
0 0	$\bar{a}\bar{b}$
0 1	$\bar{a}b$
1 0	$a\bar{b}$
1 1	ab

0 - complemented function

1 - uncomplimented function

a b c	min term
0 0 0	$\bar{a}\bar{b}\bar{c}$
0 0 1	$\bar{a}\bar{b}c$
0 1 0	$\bar{a}b\bar{c}$
0 1 1	$\bar{a}bc$
1 0 0	$a\bar{b}\bar{c}$
1 0 1	$a\bar{b}c$
1 1 0	$ab\bar{c}$
1 1 1	abc

Σm – sum of min term

2 variable K map

K cell $- 2^2 = 4$

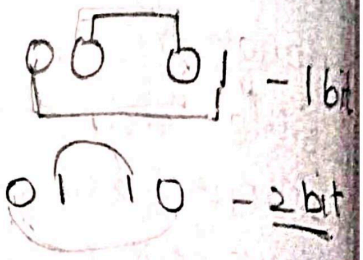
a \ b	$\bar{b}$ (0)	b (1)
$\bar{a}$ (0)	0	1
a (1)	2	3

3 variable K map

K cell $- 2^3 = 8$

a \ bc	$\bar{b}\bar{c}$ (00)	$\bar{b}c$ (01)	bc (11)	$b\bar{c}$ (10)
$\bar{a}$ (0)	0	1	3	2
a (1)	4	5	7	6

→ chain rule



4 variables K map

ab \ cd	$\bar{c}\bar{d}$ (00)	$\bar{c}d$ (01)	cd (11)	$c\bar{d}$ (10)
$\bar{a}\bar{b}$ (00)	0	1	3	2
$\bar{a}b$ (01)	4	5	7	6
ab (11)	8	9	11	10
$a\bar{b}$ (10)	12	13	15	14

For a given function $f(a,b) = \Sigma m(0,2)$ simplify using K map

1) $f(a,b) = \Sigma m(0,2)$

variable cell

$2^2 = 4$

a \ b	$\bar{b}$	b
$\bar{a}$	0	1
a	2	3

a \ b	$\bar{b}$	b
$\bar{a}$	1	1
a	1	2

1 - isolated cell
2 - pairs
4 - quad
8 - octal

1) can be grouped in the form
1, 2, 4, 8 ...

2) can be grouped adjacent cell

$f(a,b) = \bar{b}$

$\bar{a}\bar{b} + a\bar{b}$

$\bar{b}(\bar{a} + a)$

$\bar{b}(1)$

$\bar{b}$

2) $f(a,b) = \Sigma m(0,3)$

variable cell $2^2 = 4$

a \ b	$\bar{b}$	b
$\bar{a}$	1	1
a	2	1

$f(a,b) = \bar{a}\bar{b} + ab$

3 variable K map

$f(a,b,c) = \Sigma m(0,1,4,5)$

variable cell $= 2^3 = 8$

a \ bc	00	01	11	10
$\bar{a}$	1	1	3	2
a	1	1	7	6

$f(a,b,c) = \bar{b}$

$\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + a\bar{b}\bar{c} + a\bar{b}c$

$\bar{b}\bar{c}(\bar{a}+a) + c\bar{b}(a+\bar{a})$

$\bar{b}(c+\bar{c})$

$\bar{b}$

4 variable K map

$f(a,b,c,d) = \Sigma m(0,1,4,5,7,8,9,12,13,15)$

ab \ cd	$\bar{c}\bar{d}$	$\bar{c}d$	cd	$\bar{c}\bar{d}$
$\bar{a}\bar{b}$	1	1	3	2
$\bar{a}b$	1	1	1	6
ab	1	1	11	10
$a\bar{b}$	1	1	1	14

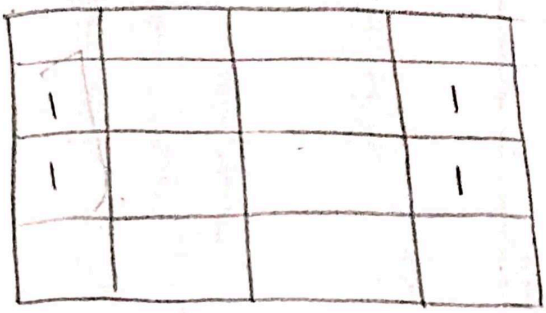
$2^4 = 16$

$\bar{a}b\bar{c}d + \bar{a}bcd + ab\bar{c}d + abcd$

$bd(\bar{a}\bar{c} + \bar{a}c + a\bar{c} + ac)$

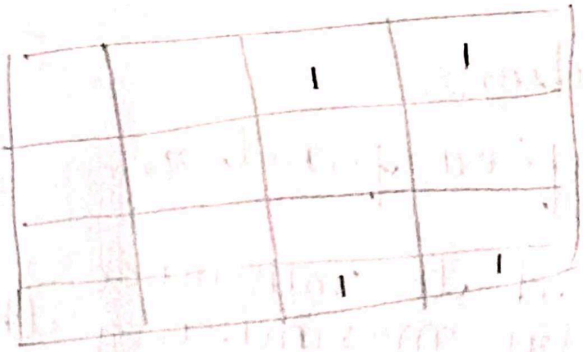
$f(a,b,c,d) = \bar{c} + bd$

Adjancies in Karnaugh Map

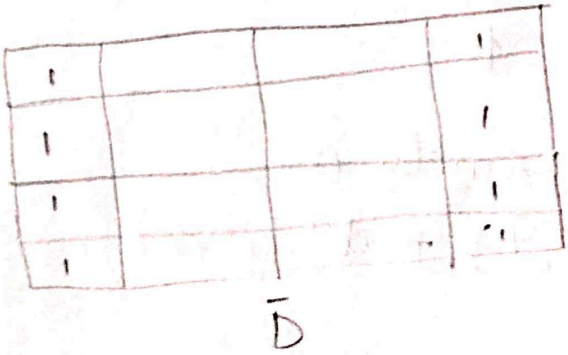


$B\bar{D}$

$\bar{B}$

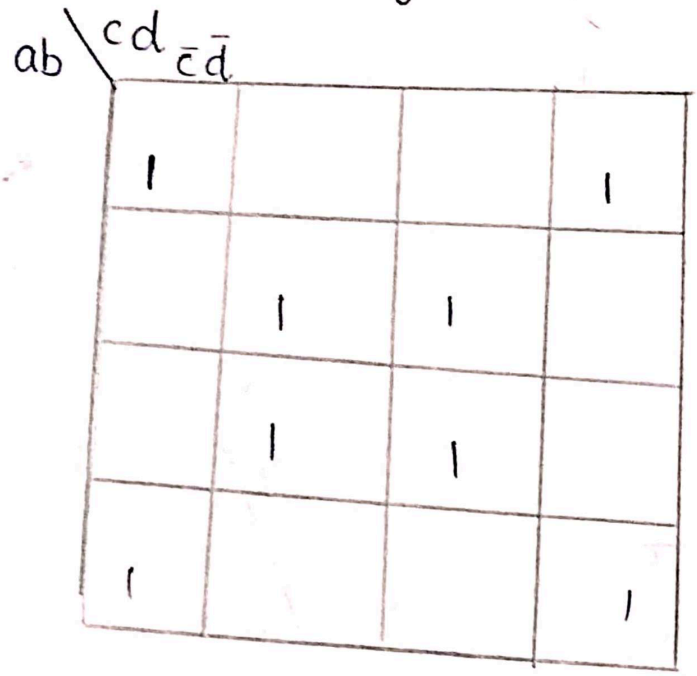


$\bar{B}C$



$\bar{D}$

consider a 4 variable k map and generate a logic expression



$$f(a,b,c,d) = bd + \bar{b}\bar{d}$$

$$b \odot d$$

Don't care conditions

incompletely specified function

Don't cares are used to minimize boolean function by maximizing the cube

On K-map don't cares are labelled with symbol x, –, ϕ

Simplify the given function

$$f(a,b,c) = \Sigma m(0,4,7) + d(1,2,6)$$

a \ bc	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$	1	1	3	2
a	1	5	1	6

(without don't case)

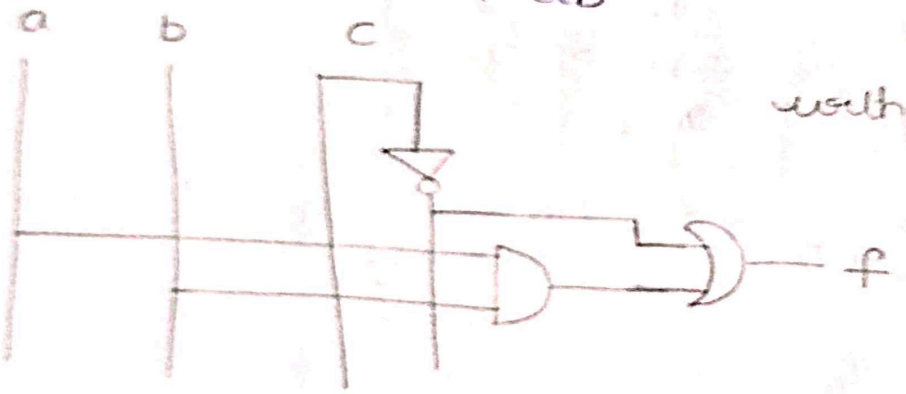
$f(a,b,c) = \bar{b}\bar{c} + abc$

a \ bc	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$	1	X	3	X
a	1	5	1	X

(with don't care)

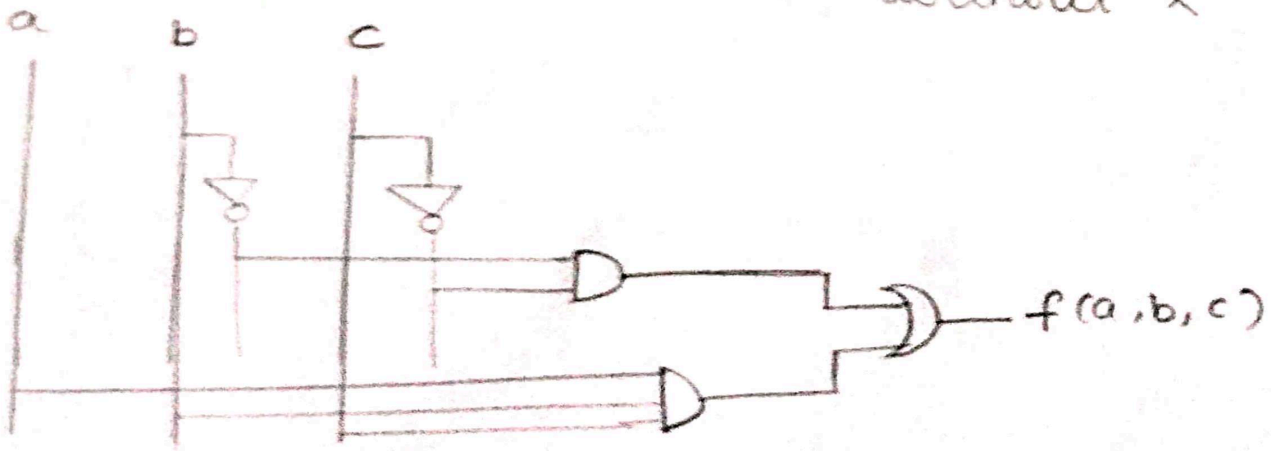
$x = 1$

$f(a,b,c) = \bar{c} + ab$



with x

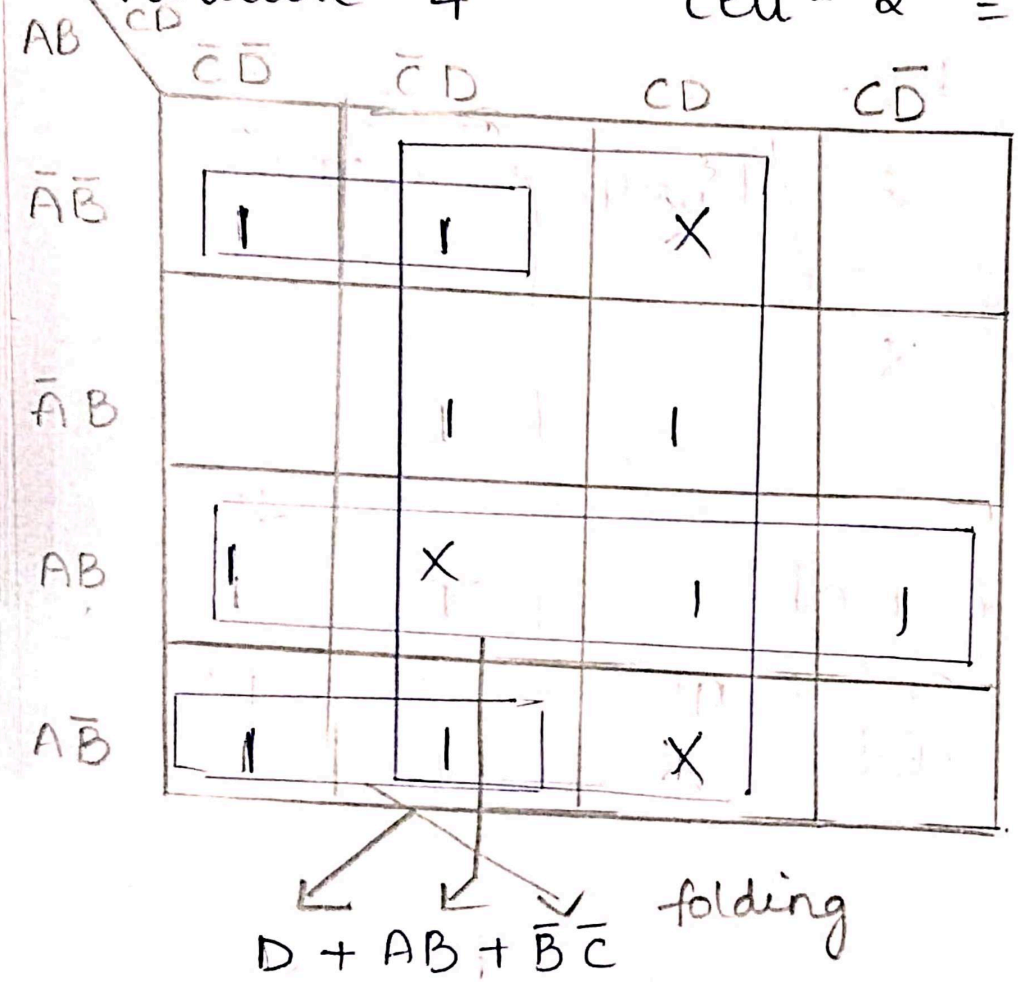
without x



Minimize the flg boolean function using K-Map

$$f(A,B,C,D) = \sum m(0,1,5,7,8,9,12,14,15) + d(3,11,13)$$

variable - 4 cell - $2^4 = 16$



5-Variable K Map

1. $f(A,B,C,D,E) = \Sigma m(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$

2. $f(A,B,C,D,E) = \Sigma m(1,4,8,10,11,20,22,24,25,26) + d(0,12,16,17)$

$\bar{a}$

be \ de	$\bar{d}\bar{e}$	$\bar{d}e$	de	$d\bar{e}$
$\bar{b}\bar{c}$	0 1	1	3	2
$\bar{b}c$	4 1	5	7	6
bc	12	13 1	15 1	14
$b\bar{c}$	8	9 1	11 1	10

$\ddot{a}$

be \ de	$\bar{d}\bar{e}$	$\bar{d}e$	de	$d\bar{e}$
$\bar{b}\bar{c}$	0	1 1	3	2
$\bar{b}c$	4	5 1	7	6
bc	12	13 1	15	14
$b\bar{c}$	8	9 1	11 1	10

$$be + \bar{a}\bar{b}\bar{e} + \ddot{a}\bar{d}e$$

$$\bar{A}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}d + a\bar{b}c\bar{e} + ab\bar{c}\bar{e} + a\bar{c}\bar{d}$$

2. $f(a,b,c,d,e) = \Sigma m(1,4,8,10,11,20,22,24,26,25) + d(0,12,16,17)$.

$\bar{a}$

bc \ de	$d\bar{e}$	de	de	$d\bar{e}$
$\bar{b}\bar{c}$	0	1	3	2
$\bar{b}c$	4	5	7	6
bc	12	13	15	14
$b\bar{c}$	8	9	11	10

a

bc \ de	$d\bar{e}$	de	de	$d\bar{e}$
$\bar{b}\bar{c}$	16	17	19	18
$\bar{b}c$	20	21	23	22
bc	28	29	31	30
$b\bar{c}$	24	25	27	26

$\bar{a}\bar{d}\bar{e} + \bar{a}\bar{b}\bar{c}\bar{d} + \bar{b}\bar{c}\bar{d} + \bar{a}b\bar{c}\bar{e} +$

(Folding)

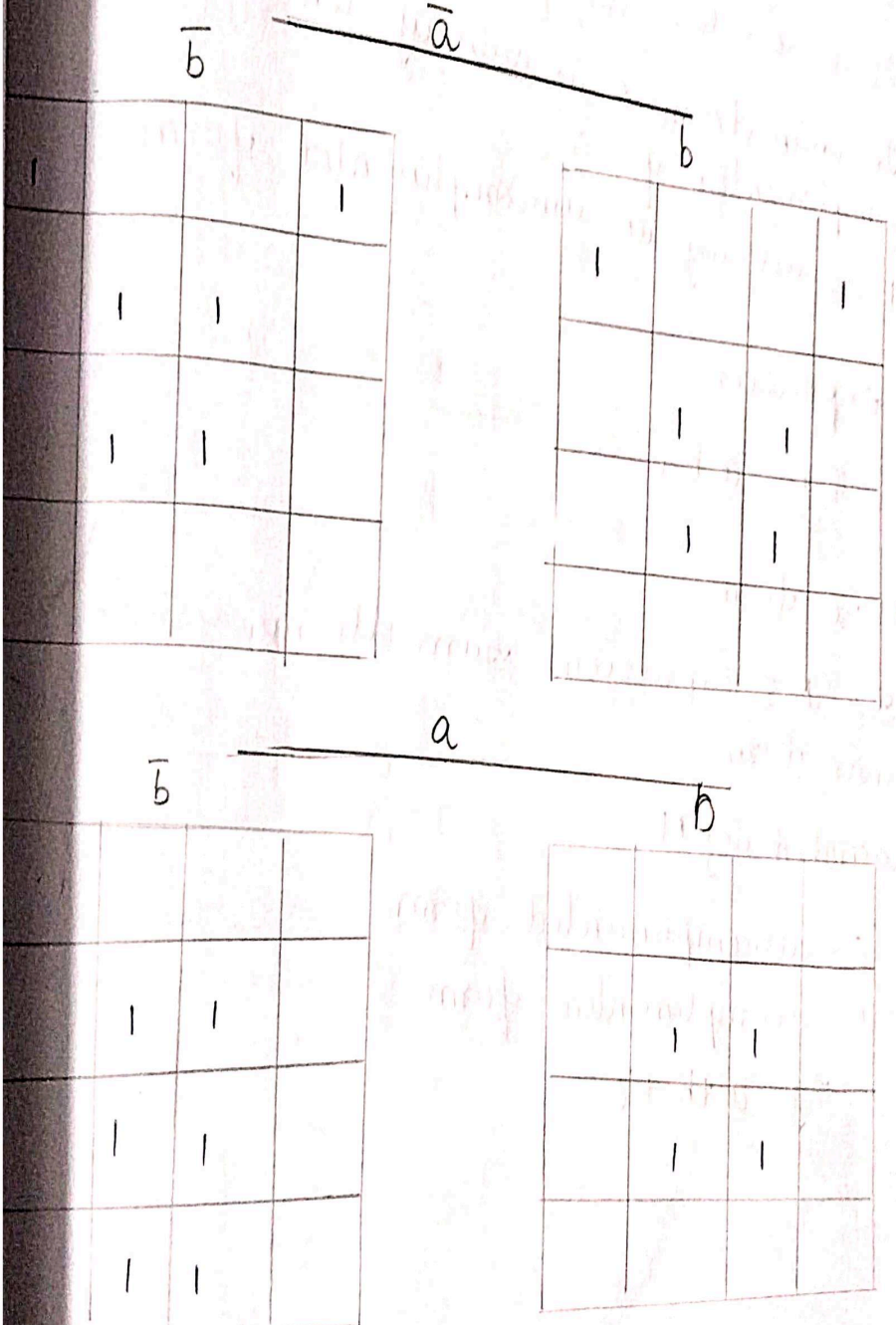
$a\bar{c}\bar{d} + a\bar{b}c\bar{e} + ab\bar{c}\bar{e}$

$\bar{a}\bar{d}\bar{e} + \bar{b}\bar{c}\bar{d} + \bar{a}b\bar{c}d + a\bar{c}d +$

$a\bar{b}c\bar{e} + ab\bar{c}\bar{e}$

6 variable K-map

f(a, b, c, d, e, f)



$$y = d\bar{f} + \bar{a}\bar{c}\bar{d}\bar{f} + a\bar{b}cf$$

Minterm and Max term

In SOP expression product term is called Min term

It is denoted by m

In min term 0 is always in complemented form

1 is always in uncomplimented form

~~expression~~

Eg: $\bar{a}\,\bar{b}\,c$

Max term

In POS expression sum terms called max term

denoted by M

0 - uncomplimented form

1 - complimented form

Eg: $a + b + \bar{c}$

	a b c	min term	Max term
0	000	$\bar{a}\bar{b}\bar{c}$	$a+b+c$
1	001	$\bar{a}\bar{b}c$	$a+b+\bar{c}$
2	010	$\bar{a}b\bar{c}$	$a+\bar{b}+c$
3	011	$\bar{a}bc$	$a+\bar{b}+\bar{c}$
4	100	$a\bar{b}\bar{c}$	$\bar{a}+b+c$
5	101	$a\bar{b}c$	$\bar{a}+b+\bar{c}$
6	110	$ab\bar{c}$	$\bar{a}+\bar{b}+c$
7	111	abc	$\bar{a}+\bar{b}+\bar{c}$

Realization of POS expression

- $F(A,B,C) = \pi M(0,1,6,7)$
- $F(A,B,C) = \pi M(0,1,6,7) \cdot d(2,3)$

variable - 3 cell - $2^3 = 8$

a \ bc	bc	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$
a	0 0	0 1	3	2
$\bar{a}$	4	5	0 7	6 0

$f(a,b,c) = (a+b) \cdot (\bar{a}+\bar{b})$

a \ bc	bc	$b\bar{c}$	$\bar{b}\bar{c}$	$\bar{b}c$
a	0 (0)	1 (0)	3 (x)	2 (x)
$\bar{a}$	4	5	7 (0)	6 (0)

$f(a,b,c) = a \cdot \bar{b}$

Design of Multiple operation output circuit

Design a ckt with 4 input and 3 output which realise the function

$F1(a,b,c,d) = \Sigma m(11,12,13,14,15)$

$F2(a,b,c,d) = \Sigma m(3,7,11,12,13,15)$

$F3(a,b,c,d) = \Sigma m(3,7,12,13,14,15)$

Limitations
PIs and essential PIs

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

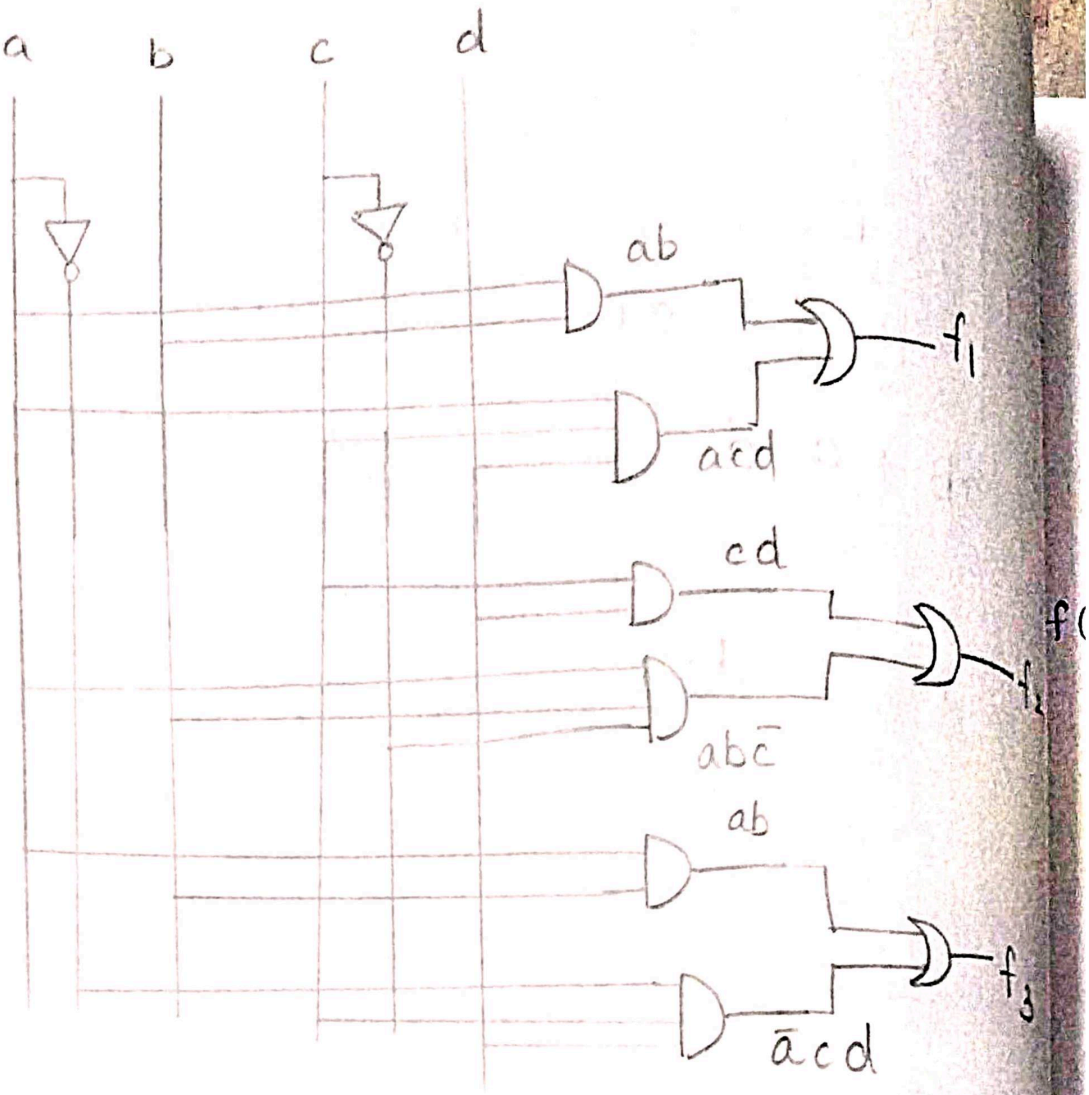
$f_1 = ab + acd$

		1	
		1	
1	1	1	
		1	

$f_2 = cd + ab\bar{c}$

		1	
		1	
1	1	1	1

$f_3 = ab + \bar{a}cd$



For a given function minimize using K-Map

$$f(a,b,c) = \bar{a}bc + ab + \bar{a}c$$

Note:- 1. change the equation into standard exp

2. generate min term

3. use K-Map generate minimize eq

$$f(a,b,c) = \bar{a}bc + ab(c+\bar{c}) + \bar{a}c(b+\bar{b})$$

$$= \bar{a}bc + abc + ab\bar{c} + \bar{a}bc + \bar{a}\bar{b}c$$

$$\bar{a}bc + \bar{a}bc = \bar{a}bc$$

$$f(a,b,c) = \bar{a}bc + abc + ab\bar{c} + \bar{a}\bar{b}c$$

011	111	110	001
3	7	6	1

$$f(a,b,c) = \Sigma m(1,3,6,7)$$

0	1 1	3 1	2
4	5	7 1	6 1

$$f = \bar{a}c + ab$$

Limitations

If number of variables increased it is difficult to analyze K-Map

It is a trial and error method

It is a unprogrammable method

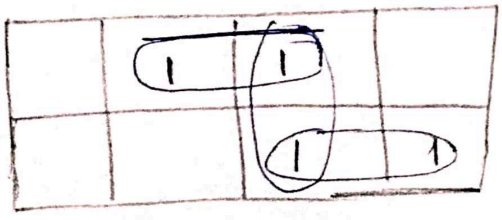
To overcome above limitation the QM method or tabular method can be used

Prime Implicants and Essential Prime Implicants (PI)

Prime implicants (PI)

Prime implicants are all possible cubes on the K-Map are called PI's

Essential prime implicants are the minimum number of PI's available on K Map to cover all one's or zero's



$\bar{a}c$

ab

bc

3 PI's

$\bar{a}c$

ab

Tabular Method

Step 1 :- List all min terms in binary form

Step 2 :- Arrange the min terms according to number of ones

Step 3 :- Compare each binary number with every term in adjacent next higher category and if they differ by only one position put a check mark '–' on the position that they differ

Step 4 Apply the same process described in step 3 for resultant column until the further elimination of literals

Step 5 List all prime implicants

Step 6 :- Select the minimum no. of PI's which must cover all the min term

For a given function $f(a, b, c, d) = \Sigma m(0, 2, 3, 6, 7, 8, 10, 12, 13)$ using Quine - Mccluskey method or tabular method.

	8 A	4 B	2 C	1 D	
0	0	0	0	0	– 0
2	0	0	1	0	– 1
3	0	0	1	1	– 2
6	0	1	1	0	– 2
7	0	1	1	1	– 3
8	1	0	0	0	– 1
10	1	0	1	0	– 2
12	1	1	0	0	– 2
13	1	1	0	1	– 3

group	min term	Min term	Remarks
1	0	0 0 0 0	✓
2	2	0 0 1 0	✓
	8	1 0 0 0	✓
3	3	0 0 1 1	✓
	6	0 1 1 0	✓
	10	1 0 1 0	✓
	12	1 1 0 0	✓
4	7	0 1 1 1	✓
	13	1 1 0 1	✓

group	min term	Min term	Remarks
1	0,2	0 0 – 0	✓
	0,8	– 0 0 0	✓
2	2,3	0 0 1 –	✓
	2,6	0 – 1 0	✓
	2,10	– 0 1 0	✓
	8,10	1 0 – 0	✓
	8,12	1 – 0 0	
3	3,7	0 – 1 1	✓
	6,7	0 1 1 –	✓
	12,13	1 1 0 –	
1	0,2,8,10	– 0 – 0	
	0,8,2,10	– 0 – 0	
2	2,3,6,7	0 – 1 –	
	2,6,3,7	0 – 1 –	

Conclusion

The uncovered midterms are

8, 12

12, 13

0, 2, 8, 10

2, 3, 6, 7

Draw a PI chart using uncovered minterm

Prime implicant	Minterms								
	0	2	3	6	7	8	10	12	13
8,12 $A\bar{C}\bar{D}$						X		X	
12,13 $AB\bar{C}$								X	(X)
0,2,8,10 $\bar{B}\bar{D}$	(X)	X				X	(X)		
2,3,6,7 $\bar{A}C$		X	(X)	(X)	(X)				

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

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

for a given function $f(a,b,c,d) = \Sigma m(2,4,5,9,12,13)$. Simplify using QM method.

Sol:

	8	4	2	1	
2	0	0	1	0	- 1
4	0	1	0	0	- 1
5	0	1	0	1	- 2
9	1	0	0	1	- 2
12	1	1	0	0	- 2
13	1	1	0	1	- 3

Group	Min term	Min term	Remarks
1	2	0 0 1 0	
	4	0 1 0 0	✓
2	5	0 1 0 1	✓
	9	1 0 0 1	✓
	12	1 1 0 0	✓
3	13	1 1 0 1	✓
1	4, 5	0 1 0 —	✓
	4, 12	— 1 0 0	✓
2	5, 13	— 1 0 1	✓
	9, 13	1 — 0 1	
	12, 13	1 1 0 —	✓
1	4, 5, 12, 13	— 1 0 —	
	4, 12, 5, 13	— 1 0 —	

Conclusion

2

9, 13

4, 5, 12, 13.

Draw PI chart using uncovered min term

PI	2	4	5	9	12	13
2 $\bar{A}B\bar{C}\bar{D}$	(x)					
$A\bar{C}D$ 9,13				(x)		x
4,5,12,13 $B\bar{C}$		(x)	(x)		(x)	x

$m(2, 4, 5, 9, 12, 13)$

ab \ cd	$\bar{c}\bar{d}$	$\bar{c}d$	cd	$c\bar{d}$
$a\bar{b}$	0	1	3	21
ab	41	51	7	6
ab	12[1]	13[1]	15	14
$a\bar{b}$	8	91	11	10

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

for a given function $f(a,b,c,d)$

$=\sum m(0,1,2,5,6,7,8,9,10,14)$

	8	4	2	1	
0	0	0	0	0	-0
1	0	0	0	1	-1
2	0	0	1	0	-1
5	0	1	0	1	-2
6	0	1	1	0	-2
7	0	1	1	1	-3

8	1	0	0	0	-1
9	1	0	0	1	-2
10	1	0	1	0	-2
14	1	1	1	0	-3

Group	Min term	Min term	Remarks
1	0	0 0 0 0	
2	1	0 0 0 1	
	2	0 0 1 0	
	8	1 0 0 0	
3	5	0 1 0 1	
	6	0 1 1 0	
	9	1 0 0 1	
	10	1 0 1 0	
4	7	0 1 1 1	
	14	1 1 1 0	
1	0, 1	0 0 0 -	
	0, 2	0 0 - 0	
	0, 8	- 0 0 0	
	1, 5	0 - 0 1	
	1, 9	- 0 0 1	

incomplt

Draw a PI chart using uncovered min terms

PI's	0	1	2	5	6	7	8	9	10	14
1, 5		×		×						
5, 7 $A\bar{B}D$				×		×				
6, 7						×				
8, 9 $B\bar{C}$										
0, 2, 8, 10 $C\bar{D}$										
6, 10, 14										

$$f(a,b,c,d) = \bar{A}BD + \bar{B}\bar{C} + C\bar{D}$$

The given function ~~f(a,b,c,d)~~ =

ab \ cd	$\bar{c}\bar{d}$	$\bar{c}d$	cd	$c\bar{d}$
$\bar{a}\bar{b}$	0	1	3	2
$\bar{a}b$	4	5	7	6
ab	12	13	15	14
$a\bar{b}$	8	9	11	10

$$f(a,b,c,d) = C\bar{D} + D\bar{A}B + \bar{C}\bar{B}$$

PI's	0	1	2	5	6	7	8	9	10
(0,1,8,9) $\bar{B}\bar{C}$	x	x					x	(x)	
(0,2,8,10) $\bar{B}\bar{D}$	x		x				x		x
(2,6,10,14) $C\bar{D}$			x		x				x
(1,5) $\bar{A}\bar{C}D$		x		x					
(5,7) $\bar{A}BD$				x		x			
(6,7) $\bar{A}BC$					x	x			

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

Q) For a given function $f(a,b,c,d) = \Sigma m(0,1,2,5,6,7)$

PI's	0	1	2	5	6	7
(0,1) $\bar{A}\bar{B}$	x	x				
(0,2) $\bar{A}\bar{C}$	x		x			
(1,5) $\bar{B}C$		x		x		
(2,6) $B\bar{C}$			x		x	
(5,7) AC				x		x
(6,7) AB					x	x

Trial & Error method

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

K-Map

C

Q) For the following given function

$$f(A,B,C,D) = \Sigma m(2,3,7,9,11,13) + \Sigma d(1,10,15)$$

Note:- The don't care columns are omitted when forming the prime implicant chart.

	8	4	2	1	
	A	B	C	D	
2	0	0	1	0	- 1
3	0	0	1	1	- 2
7	0	1	1	1	- 3
9	1	0	0	1	- 2
11	1	0	1	1	- 3
13	1	1	0	1	- 3

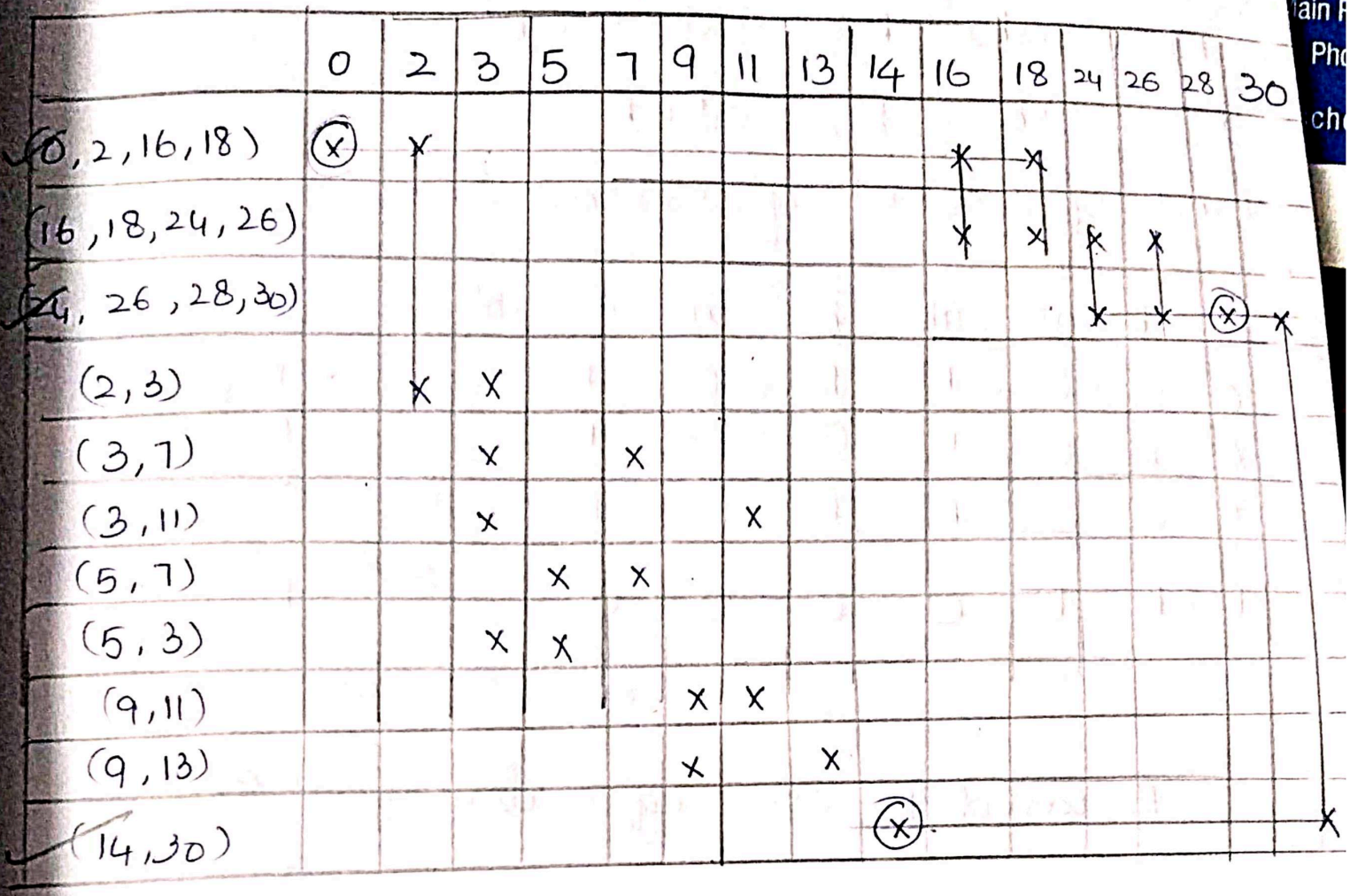
PI's chart

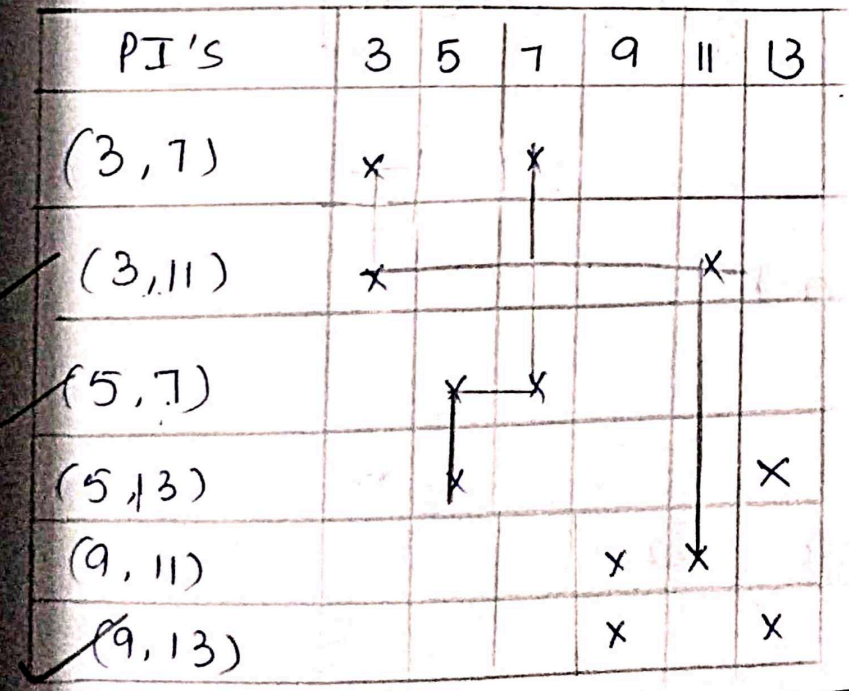
PI's	2	3	7	9	11	13
(1, 3, 9, 11)		x		x	x	
(2, 3, 10, 11)	(x)	x			x	
(3, 7, 11, 15)		x	(x)		x	
(9, 11, 13, 15)				x	x	(x)

$\bar{b}c + cd + ad$

Q Find a minimum sum of products expression for the following function.

$f(a, b, c, d) = \Sigma m(0, 2, 3, 5, 7, 9, 11, 13, 14, 16, 18, 24, 26, 28, 30)$.





$\bar{B}\bar{C}\bar{E} + AB\bar{E} + \bar{A}\bar{C}DE + \bar{A}\bar{B}CE + \bar{A}B\bar{D}E + BCD\bar{E}$

Equivalent functions

$F1 = (ab)'$ $F2 = (ab)' + ab'$

$F3 = ab$ $F4 = a' + b$

Find equivalent function

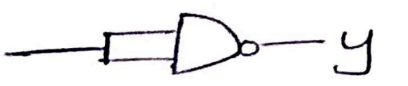
a	b	f_3 ab	f_1 $\overline{ab}$	$\bar{b}$	$a\bar{b}$	f_2 $(ab)' + ab'$	$\bar{a}$	$f_4 = a' + b$
0	0	0	1	1	0	1	1	1
0	1	0	1	0	0	1	1	1
1	0	0	1	1	1	1	0	0
1	1	1	0	0	0	0	0	1

F_1 and F_2 are equivalent functions

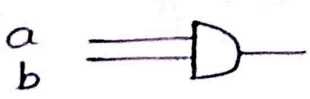
Realization of basic gates using universal gates

NAND

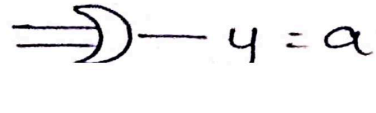
NOT

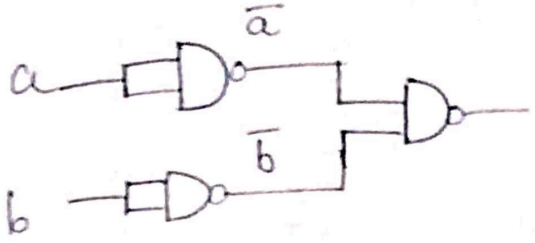


AND



OR



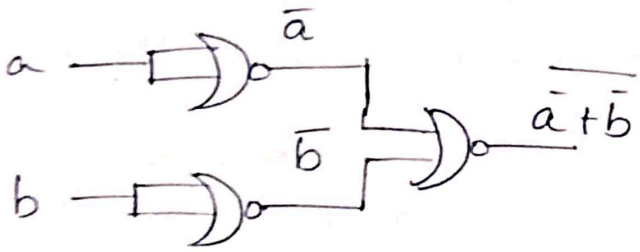


$a \cdot b \cdot \quad \bar{a} + \bar{b} \neq a + b$

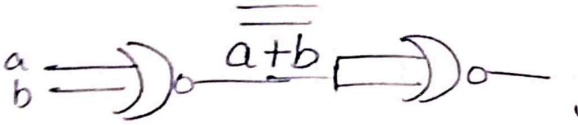
NOR

Not

AND



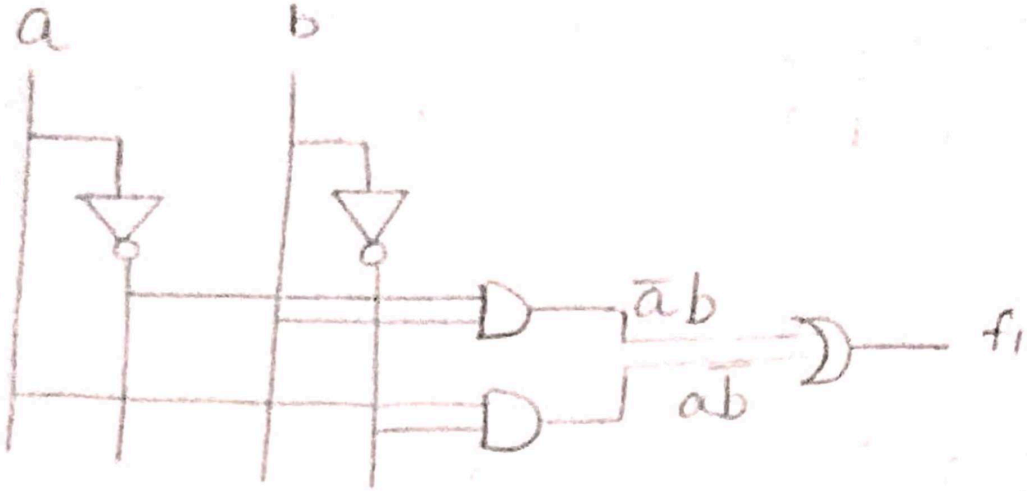
OR



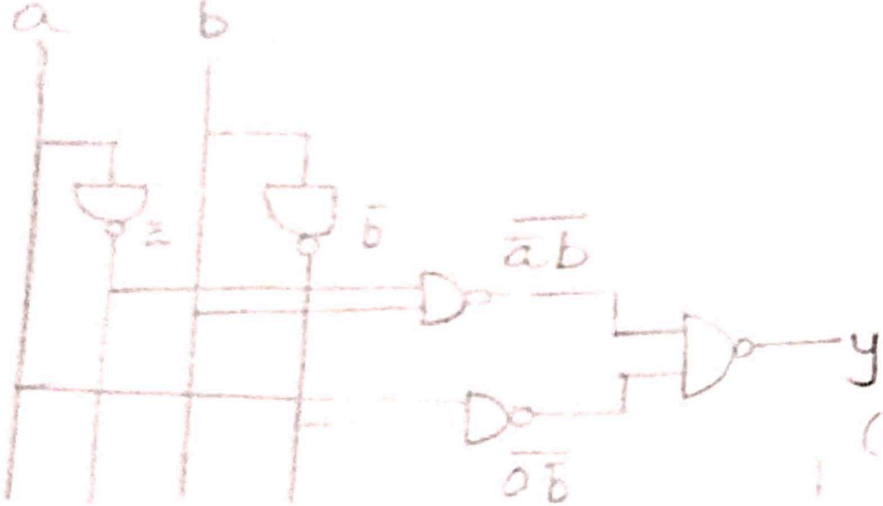
Realize the given function $y(a,b) = \bar{a}b + a\bar{b}$ using basic gates.

Steps

1. Realizing using basic gates
2. change all basic gates with NAND gate
3. check the functionality


a
b
$\bar{a}b$
$a\bar{b}$
f_1

NAND


a
b
$\bar{a}$
$\bar{b}$
$\overline{\bar{a}b}$
$\overline{a\bar{b}}$
$y = \overline{(\bar{a}\cdot b)\cdot(a\bar{b})}$
$\bar{\bar{a}}\cdot\bar{b} + a\bar{b}$
$\bar{a}b + a\bar{b}$

UNIT-2

Part - 1
Number System

Part - 1
Introduction to combination logic circuits

Number System

In a digital system deals with binary number system. It consists of only 2 digits 0 and 1 ON and OFF respectively. The base or radix for binary number system is 2.

One single digit in a binary no. system is called bit.

Considered a string of binary number

0101 1010 1010 1111
MSB Nibble Bit LSB

- Group of 4 binary bits is called nibble
- Group of 8 binary bits is called byte
- Group of 16 binary bits is called word
- Group of 32 binary bits is called double word
- Extreme right side is called least significant bit
- Extreme left-most significant bit

Types of Number system

1) Decimal Number System :- base or radix (10)
2) Binary Number System - base or radix (2)
3) Hexadecimal Number system - base/radix (16)
4) Octal Number System - base/radix (8)

Binary Number System

It is the simplest form of number system. It consists of only two digits in base which are 0 and 1 or on and off respectively

- For a given deci number.

For a given binary number system convert into Decimal

Procedure :- To obtain decimal number multiply with base for the given number

Ex :- $10110_{(2)} = \_\_\_\_\_{(10)}$

1	0	1	1	0
2^4	2^3	2^2	2^1	2^0

$0 \times 1 = 0$
$1 \times 2 = 2$
$1 \times 4 = 4$
$0 \times 8 = 0$
$1 \times 16 = 16$
22

For a given binary number convert into decimal number

$1001.1101_2 = \_\_\_\_\_ _{10}$

1	0	0	1	.	1	1	0	1
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}

$8+0+0+1 \cdot 1\times(0.5)+1(0.25)+0+1\times(0.0625)$

9.8125_{10}

Decimal Number System

Decimal system consists of 10 digits starting from 0 to 9. This number system has ten as its base.

Ex: 246.45

$2*10^2+4*10^1+6*10^0 \cdot 4*10^{-1}+5*10^{-2}$

(1) For a given decimal number system 156 convert into binary number system

Procedure :- To obtain binary number system divide base into the given number

2 | 156
2 | 78 - 0
2 | 39 - 0
2 | 19 - 1
2 | 9 - 1
2 | 4 - 1
2 | 2 - 0
2 | 1 - 0

10011100_2

(ii) 145

2 | 145
2 | 72 - 1
2 | 36 - 0
2 | 18 - 0
2 | 9 - 0
2 | 4 - 1
2 | 2 - 0
1 - 0

10010001_2

Binary	Decimal	Hexadecimal	Octal
0 0 0 0	0	0	0
0 0 0 1	1	1	1
0 0 1 0	2	2	2
0 0 1 1	3	3	3
0 1 0 0	4	4	4
0 1 0 1	5	5	5
0 1 1 0	6	6	6
0 1 1 1	7	7	7
1 0 0 0	8	8	10
1 0 0 1	9	9	11
1 0 1 0	10	A	12
1 0 1 1	11	B	13
1 1 0 0	12	C	14
1 1 0 1	13	D	15
1 1 1 0	14	E	16
1 1 1 1	15	F	17

Octal Number System

- Characteristics of the octal number system are as follows
  - Uses eight digits - 0, 1, 2, 3, 4, 5, 6, 7
  - Also called as base 8 number system

For a given octal number system convert to decimal

(i) 19570_8

1	2	5	7	0
8^4	8^3	8^2	8^1	8^0

$4096 + 1024 + 320 + 56$

5496_{10}

Hexadecimal Number system

- Characteristics of hexadecimal number system are as follows -
  - Uses 10 digits and 6 letters,
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Letters represent the numbers starting from 10.
    A = 10, B = 11, C = 12, D = 13, E = 14, F = 15
  - Also called as base 16 number system

For a given hexadecimal number convert to decimal

(i) 89

$8 \times 16^1 + 9 \times 16^0 = 137$

(ii) CFF_{16} =

$12 \times 16^2 + 15 \times 16^1 + 15 \times 16^0$

$3072 + 240 + 15$

3327

(iii) $8F_{16}$ =

$8 \times 16^1 + 15 \times 16^0$

$128 + 15$

143

(iv) $19FDE_{16}$ =

$1 \times 16^4 + 9 \times 16^3 + 15 \times 16^2 + 13 \times 16^1 + 14 \times 16^0$

$65536 + 36864 + 3840 + 208 + 14$

106462.

(ii) CFF_{16} =

$12 \times 16^2 + 15 \times 16^1 + 15 \times 16^0$

$3072 + 240 + 15$

3327

(iii) $8f_{16}$ =

$8 \times 16^1 + 15 \times 16^0$

$128 + 15$

143

(iv) $19fDE_{16}$ =

$1 \times 16^4 + 9 \times 16^3 + 15 \times 16^2 + 13 \times 16^1 + 014 \times 16$

$65536 + 36864 + 3840 + 208 + 14$

106462.

Convert the given number 24.3_8 to binary ~~and~~ to decimal number system

(i) 24.3_8

421 2 4 . 3

010 100 . 011

010100.011_2

0 1 0 1 0 0 . 0 1 1

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 . 2^{-1}\ 2^{-2}\ 2^{-3}$

$1 \times 16 + 1 \times 4 + 1 \times 0.25 + 1 \times 0.125$

$16 + 4 + 0.25 + 0.125$

20.375

(ii) 25_8

2 5
010 101

010101_2

convert the given decimal number 39.12 to Binary

$39.12_{10} =$ _____ $_2$

2 | 39
2 | 19 - 1
2 | 9 - 1
2 | 4 - 1
2 | 2 - 0
1 - 0

100111

Taking the fraction part

Fraction	fraction x 2	Rem new f	Integer
0.12	0.24	0.24	0
0.24	0.48	0.48	0
0.48	0.96	0.96	0
0.96	1.92	0.92	1
0.92	1.84	0.84	1
0.84	1.68	0.68	1

100111.000111 ...

a) Convert decimal number into binary numbers b) add the binary number and convert the result into decimal equivalent.

Binary addition : The rules of binary addition

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10$ sum 0 carry 1

$1 + 1 + 1 = 11$ sum 1 carry 1

Addition

(i)

1011	— 11	1 1 1011 1011
0001	— 1	
1100	⇐ 12	10110

1011	— 11
0011	— 3
1110	14

Binary subtraction : The rules of binary subtraction

$0 - 0 = 0$

$1 - 0 = 1$

$1 - 1 = 0$

$10 - 1 = 1$
$(2 - 1 = 1)$

Convert decimal number 15 and 31 and add

```
2 | 15
2 | 7  - 1
2 | 3  - 1
  | 1  - 1
```

1111_2

```
2 | 31
2 | 15 - 1
2 | 7  - 1
2 | 3  - 1
  | 1  - 1
```

11111_2

```
  1 1 1 1 1
    1 1 1 1
-----------
  1 0 1 1 1 0
```

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$32 + 8 + 4 + 1$

$= 46$

Subtract 10001 from 11001

```
 16  8  4  2  1
  1  1  0  0  1     25
  1  0  0  0  1     17
-----------------
  0  1  0  0  0
```

```
  0  1  0  0  0
 32  8  4  2  1
        1
```

$= 8$

. 1's complement

The 1's complement of the binary number obtained by complementing each bit (0 to 1 and 1 to 0)

Ex:- 1's complement of 1100 is

0011

2's complement

The signed binary number required large electronic circuitry for addition and subtraction. Therefore a positive decimal number are expressed in sign - magnitude form

- But negative numbers are expressed in 2's complement form.

Eg:- For a given binary number convert into 2's complement.

```
      1 0 1 0 1
1's comp - 0 1 0 1 0
add 1            1
-----------------------
BN    0 1 0 1 1
```

Obtain 2's complement for given number

```
  0 1 0 1 1 0
  1 0 1 0 0 1
          1 1
        1
-----------------
  1 0 1 0 1 0
```

Introduction to Combination Circuits

- Combinational circuits
  - → Output is a function of the present inputs.
  - → Does not have state information
  - → Does not require memory

Combinational circuits provide a higher level of abstraction

- → Help in reducing design complexity
- → Reduce chip count

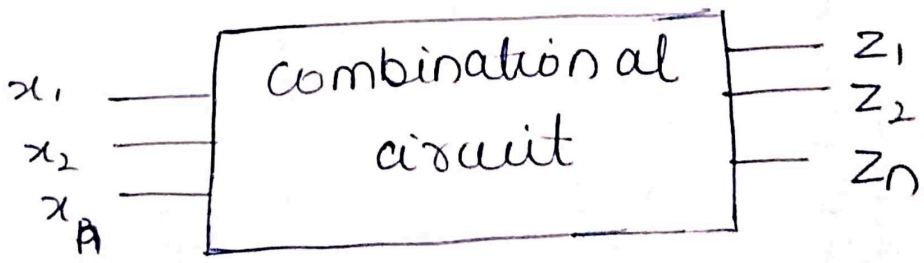
Ex:- switch, adders, substractor, comparators, decoders, encoders multiplexers, de-multiplexers (basic gates)

Block diagram

Here $x_1, x_2 \cdots x_n$ are primary inputs

Here $z_1, z_2 \cdots z_n$ are primary outputs

$$z = f(x)$$



Design of combinational circuits

Step 1 :- Draw block diagram and mention inputs and outputs

Step 2 :- Obtain logical table or behaviour table

Step 3 :- Generate output logic equations using K-Map

Step 4 :- Draw combinational circuit using logic gates

Eg :- Design 3 input AND gate.

① 

Block diagram

② logic table.

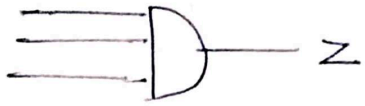
a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

③

a \ bc	$b\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$	0	1	3	2
a	4	5	(7)	6

$z = abc$

④ combination logic circuit



Design a combinational circuit for the given task.

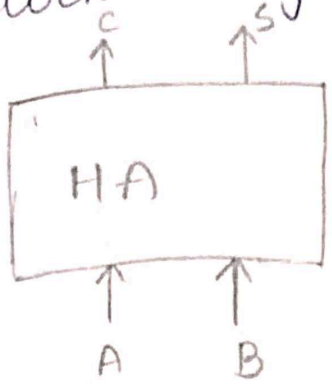
A	C	K	M	LED
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	1
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

$2^4 = 16$

Design of half adder

addition of 2 binary values with sum & carry output

Block diagram



2^2

Truth table

	A	B	C	S
0	0	0	0	0
1	0	1	0	1
2	1	0	0	1
3	1	1	1	0

a \ b

0	1
2	31

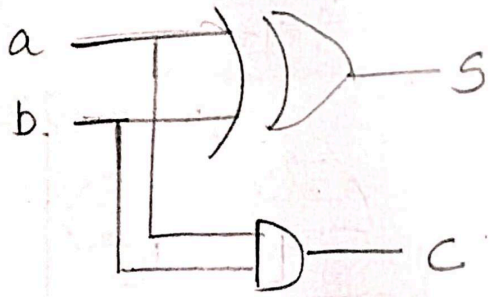
$C = a \cdot b$

0	11
21	3

$S = \bar{a}b + a\bar{b}$

$= a \oplus b$

Circuit diagram



XOR

9

Design for full adder

- Adds three 1-bit values
  - Like half-adder, produces a sum and a carry
- Adds building N-bit adders
  - Simple technique

Exercise - Design of FA using TWO HA

Logic table

	a	b	c_i	c_o	S
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

a \ bc_i	$b\bar{c}_i$	$\bar{b}c_i$	bc_i	$b\bar{c}_i$
$\bar{a}$	0	1	3 1	2
a	4	5 1	7 1	6 1

$$c_0 = bc_i + ac_i + ab_i$$

a \ bc_i	$\bar{b}\bar{c}_i$	$\bar{b}c_i$	bc_i	$b\bar{c}_i$
$\bar{a}$	0	1 1	3	2 1
a	4 1	5	7 1	6

$a \odot b$

$\overline{a \oplus b}$

$S = \bar{a}\,b\,c_i + a\,b\,\bar{c}_i + a\,b\,c_i + \bar{a}\,b\,\bar{c}_i$

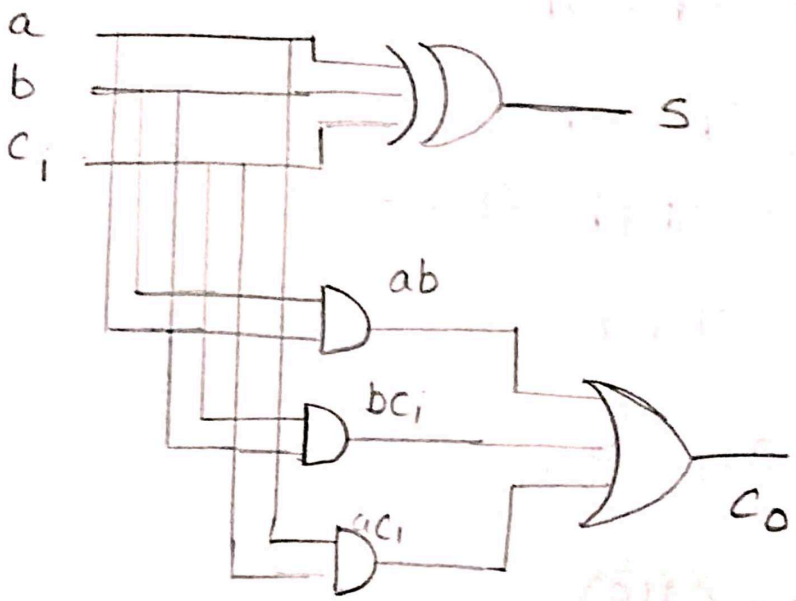
$= \bar{a}(\bar{b}\,c_i + b\,\bar{c}_i) + a(\bar{b}\,\bar{c}_i + b\,c_i)$

$= \bar{a}(b \oplus c_i) + a(b \odot c_i)$

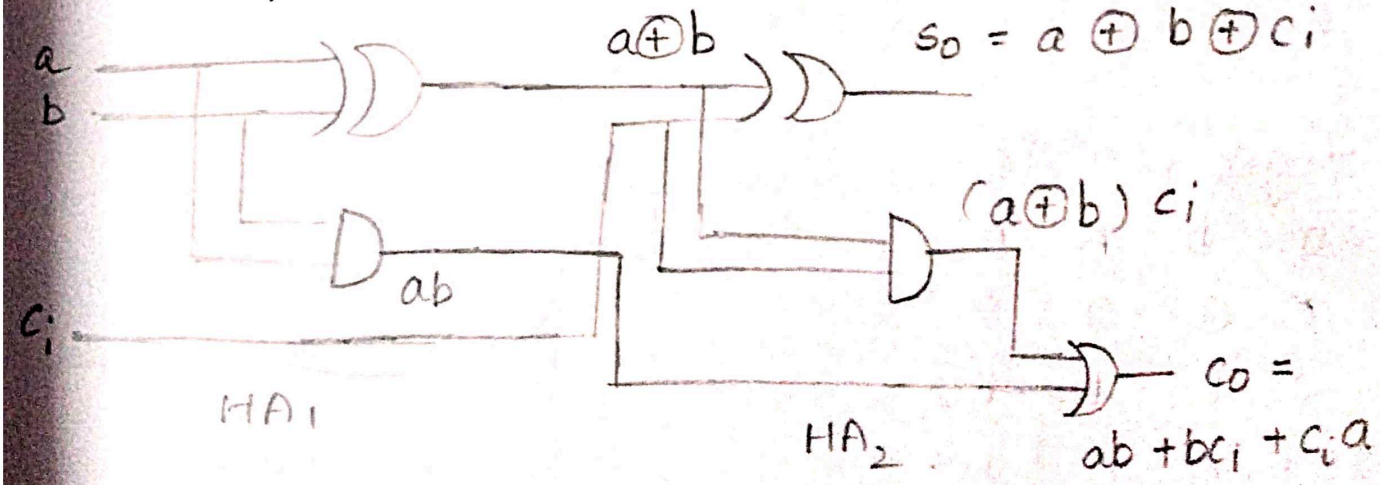
$= \bar{a}(b \oplus c_i) + a\overline{(b \oplus c_i)}$

$S = a \oplus b \oplus c$

Logic diagram



Realization of full adder using half adder.



$$C_0 = (a \oplus b) c_i + ab$$

$$(\bar{a}b + a\bar{b}) c_i + ab$$

$$\bar{a}bc_i + a\bar{b}c_i + ab$$

$$b(\bar{a}c_i + a) + a\bar{b}c_i$$

$$b(\bar{a}+a)(a+c_i) + a\bar{b}c_i$$

$$b(a+c_i) + a\bar{b}c_i$$

$$= ab + bc_i + a\bar{b}c_i$$

$$= ab + c_i(b + a\bar{b})$$

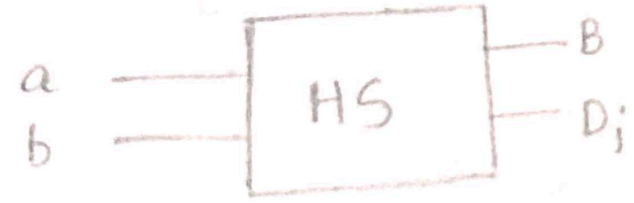
$$= ab + c_i(b + a\bar{b})$$

$$= ab + c_i(b+\bar{b})(b+a)$$

$$= ab + c_i(b+a)$$

$$= ab + ac_i + bc_i$$

Half subtractor (HS)



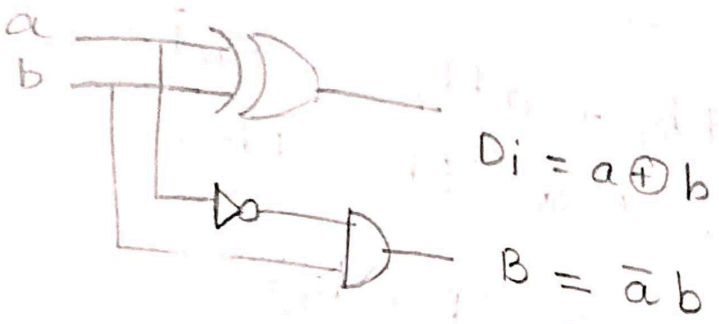
Logic table

	a	b	B	D_i
0	0	0	0	0
1	0	1	1	1
2	1	0	0	1
3	1	1	0	0

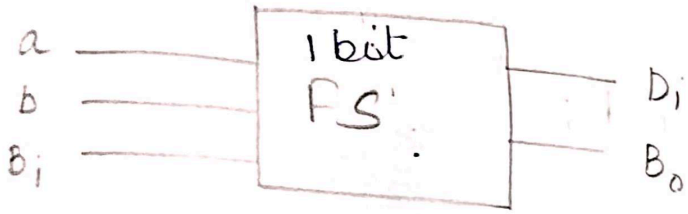
$B = \bar{a}b$

$D_i = \bar{a}b + a\bar{b}$

$= a \oplus b$



Full Substractor



Borrow = lesser - higher

	a	b	b_i	B_0	D_i
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

	$\bar{b}\bar{b}_i$	$\bar{b}b_i$	bb_i	$b\bar{b}_i$
$\bar{a}$	0	1	3	2
a	4	5	7	6

$B_0 = \bar{a}b_i + \bar{a}b + bb_i$

	$\bar{b}\bar{b_i}$	$\bar{b}b_i$	bb_i	$b\bar{b_i}$
$\bar{a}$	0	1 1	3	2 1
a	4 1	5	7 1	6

$$a\bar{b}\bar{b_i} + \bar{a}\bar{b}b_i + abb_i + \bar{a}b\bar{b_i}$$

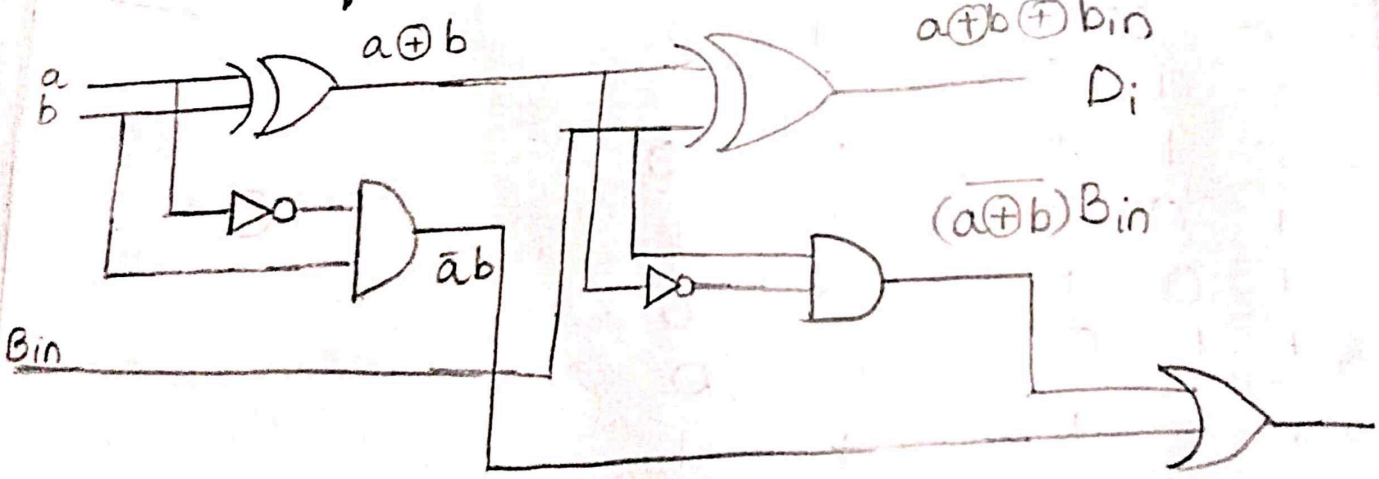
$$a(\bar{b}\bar{b_i} + bb_i) + \bar{a}(\bar{b}b_i + b\bar{b_i})$$

$$a(b \odot b_i) + \bar{a}(b \oplus b_i)$$

$$a(\overline{b \oplus b_i}) + \bar{a}(b \oplus b_i)$$

$$D_i = a \oplus b \oplus b_i$$

Realize of full subtractor using two half subtractor



$$B_0 = (\overline{a \oplus b})B_{in} + \bar{a}b$$

$$(a \odot b)B_{in} + \bar{a}b$$

$$(\bar{a}\bar{b} + ab)B_{in} + \bar{a}b$$

$$\bar{a}\bar{b}B_{in} + abB_{in} + \bar{a}b$$

$$\bar{a}(\bar{b}B_{in} + b) + abB_{in}$$

$$\bar{a}(\bar{b}+b)(\bar{b}+B_{in}) + abB_{in}$$

$$(\bar{b} \wedge B_i) \vee b$$

$$(\bar{b} \vee b) \wedge (b \vee B_i)$$

$\bar{a}b + \bar{a}B_{in} + abB_{in}$

$b(\bar{a} + aB_{in}) + \bar{a}B_{in}$ $\quad \bar{a} \vee (a \wedge B_i)$

$b(\bar{a}+a)(\bar{a}+B_{in}) + \bar{a}B_{in}$ $\quad (\bar{a} \vee a) \wedge (\bar{a} \vee B_i)$

$b(\bar{a}+B_{in}) + \bar{a}B_{in}$

$B_o = \bar{a}b + bB_{in} + \bar{a}B_{in}$

$D_i = a \oplus b \oplus B_{in}$

Design of 1 bit comparator circuit

$2^1 : 2$

1:

a	b	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

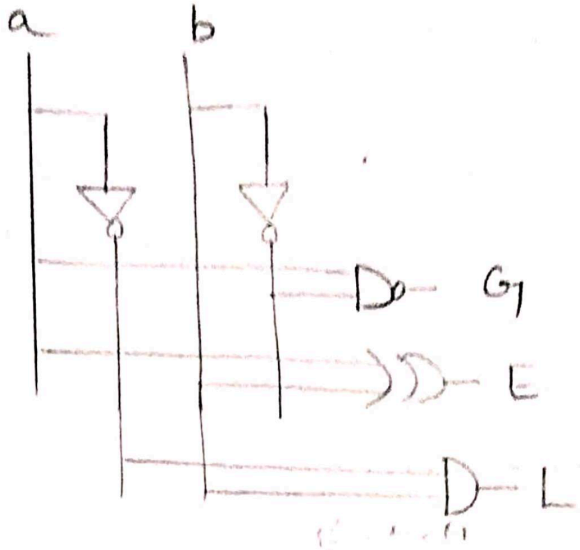
$G = a\bar{b}$

$E = \bar{a}\bar{b} + ab = a \odot b$

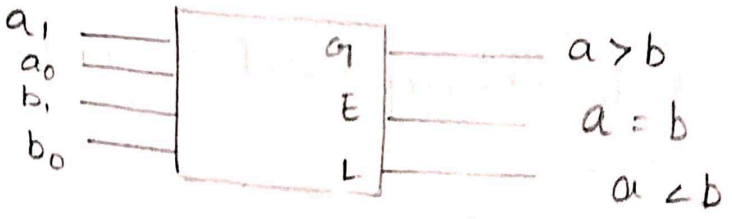
$L = \bar{a}b$

Bi)

Logic circuit



Design of 2-bit comparator circuit



	a_1	a_0	b_1	b_0	G	E	L
	2	1	2	1			
0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1
4	0	1	0	0	1	0	0
5	0	1	0	1	0	1	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	0	1
8	1	0	0	0	0	0	1
9	1	0	0	1	1	0	0
10	1	0	1	0	1	0	0
11	1	0	1	1	0	1	0
12	1	1	0	0	0	0	1
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	1	0

$a_1a_0 \backslash b_1b_0$	$\bar{b}_1\bar{b}_0$	$\bar{b}_1 b_0$	$b_1 b_0$	$b_1\bar{b}_0$
$\bar{a}_1\bar{a}_0$	0	1	3	2
$\bar{a}_1 a_0$	4 1	5	7	6
$a_1 a_0$	12 1	13 1	15	14 1
$a_1\bar{a}_0$	8 1	9 1	11	10

greater

$$\bar{b}_1 a_1 + a_0 \bar{b}_1 \bar{b}_0 + a_1 a_0 \bar{b}_0$$

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

equal

$$a_1 a_0 b_0 b_1 + \bar{a}_1 a_0 \bar{b}_1 b_0 + \bar{a}_1 \bar{a}_0 \bar{b}_1 \bar{b}_0$$

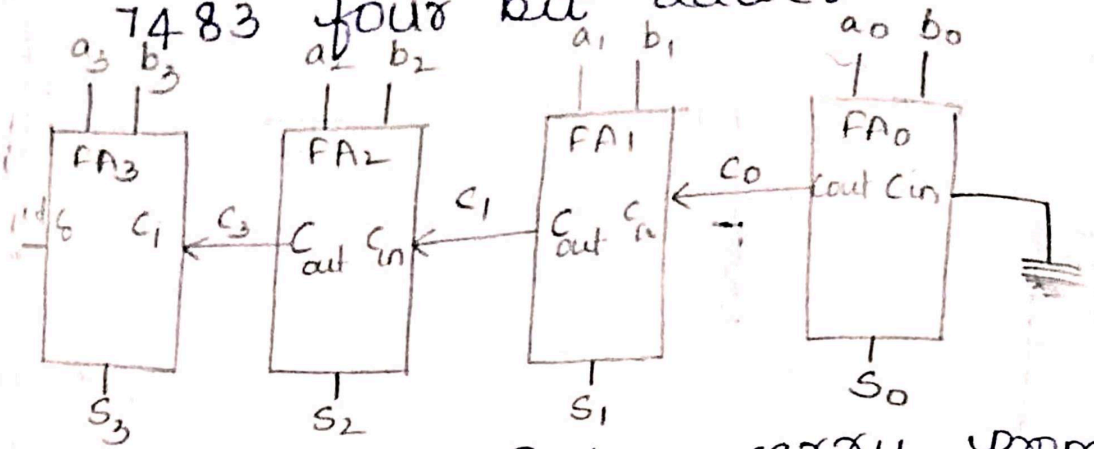
Less

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

$$b_1\bar{a}_1 + b_0\bar{a}_1\bar{a}_0 + \bar{a}_0 b_1 b_0$$

Ripple carry adder

7483 four bit adder



Limitations: Delay carry propagation or carry propagation delay

can be overcome using carry lookahead adder

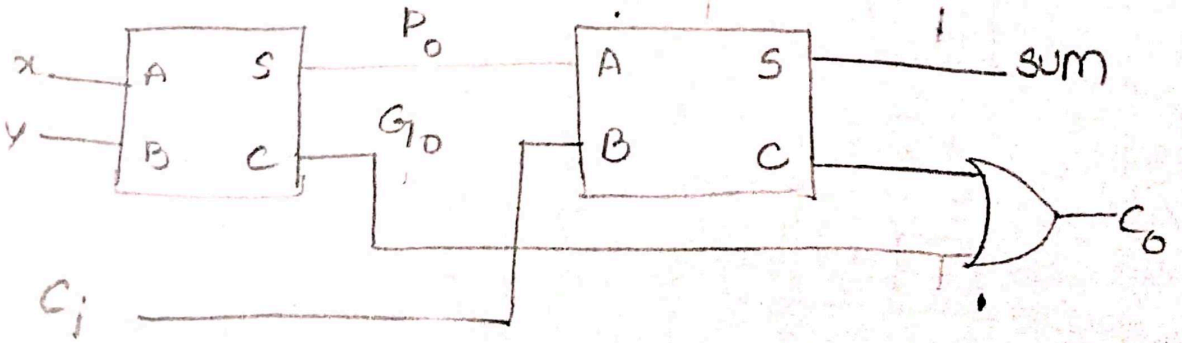
Carry look ahead adder

The process by eliminating inter stage carry delay is carry-lookahead adder.

It uses 2 functions carry generator & carry propagator

It is a high speed adder

Let us consider, FA using 2 HA to illustrate carry look ahead adder



we define Pi and Gi as the carry propagate and carry generate signals for the i^{th} stage of the adder

$P_i = A_i$ xor B_i and $G_i = A_i B_i$

$S = A_i$ xor B_i xor C_{in}

$C_{i+1} = G_i + P_i C_i$ Let $i = 0, 1, 2$

Sol

$i = 0$

$C_1 = G_0 + P_0 C_0$

$i = 1$

$C_2 = G_1 + P_1 C_1$ Now C_1 in Prev eq

$C_2 = G_1 + P_1 (G_0 + P_0 C_0)$

$G_1 + P_1 G_0 + P_0 P_1 C_0$

$i = 2$ $C_3 = G_2 + P_2 C_2$

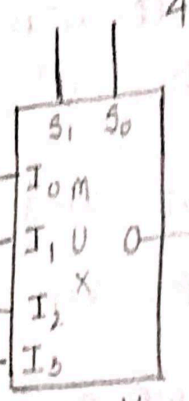
$\Rightarrow C_3 = G_2 + P_2 (G_1 + P_1 G_0 + P_0 P_1 C_0)$

$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_0 P_1 C_0$

Multiplexers or data selectors

Multiplexer 4 x 1

- 2^n data inputs
- n selection inputs
- a single output

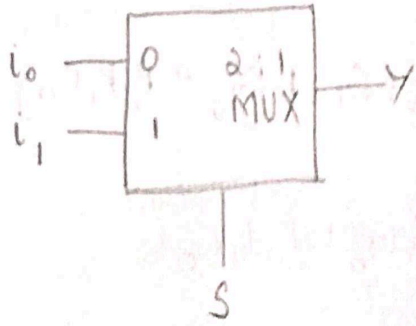


- selection input determines the input that should be connected to the output

Design of 2:1 MUX

Block diagram

$2^n : 1$



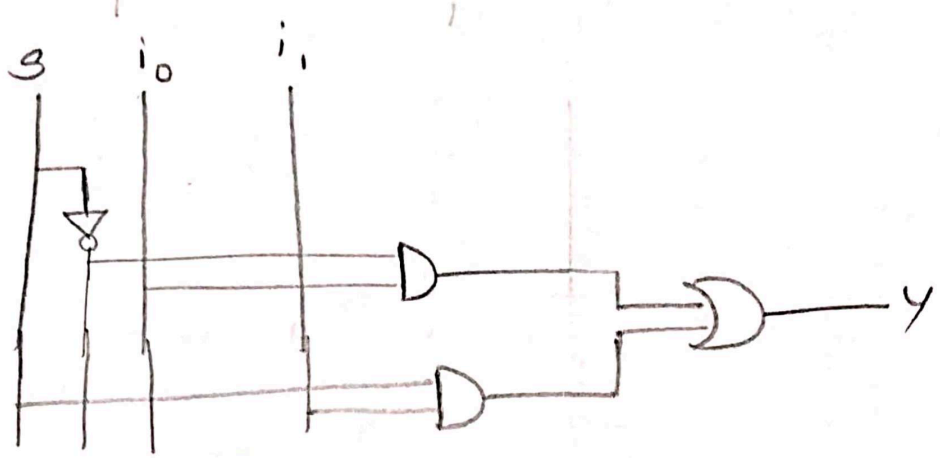
S	Y
0	i_0
0	i_1

Logic equation

$Y = \bar{S} i_0 + S i_1$

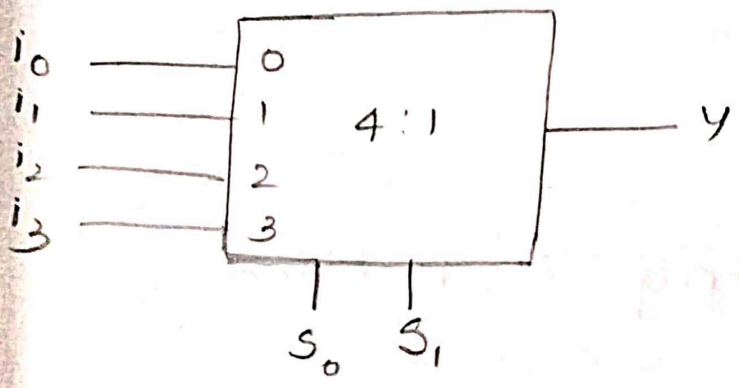
Logic diagram 2:1

mixing of signal
MUX
multiplex



Design of 4:1 MUX

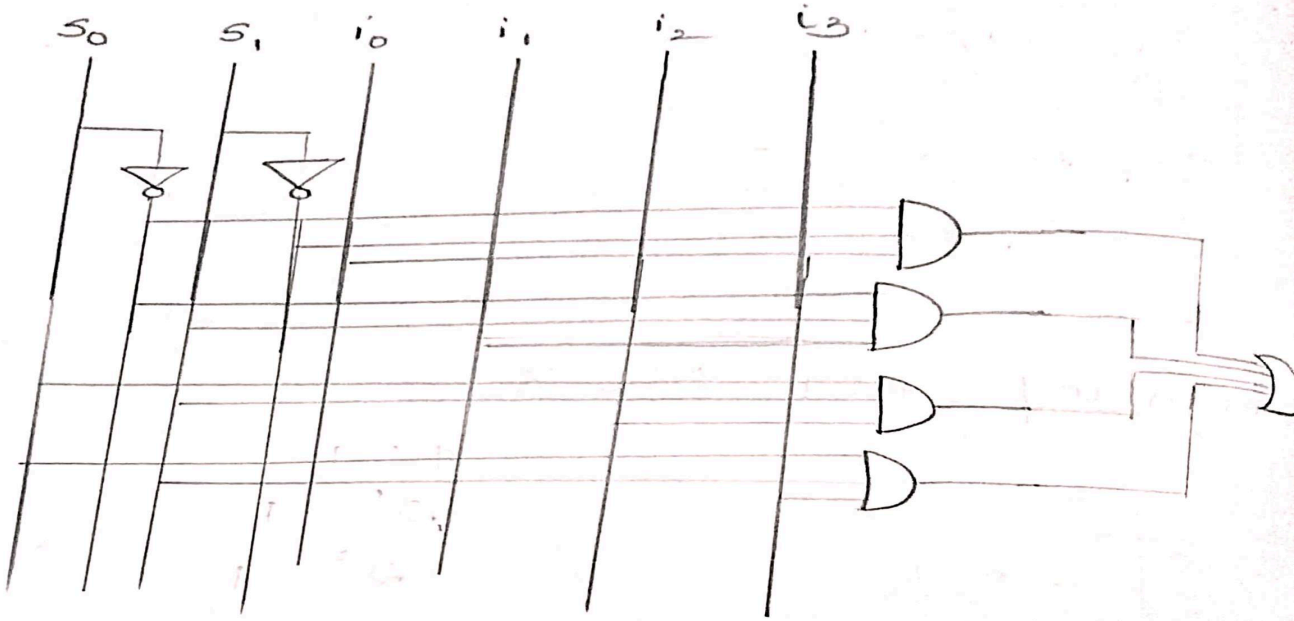
4:1
$2^n : 1$
$2^2 : 1$
4:1



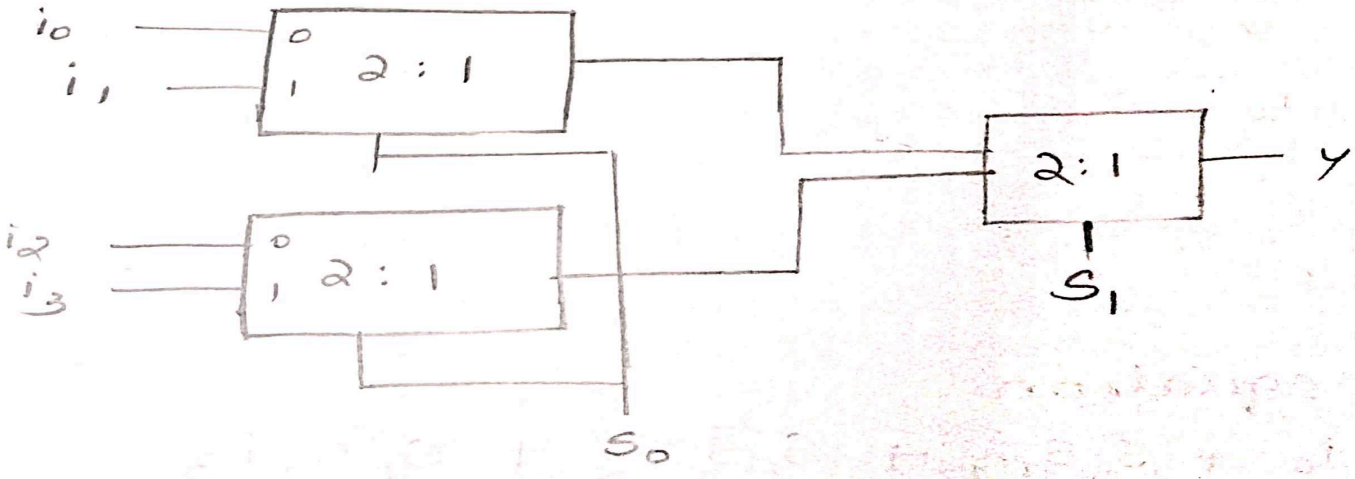
S_0 S_1	Y
0 0	i_0
0 1	i_1
1 0	i_2
1 1	i_3

Logic equation

$$\bar{S}_0 \bar{S}_1 i_0 + \bar{S}_0 S_1 i_1 + S_0 \bar{S}_1 i_2 + S_0 S_1 i_3$$

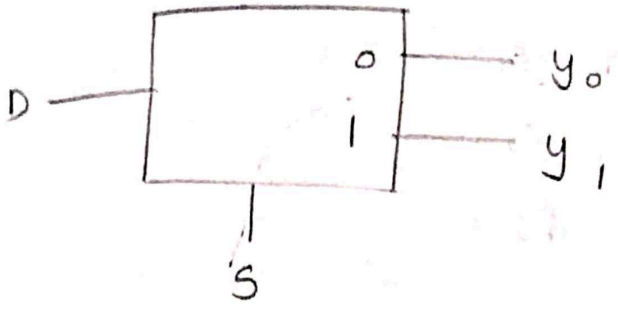


Design of 4:1 using 2:1 MUX



De Mux

Designing 1:2 De Mux



$1:2^n$

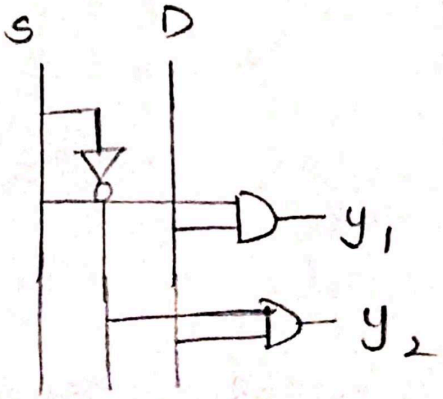
$1:2^n \rightarrow$ selection S

$1:2^1 \rightarrow$ 1 select line

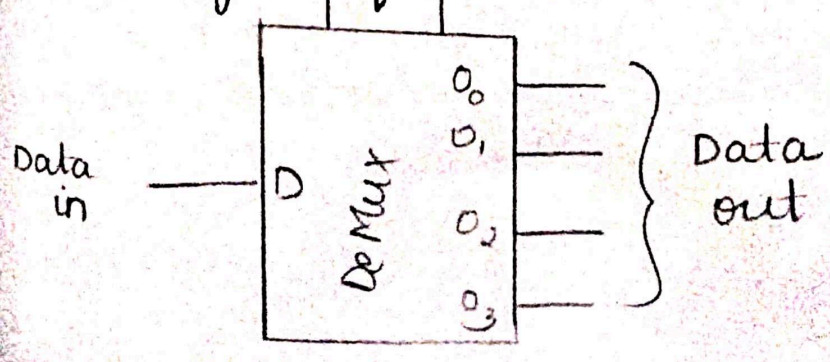
S	y_0	y_1
0	D	0
1	0	D

$y_0 = \bar{S} D$

$y_1 = S D$



Design of 1:4 De Mux



	S_1	S_0	O_0	O_1	O_2	O_3
0	0	0	D	0	0	0
1	0	1	0	D	0	0
2	1	0	0	0	D	0
3	1	1	0	0	0	D

Logic equation

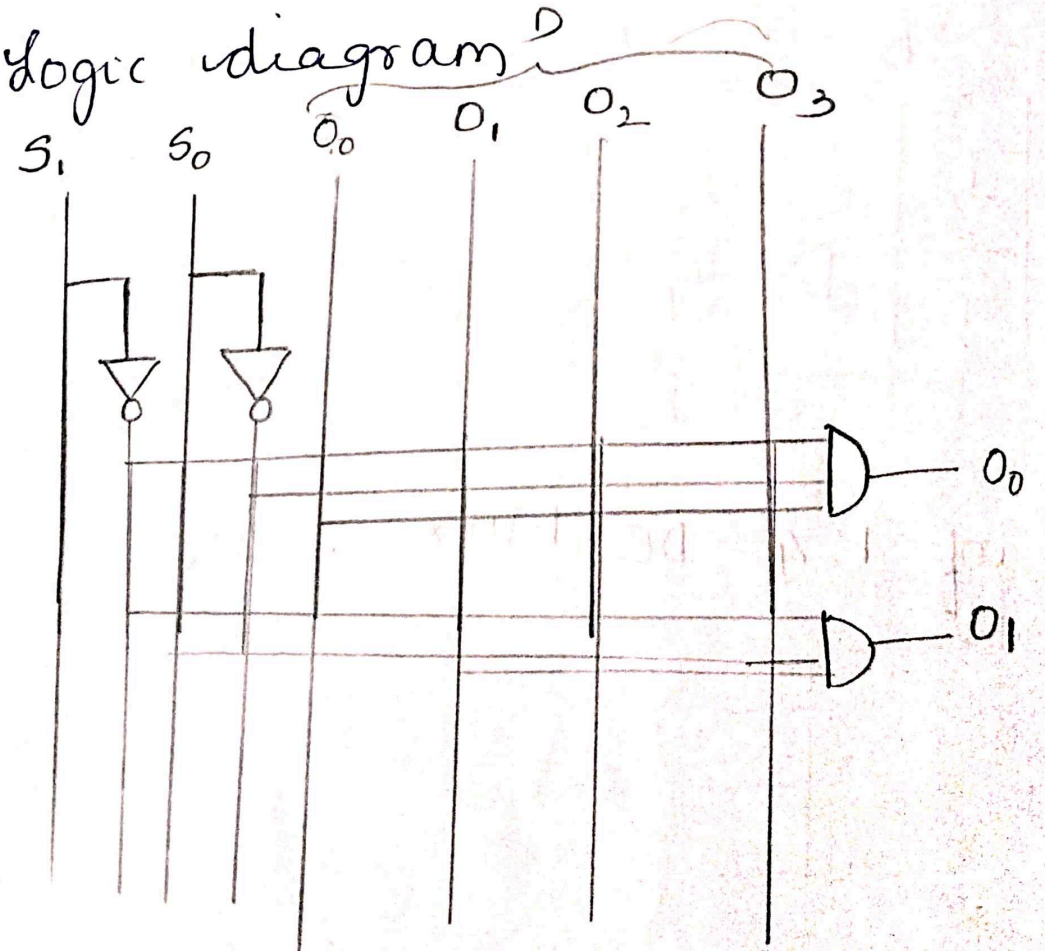
$O_0 = \bar{S_1} \bar{S_0} D$

$O_1 = \bar{S_1} S_0 D$

$O_2 = S_1 \bar{S_0} D$

$O_3 \quad S_1 S_0 D$

Logic diagram



Decoder

Decoder selects one out of n inputs

The configuration of decoders

$N \times 2^n$

Here N = no. of inputs

2^n = No. of outputs

Application

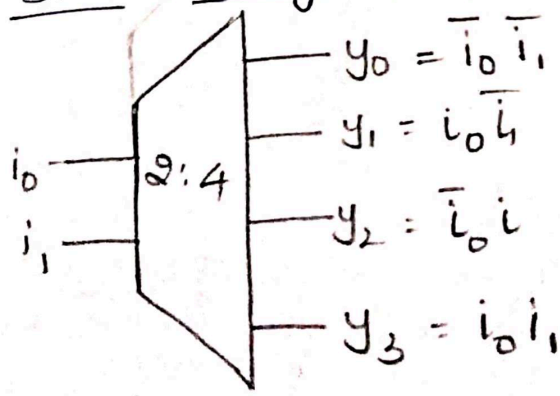
→ Decoder is min term generator because outputs represents all possible inputs variable.

→ It is used in display decoders

→ Decoders used in memory addressing

enable dis
no o/p

Block diagram



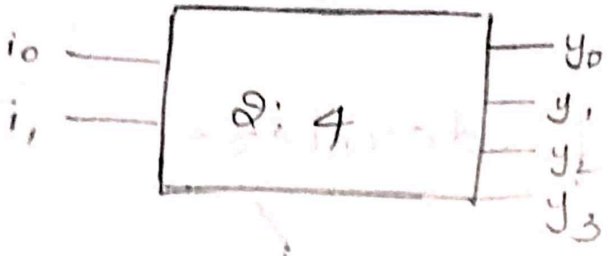
Min term generation

→ display decoding

→ Data converter

→ memory addressing

Design of 2:4 decoder



i_0	i_1	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

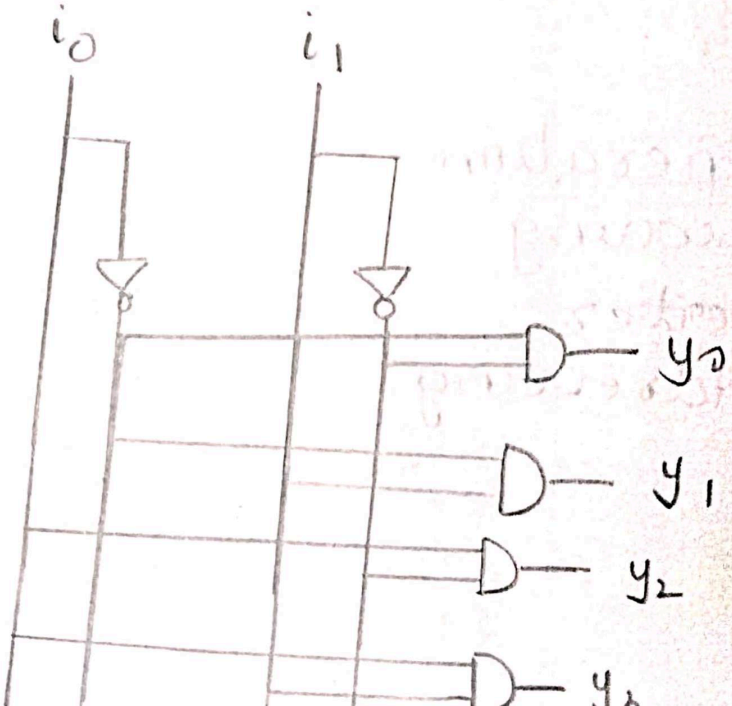
Logic equation

$y_0 = \bar{i}_0 \bar{i}_1$

$y_1 = \bar{i}_0 i_1$

$y_2 = i_0 \bar{i}_1$

$y_3 = i_0 i_1$

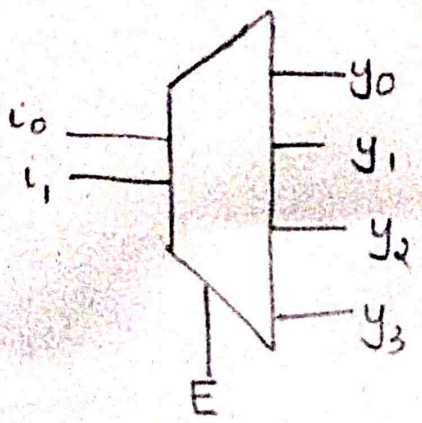


Design of 3:8 Decoder

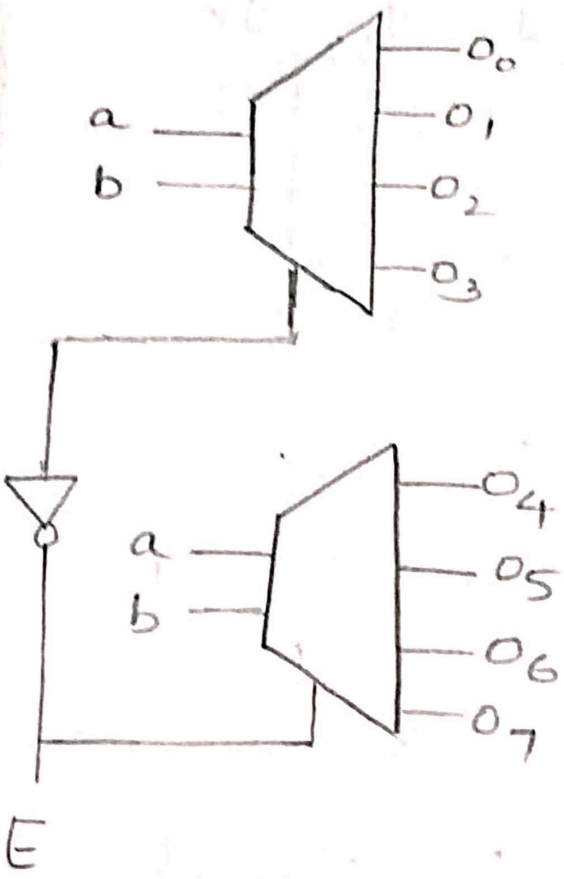
i_0	i_1	i_2	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

A decoder which enable E signal

E	i_0	i_1	y_0	y_1	y_2	y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

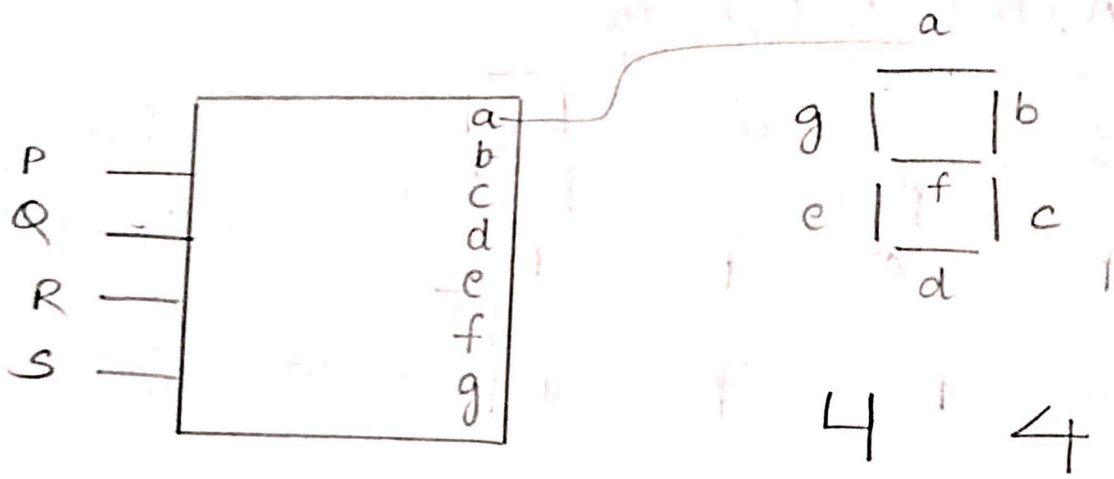


Design of 3:8 using 2:4 deco



E	a	b	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

BCD to seven segment display



4 4

	P Q R S	a b c d e f g
0	0 0 0 0	1 1 1 1 1 0 1
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 1 0
3	0 0 1 1	1 1 1 1 0 1 0
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

Using K map generate logic eq for A, B, C, D, E, F, g

a

PQ \ RS	$\bar{R}\bar{S}$ 00	$\bar{R}S$ 01	RS 11	$R\bar{S}$ 10
$\bar{P}\bar{Q}$ 00	0	1	3	2
$\bar{P}Q$ 01	4	5	7	6
PQ 11	12	13	15	14
$P\bar{Q}$ 10	8	9	11	10

$$P + R + S\bar{P}Q + \bar{R}\bar{S}\bar{Q}$$

b

	$\bar{R}\bar{S}$	$\bar{R}S$	RS	$R\bar{S}$
$\bar{P}\bar{Q}$	0 1	1 1	3 1	2 1
$\bar{P}Q$	4 1	5	7 1	6
PQ	12 X	13 X	15 X	14 X
$P\bar{Q}$	8 1	9 1	11 X	10 X

$$P + \bar{Q} + \bar{R}\bar{S} + RS$$

c

01	11	31	20
41	51	71	61
12 X	13 X	X 15	14 X
81	91	11 X	10 X

$C = P + \bar{R} + S + Q$

d

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

$D = \bar{Q}\bar{R}\bar{S} + R\bar{P}\bar{Q} + \bar{P}R\bar{S} + \bar{P}Q\bar{R}S$

Encoders

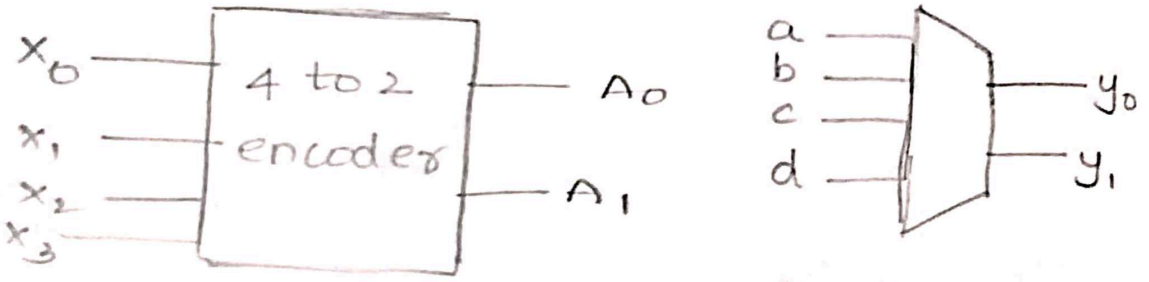
for $n = 2$

Encoder 4 to 2 $(2^n \times n)$

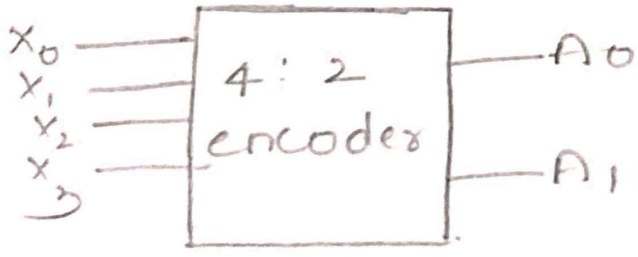
The configuration of encoder $(2^n \times n)$

2^n = no of input

n : no of output



Design 4 : 2 encoder



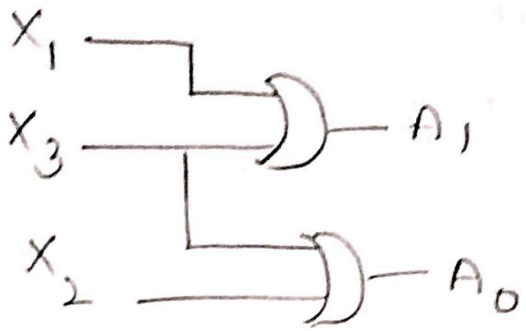
X_0	X_1	X_2	X_3	A_0	A_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Logic equation

$A_0 = X_2 + X_3$

$A_1 = X_1 + X_3$

Logic digram 4:2 encoder



Design a four line encoder that outputs a zero code unless one and only one input line is active

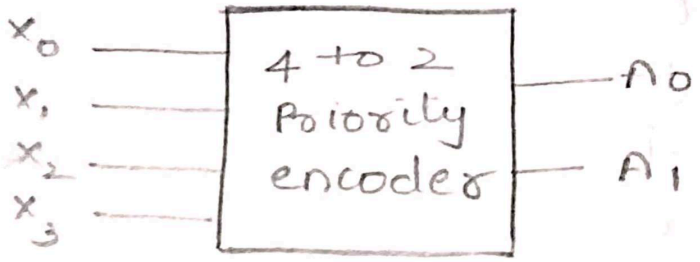
	X_3	X_2	X_1	X_0	A_0	A_1
0	0	0	0	0	d	d
1	0	0	0	1	0	0
2	0	0	1	0	0	1
3	0	0	1	1	d	d
4	0	1	0	0	1	0
5	0	1	0	1	d	d
6	0	1	1	0	d	d
7	0	1	1	1	d	d
8	1	0	0	0	1	1
9	1	0	0	1	d	d
10	1	0	1	0	d	d
11	1	0	1	1	d	d
12	1	1	0	0	d	d
13	1	1	0	1	d	d
14	1	1	1	0	d	d
15	1	1	1	1	d	d

X_2X_3 \ A_0 X_1X_0	0	1	3	2
	0 X	1 1	3 X	2
	4 1	5 X	7 X	6 X
	12 X	13 X	15 X	14 X
	8 1	9 X	11 X	10 X

$$A_1 = X_2 + X_3$$

$$A_0 = X_1 + X_3$$

Priority Encoder



X_0	X_1	X_2	X_3	A_0	A_1
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

$$A_0 = X_2 + X_3$$

$$A_1 = X_1 + X_3$$

Code converters

1. Binary - excess 3 code converter
2. Binary - gray code converter
3. Gray code - Binary converter

a	b	c	d	E_1	E_2	E_3	E_4
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

ab \ cd	$\bar{c}\bar{d}$	$\bar{c}d$	cd	$c\bar{d}$
$a\bar{b}$	0	0	0	0
$\bar{a}b$	0	1	1	1
ab	X	X	X	X
$\bar{a}\bar{b}$	1	1	X	X

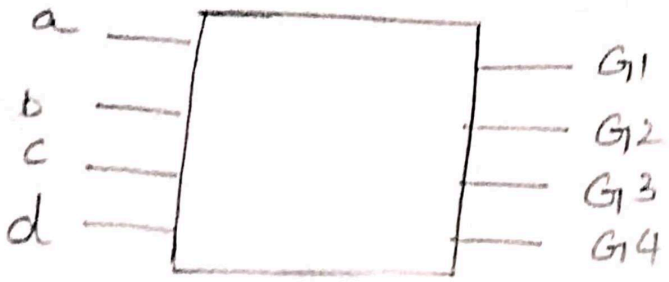
$$E_1 = a + bc + bd.$$

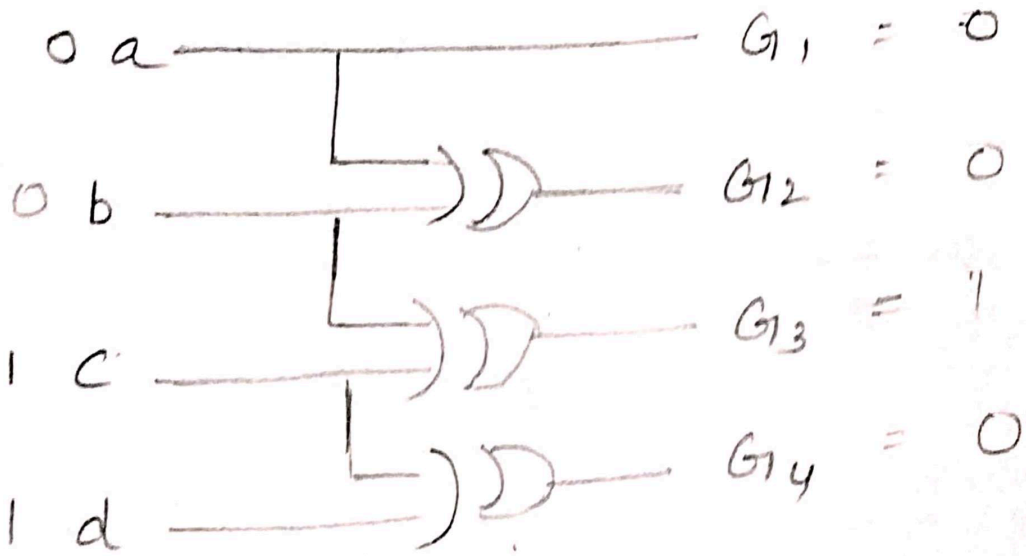
	$\bar{c}\bar{d}$	$\bar{c}d$	cd	$c\bar{d}$
$a\bar{b}$	0	1	1	1
$\bar{a}b$	1	0	0	0
ab	X	X	X	X
$\bar{a}\bar{b}$	0	1	X	X

$$\bar{b}d + \bar{b}c\bar{d} + b\bar{c}\bar{d}$$

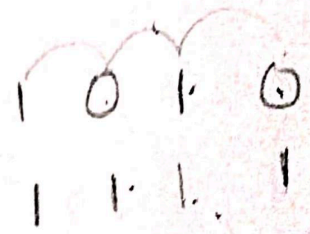
Binary to Gray code converter

↳ use XOR gate





a	b	y
0	0	0
0	1	1
1	0	1
1	1	1



Diff
i/p = 1

	a	b	c	G_1	G_2	G_3
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

obtain logic eq for G_1, G_2, G_3 and draw logic diagram

	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$	0 0	1 0	3 0	2 0
a	4 1	5 1	7 1	6 1

$G_1 = a$

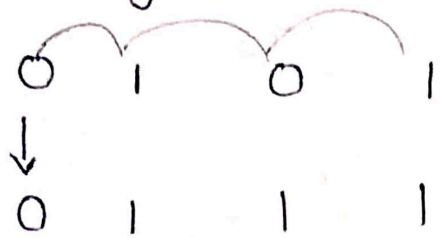
0 0	1 0	3 1	2 1
4 1	5 1	7 0	6 0

$G_2 = \bar{a}b + \bar{b}a$

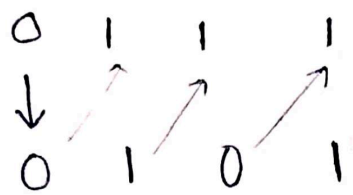
0 0	1 1	3 0	2 1
4 0	5 1	7 0	6 1

$G_3 = \bar{b}c + b\bar{c}$

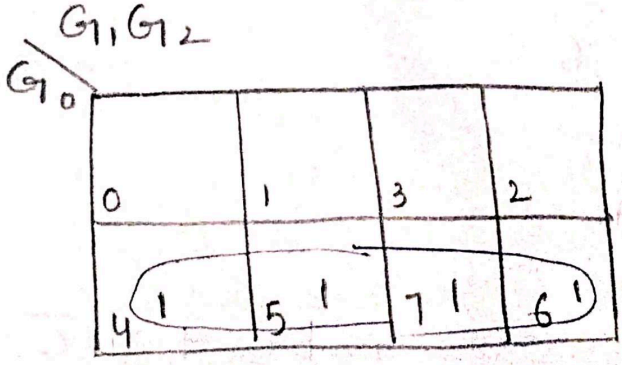
Binary to Gray code



Gray code to Binary



	G_{I_0}	G_{I_1}	G_{I_2}	B_0	B_1	B_2
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	1
5	1	0	1	1	1	0
6	1	1	0	1	0	0
7	1	1	1	1	0	1



$$B_0 = G_0$$

0	1	3 (1)	2 (1)
4 (1)	5 (1)	7	6

$$B_1 = \bar{G}_0 G_1 + G_0 \bar{G}_1 = G_0 \oplus G_1$$

0	1 (1)	3	2 (1)
4 (1)	5	7 (1)	6

$$B_2 = \bar{G}_0 \bar{G}_1 G_2 + \bar{G}_0 G_1 \bar{G}_2 + G_0 \bar{G}_1 \bar{G}_2 + G_0 G_1 G_2$$

$$\bar{G}_0 (\bar{G}_1 G_2 + G_1 \bar{G}_2) + G_0 (\bar{G}_1 \bar{G}_2 + G_1 G_2)$$

$$\bar{G}_0 (G_1 \oplus G_2) + G_0 (G_1 \odot G_2)$$

$$\bar{G}_0 (G_1 \oplus G_2) + G_0 \overline{(G_1 \oplus G_2)}$$

$\bar{a}b + a\bar{b}$

$$G_0 \oplus G_1 \oplus G_2$$

UNIT - 3

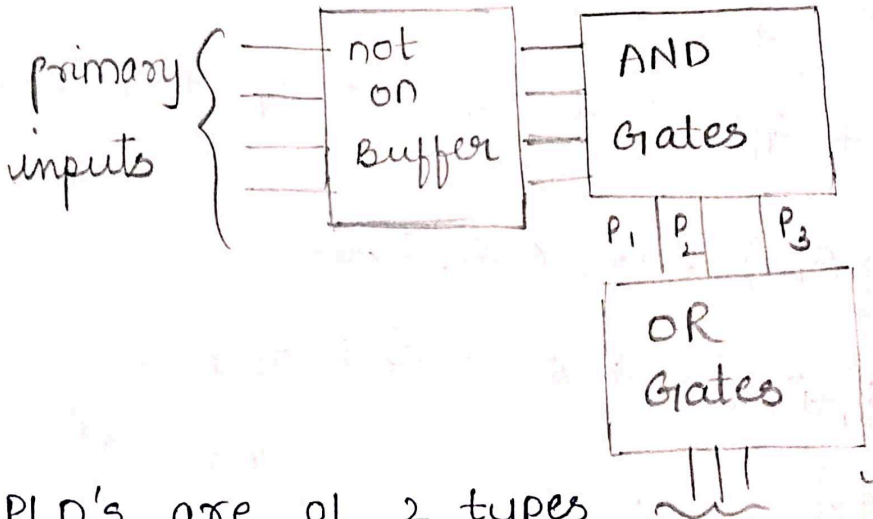
Design of combinational circuits using PLD's

PLD's → Programmable logic devices

Structure of PLD's

It consists of array of AND Gates OR Gates and NOT gates

It is readily available offshell integrated circuit



PLD's are of 2 types

Simple PLD's and complex PLD's

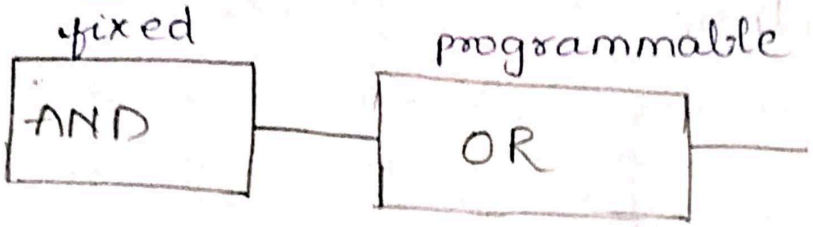
1. Simple PLD's are classified into three types

a) PROM :- Programmable read only memo

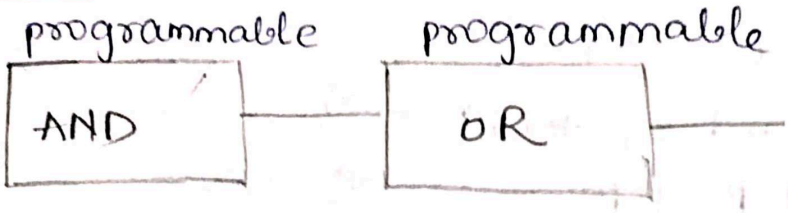
b) PLA :- Programmable logic array

c) PAL :- Programmable array logic

PROM

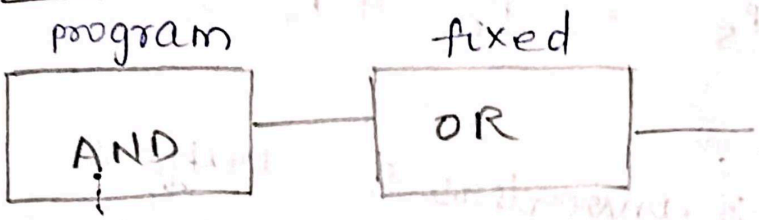


PLA



fixed.
prog x

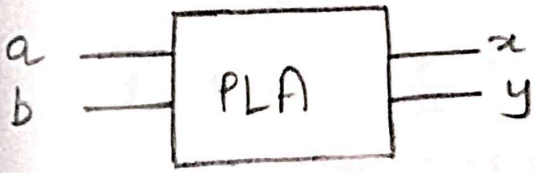
PAL

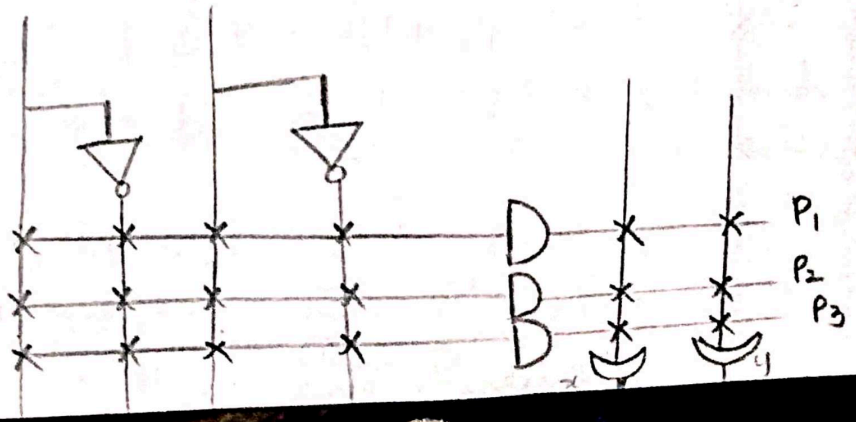


→ PLA is more efficient because it has option for programmability

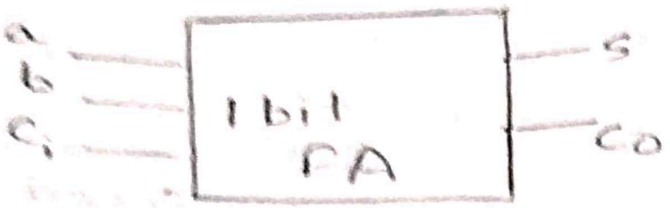
→ General structure of programmable logic array

1) PLA





Design of 1 bit full adder using PLA structure



$$s = a \oplus b \oplus c_i$$

$$c_o = ab + bc_i + c_i a \quad \text{—①}$$

$$c_o = P_1 + P_2 + P_3$$

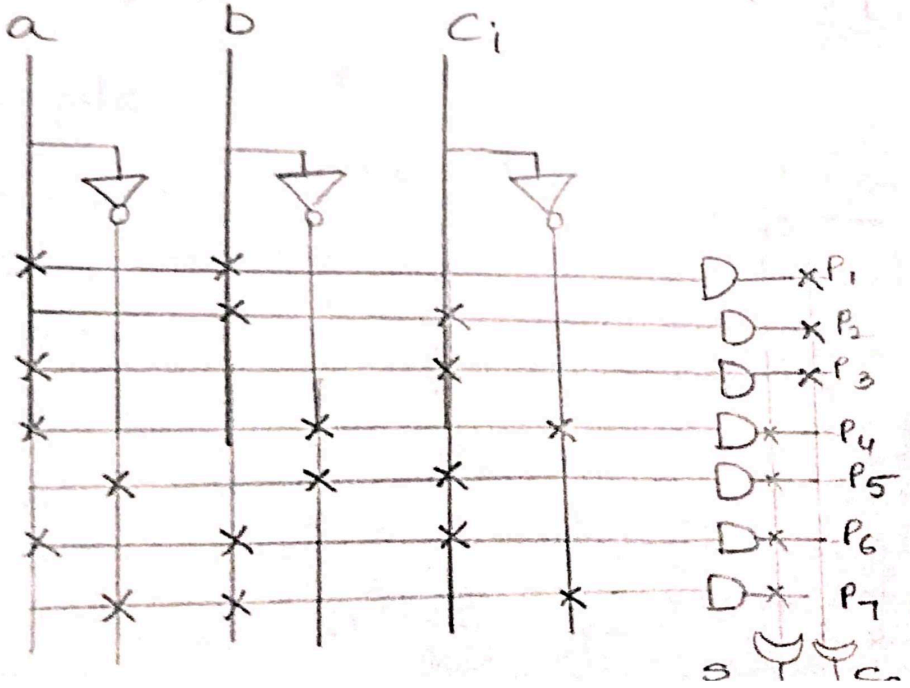
$$s = a\bar{b}\bar{c}_i + \bar{a}\bar{b}c_i + abc_i + \bar{a}b\bar{c}_i \quad \text{—②}$$

$$P_4 + P_5 + P_6 + P_7$$

No. of —▷— 3 inverters & 3 buffers

No. of —D— no. of product terms 7

No. of —D— no. of OR gates (sum) . 2 o/p



for a given function f_1, f_2, f_3 realize using PLA

$f_1(a,b,c) = \Sigma m(0,1,3,5)$

$F_2(a,b,c) = \Sigma (0,3,7,4)$

$f_3(a,b,c) = \Sigma (2,3,5,7)$

	$\bar{b}\bar{c}$	$\bar{b}c$	bc	$b\bar{c}$
$\bar{a}$	0 (1)	1 (1)	3	2 (1)
a	4	5 (1)	7	6

$f_1 = \quad + \bar{b}c + \bar{a}\bar{c}$

0 (1)	1	3 (1)	2
4 (1)	5	7 (1)	6

$f_2 = bc + \bar{b}\bar{c}$

0	1 (1)	3 (1)	2 (1)
4	5 (1)	7 (1)	6

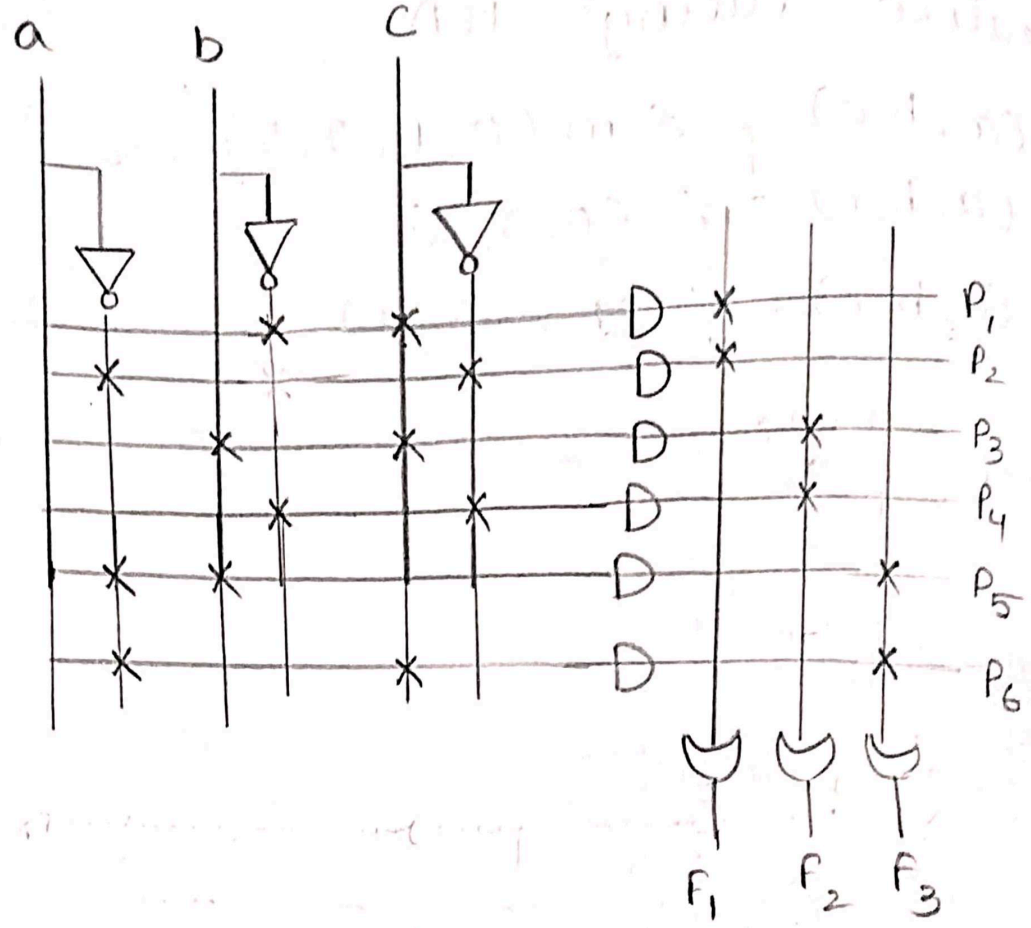
$f_3 = \bar{a}b + ac$

$f_1 = \bar{b}c + \bar{a}\bar{c} = P_1 + P_2$

$f_2 = bc + \bar{b}\bar{c} = P_3 + P_4$

$f_3 = \bar{a}b + ac = P_5 + P_6$

3 inverters
6 product terms
3 sum terms



Design of 1 bit comparator using PLA

$G = a\bar{b}$

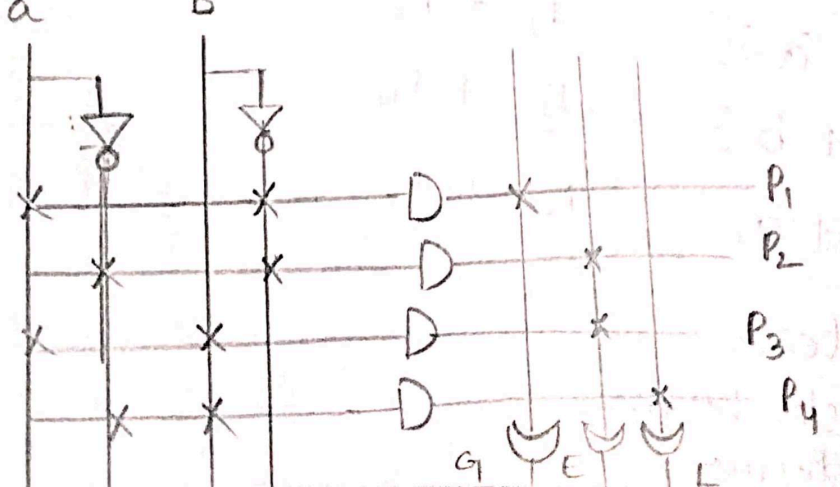
$E : \bar{a}\bar{b} + ab$

$L = \bar{a}b$

2 inverters

4 product terms

3 sum terms

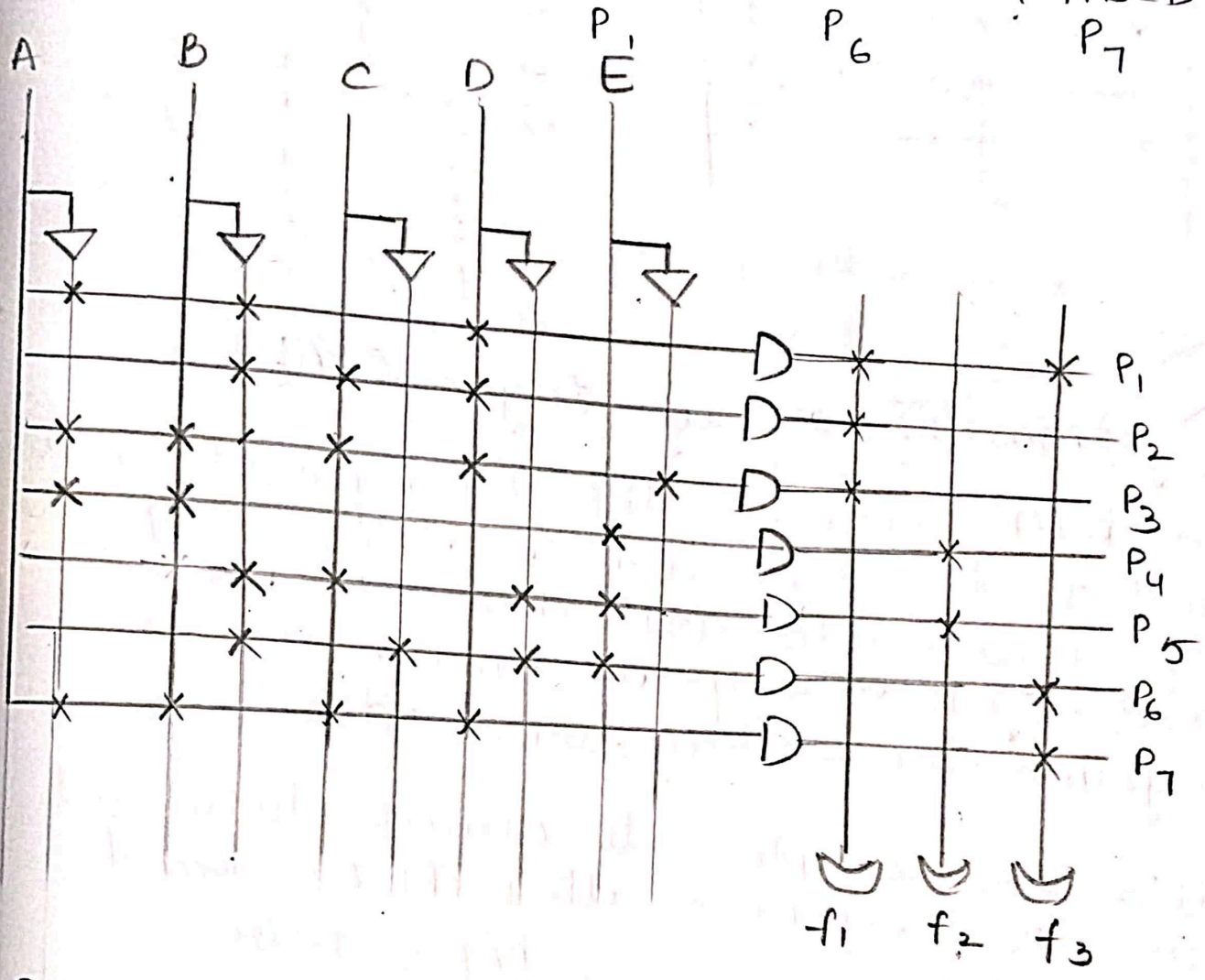


Design a PLA to realize the fig 3 logic functions and show the internal connections

$$f_1(A,B,C,D,E) = \underset{P_1}{\bar{A}\bar{B}\bar{D}} + \underset{P_2}{\bar{B}C\bar{D}} + \underset{P_3}{\bar{A}BCD\bar{E}}$$

$$f_2(A,B,C,D,E) = \underset{P_4}{\bar{A}BE} + \underset{P_5}{\bar{B}C\bar{D}E}$$

$$f_3(A,B,C,D,E) = \underset{P_1}{\bar{A}\bar{B}\bar{D}} + \underset{P_6}{\bar{B}\bar{C}\bar{D}E} + \underset{P_7}{\bar{A}BCD}$$



Realize the following 3 switching functions with a 3 input, 3-output PLA

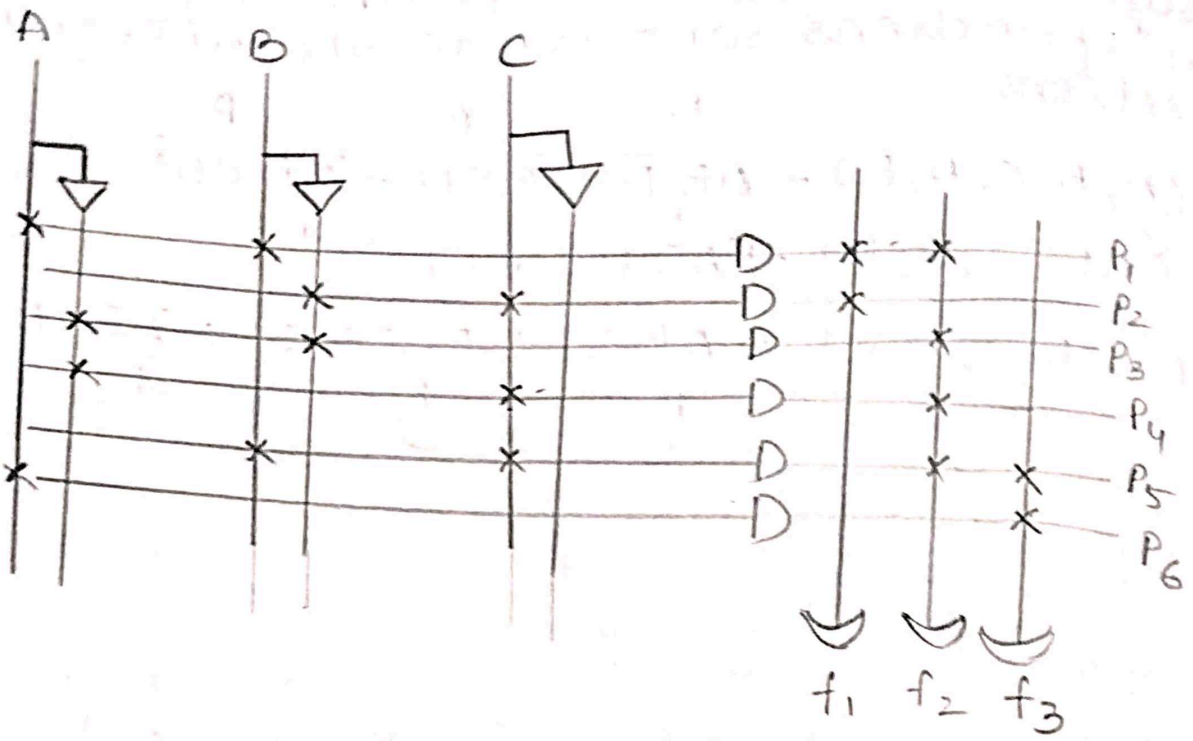
$$f_1(A,B,C) = \underset{P_1}{AB} + \underset{P_2}{\bar{B}C}$$

$$f_2(A,B,C) = (A+\bar{B}+C)(\bar{A}+B)$$

$$f_3(A,B,C) = \underset{P_6}{A} + \underset{P_5}{BC}$$

$$f_2 : A\bar{A} + AB + \bar{A}\bar{B} + \bar{B}B + \bar{A}C + BC$$

$$f_2 : AB + \bar{A}\bar{B} + \bar{A}C + BC$$



Programmable array logic (PAL)

(2)

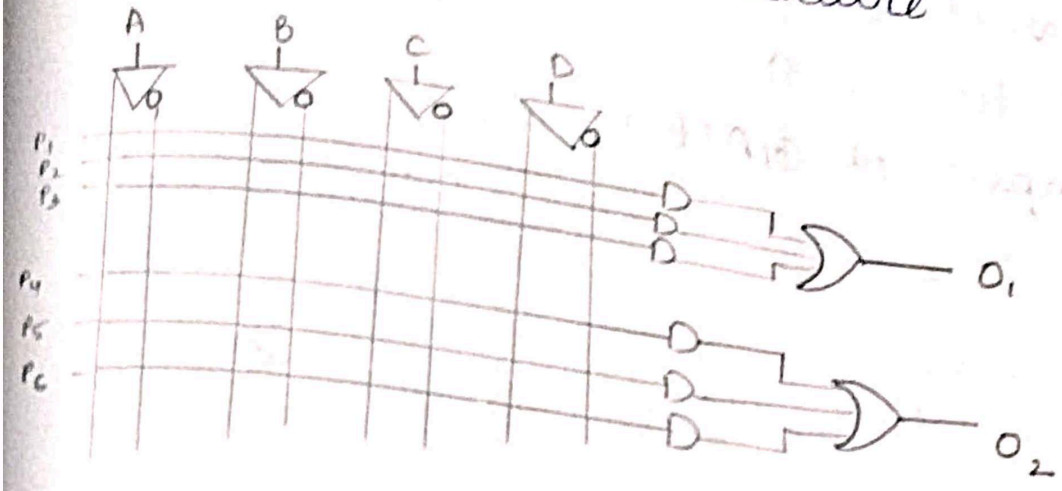
Drawbacks led to the development of a similar device in which the AND plane is programmable, but the OR plane is fixed. Such a chip is known as programmable array logic device.

PAL's are simpler to manufacture, and thus less expensive than PLAs, and offer better performance, PAL's have become popular in practical applications

(1)

- In a PLA both the AND and OR planes are programmable
- The programmable switches presented two difficulties for manufacturers of these devices they were hard to fabricate correctly, and they reduced the speed performance of circuit implemented in the PLA's

Standard PAL structure



Realize functions using PAL

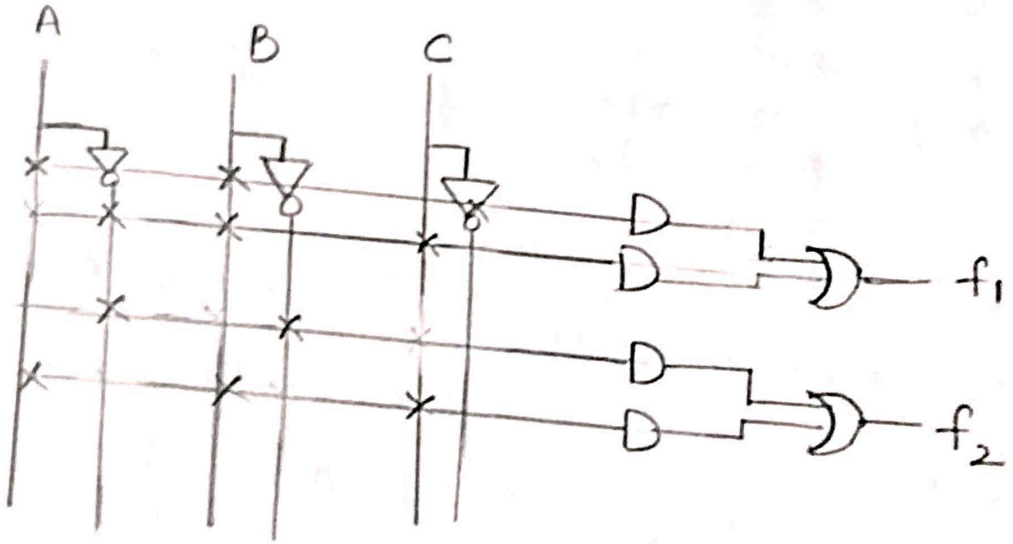
$f_1 = x_1 x_2 x_3' + x_1' x_2 x_3$

$f_2 = x_1' x_2' + x_1 x_2 x_3$

(or)

$f_1(a,b,c) = ab\bar{c} + \bar{a}bc$

$f_2(a,b,c) = \bar{a}\bar{b} + ab$



Realize functions using PLA

$f_1(a,b,c) = \bar{a}b + a\bar{b}c + \bar{a}c$

$f_2(a,b,c) = a\bar{b} + a\bar{b}c$

$f_3(a,b,c) = a\bar{b}c + \bar{a}\bar{b} + abc$

$f_1(a,b,c) = P_1 + P_2 + P_3$

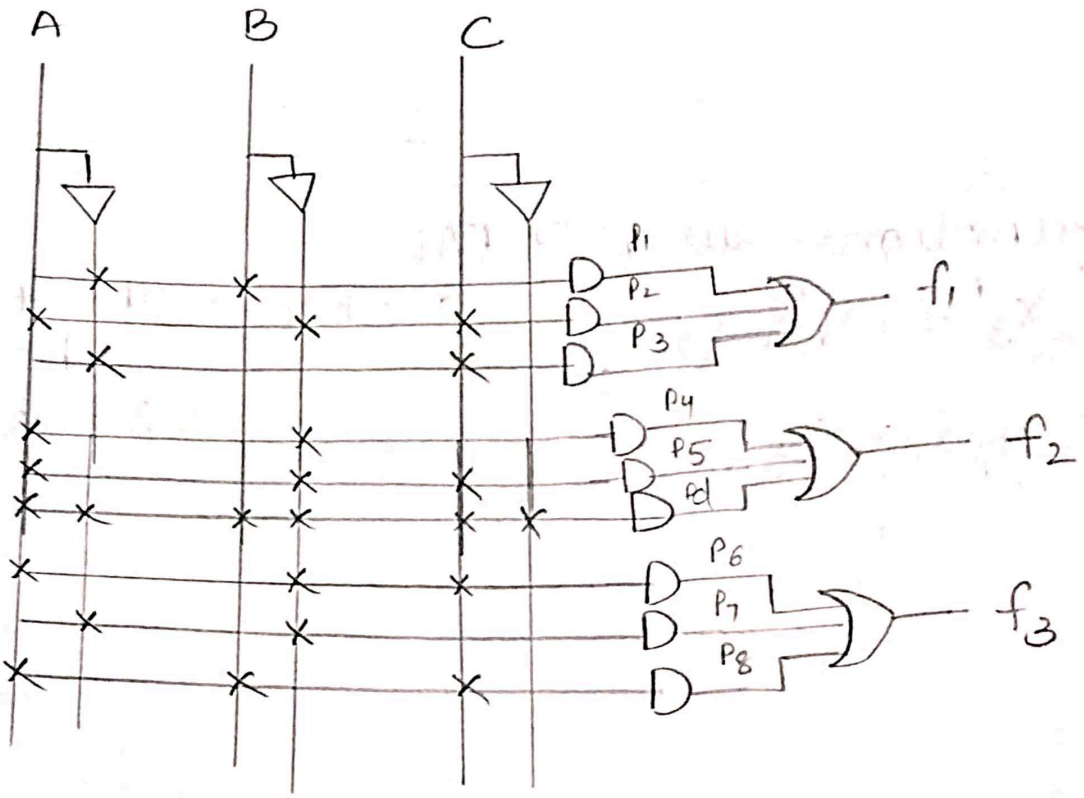
$f_2(a,b,c) = P_4 + P_5 + P_6$

$f_3(a,b,c) = P_6 + P_7 + P_8$

NO. of inverters

AND Gates - 9

NO. of 3 input OR GATES - 3



Realize the given function using PAL

$f_\alpha(A,B,C,D) = \Sigma m(0,2,7,10) + d(12,15)$

$f_B(A,B,C,D) = \Sigma m(2,4,5) + d(6,7,8,10)$

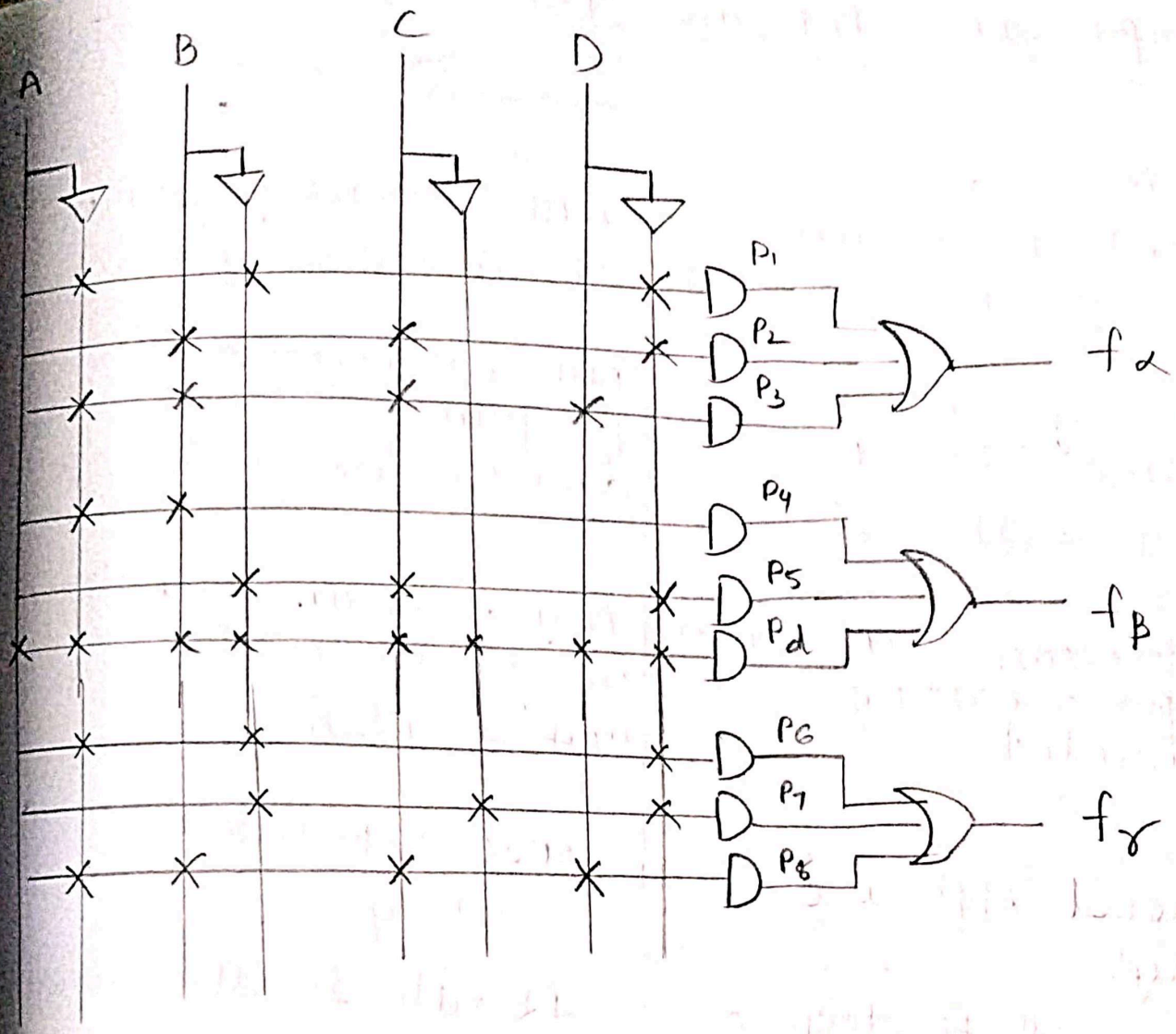
$f_C(A,B,C,D) = \Sigma m(2,7,8) + d(0,5,13)$

The result of the fig 3 exp

$f_\alpha(A,B,C,D) = \underset{P_1}{\bar{A}\bar{B}\bar{D}} + \underset{P_2}{\bar{B}C\bar{D}} + \underset{P_3}{\bar{A}BCD}$

$f_B(A,B,C,D) = \underset{P_4}{\bar{A}B} + \underset{P_5}{\bar{B}C\bar{D}} + P_d$

$f_\gamma(A,B,C,D) = \underset{P_6}{\bar{A}\bar{B}\bar{D}} + \underset{P_7}{\bar{B}\bar{C}\bar{D}} + \underset{P_8}{\bar{A}BCD}$



Limitations

- Limited number of switching functions can realiz
- Selection of device to a particular applications. Single product term can not be shared between two sum terms
- If two sum contains a common product term that product must be generate twice
- More cost effective than PROM and PLA

Comparison PLA and PAL

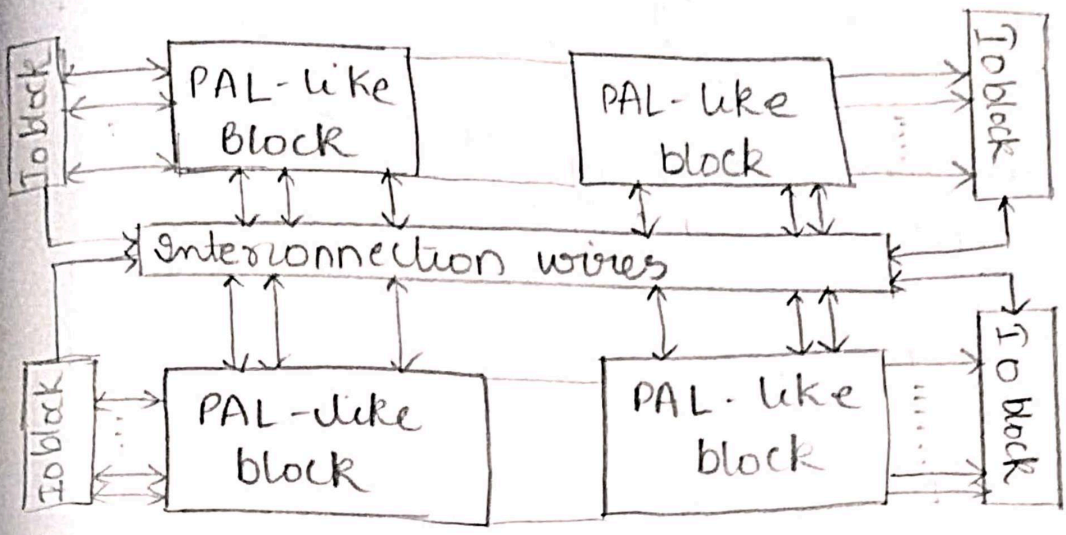
PLA	PAL
AND and OR array are programmable	AND array programmable OR array is fixed
AND array can be programmed to get desired midterms	AND array can be programmed to get desired min terms
Any boolean function in SOP form can be implemented	Any boolean function in SOP form can be implemented
less cost effective (costly)	cost effective (cheap)
More flexible to designer	Flexible to the designer

CPLD

For implementation of circuits, that require more i/p and o/p, either multiple PLA's or PAL's can be employed or is called CPLD

A CPLD comprises multiple circuit blocks on a single chip with internal wiring resources to connect the circuit blocks. Each circuit block is similar to a PLA or a PAL.

Architecture of CPLD



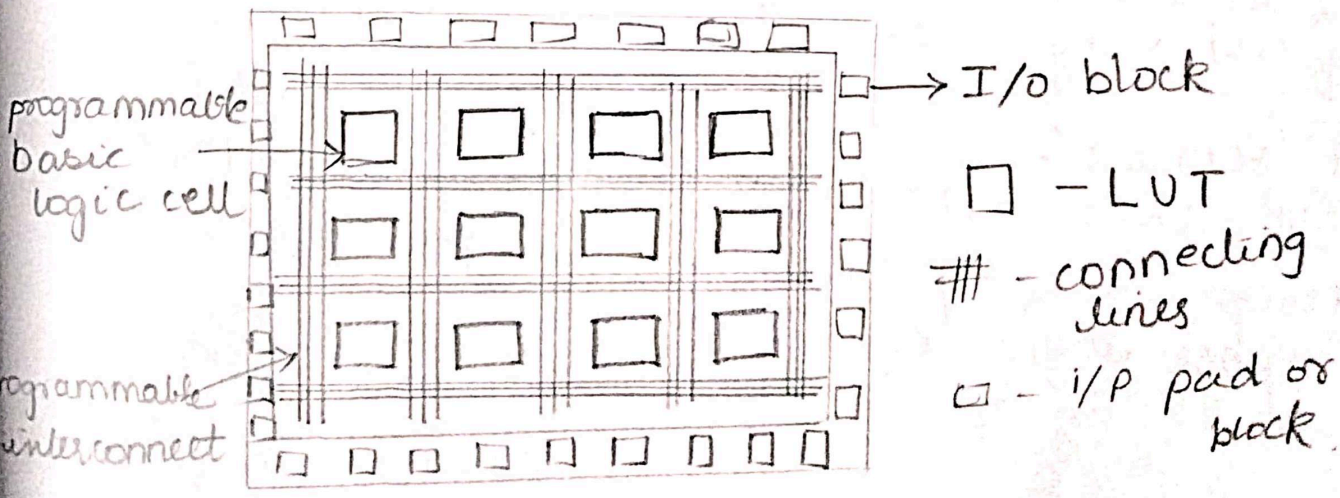
☆☆

Field programmable gate array (FPGA)

All FPGA contain regular structure of programmable basic logic cells surrounded by programmable interconnects

FPGA used to implement logic circuits of more than a few hundred thousand equivalent gates in size.

Two examples of FPGA's, called as the Altera FLEX 10K and the Xilinx XC4000



FPGA - characteristics

- None of the mask layers are customised
- A method for programming the basic logic cells and the interconnection

LUT (look up Table)

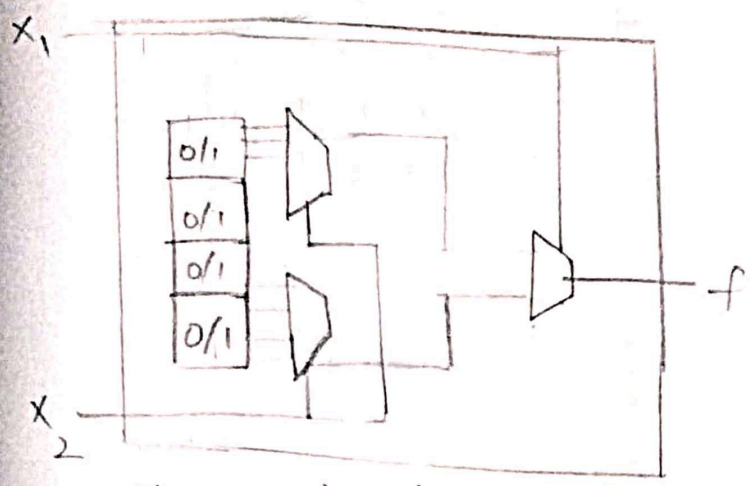
The most commonly used logic block is LUT which contains storage cells that are used to implement a small logic function

Each cell is capable of holding a single logic value, either 0 or 1

The stored value is produced as o/p of storage cell

It consists of memory block and MUX combination, it is used in FPGA design to implement any general purpose logic equations

Structure of LUT



Two input LUT

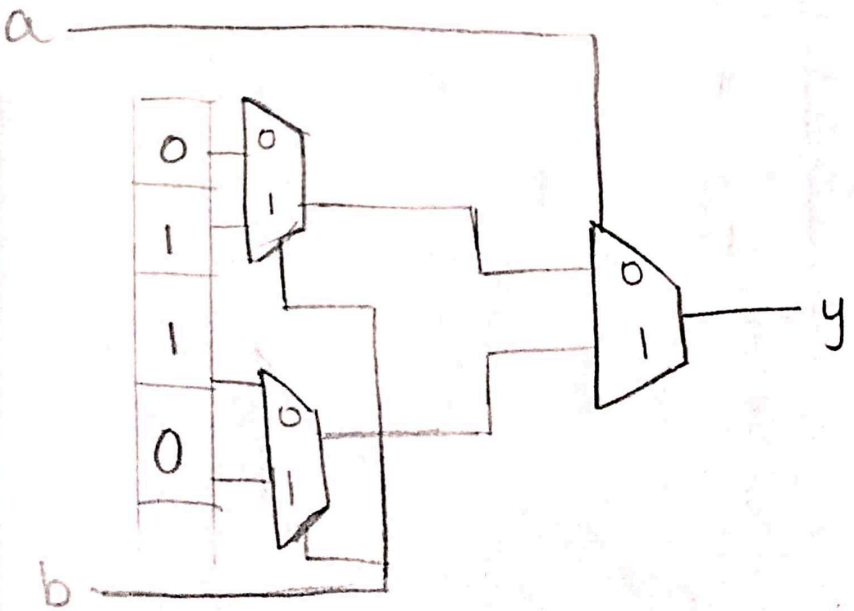
Eg1:- 2 input LUT, implement XOR and XNOR function using LUT

XOR

Choose 2 input LUT

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

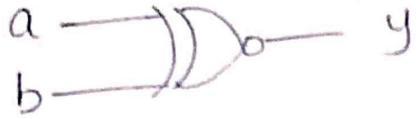
Eg:-



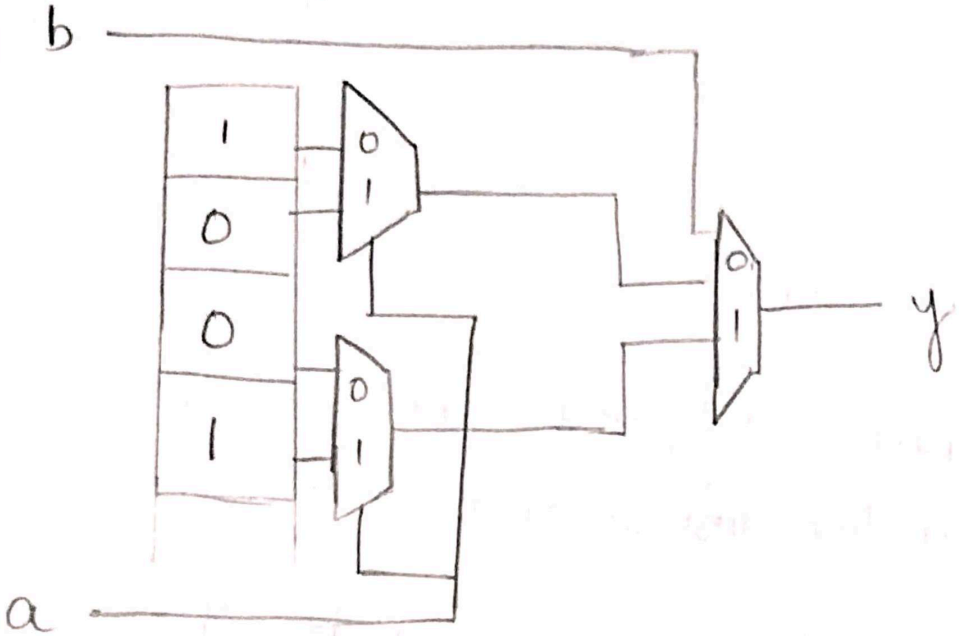
11 + 00

1 + 0 1

XNOR

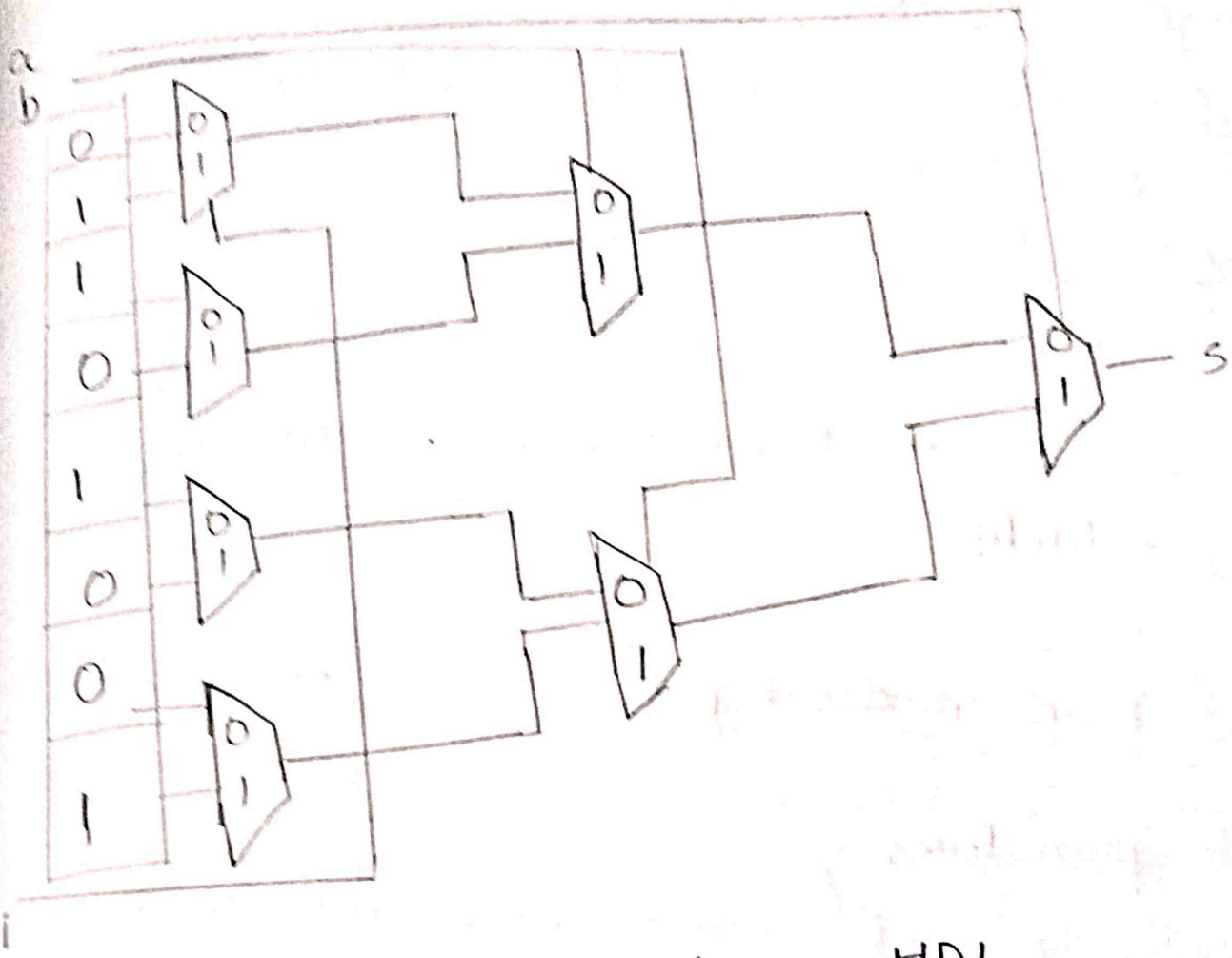


a	b	y
0	0	1
0	1	0
1	0	0
1	1	1



Eg 2 : 3 bit adder

a	b	C_1	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



Introduction to verilog . HDL

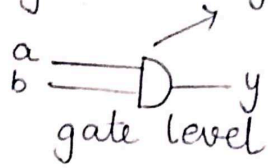
HDL (Hardware description language)

- verilog
- VHDL - very high speed of Description language

verilog

- Switch level modelling
- Gate level
- Data flow
- Behavioral - Truth table
- RTL

Eg:- gatelevel module



gate level

$y = a \cdot b$ — Data-flow module

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

Logic Table → Behavioural module * (uses if)

Gate level moduling

Verilog - case sensitive

gate primitives

and or not nand nor xor xnor

Syntax

① Gate instance creation - without instance name

② Gate instance creation - with instance name

① and (list of o/p, list of i/p);

and (y, a, b);

② 

and G1, (y, a, b);

write a verilog code for basic gates using gate level modeling.

structure / template of verilog

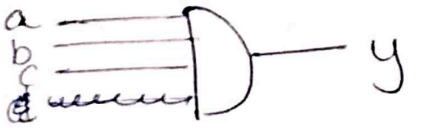
```
module <name of module> (port list);
input list of inputs;
output list of outputs;

// logic - comments

endmodule
```

---

eda playground
electronic design automation

---

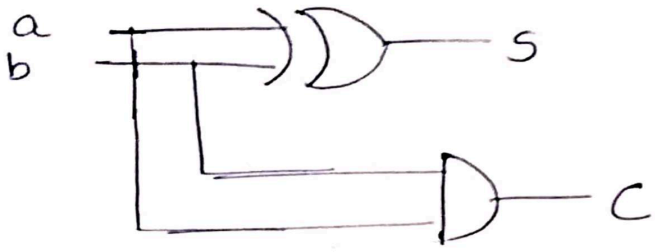
write a program for basic gates




```
module and_p(a, b, c, y);
input a, b, c;
output y;
// logic . 3 input and prog in gate level module
  and (y, a, b, c);
endmodule
```

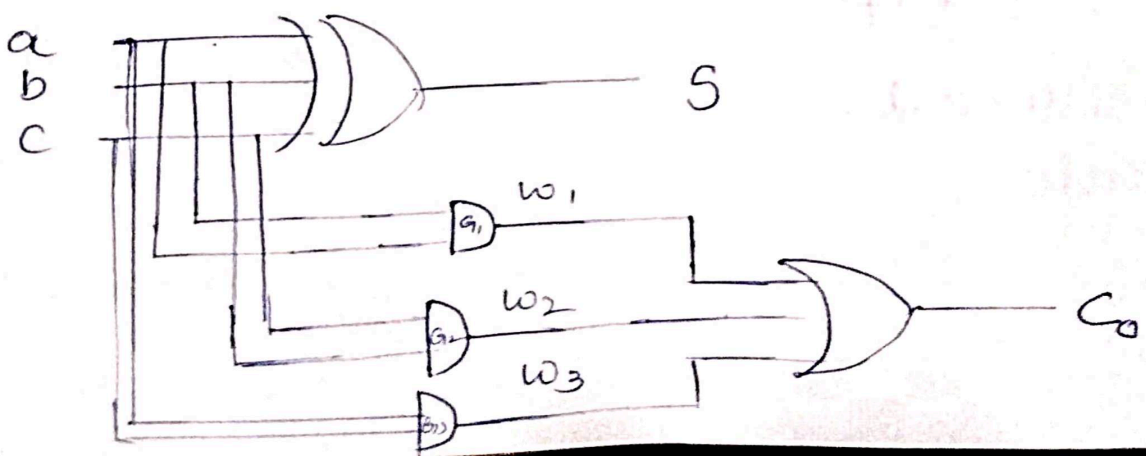
```
// by
 xor (y,a,b);
nand (y,a,b);
 not (y,a);
nor (y,a,b);
xnor (y,a,b);
```

Verilog code for half adder



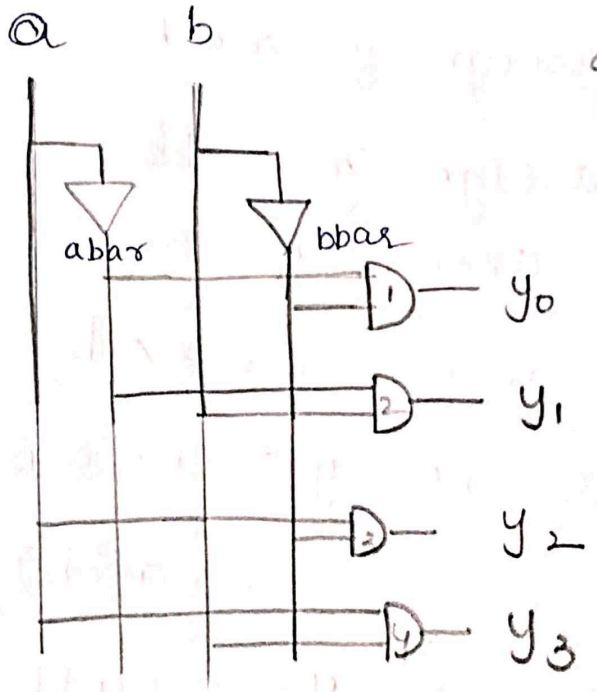

```
module ha(a,b,s,c);
 input a, b;
 output s, c;
// logic - 2 input, 2 output
    xor (s,a,b);
    and (c,a,b);
  endmodule
```

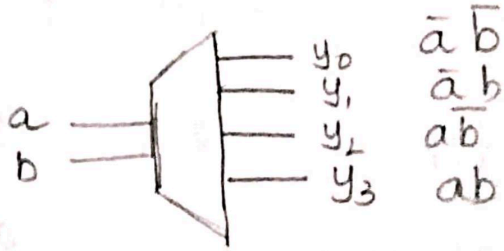
Verilog code for full adder



```
module fa( a, b, ci, s, co);
 input  a, b, ci;
 output  co, s;
 wire  w1, w2, w3;
 // logic
  xor  (s, a, b, ci);
 and G1 ( w1, a, b);
 and  G2 (w2, b, ci);
 and  G3 (w3, a, ci);
  or  (co, w1, w2, w3);
```

Verilog code for a decoder 2:4 for gate level module






```
module dec (a, b, y0, y1, y2, y3);
 input a, b;
 output y0, y1, y2, y3
 wire abar, bbar;
 // logic - gate level modelling
```

```
not (abar, a);
not (bbar, b);
and 1 (y0, abar, bbar);
and 2 (y1, abar, b);
and 3 (y2, a, bbar);
and 4 (y3, a, b);
endmodule
```

Data flow modelling

→ logic equation used to write program in data flow modelling

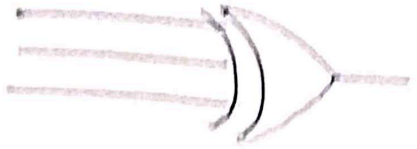
→ This modelling requires verilog operator to define logic equations.

$y = a.b$	→ &	assign y = a & b;
$y = a+b$	→ |	assign y = a | b;
$y = \bar{a}$	→ ~(or) !	assign y = ~a;
$y = a \oplus b$	→ ^	assign y = a ^ b;
$y = \overline{a.b}$	→ ~&	assign y = a ~& b; or y = ~(a&b);
$y = \overline{a+b}$	→ ~|	assign y = ~(a|b)
$y = a \odot b$	→ ~^	assign y = ~(a^b);

Eg:- $y = ab + bc + \bar{c}a$

```
assign y = a&b | b&a | ~c & a
```

Write a verilog code for 3 gate using data-flow modelling

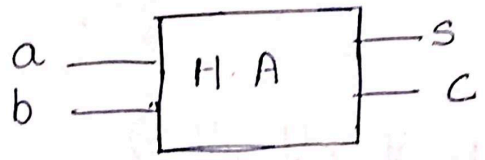


$y : a \oplus b \oplus c$

```
module xor_d(a, b, c, y)
input  a, b, c;
output y;
// logic - dataflow modelling
 assign y = a ^ b ^ c;
endmodule
```

Similarly, write program for remaining gates (3 i/p) and, or, not, xnor, nand, nor

8) Half adder program in data modelling

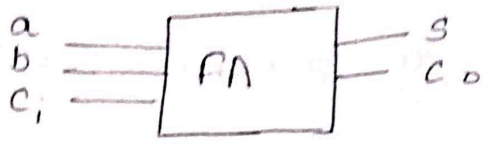


$S = a \oplus b$

$C = a \cdot b$

```
module ha_d (a, b, s, c);
input  a, b;
output s, c;

// logic -
 assign s = a ^ b;
 assign c = a & b;
endmodule
```

Q Full adder program in data flow modelling



$S = a \oplus b \oplus c_i$

$C_o : a \cdot b + b \cdot c_i + c_i \cdot a$

```
module fa_d(a, b, ci, s, co);
input a, b, ci;
output s, co;
assign s = a ^ b ^ ci;
assign co = a & b | b & ci | ci & a;
endmodule
```

Q. Decoder 2:4 program in dataflow modelling

```
module dec(a, b, y0, y1, y2, y3);
input a, b;
output y0, y1, y2, y3;
to
// logic
assign y0 = ~a & ~b;
assign y1 = ~a & b;
```

```
assign y2 = a & ~b;
assign y3 = a & b;
endmodule
```

Behavioural modelling

This modelling is similar to C language. It uses procedural constructs called always block.

Syntax

always @ (sensitivity list)

(or)

always @ *

```
module module_name ( Port list );
input a , b ;
output reg s, c;
always @ *
  begin
  ____
  ____   } → Multiway
  ____       if
             if-else
end          case
             while
endmodule    for
             repeat
```

Note:- No specification in verilog

Eg:- 1010

<size>'<format><number>

4'b1010

Eg: AND Gate

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

```
if (a==0 && b==0)
  y = 1'b0;
  else
    y = 1'b1;
```

```
case ({a,b})
2'b00 : y = 1'b0;
2'b01 : y = 1'b0;
2'b10 : y = 1'b0;
2'b11 : y = 1'b1;
endcase
```

Write a verilog code for ~~beha~~ basic gates in behavioural modelling

1. NOT

```
module not_b (a, y);
input a;
output reg y;
always @*
begin
if (a==0)
 y = 1'b1;
 else
```

```
    y = 1'b0;
end
endmodule
```

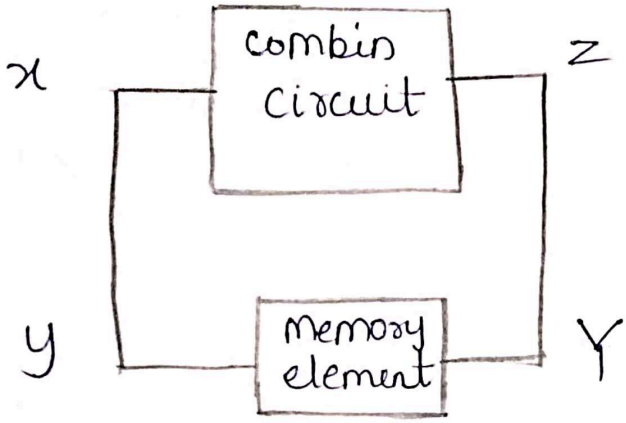
2. AND

```
module and_b(a, b, y);
input a, b;
output reg y;
always@(a, b)
begin
if (a == 0 && b == 0)
    y = 1'b0;
else
y: 1'b1;
end
endmodule
```

Unit - IV

Sequential circuits :- It is consists of combinational circuit and memory element.

The output of sequential circuit depends on present input and passed output



Y - next state
y : present state
x : present input
z = output

It depends on present input and past output.

Examples of sequential circuits

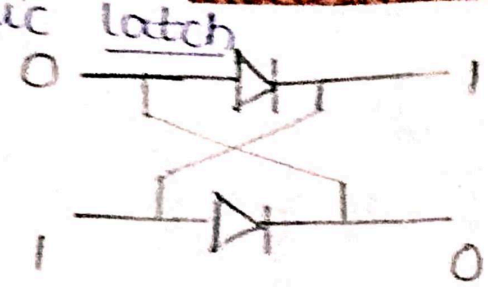
Lift elevator, tea, coffee vending machine, counters, etc.

Memory elements is used to store one bit of information it is called flip-flop or latch.

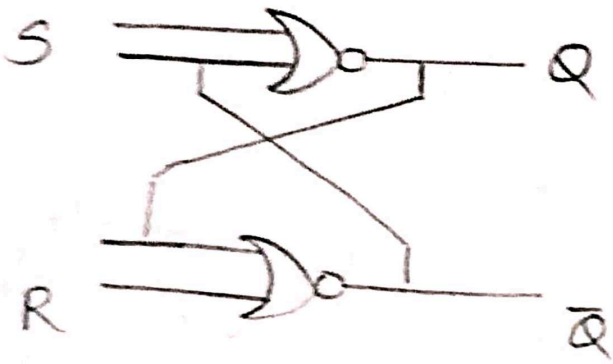
Flip-Flop - It is used to store one bit of information

Latch or flip-flop can be constructed using two gates connecting in cross coupled manner

Basic latch



SR latch



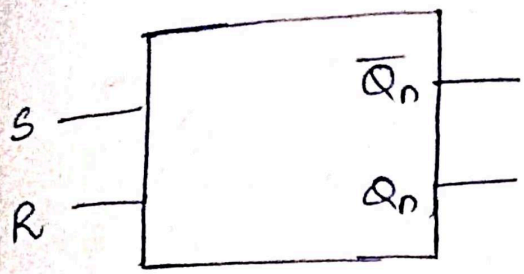
S - set
R - reset
Q - present state latch
$\bar{Q}$ - complement

Syllabus Unit - IV

Sequential circuits

- Basic latch
- Gated SR latch
- Gated D latch
- master - slave edge
- triggered flip - flop
- T Flip-Flop
- D-Flipflop
- JK Flip-Flop
- Excitation tables
- Registers and counters
- Verilog code for flip-flop

Basic SR latch using NOR gate



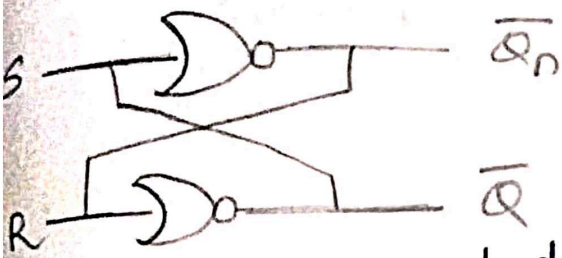
Block diagram

S = set
R = reset } inputs

Q_n = present state of latch

$\bar{Q}_n$: compliment of present state

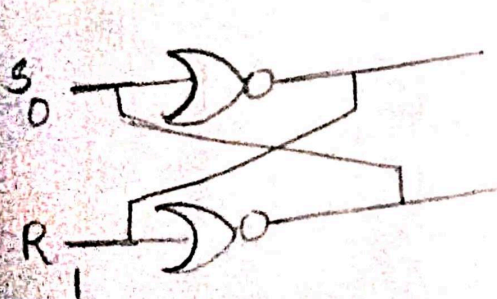
Q_{n+1} = next state of latch



Logic circuit diagram

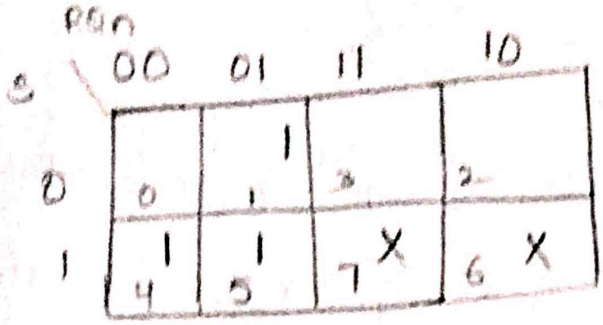
Inputs

S	R	Q_n	Q_{n+1}	state
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	.	undefined
1	1	1	.	



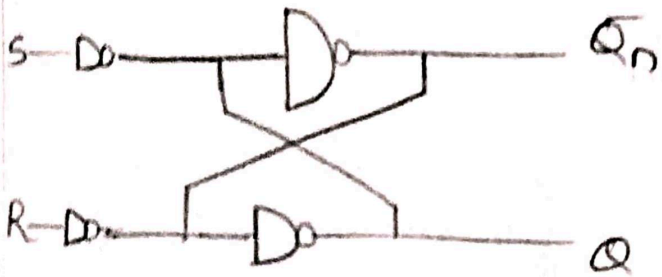
Q_{n+1}

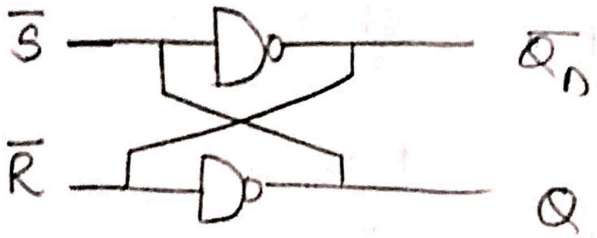
0 0 1
0 1 0
1 0 0
1 1 0



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

SR latch using NAND structure





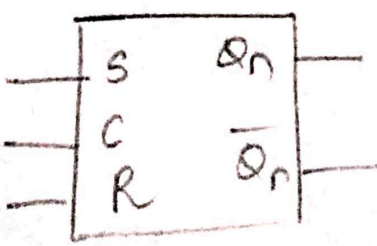
same as

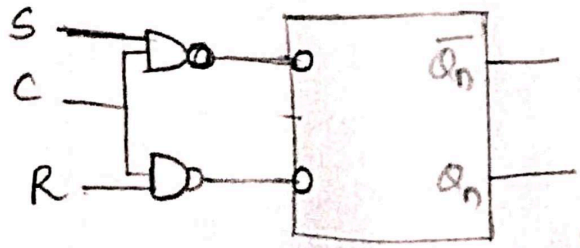
Limitations

1. more sensitive to inputs
2. undefined state

Gated SR latch using NAND structure

(To improve controll





$$Q_{n+1} = CS + \bar{R} Q_n + \bar{C} Q_n$$

C	S	R	Q_n	Q_{n+1}	state
0	x	x	0	0	hold
0	x	x	1	1	
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	0	reset
1	0	1	1	0	
1	1	0	0	1	set
1	1	0	1	1	
1	1	1	0	x	undefined
1	1	1	1	x	

Statement:- Here if C = 0 circuit acts in hold state.
If C = 1 it acts as a SR latch

CS \ RQn	00	01	11	10
00	0 x	1 x	3 x	2 x
01	4 x	5 x	7 x	6 x
11	12 1	13 1	15 x	14 x
10	8 0	9 1	11 0	10 0

0 x x
0 0 0 0
0 0 1 0
0 1 0 0
0 1 1 0

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

$Q_{n+1} = S + \bar{R}P_n$ (with DC's)

Latch

Latch circuit without clock signal

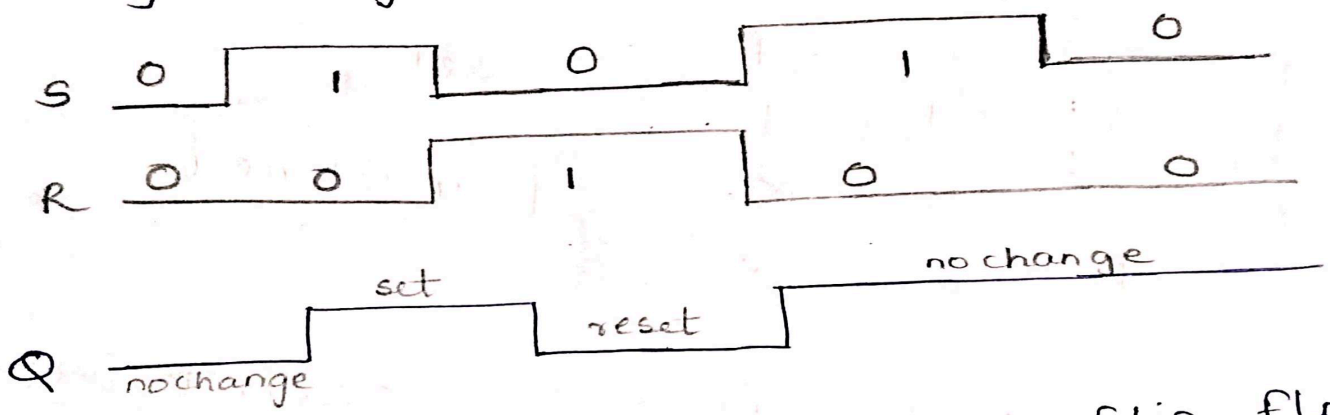
Latch output depends only on input

Flip-Flop

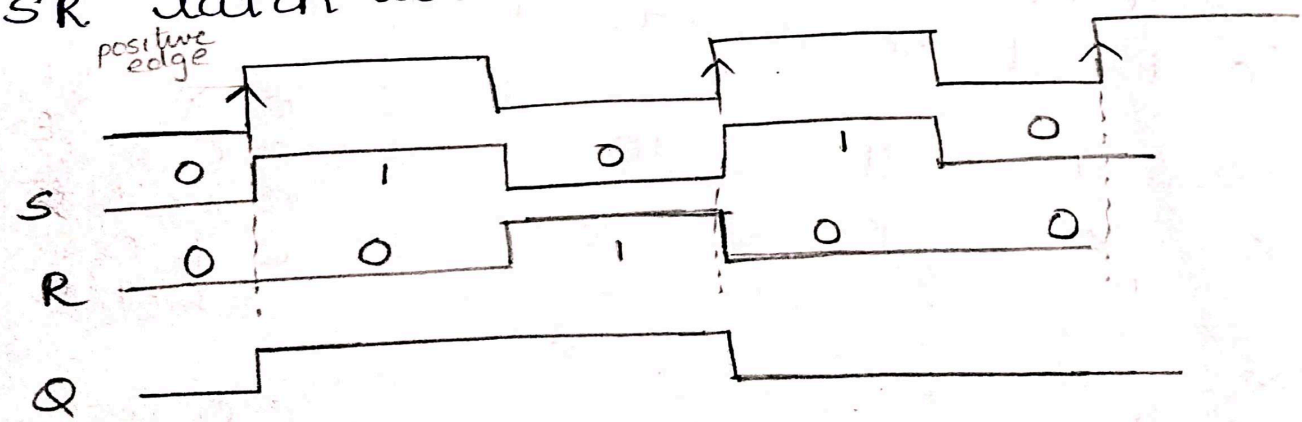
Flip-Flop circuit is with clock signal

Flip-Flop depends on clock input as well as input

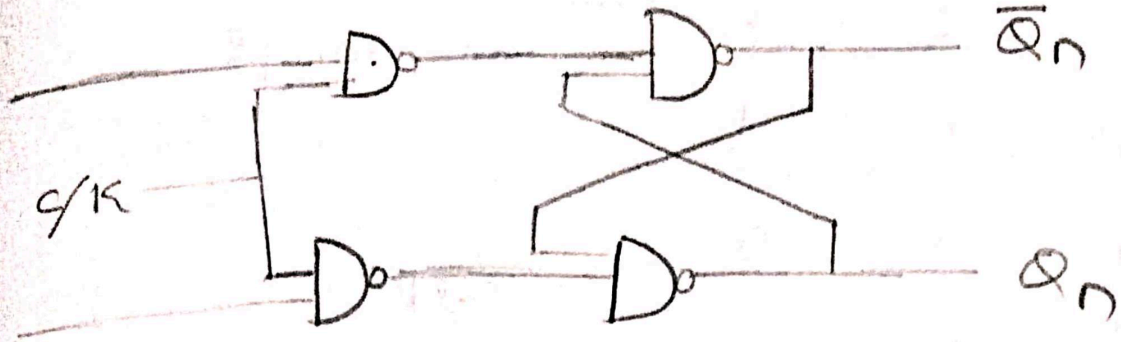
Timing diagram



SR latch with clock or SR Flip-flop



SR Flip-Flop

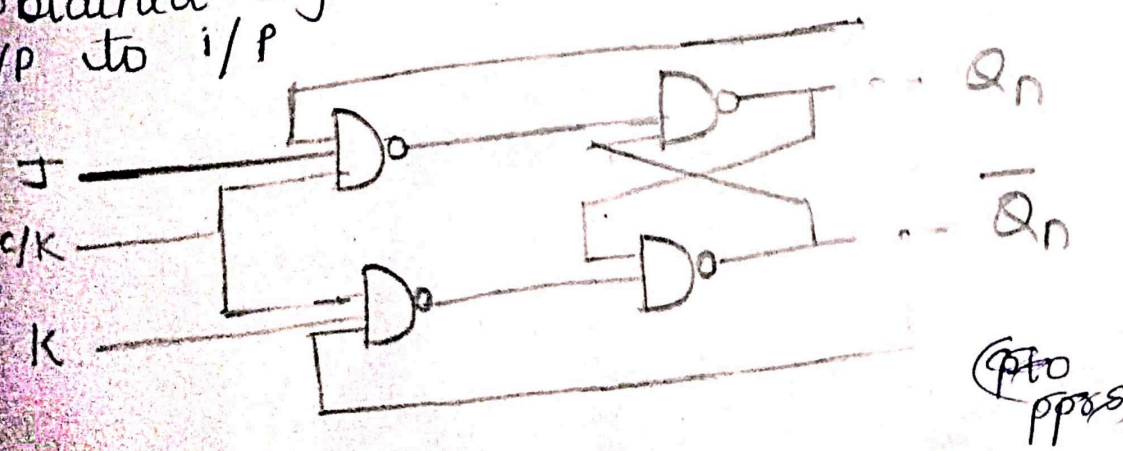


S	R	Q_n	Q_{n+1}	state
0	0	0	0	No change
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	x	not defined
1	1	1	x	

JK Flip Flop

The undefined state in SR flip flop can be overcome using JK flip-flop.

JK flip-flop circuit diagram is obtained by taking feedback from o/p to i/p



(Pto ppts)

J	K	Q_n	Q_{n+1}	vstate
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	1	toggle vstate or define vstate
1	1	1	0	

Summary of all flip-flops (1)

Q_n	Q_{n+1}	T	D	S	R	J	K
0	0	0	0	0	x	0	x
0	1	1	1	1	0	1	x
1	0	1	0	0	1	x	1
1	1	0	1	x	0	x	0

1. Flip flop conversions
2. Design sequential circuit.

a) Realize the JK FF into D

Desired: D Given: JK

b) Convert T FF into D

Desired: D Given: T

Procedure

1. Obtain transition table for the desired flip flop.
2. Generate given FF inputs from step 1
3. Generate eq flip flop inputs using k-map from given ff inputs
4. Draw the logic diagram for the given ff

Realize JK Flip Flop using SR FF

desired JK — given SR

	J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	0	X
1	0	0	1	1	X	0
2	0	1	0	0	0	X
3	0	1	1	0	0	1
4	1	0	0	1	1	0
5	1	0	1	1	X	0
6	1	1	0	1	1	0
7	1	1	1	0	0	1

Generate eqn for S & R in terms of JK, Q_n using K-Map

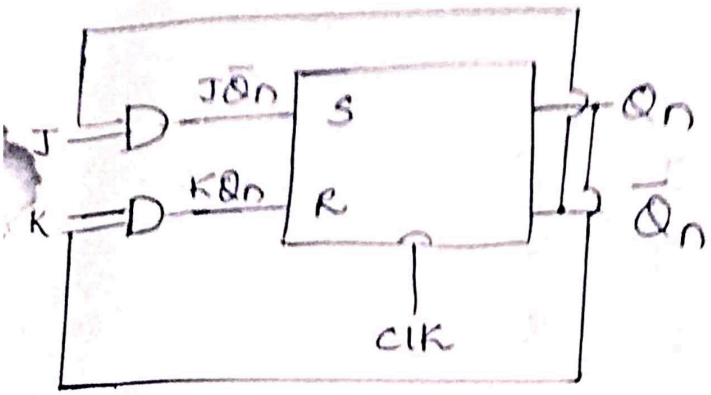
J \ KQn	00	01	11	10
0	0	X 1	3	2
1	4 [1]	X 5	7	6 [1]

$S = J\overline{Q_n}$

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

$R = KQ_n$

$S: J\bar{Q}_n$ $R: KQ_n$

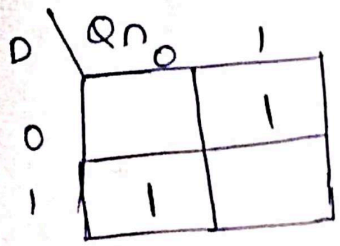


convert T flipflop into D

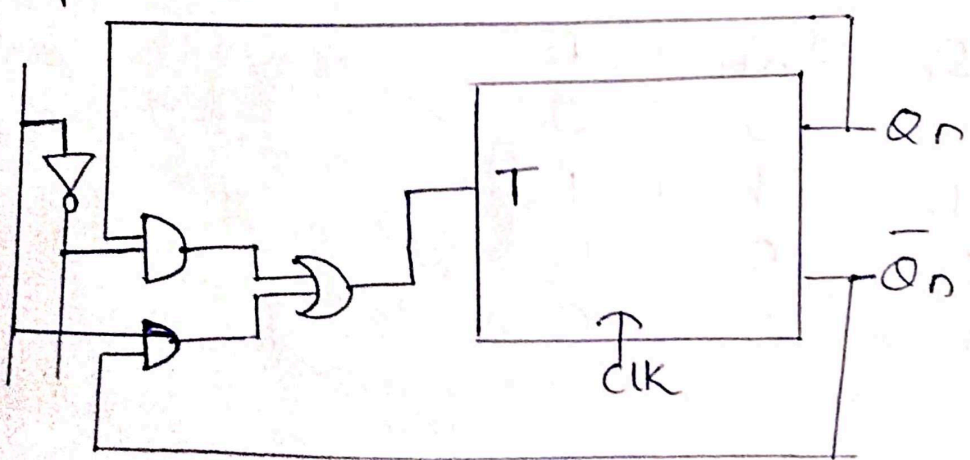
desired D

given T

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



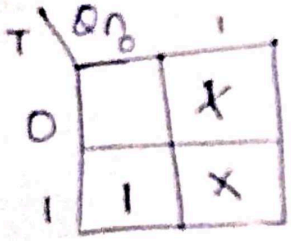
$T: \bar{D}Q_n + D\bar{Q}_n$



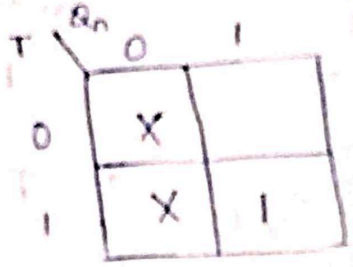
Q) Realize t flip flop using JK

desired t | given JK

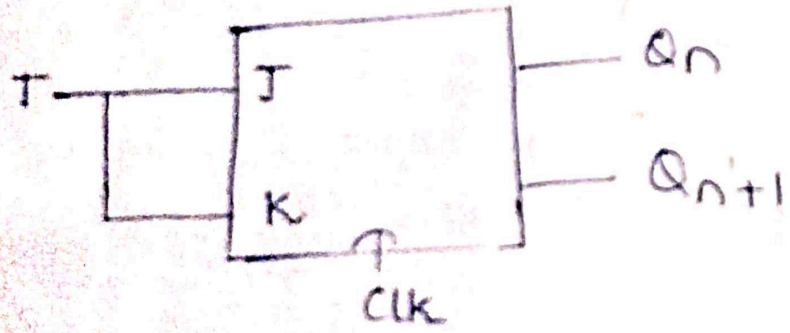
T	Q_n	Q_{n+1}	J	K
0	0	0	0	x
1	0	1	1	x
1	1	0	x	1
0	1	1	x	0



J = T



R = T

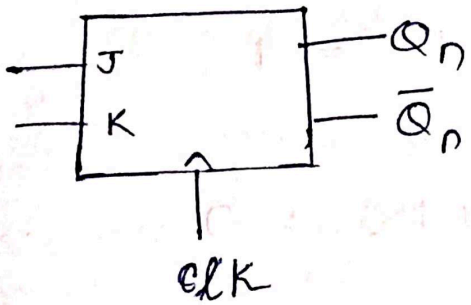


Q) Realize JK ff using D

desired JK | given D

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

JK Flip Flop



Excitation table.

J	K	Q_n	Q_{n+1}	state
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	1	toggle.
1	1	1	0	

Characteristic equation

J \ KQn	00	01	11	10
0	0	1	3	2
1	4	5	7	6

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

toggle case ... n circuit

$Q_n = 0$ $\quad Q_{n+1} = \bar{K} Q_n + J \bar{Q}_n$

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

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

Transition Table

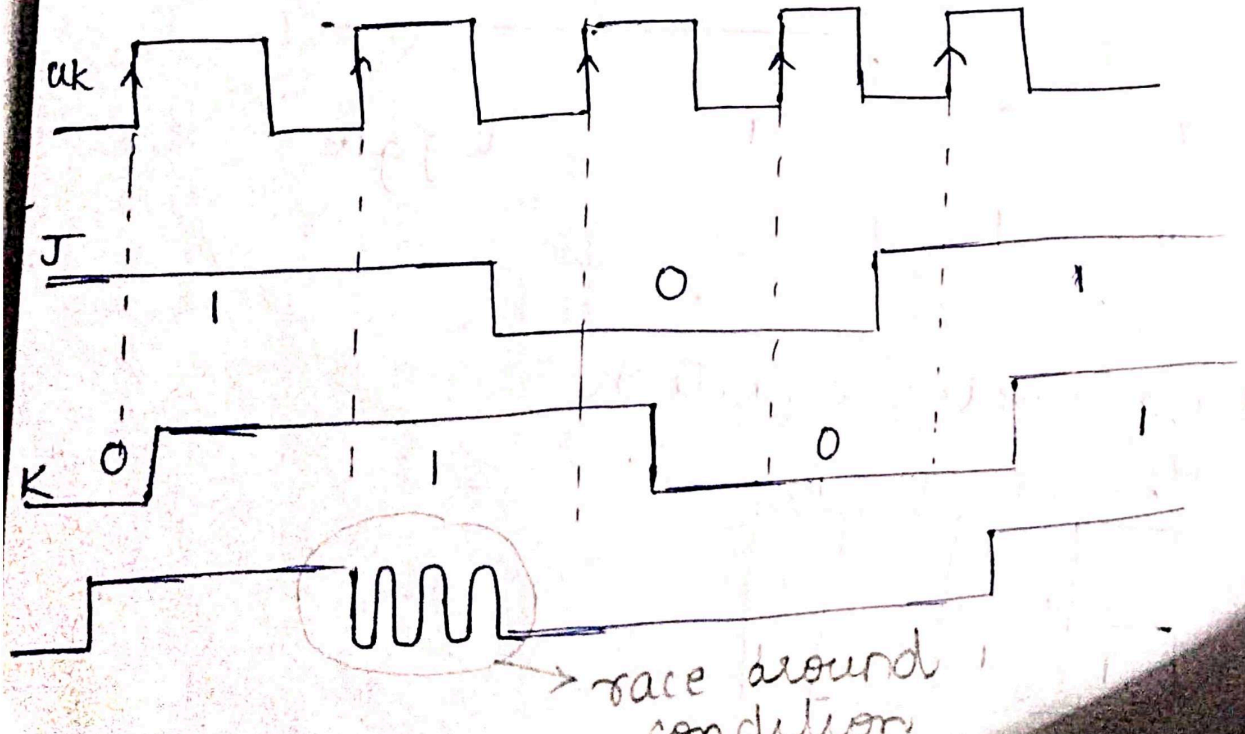
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

→

0	0
0	1
0	X

Timing diagram

Let us consider a JK flip flop



2m(☆)
What is race around cond in JK FF

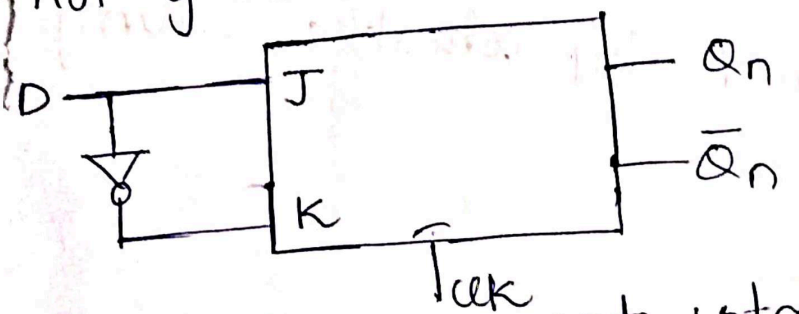
If J=1, K=1 the next state Q_{n+1} is compliment of the present state

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

In race around condition JK flip flop is more unstable. This prob can be overcome using master - slave triggered flip flop

D-Flip Flop (Delay Flip-Flop)

D-Flip Flop is obtained by short circuiting JK flip-flop terminal through NOT gate



In D Flip-Flop next state Q_{n+1} is equal to D input

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

$$J = D \quad K = \bar{D}$$

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

$$= D\bar{Q}_n + DQ_n$$

$$= D(\bar{Q}_n + Q_n)$$

$$= D$$

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

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

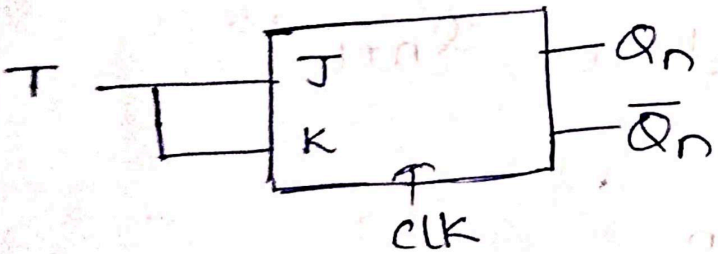
$$D = \bar{Q}_n Q_{n+1} + Q_n Q_{n+1}$$

$$= Q_{n+1} (\bar{Q} + Q)$$

$$= Q_{n+1}$$

Toggle flip Flop

T flip-flop is obtained by short circuiting JK terminals



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

$$T = J = K$$

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

$$Q_{n+1} = T \oplus Q_n$$

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

$$T = \bar{Q}_n Q_{n+1} + Q_n \bar{Q}_{n+1}$$

$$T = Q_n \oplus Q_{n+1}$$

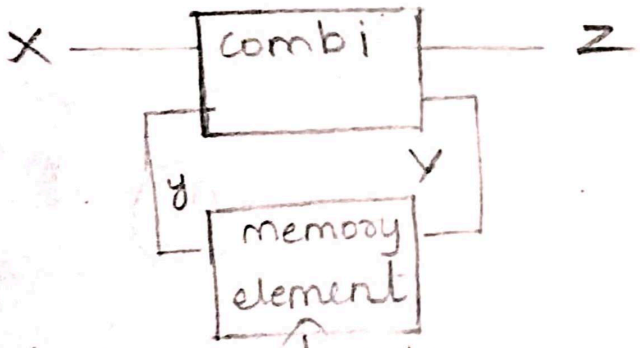
UNIT - V
Synchronous sequential Circuit

Syllabus

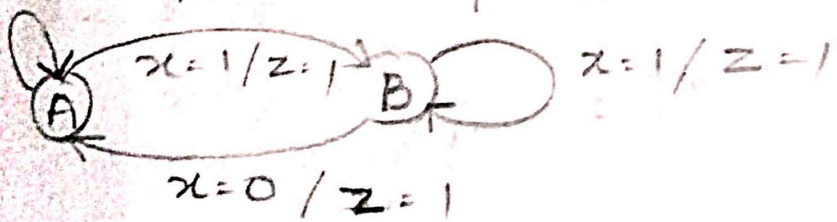
. Basic design steps
. Finate state machine (FSM) - moore & mealy state models
. State minimization
. Sequence generation and detection
. ASM charts (Algorithmic state machine)

Representation of sequential circuits

State diagram



x : input
y : present state
Y = Next state
z : output



Mealy FSM modal

x/z input/output

(A) → Name of the state

State table

PS (y)	x (input)	
	Y/z	NS/output

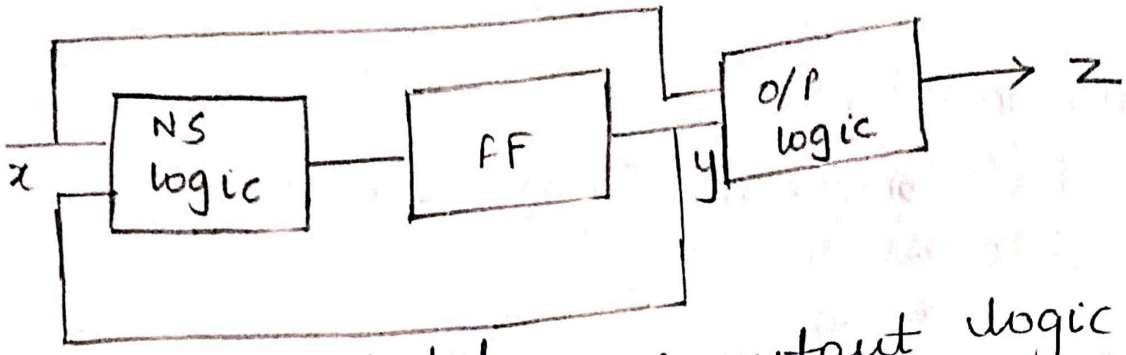
y \ x	0	1
A	A/0	B/1
B	A/0	B/1

JK Flip Flop

$z = f(x, y)$

x = input
y = present stati

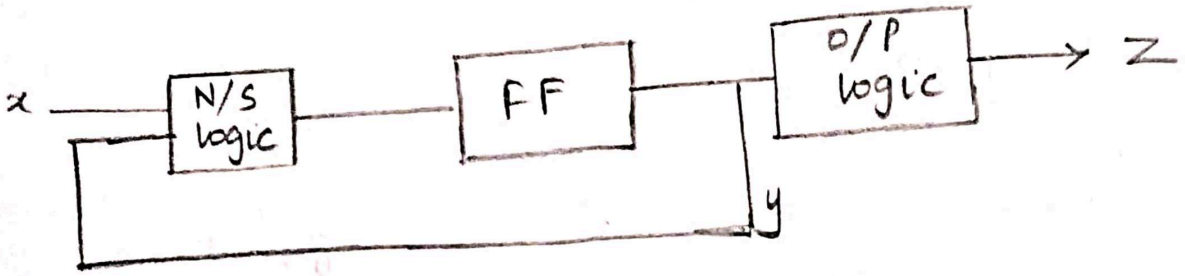
Block diagram



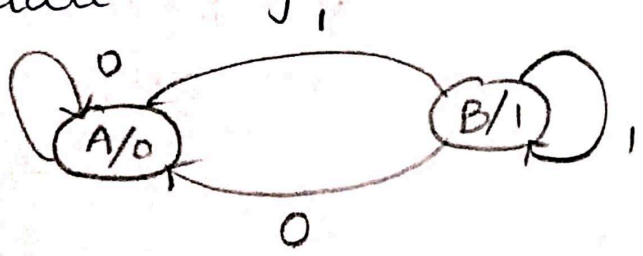
Moore FSM Modal

In Moore FSM modal output logic depends on present state of logic and independent on present input

$z = f(y)$



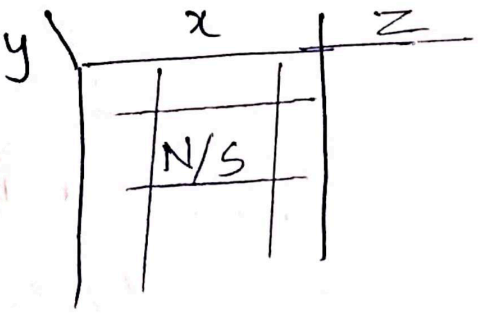
state diagram



A = name of state
z = output

state table

y	0	1	z
A	A	B	0
B	A	B	1



Design steps

For the given state table or state diagram

1. Perform state minimization (optional)
2. Find no. of states and decide number of flip flops
3. State assignment. It is used to convert alphanumerical values into binary values
4. Choose any one flip flop and draw transition table
5. Obtain logic equations for output and flip flop inputs using K-Map
6. Draw a sequential circuit
7. Draw timing diagram

Q) For the given state diagram design a sequential model using

a) D Flip Flop
b) Jk flip flop
c) T Flip flop

No. of states	No. of ff
2	1
$4 = 2^2$	2
$8 = 2^3$	3
$5 = 2^{2.1}$	3
$9 = 2^{3.1}$	4

y \ x	A	B
A	A/0	B/1
B	A/0	B/1

No. of states $= 2^{ff}$

No. of states = 2

No. of ff = 1

State assignment

A - 0

B - 1

y \ x	0	1
0	0/0	1/0
1	0/0	1/1

PS	IN	NS	FF IN	output
y	x	Y	D	Z
0	0	0	0	0
0	1	1	1	0
1	0	0	0	1
1	1	1	1	

$D = Q_{n+1}$

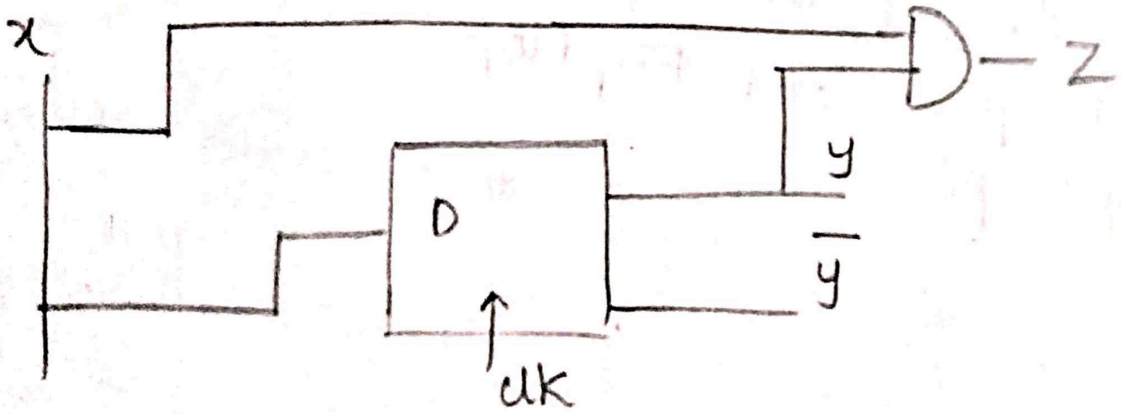
$D = Y$

$D = \bar{y}x + y x$

$D = x(\bar{y} + y)$

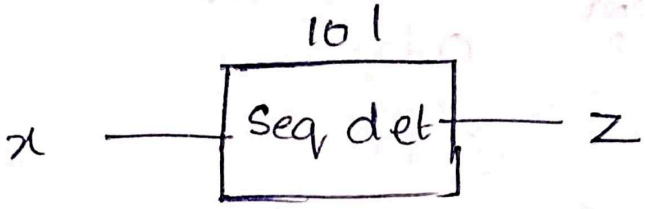
$D = x$

$Z = xy$



FSM design for sequence generator and detection

Eg:- sequence detector



if seq de
z = 1
else
z = 0

x : 0 1 0 1 0 1 0 1 0 0 0 1 0 1

z : 0 0 0 1 0 0 0 1 0 0 0 0 0 1

detect a seq 001 in the given input seq

x = $\overline{0\,0\,0\,1\,0\,0\,1\,0\,0}$ 1 $\overline{0\,0\,1}$

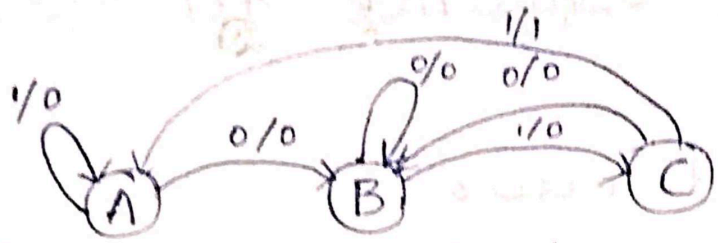
z : 0 0 0 1 0 0 1 0 0 1 0 0 1

- For a given sequence 011 draw a state diagram

0 1 1
1st 2nd 3rd – correct bits

1 0 0
1st 2nd 3rd – wrong bits

Note:- Machine starts detection process when it receives correct bits, machine goes to same state when it receives wrong first bit.



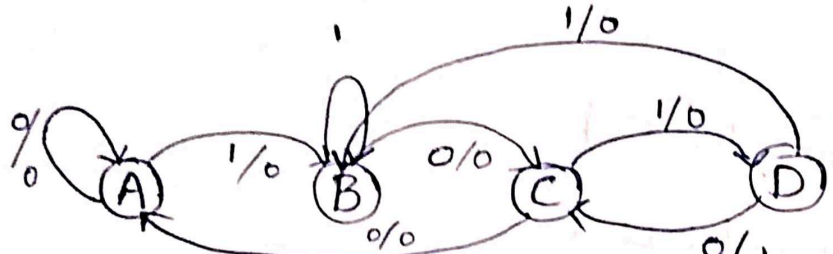
1st 2nd

0 | 0 ? 0 1 | 0

1 0 1 (0 1 1)

2. 1 0 1 0 0 1 0 1

1st 2nd 3rd 4th

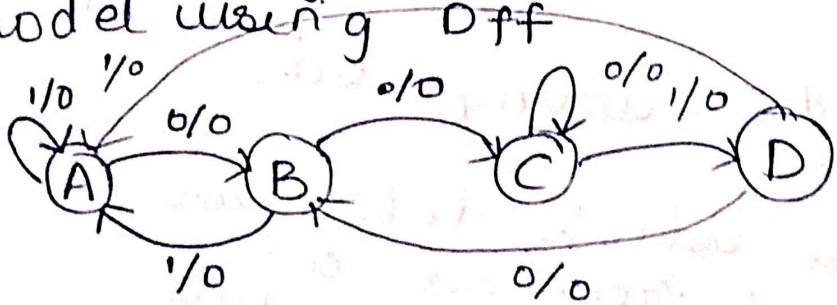


0 1 0 1 0 0 1 0 1 0

1st 1st

1 1 1 0 1 1

3. For a given sequence 0011, draw state diagram and draw sequential model using Dff



0 0 0 0 0 0 1 0

1 0 1 0 0 1 0 0 1 1

State table

y	0	1
A	B/0	A/0
B	C/0	A/0
C	C/0	D/0
D	B/0	A/1

state assi

A - 0 0

B - 0 1

C - 1 0

D - 1 1

No. of states - 4

No. of flip-flop - $2^{②} = 2$

state table

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

Y_1Y_2	x	Y_1, Y_2	D_1D_2	Z
0 0	0	0 1	0 1	0
0 0	1	0 0	0 0	0
0 1	0	1 0	1 0	0
0 1	1	0 0	0 0	0
1 0	0	1 0	1 0	0
1 0	1	1 1	1 1	0
1 1	0	0 1	0 1	0
1 1	1	0 0	0 0	1

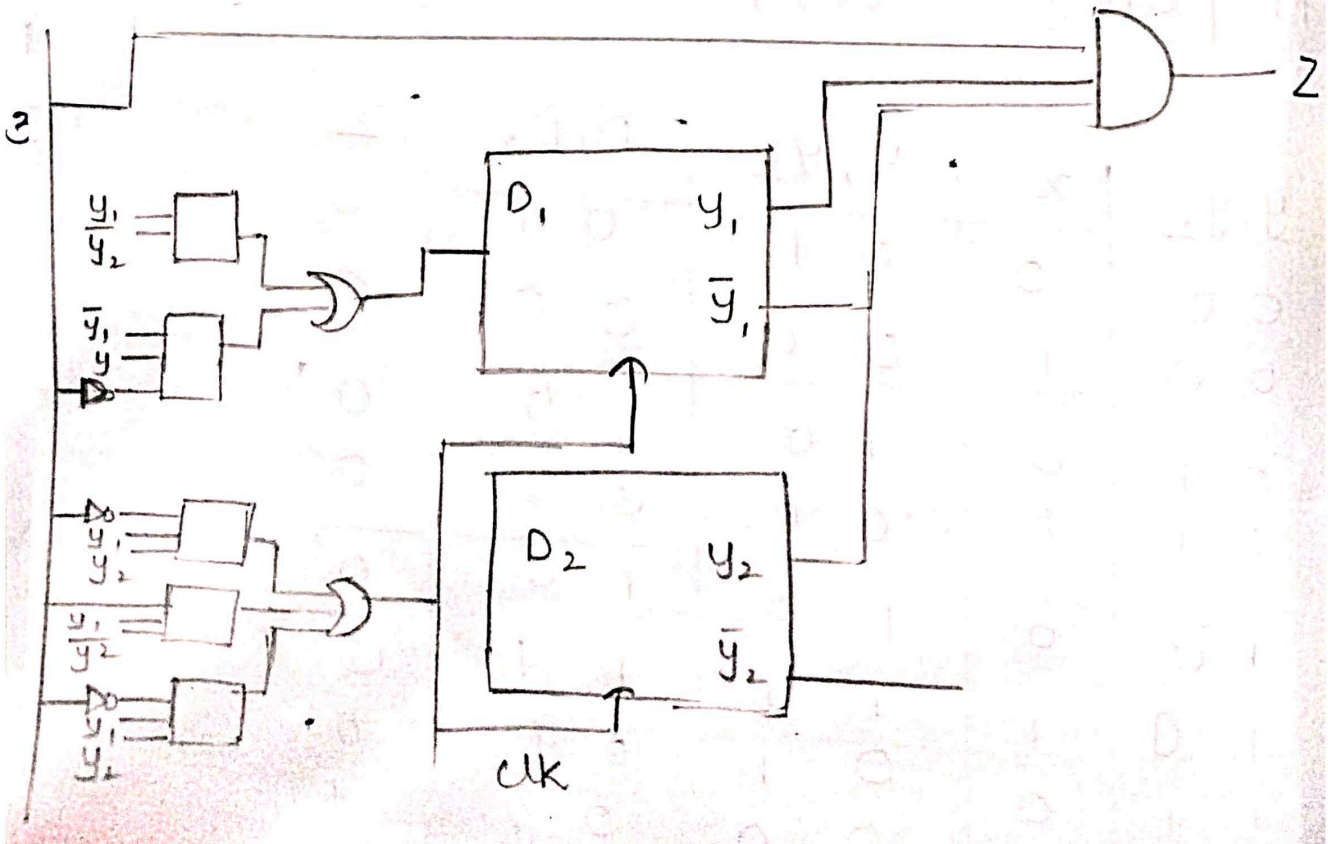
y \ y_2x	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$$D_1 = y_1\bar{y}_2 + \bar{y}_1 y_2 \bar{x}$$

y_1 \ y_2x	00	01	11	10
0	(1)			
1		(1)		(1)

$$D_2 = \bar{y}_1\bar{y}_2\bar{x} + y_1\bar{y}_2 x + y_1 y_2 \bar{x}$$

$$z = x y_1 y_2$$



State minimization

It is used to perform to minimize hardware (flip flops and logic gates) in a designed sequential circuit.

Reduction in hardware leads to improve performance, reduces area, decrease power dissipation and reduces time delay

State minimization uses two methods

1) Inspection method
2) Partition method

Q) For the given state table minimize the states using inspection method.

y	0	1
A	A/0	C/0
B	C/0	D/1
C	E/0	C/1
D	A/0	C/0
E	D/0	B/1

State A and D are equivalent states because A and D are having same next state and O/P for the given input x 0 and 1

∴ state D can be removed from

original table and replace D with A

y	0	1
A	A/0	C/0
B	C/0	A/1
C	E/0	C/1
E	A/0	B/1

A - a
B - b
C - c
E - d

y \ x	0	1
a	a/0	c/0
b	c/0	a/1
c	d/0	c/1
d	a/0	b/1

2.

y	0	1
a	a/0	b/1
b	c/1	c/0
c	c/0	e/0
d	d/0	e/0
e	a/1	b/0

C and D are equivalent state because for $x=0$ C and D goes to same state and produce same output for $x=1$ C and D goes to e state and produce same output 0

∴ D can be removed from original circuit and replace D with C

d = C
a = A
b = B
c = C
e = D

y \ x	0	1
A	A/0	B/1
B	C/1	C/0
C	C/0	D/0
D	A/1	B/0

Partition method

For a given table,

y \ x	0	1
A	C/1	D/0
B	C/1	E/0
C	B/1	E/0
D	D/0	B/1
E	E/0	A/1

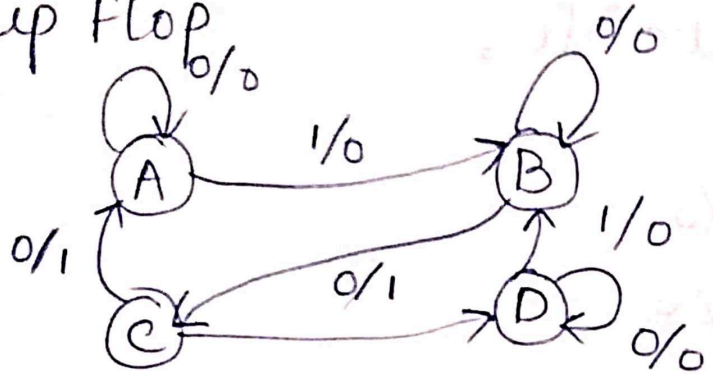
	Partitioning	Status
P_0	(A B C D E)	separate (A B C) (D E)
o/p x = 0	1 1 1 0 0	
o/p x = 1	0 0 0 1 1	
P_1	(A B C) (D E)	No separation possible
NS x = 0	C C B D E	
NS x = 1	D E E B A	

Here A and B and C are equivalent states
∴ remove state B and C replace B & C with A

D & E are equivalent states, remove E from the table. Replace E with D

y	0	1
A	A/1	D/0
D	D/0	A/1

Q) Minimize the given state diagram and design sequential ckt using T flip flop



y	x = 0	x = 1
A	A/0	B/0
B	B/0	C/1
C	A/1	D/1
D	D/0	B/0

Here state A & D are equivalent because for input $x=0$; when next state for A & D are self state

for $x=1$, both the states A & D going to state B

Hence A & D are equivalent

∴ Remove 'D' and replace 'D' with 'A' using inspection

$A = D$

Minimize table

y \ x	0	1
A	A/0	B/0
B	B/0	C/1
C	A/1	A/1

state assi

A — 00

B - 01

C - 10

State transition table

$y_1 y_0$	x	$T_1 T_0$	z	$T_1 T_0$
00	0	00	0	00
00	1	01	0	01
01	0	01	0	00
01	1	10	1	11
10	0	00	1	10
10	1	00	1	10
11	0	xx	x	xx
11	1	xx	x	xx

$y_1 y_0$ \ x	0	1
00	00/0	01/0
01	01/0	10/1
10	00/1	00/1

Generate logic eq for T_1, T_0 & z using K. Map

y_1 \ y_0x	00	01	11	10
0	0	1	1 3	2
1	4	5	x7	x6

$$z = y_1 + y_0 x$$

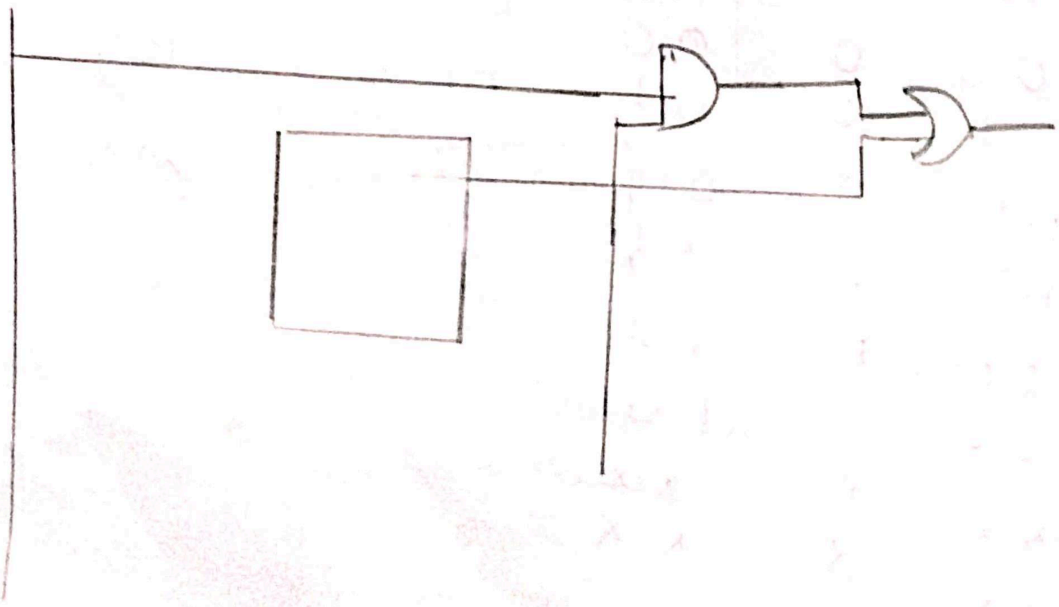
y_1 \ y_0x				
			1	
	1	1	X	1

$$T_1 = y_1 + y_0 x$$

y_1 \ y_0x				
		1	1	
			X	X

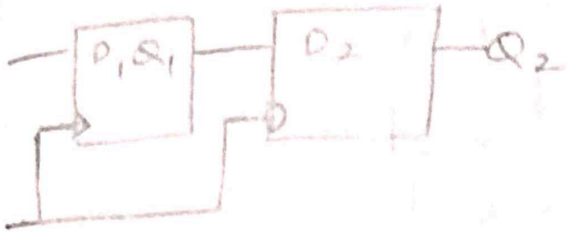
$$T_0 = \bar{y}_1 x$$

sequential ckt

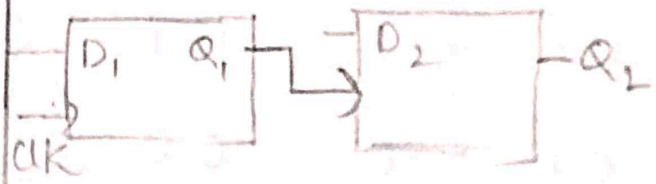


Design synchronous 2 bit up-down counter

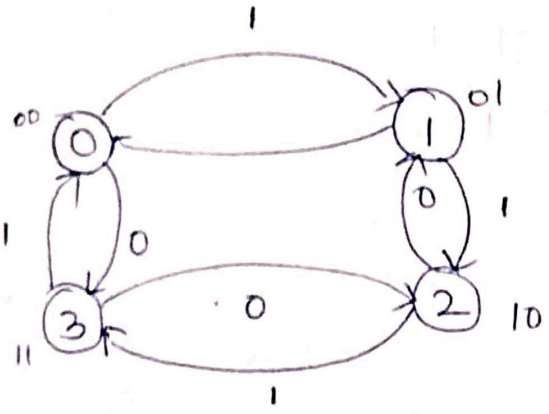
Synchronous



Asynchronous



- Easy to design
- stable



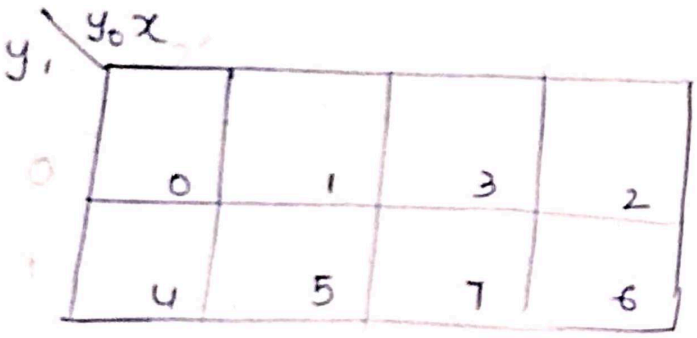
$x=0$ down $x=1$ up

state assign

0 - 00
1 - 01
2 - 10
3 - 11

y_1y_0	x = 0	x = 1	Z_1Z_2
00	11	01	00
01	00	10	01
10	01	11	10
11	10	00	11

$y_1 y_0$	x	$Y_1 Y_0$	$D_1 D_0$	$z_1 z_0$
0 0	0	1 1	1 1	0 0
0 0	1	0 1	0 1	0 0
0 1	0	0 0	0 0	0 1
0 1	1	1 0	1 0	0 1
1 0	0	0 1	0 1	1 0
1 0	1	1 1	1 1	1 0
1 1	0	1 0	1 0	1 1
1 1	1	0 0	0 0	1 1

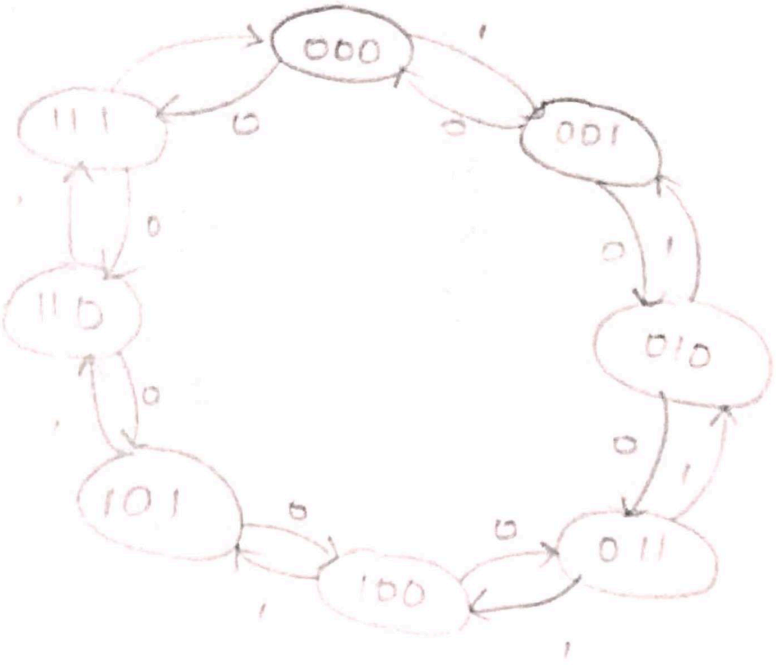


Design of 3 bit up-down counter

state

$2^3 = 8$

moore model



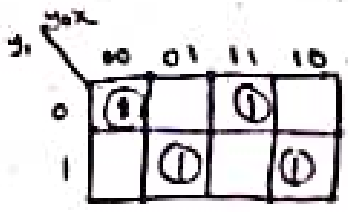
y_2 y_1 y_0	w	T_2 T_1 T_0	D_1 D_2 D_3
0 0 0	0	1 1 1	1 1 1
0 0 0	1	0 0 1	0 0 1
0 0 1	0	0 0 0	0 0 0
0 0 1	1	0 1 0	0 1 0
0 1 0	0	0 0 1	0 0 1
0 1 0	1	0 1 1	0 1 1
0 1 1	0	0 1 0	0 1 0
0 1 1	1	1 0 0	1 0 0
1 0 0	0	0 1 1	0 1 1
1 0 0	1	1 0 1	1 0 1
1 0 1	0	1 0 0	1 0 0
1 0 1	1	1 1 0	1 1 0
1 1 0	0	1 0 1	1 0 1
1 1 0	1	1 1 1	1 1 1

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

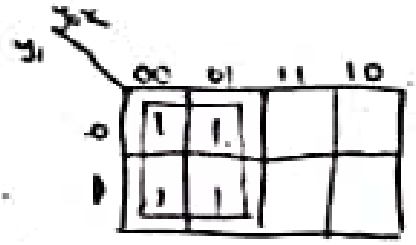
Logic equation

06/01/23
Friday

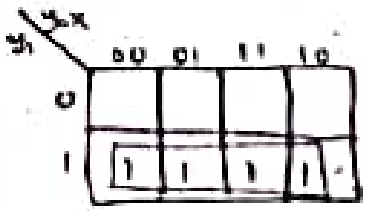
2bit up/down counter ---



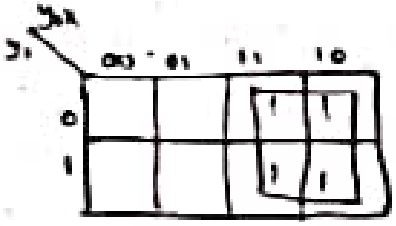
$D_1 = y_1 \oplus y_0 \oplus x$



$D_0 = \bar{y}_0$

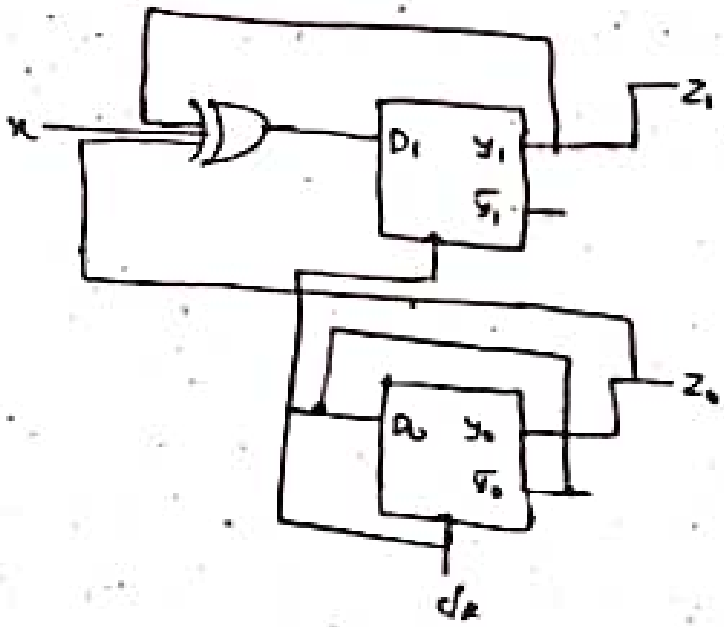


$Z_1 = y_1$



$Z_0 = y_0$

Circuit Diagram / Sequential Circuit



Algorithmic state Machine charts (ASM chart)

ASM stands for Algorithmic State Machine used to Control digital system. It carries input and output information, it is a step-by-step procedure.

Components of ASM

- State Box
- Decision box
- Conditional output box

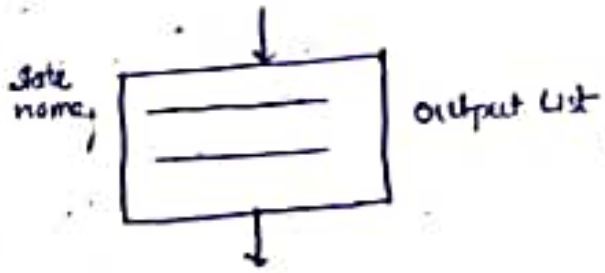
→ Conditional output box also known as Meeley output box

Advantages of ASM

- Operation of a digital system can be easily understood by inspection of the SM chart.
- ASM charts represent physical hardware
- The SISM chart are equivalent to a state graph, and it leads directly to a hardware realization
- ASM charts can be described the operation Sequential Circuits
- ASM charts are easier to understand and can be Converted Several equivalent form
- The ASM Chart may be equivalently expressed as a State and output table.

ASM Components

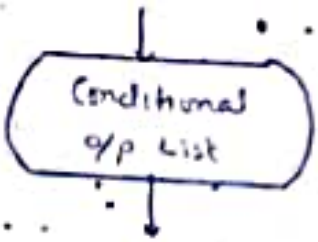
State Box: The state of the system is represented by a state box. It is a rectangular box. At the top left hand corner the name of the state and within the state box the output signals are listed



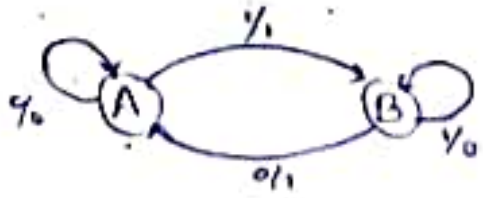
State Box

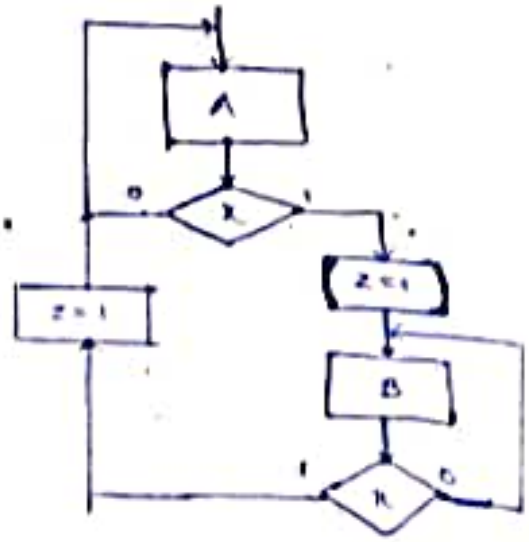
Decision Box: It is a diamond-shaped box with true false branches. Boolean Condition is placed in the box and the decision is made from the value of one or more inputs

Conditional box:



For a given mealy FSM draw the ASM chart

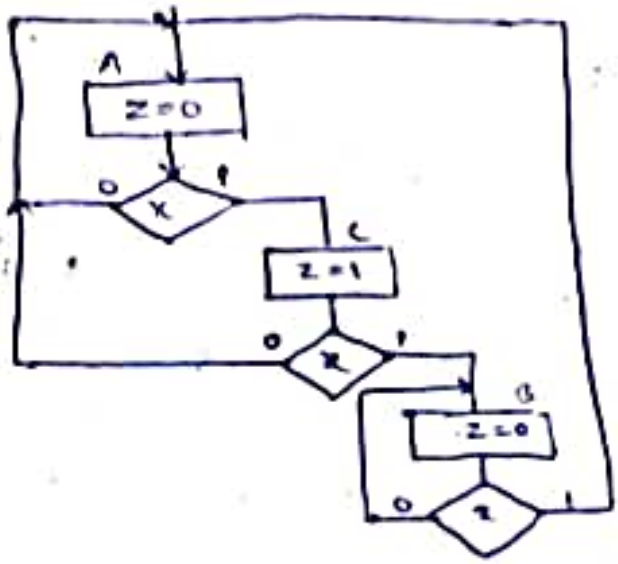




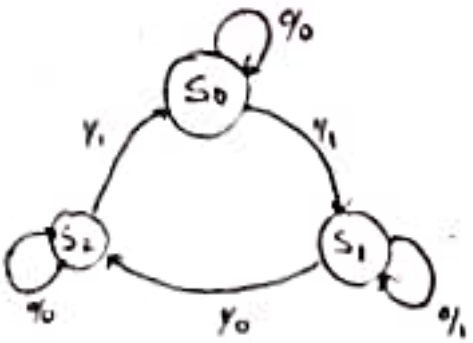
For the given Moore FSM Table draw a ASM chart

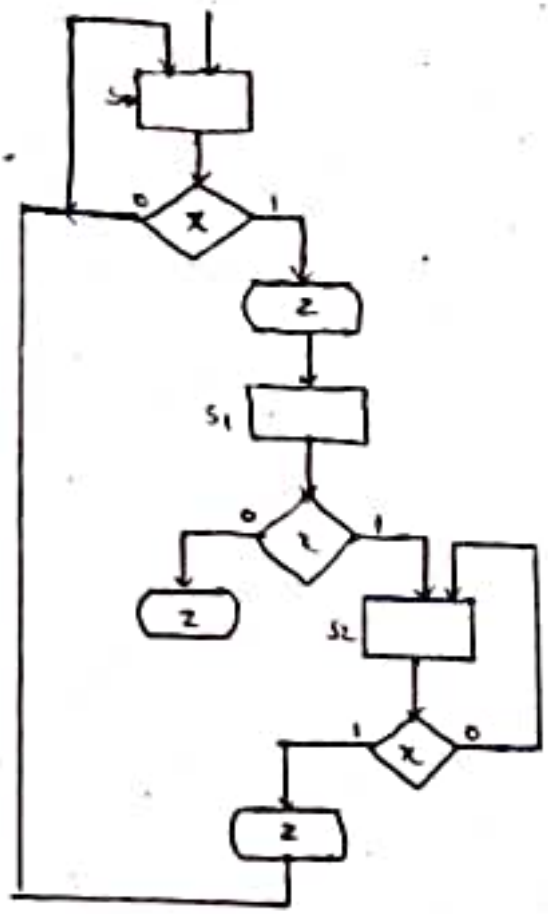
PS	x=0	x=1	z
A	A	C	0
B	B	A	0
C	A	B	1

NOTE: ASM chart for Moore FSM use only State Box and decision box



Mealy Diagram to ASM





AMoore SISM with Multiple Inputs

Moore diagram to ASM