Digital Electronics - 01 (13/07/2

Number system	Symbol	Base / radix
Decimal	(0-9)	10
Binary	(0, 1)	2
octal	(0-7)	8
Hexadecimal	(0-9, A, B, C, D, E, F)	16

512
3
1536
64
16
6
1622

✓ Any number system ⟶ decimal number sys.

$(0026)_{16} = 0\times16^3 + 0\times16^2 + 2\times16 + 6\times16^0$

$= (38)_{10}$

$(3126)_{10} = 3\times10^3 + 1\times10^2 + 2\times10^1 + 6\times10^0$

$= \sum D_i r^i$

$(1101)_{(4bit)} \longrightarrow 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

(8 4 2 1)

$= (13)_{10}$

$(3126)_8 = 3\times8^3 + 1\times8^2 + 2\times8^1 + 6\times8^0$

$= (1622)_{10}$

8 4 2 1

0000 → 0
0001 → 1
0010 → 2
0011 → 3
0100 → 4
0101 → 5
0110 → 6
0111 → 7
1000 → 8
1001 → 9
1010 → A
1011 → B

Binary no. from 4 bits

1100 → C
1101 → D
1110 → E
1111 → F

$(0.51)_{10} = 5\times10^{-1} + 1\times10^{-2}$

After point → $(0.10)_2 = 1\times2^{-1} + 0\times2^{-2} = (0.5)_{10}$

$(0.41)_8 = 4\times8^{-1} + 1\times8^{-2} = 0.5 + 0.015$

$= (0.515)_{10}$

$(0.80)_{16} = 8\times16^{-1} = (0.5)_{10}$

⇒ Fractional Decimal to hexadecimal / Binary ---

fraction → $f \times r = I_1 \cdot f_1$

Base → $f_1 \times r = I_2 \cdot f_2$

$f_2 \times r = I_3 \cdot f_3$

$\vdots$

$(\cdot I_1 I_2 I_3)$ req. no. sys.

e.g. $(0.10)_2 = (0.1)\times2 = 0.2$

$0.2\times$

in decimal

$(0.5)_{10}$ → $0.5\times2 = 1.0$

$0\times2 = 0.0$ ↓

$= (0.10)_2$

$(0.5)_{10}$ in octal

$0.5\times8 = 4.0$ ↓

$0\times8 = 0.0$

$(0.40)_8$

$(0.5)_{10}$ in hexadecimal

$0.5\times16 = 8.0$

$0\times16 = 0$

$(0.80)_{16}$

Decimal to any number system → Number

	Remainder	
r	I	R1
r	I1	R2
r	I2	R3
		R4

$(R_4 R_3 R_2 R_1)_r$ →

base

e.g. $(13)_{10}$ → binary

2 | 13		1
2 | 6		0
2 | 3		1
2 | 1		1
	0	

$(1101)_2$

e.g. $(1622)_{10}$

		remainder
8 | 1622		6
8 | 202		2
8 | 25		1
8 | 3		3
	0	

$(3126)_8$

16 | 38		6
16 | 2		2
	0	

⇒ $N = (I \cdot F)$

e.g. $(1622)_{10} \longrightarrow (?)_8$

(3126)

8 | 1622		6
8 | 202		2
8 | 25		1
8 | 3		3
0		

$(3126)_8$

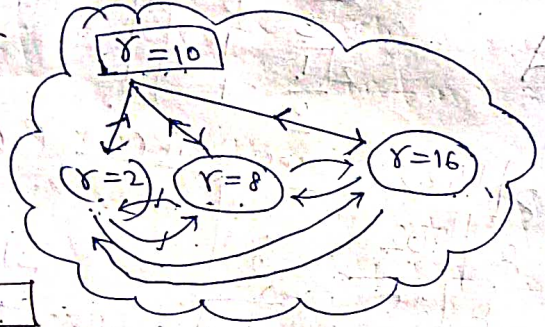
but $(1622.5)_{10} \longrightarrow (3126.4)_8$

1) convert the Int part
2) convert ⁺ fractional ⁺
and combine the result.

e.g. $(38.5)_{10} \longrightarrow (26.8)_{16}$

16 | 38		6
16 | 2		2
0		

$.5 \times 16 = 8.0$
$0 \times 16 = 0.0$
$.80$



Binary ⟷ octal

e.g. $0b1010.0111_2 \longrightarrow (?)_8$

→ Make grp of 3 : 0010.010 . 011 . 100

1 2 3 4

$(12.34)_8$

octal	Binary (4 2 1)	weightage
0	000	$2^2 \times 0 + 2^1 \times 0 + 2^0 \times 0$
1	001	
2	010	
3	011	
4	100	
5	101	
6	110	
7	111	

(Battle)

$2^{30} = 1K \times 1K \times 1K = 1G$

$2^{20} = 1K \times 1K = 1M$
$= 1048576 = 1M$

$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$ (1K) → Kilo

1KB = 1024 x 8 bits

$\rightarrow 2^{10} \times 2^3 = 2^{13}$

e.g. $(1010.0111)_2 \longrightarrow (?)_{16}$

8421
0111
1+2+4 = 7

8421
1010
0+2+0 +8 = A

make grp of 4 → $(A.7)_{16}$

weitage
8,4,2,1

4 GB = $2^2 \times 2^{30}$ 32 bit

1KB = 2^{10} (8+8

hexadecimal to octal

octa ~~decimal~~

decimal

binary → decimal

e.g. $(A.7)_{16}$

8421
0111

$00(1010.0111)_2$

1 2 3 4

$(12.34)_8$

hexadecimal ⟷ to ⟷ octal via binary

Doubt

{ 32 bit : Adress ✓
  8 bit : data ✓

* 4 GB → $2^2 + 2^{30}$ (8 bit)

* (A)$_{24}$ → A/D → data (8)
  Adress

If 32 bit Adress & 32 bit data
↓
(A/D)$_{32}$

2^{32} (32 bits)
↓
No. of Adress → each adress contain 32 bit

** 1 TB = 1K x 1K x 1K x 1K x 1K
= 2^{40}

40 bit adress
64 bit ~~data~~/Bus

2^{64} (32/64)
→ system

P RAJA → Digital Elec.
(GATE)

BCD

Binary coded decimal.

$(123)_{10} \xrightarrow{BCD} (0001\ 0010\ 0011)_{BCD}$

each digit is repr. by 4 bit '2'

Decimal → Binary for Hexa.

A, B, C, D, E → 1010 unused in BCD

only used in BCD: 0 – 0000 ... 9 – 1001

'N' bit → 2^N symbols

so in '4' bits

$2^4 = 16$ symbols

out of which 6 are unused : A → E

ASCII character code

capital A starts with 100

A → (100 0001), B → 100 0010

small a starts with 110

a → (110 0001), b → 110 0010

0 → (011 0000)

9 → (011 1001)

5 → (011 0101)

using 7 bit

↓

2^7

$= 128$ symbols

A → 100 0001

B → 10

→ (010 0011) } special char.

$ → 010 0100)

---

x AND y

↓

intersection



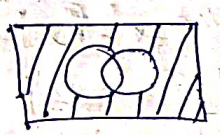
→ x & y

OR → union



→ x OR y

NOT →



or



x' / $\bar{x}$ / x!

AND:



x.y

OR



x + y

NOT



x	y	AND	OR	NOT x'	NOT y'
0	0	0.0 = 0	0+0 = 0	1	1
0	1	0.1 = 0	0+1 = 1	1	0
1	0	1.0 = 0	1+0 = 1	0	1
1	1	1.1 = 1	1+1 = 1	0	0

*** test in LAB

✓ done

---

discrete :-

$f : A \to B$

$g : B \to C$

$f(a) = a - 1$

$g(y) = b^r$

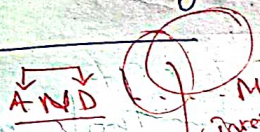
$fog = f(g(b)) = (a-1)^r \quad b^r - 1 = x^r - 1$

$gof = g(f(a)) = (a-1)^r = (x-1)^r$

as $fog \neq gof$ (not commutative)

AND → MULTIPLICATION

OR → ADDITION



Digital logic-03 (25/07)

→ Boolean Algebra :-

operator ~~procedure~~ : precedence :-

( ), NOT, AND, OR

$\bar{x}$ / x', '.', '+'

$0 \cdot y = 0$

$1 \cdot y = y$

x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1

$0 + y = y$ ✓

$1 + y = 1$ ✓

→ $(y + 0) = (0 + y) = y$

$1 \cdot y = y \cdot 1 = y$

$x + y = y + x$ or $x \cdot y = y \cdot x$ } commutative

→ $x(y + z) = x \cdot y + x \cdot z$ (distributive)

→ $x + y \cdot z = (x + y)(x + z)$

→ $x + (y + z) = (x + y) + z$

$x \cdot (y \cdot z) = (x \cdot y) \cdot z$ } Associative

→ $\bar{\bar{x}} = x$ (complement)

✓ $x + \bar{x} = 1$ } OR

✓ $x \cdot x' = 0$ } AND

$x \cdot x = x$

→ dual : '.' ↔ '+'

0 ↔ 1

$0 + y = y$ — (1)

$1 + y = 1$ — (2)

$1 \cdot y = y$ — (5)

$0 \cdot y = 0$ — (6)

$x + x' = 1$ — (3)

$x + x = x$ — (4)

$x \cdot x' = 0$ — (7)

$x \cdot x = x$ — (8)

$\bar{\bar{x}} = x$ — (9)

$x + y = y + x$ — (14)

$x + (y + z) = (x + y) + z$ — (10)

$x \cdot y = y \cdot x$ — (15)

$x \cdot (y + z) = (x \cdot y) \cdot z$ — (11)

$x \cdot (y + z) = x \cdot y + x \cdot z$ — (12)

$x + y \cdot z = (x + y)(x + z)$ — (13)

* $(x + y)' = \overline{x + y} = x' \cdot y'$ (NOR)

* $(A \cdot B)' = \overline{A \cdot B} = x' + y'$ (NAND)

} DeMorgan's theorem

x	y	x'	y'	x'.y'	x'+y'
0	0	1	1	1	1
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	0

OR	NOR $\overline{x+y} = x'.y'$
0	1
1	0
1	0
1	0

AND	NAND $\overline{x.y} = \bar{x}+\bar{y}$
0	1
0	1
0	1
1	0

eqv

eqv

$x'.y' = \overline{x+y}$ → (16)

$x'+y' = \overline{x.y}$ — (17)

⇒ NOR :



⇒ NAND :



AND priority ↑

⇒ Ab-sorp-tion

$\left(x + (x.y)\right) = x.(1+y) = x.1 = x$ — (18)

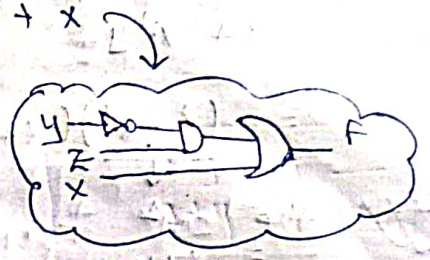
(1) or (y) = 1

x common

$x.(x+y) = x$ — (19)

⇒ $f = \left(x + ((y)'.z)\right)$ ⟶ Next pg

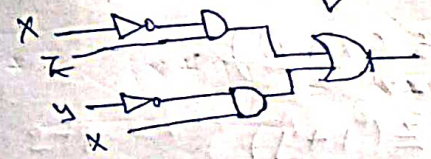
z	y	x	$f = y'z + x$
0	0	0	0+0 = 0
0	0	1	0+1 = 1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

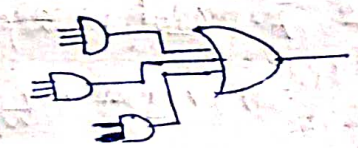


⇒ $f_2 = x'y'z + x'yz + xy'$ ✓ (same result in T.T)

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

$= x'z + xy'$ ✓





3(input)

⇒ $x(x'+y) = xx' + xy = xy$ } minimization

Q) $x + x'y$ (minimize this

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

$= 1$

$= x + y$ (Ans)

Q) $(x+y)(x+y') = xx + xy' + yx + yy'$ → 0

$= x + yy'$ → 0

$= x$

$= x + xy' + xy$

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

$= x$ (1)

Q) $xy + x'z + yz$

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

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

$= xy + x'z + yzx + yzx'$

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

$= xy + x'z$

dual →

$(x+y)(x'+z)(y+z)$

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

Digital logic - 04 (31/07)

* Demorgan's theorem :-

NOT of OR NOR

$$(A+B)' = A' \cdot B'$$

$$(A+B+C)' = A' \cdot B' \cdot C'$$

$$(A+B+C---Z)' = A' \cdot B' \cdot ---Z'$$

NAND

$$(AB)' = A' + B'$$

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

$$(ABC--Z)' = A' + B' + C' --Z'$$

* complement of a fn :-

'+' $\leftrightarrow$ '.'

complement literal

e.g. $(A'B')' = A'' + B''$

$= A + B$

⇒ Dual of function :-

① '+' $\leftrightarrow$ '.'

$F_1 = x'yz' + x'y'z$

dual of $F_1 = (x'+y+z')(x'+y'+z)$

$\overline{F_1} = (x+y'+z)(x+y+z')$

* $F_2 = x(y'z' + yz)$

dual of F_2

$= x + ((y'+z') \cdot (y+z))$

$\overline{F_2} = x' + (y+z)(y'+z')$

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

- $A+B \rightarrow$ OR
- $A \cdot B \rightarrow$ AND
- $\overline{A+B} \rightarrow$ NOR
- $\overline{A \cdot B} \rightarrow$ NAND
- $AB' + A'B \rightarrow$ XOR
- $AB + A'B' \rightarrow$ XNOR

x : exclusive XOR XNOR

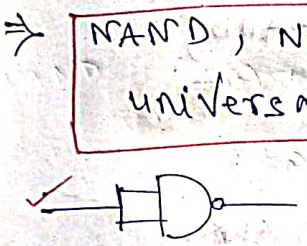
	A	B	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
$A'B'$	0	0	0	0	0	0	0	0	0	0
$A'B$	0	1	0	0	0	0	1	1	1	1
AB'	1	0	0	0	1	1	0	0	1	1
AB	1	1	0	1	0	1	0	1	0	1
			NULL	AND		$AB' + A'B$ = A	$A'B$	B	XOR	OR

0000
0010
0011 — 3
0100 — 4

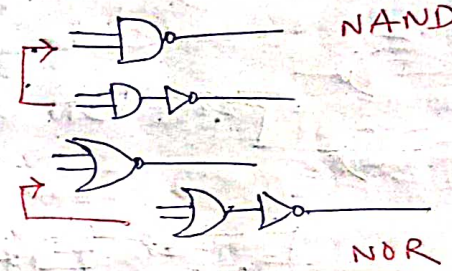
all terms are absent

F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	1
0	1	0	1	0	1	0	1
$A'B'$ ↓ NOR		XNOR	B'	$B'+A$	A'	$A'+B$	NAND	constant-1

NAND, NOR, universal gate :-



$\overline{X \cdot X} = \overline{X}$ (NOT)

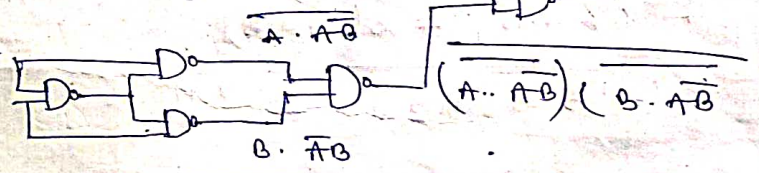




$\overline{A \cdot B} = A \cdot B$ (AND)

$\overline{A' \cdot B'} = A'' + B'' = A + B$ (OR)



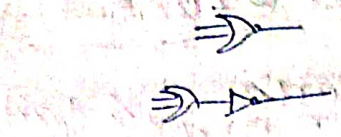


$\overline{\left(\overline{A \cdot \overline{AB}}\right)' + \left(\overline{B \cdot \overline{AB}}\right)}$

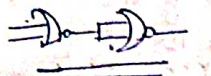
$A \cdot (A' + B') + B(A' + B')$

$AB' + A'B$

XOR

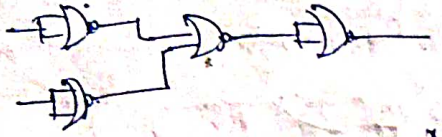


$\overline{x+x} = \overline{x}$ (NOT)

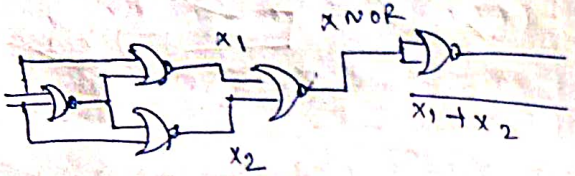


$\overline{\overline{A+B}} = A+B$ (OR)

$\overline{A'+B'} = \overline{A'' \cdot B''} = \overline{A \cdot B}$ (AND)



NAND



$x_1 = \overline{A + \overline{A+B}}$

$x_2 = \overline{B + \overline{A+B}}$

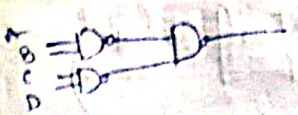
$\overline{\left(\overline{A' + \overline{A+B}}\right) + \left(\overline{B + \overline{A+B}}\right)}$ } $\overline{x_1 + x_2}$

$= \overline{A'(A+B) + B(A+B)}$

$= \overline{(A'B + AB')}$ = XNOR

e.g. $(AB + CD)$ using NAND

$\Rightarrow \overline{\overline{AB + CD}} = \overline{\overline{AB} \cdot \overline{CD}}$



e.g. $\overline{(A+B)(C+D)}$ using NOR

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



How to complement any fn:- (Apply Demorg

e.g. $\overline{\overline{A}\,\overline{B} + AB}$

$= (A+B)(\overline{A} + \overline{B})$

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

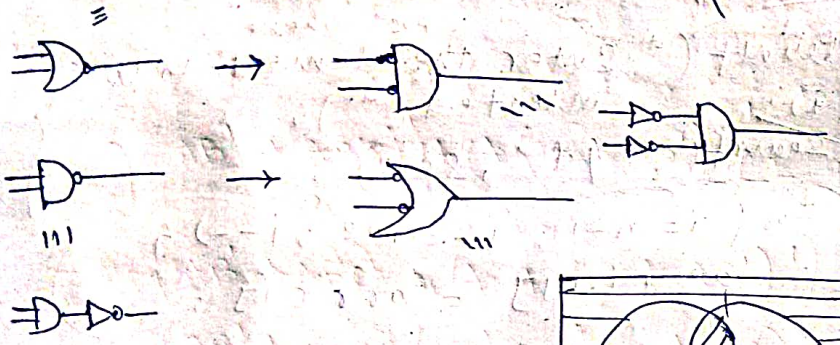
Q) $xy + x'z = (x+z)(x'+y)$

$= x\cancel{x'} + xy + x'z + yz(x+x')$

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

$= xy + x'z$

(proved)

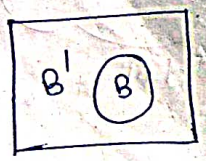


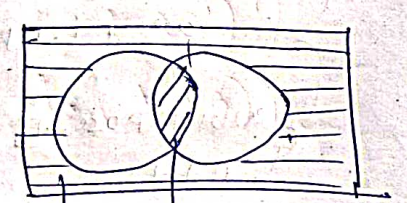
⇒ Venn diagram :-



$A+B$

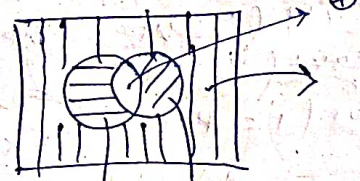
$\overline{A+B}$





$\overline{AB}$ (NAND)

AND (AB)



XNOR

$AB' + A'B$

XOR

Q) minimize

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

$= A + AB' + ABC + 0 + A'B'$

$= A(1+B' + BC) + A'B'$

$= A + A'B'$

$= (A+A')(A+B')$ (distributive law)

$= A + B'$

Q) simplify $ABC + AB'C + ABC'$

$= AB(C+C') + AB'C$

$= AB + AB'C$

$= AB + AC$

$= A(B+C)$

$ABC + AB'C + ABC' = 2ABC + AB'C + ABC'$

Q) minimize $A[B + C'(AB + AC')']$

$= A[B + C'(A'+B')(A'+C)]$

$= AB + AC'(A' + A'C + A'B' + B'C)$

$= AB + 0 + 0 + 0$

$= AB$

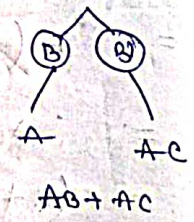
Q) Simplify $AB+(AC)'+AB'C(AB+C)$

$= AB+(AC)'+0+AB'C$

$= (AB+AC)+(AC)'$

$= AB+1$

$= 1$



*Theory: $A+A'B = (A+A')(A+B)$

$= A+B$

Q) Minimize

$\{(AB'+ABC)'+A(B+AB')\}'$

$= (AB'+ABC)''\cdot(A'+(B+AB')')$

$= (AB'+ABC)(A'+B'\cdot(A'+B))$

$= 0$

Q) Simplify $\left[\{(AB)'(BC)'\}'\{(BC)'(CD)'\}'\right]'$

$= (AB)'(BC)'+(BC)'+(CD)'$

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

distributive th.: ~~$A'B'+A'C'+B'C$~~

$= B'+A'C'+C'+B'D'$

$= B'(1+D')+C'(1+A')$

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

$\cdot A'B'+A'C'+B'+B'C'$
$= B'(1+A'+\dots)$
$\cdot \dots + A'C'$
$= B'+A'C'$

Q) minimize

$(A+B)(B+C')$ → distributive

$= B+AC$

Numerical methods :- Assignments :-

$$f(x,y) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}xy_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}\times y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}\times y_3$$

x	y
10	31
21	2
9	−4

$x = 11.5$

$\Rightarrow \frac{(11.5-21)(11.5-9)}{(10-21)(10-9)}\times 31$

$= 66.93$

$\Rightarrow \frac{(11.5-10)(11.5-9)}{(21-10)(21-9)}\times 2$

$= \frac{(1.5\times 2.5)\times 2}{1\times 12} = 0.68$

$\Rightarrow \frac{(11.5-10)(11.5-21)}{(9-10)(9-21)}\times(-4) = \frac{1.5\times 9.5\times 4}{12} = \frac{1.5\times 9.5\times 1}{3}$

$= 4.75$

x2.3

Under common bar, under ' - ve sign then sign system.

	A B C	Minterm (AND)	Maxterm = Minterm (OR)
0	0 0 0	$A'B'C'$ (m_0)	$A+B+C$ (M_0)
1	0 0 1	$A'B'C$ (m_1)	$A+B+C'$ (M_1)
2	0 1 0	$A'BC'$ (m_2)	$A+B'+C$ (M_2)
3	0 1 1	$A'BC$ (m_3)	$A+B'+C'$ (M_3)
4	1 0 0	$AB'C'$ (m_4)	$A'+B+C$ (M_4)
5	1 0 1	$AB'C$ (m_5)	$A'+B+C'$ (M_5)
6	1 1 0	ABC' (m_6)	$A'+B'+C$ (M_6)
7	1 1 1	ABC (m_7)	$A'+B'+C'$ (M_7)
		Prime = 0	Prime = 1

0 → A' B' C'
1 → A B C

$(\overline{m_i} = M_i)$

$f = AB$

$\Rightarrow (A+B)(A+B')$

$(A'+B)$

$\Rightarrow A(A'+B)$

$= AB$

A	B	m_i	M_i	$f = AB$
0	0	$A'B'$	$A+B$	0
0	1	$A'B$	$A+B'$	0
1	0	AB'	$A'+B$	0
1	1	AB	$A'+B'$	1 ✓

f (func of 3 var)

m_0 0 0 m_3 0 m_5 m_6 0

↓ ↓ ↓ ↓

1 1 1 1

$f \Rightarrow \sum_M (0, 3, 5, 6)$

$= \prod_M (1, 2, 4, 7)$

{ Minterm fn / Maxterm fn.

$\Rightarrow f = A'B'C' + A'BC + AB'C + ABC' = \sum_M (0,3,5,6)$ → SUM of min-terms (consider 1)

$\prod_M (1,2,4,7) = M_1 M_2 M_4 M_7$

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

Product of Maxterm (consider 0)

e.g. $F = (A+B)$: sum of product

(A,B)

$= A(B+B') + B(A+A')$ $\quad X + X = X$

$= AB' + A'B + AB$ $\quad 1 + X = X$

$= \sum_m (1,2,3)$

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

e.g. SOP

$F(A,B,C,D) = AB + ACD$

$= AB(C+C')(D+D') + A(B+B')CD$

$= ABCD + ABCD' + ABC'D + ABC'D' + ABCD + AB'CD$

$= \sum_m (15, 14, 13, 12, 11)$ ↳ sum of min term

e.g. POS

$F(A,B,C) = A + B'C$ ↓ distribution th. get product of sum ?

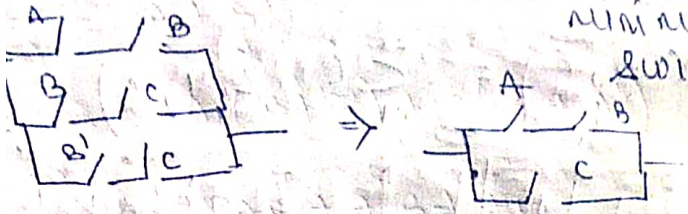
$\Rightarrow (A+B')(A+C)$ POS

product of maxterm ↳ $(A + B' + CC')(A + BB' + C)$

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

$= M_2 \times M_3 \times M_0$

$= \prod_M (0,2,3)$



Minimize the switching function

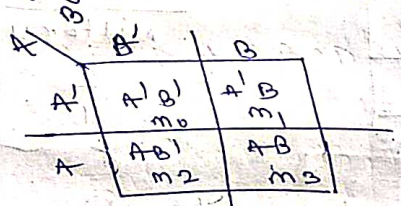
$+B + BC + B'C = AB + C$

Boolean Algebra is one technique to minimize ckt

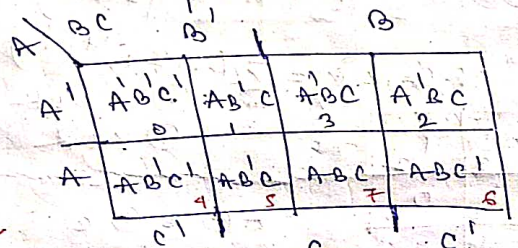
other is K-Map

Karnaugh.

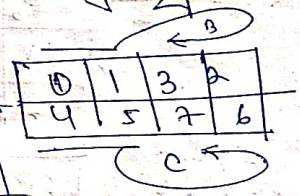
Karnaugh map (K-Map)



2 var K map



Midterm in gray code

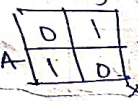




A·B (AND)



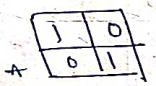
A+B (OR)



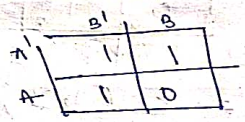
XOR



NOR

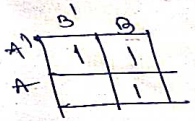


XNOR



NAND

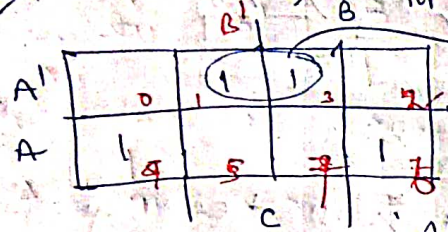
$B' + A' = \overline{AB}$



$= A' + B$

e.g. Prev. yr.

$F(A,B,C) = \sum_m (1,3,4,6)$



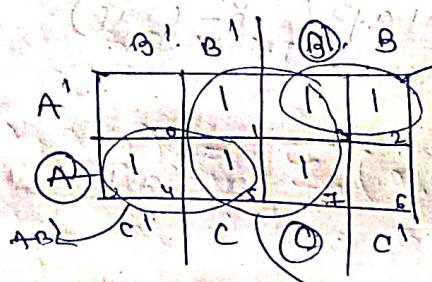
$A'C(B+B')$

$AC'(B+B')$

$A'C + AC' \rightarrow$ XOR

$= A \oplus C$

e.g.



$A'B$

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

$F = A'B + AB' + C$

$= A \oplus B + C$

e.g.

$F(A,B,C) = \sum_m (1,2,4,7)$ ⊙ XNOR ⊕ XOR

0_0	1_1	0_3	1_2
1_4	0_5	1_7	0_6

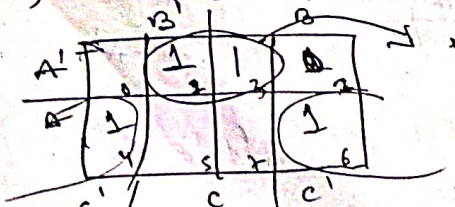
$A \oplus B \oplus C$

1_0	0_1	1_3	0_2
0_4	1_5	0_7	1_6

$\Sigma(0,3,5,6)$

$A \odot B \odot C$

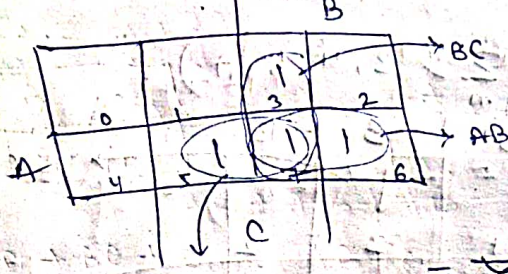
$(A,B,C) = \Sigma(1,3,4,6)$



$A'C + AC' \equiv A \oplus C$

---

*



$F = AB + BC + AC$

draw the K-map

$= \sum_m (3,5,6,7)$

$F = AB + BC + AC$

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

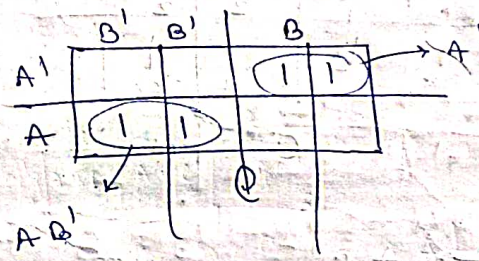
$= ABC + ABC' + ABC + A'BC + AB'C + AB'C$

$= m_7 + m_6 + m_3 + m_5$

*

$F(A,B,C) = \sum_m (2,3,4,5)$

$=$



$= A'B + AB' = A \oplus B$

XOR

2ⁿ

Midterm

2

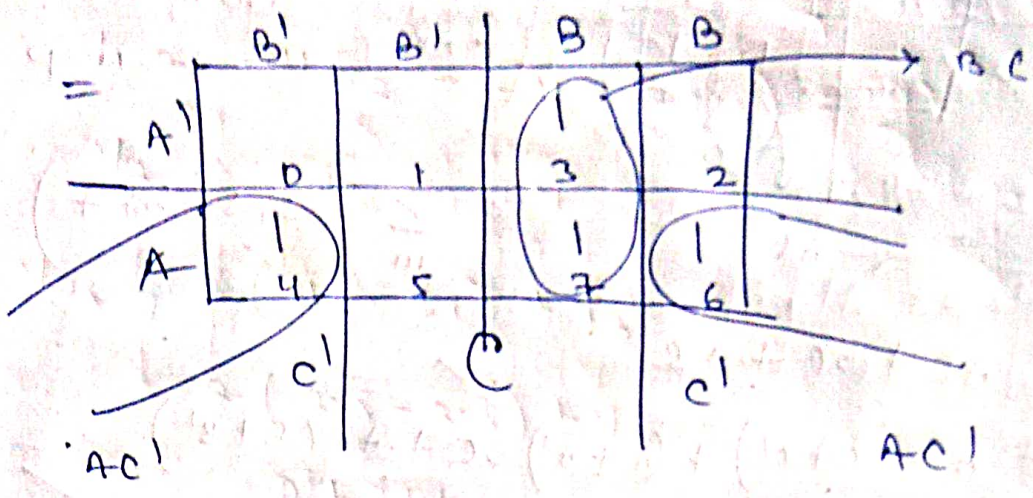
4

8

2)

* $F(A,B,C) = \sum_m (3,4,6,7)$

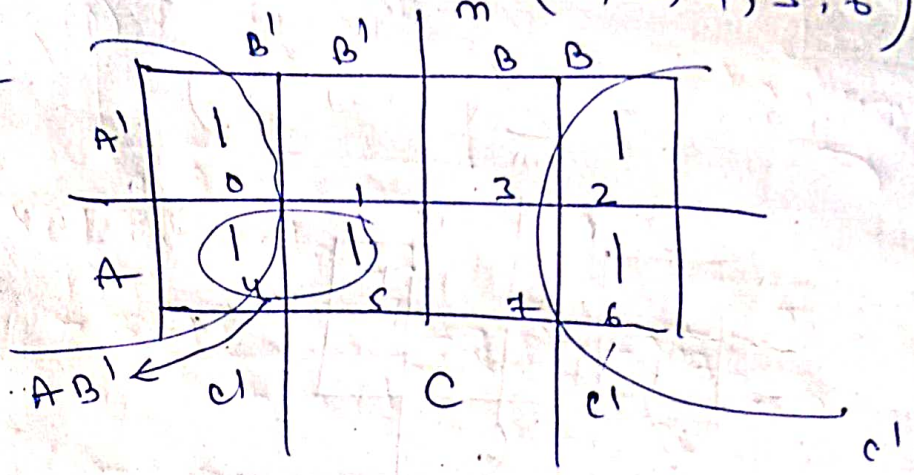
=



$= BC + AC'$

* $F(A,B,C) = \sum_m (0,2,4,5,6)$

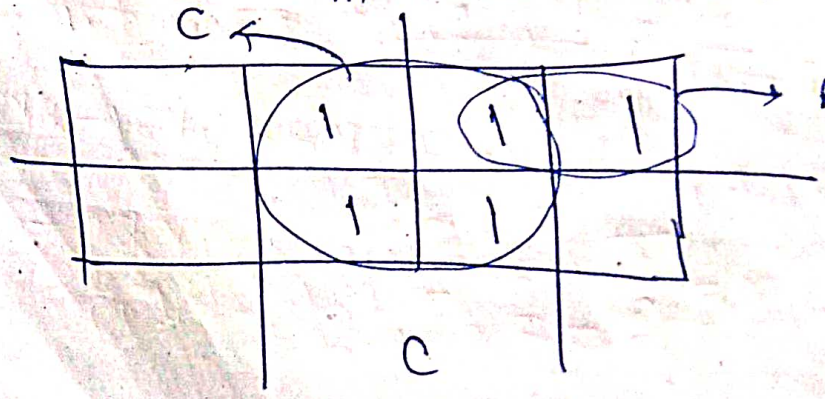
=



$= AB' + C'$

* $F(A,B,C) = \sum_m (1,2,3,5,7)$

=



$= A'B + C$

Digital logic - 07 (13 | 08 | 2024)

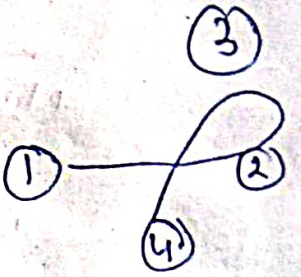
most significant bit → MSB

LSB

* 4 Variable KMAP :- $f(A, B, C, D)$

$2^3\ 2^2\ 2^1\ 2^0$
1 0 0 0

$\bar{A}\bar{B}$ \ $\bar{C}\bar{D}$	$\bar{C}$	C	C	$\bar{C}$
$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$ m0	$\bar{A}\bar{B}\bar{C}D$ m1	$\bar{A}\bar{B}CD$ m3	$\bar{A}\bar{B}C\bar{D}$ m2
$\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
AB	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
$A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$
	$\bar{D}$	D	D	$\bar{D}$



Binary code

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

$\bar{A}\bar{B}$ \ $\bar{C}\bar{D}$	$\bar{C}$	C	C	$\bar{C}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10
	$\bar{D}$	D	D	$\bar{D}$

column jump ←

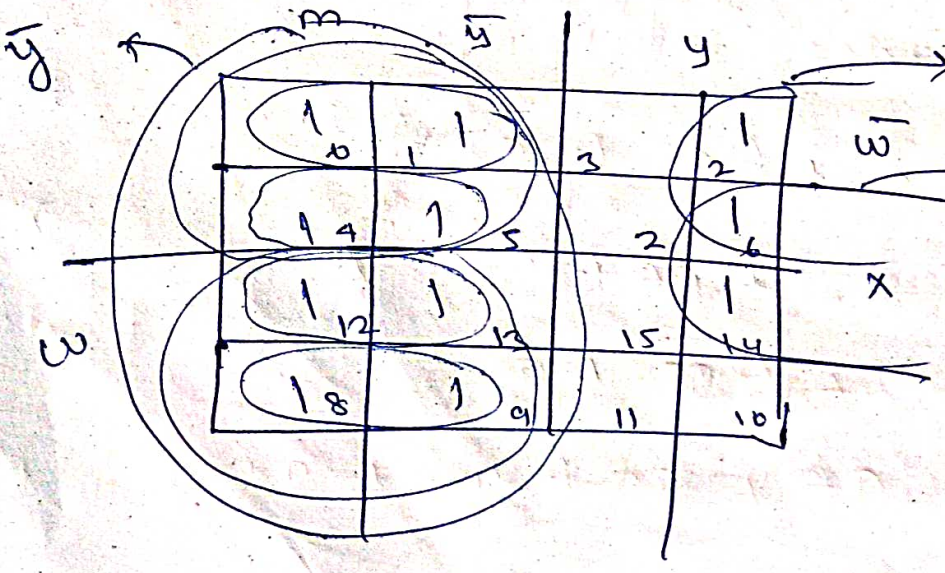
ROW JUMP

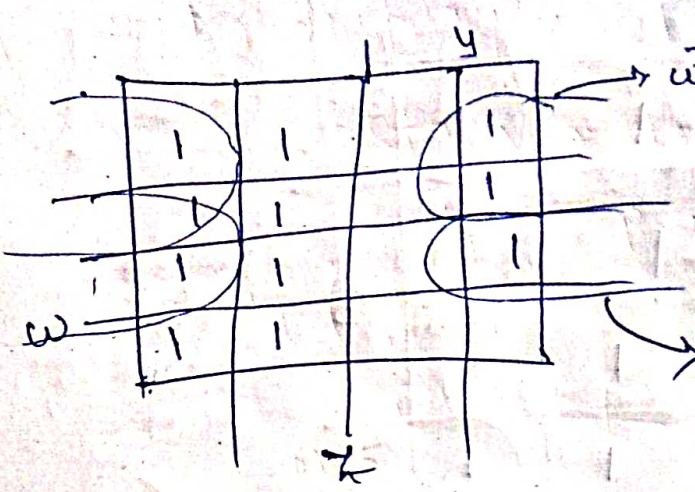
MSB ↑ → LSB

e.g. $f(w, x, y, z)$

$\log_2 2^n = n$

$= \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

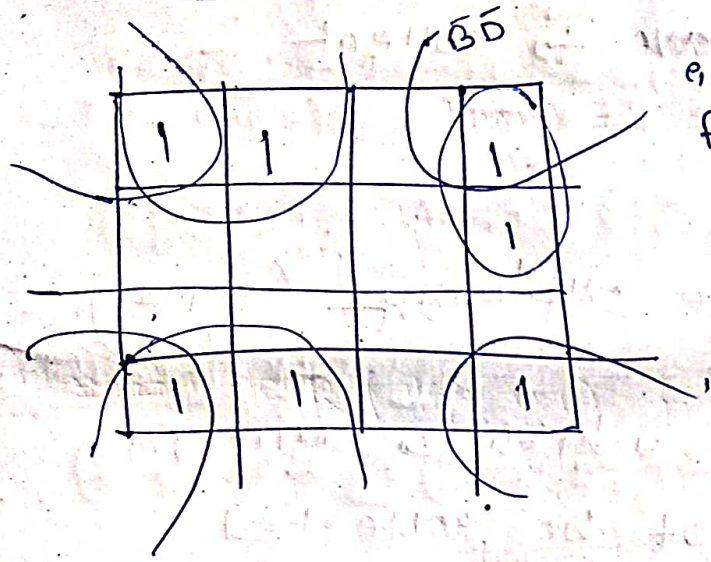




$\rightarrow \bar{w}\bar{z}(x+\bar{x})(y+\bar{y})$

$\rightarrow x\bar{z}(y+\bar{y})(w+\bar{w})$

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

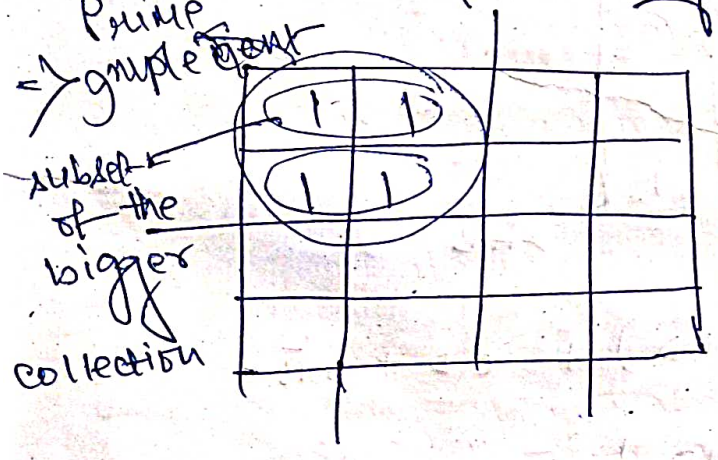


e.g.

$f = A'B'C' + B'CD' + A'BCD' + AB'C'(D+\bar{D})$

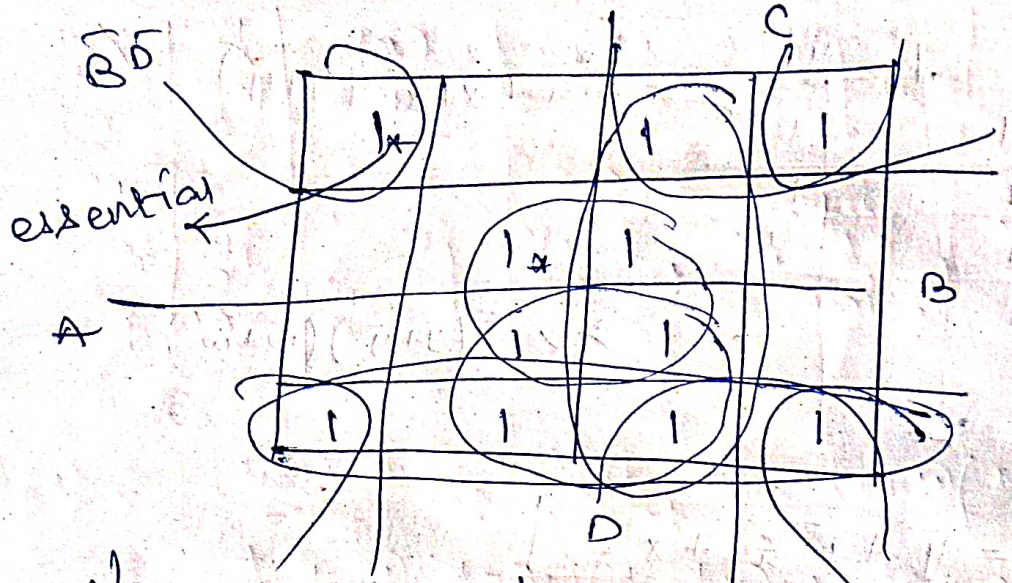
e.g. $f = A'B'C' + B'CD' + A'BCD' + AB'C'$

Minimize using KMAP :-



H.W

e.g. $f(A,B,C,D) = \sum_m (0,2,3,5,7,8,9,10,11,13,15)$

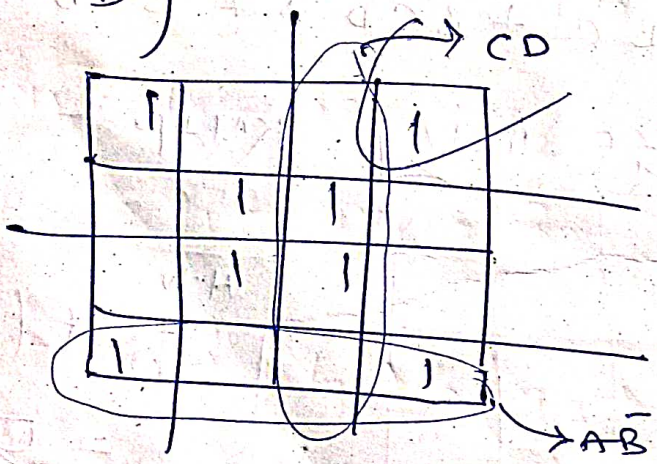


- at least once you have to collect -
- redundant.
- Essential Prime Implicant (EPI)

↓

without which a minterm will not be collected

EPI $(\bar{B}\bar{D}, BD)$



$F = \cancel{EPI}\ (\bar{B}\bar{D} + BD) + CD + A\bar{B}$

or $= \bar{B}\bar{D} + BD + AD + \bar{B}C$

or $= \bar{B}\bar{D} + BD + \bar{B}C + A\bar{B}$

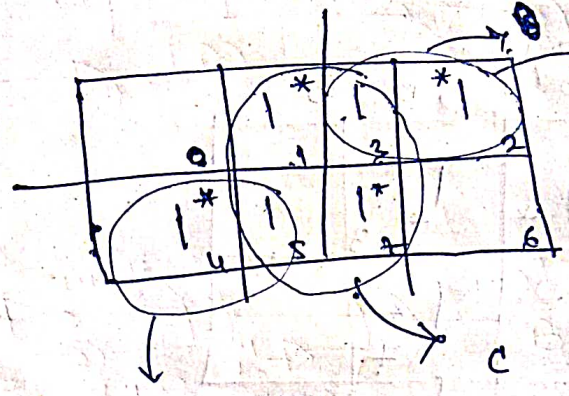
or $= \bar{B}\bar{D} + BD + CD + AD$

Digital logic-08

20/08/2024

Q1) Simplify / minimize $f = A'B'C + AB'C' + 2ABC + AC + A'BC'$ using kmap.

⇒



4	2	1
A'	B	C
0	1	1
–	–	–

m_3

(EPI) $A\bar{B}$

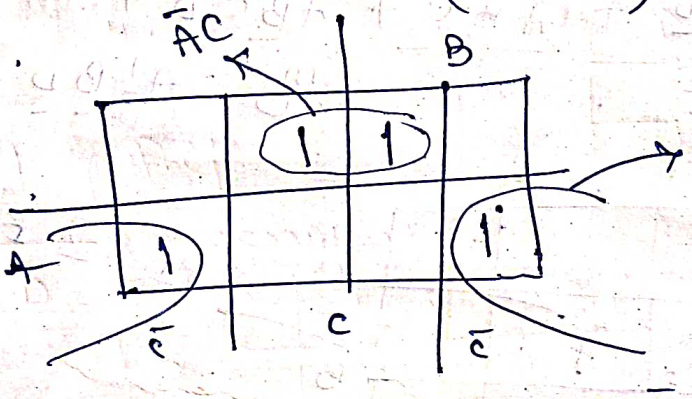
↓

Biggest possible group.

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

$= (A \oplus B) + C$

Q2) Minimize $F(A, B, C) = \sum_m (1, 3, 4, 6)$



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

Actually $F(1, 3, 4, 6)$

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

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

$= \bar{A}C$

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

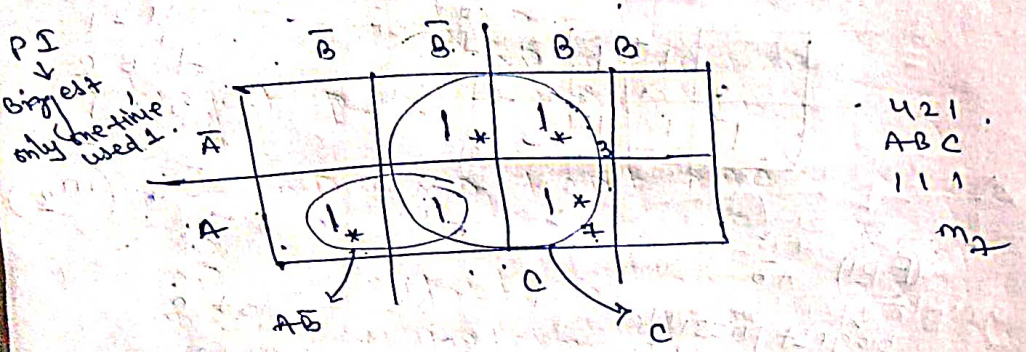
$= A\bar{C}(\bar{B} + B)$

$= A\bar{C}$

Q3) Minimize

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

$A\bar{B}$ → product SUM; $\bar{A}BC$ → m_3 → minterm; ABC → m_7

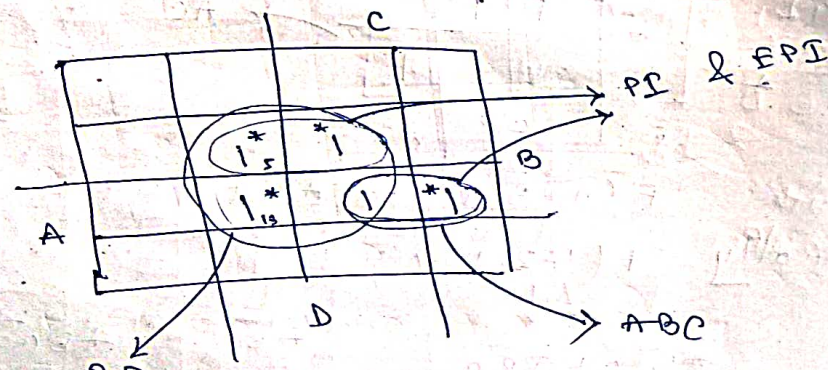


4 2 1
A B C
1 1 1
m_7

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

Q4) $f = A'\underline{B}C'\underline{D} + \underline{ABC} + \underline{AB}C'\underline{D} +$

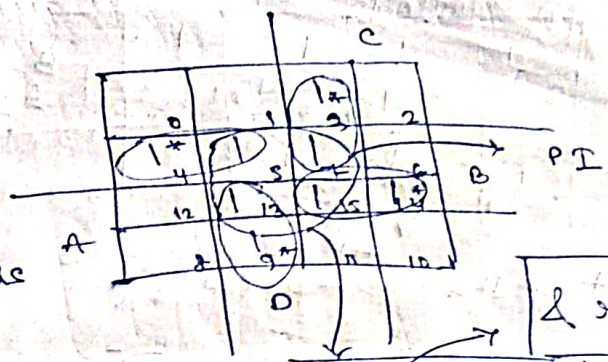
m_5 m_3 $A'\underline{BD}$



$f = \underline{BD} + \underline{ABC}$

Try to solve using boolean Algebra.?

Q5) $\sum(A,B,C,D) = \sum_m(3,4,5,7,9,13,14,15)$



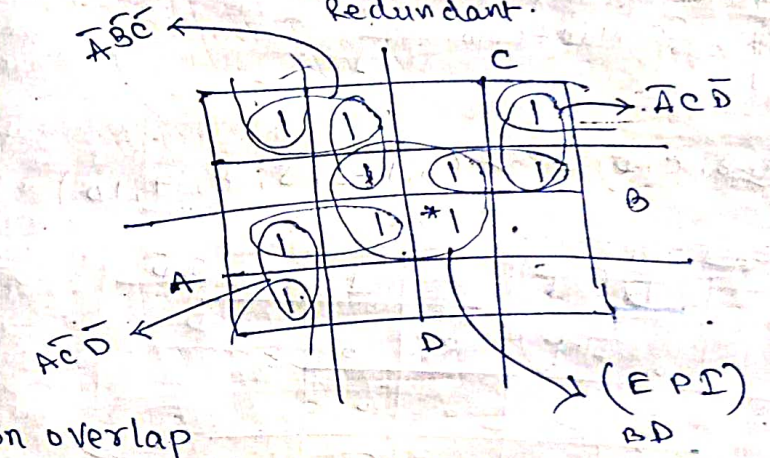
fn will be found by this EPI

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

redundant.

PI but not essential & redundant others EPI

Q6)



non overlap biggest possible loop (PI)

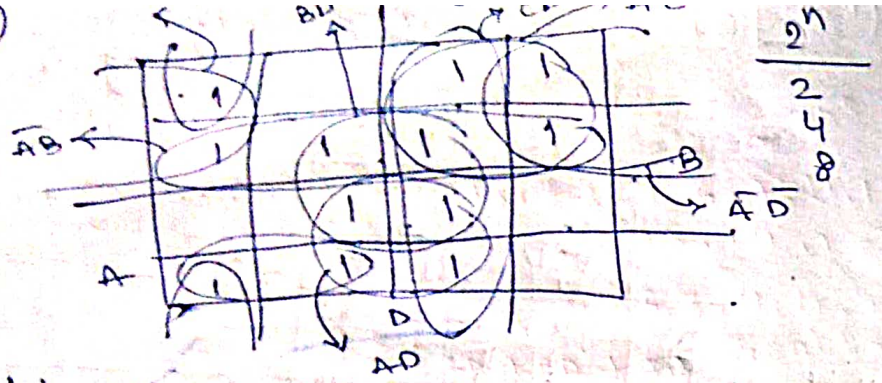
$f = BD + \bar{A}B\bar{C} + \bar{A}C\bar{D} + A\bar{C}\bar{D}$

(EPI) non overlapping PI

if you allow overlap + non overlapping → Minimal

otherwise Non minimal.

Q7)



2^n

2
4
8

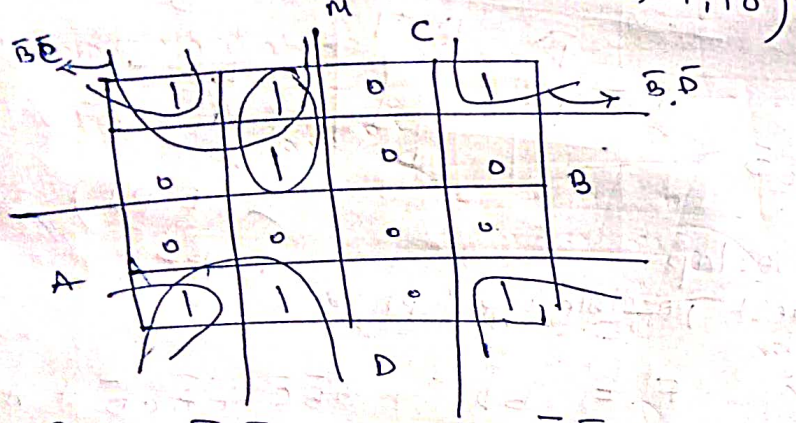
4 biggest PI.

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

* SOP | Po Maxterm. } DOP
POS | SO Minterm }

Q8) $F(A, B, C, D) = \sum_M (0, 1, 2, 5, 8, 9, 10)$

POS



SOP : $F = \bar{B}\bar{D} + \bar{B}\bar{C} + A\bar{C}D$

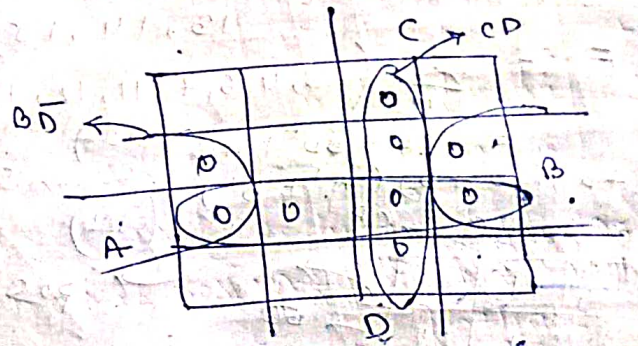
POS : $\bar{F} = (B+D)(B+C)(A+C+\bar{D})$

POS of $F =$

SOP : $\bar{F} = CD + AB + \bar{B}\bar{D}$

(group the zero)

POS : $F = \bar{F} = (\bar{C}+\bar{D})(\bar{A}+\bar{B})(\bar{B}+D)$



Digital logic - 09 (22/08/2024)

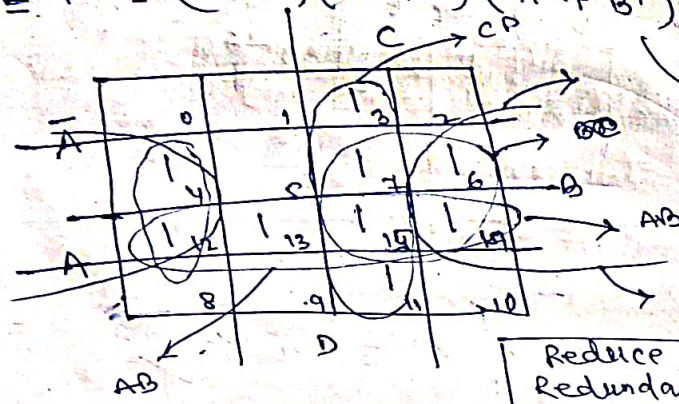
Q1) simplify boolean f/n

$F(A,B,C,D) = \prod M (3,4,6,7,11,12,13,14,15)$ → POS

$\bar{F} = \sum_m (3,4,6,7,11,12,13,14,15)$

$= CD + BD' + AB$ → SOP

$\bar{F} = F = (C'+D')(B'+D)(A'+B')$ (AND) → POS



1) EPI
2) Reduce overlap.

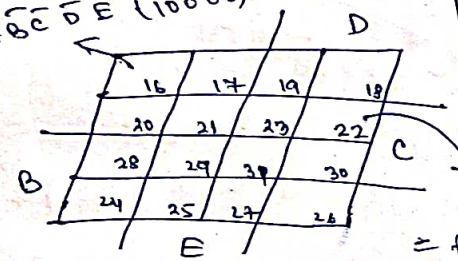
Reduce Redundancy	Mandatory to include all minterm.

→ explore all grps or prime implement

* Five variable K-map :-

$M_{16} = A\bar{B}\bar{C}\bar{D}\bar{E}$ (10000)

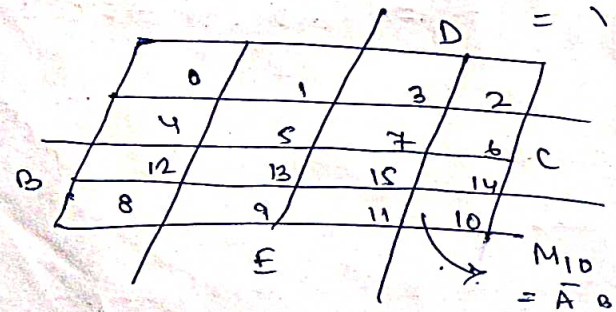
A = 1.



$M_{22} = A\bar{B}CD\bar{E}$

$= 10110$

A = 0



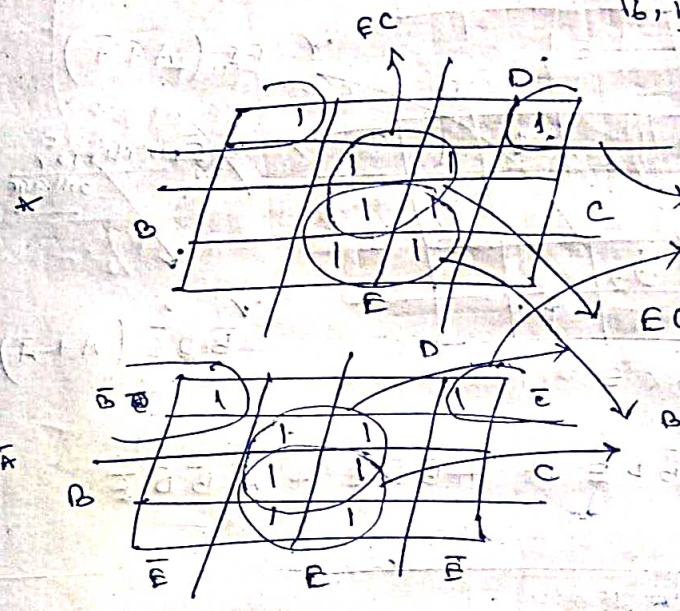
$M_{10} = \bar{A}B\bar{C}D\bar{E}$

$= 01010$

Mirror image → if you want to find mirror image use this.

P (A + Ā)

e.g. $F(A,B,C,D,E) = \sum_m (0,2,5,7,9,11,13,15,16,18,21,23,25,27,29,31)$



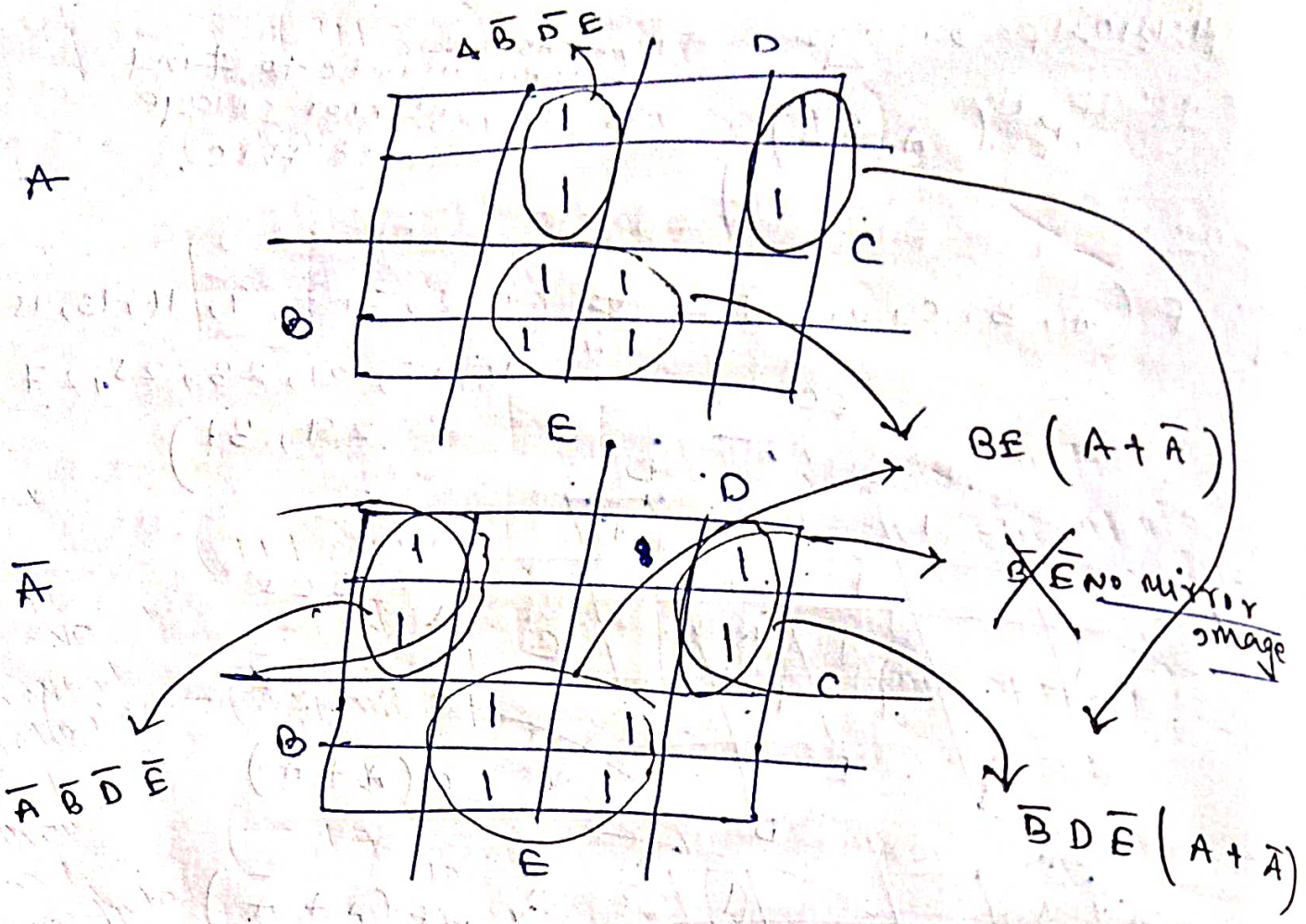
$F = \bar{B}\bar{C}\bar{E} + EC + BE$

e.g. $F(A,B,C,D,E) = \sum_m (0,2,4,6,9,11,13,15,17,18,21,22,25,27,29,31)$

* subset / subgrp - can't be prime implicant

Precedence of operator ( ), NOT, AND, OR

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



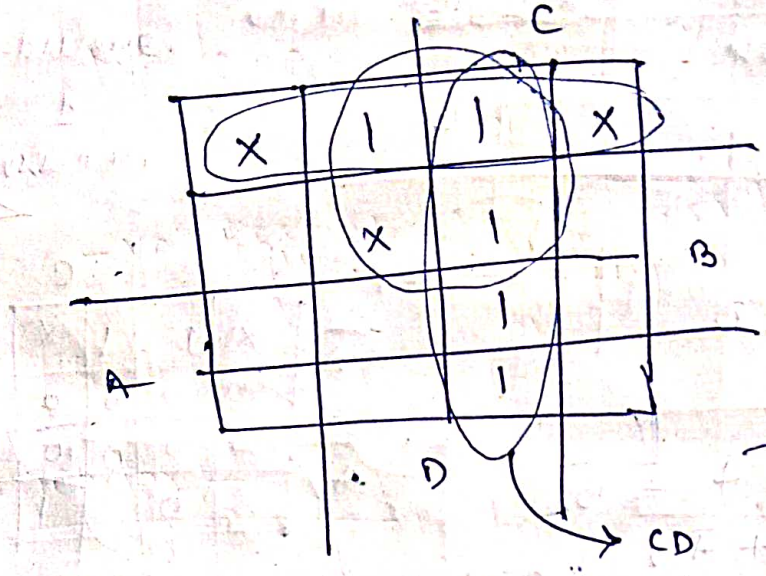
$$f = BE + \bar{B}D\bar{E} + A\bar{B}\bar{D}E + \bar{A}\bar{B}\bar{D}\bar{E}$$

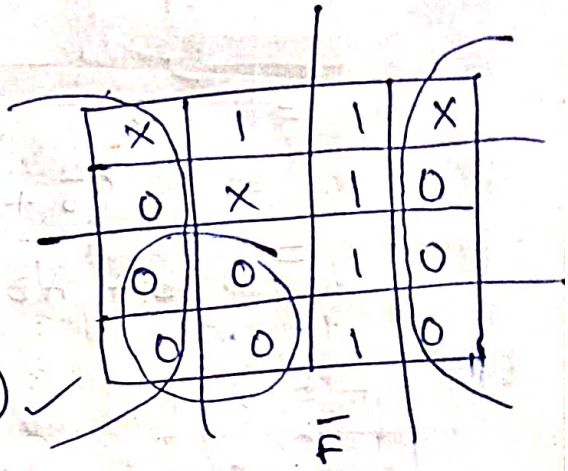
$$BE + AB'D\bar{E} + A\bar{D}E + A'B'\bar{E}$$

Digital logic-10

24/08/2024

Don't care condition:- $(X) = \sum_m (1,3,7,11,15) + \sum_d (0,2,5)$





$F_1 = CD + \bar{A}\bar{B}$

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

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

$F = ?\ F_1$ or $F_2 = F$

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

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

$= 0 + 0 + \bar{A}B\bar{D} + 0$

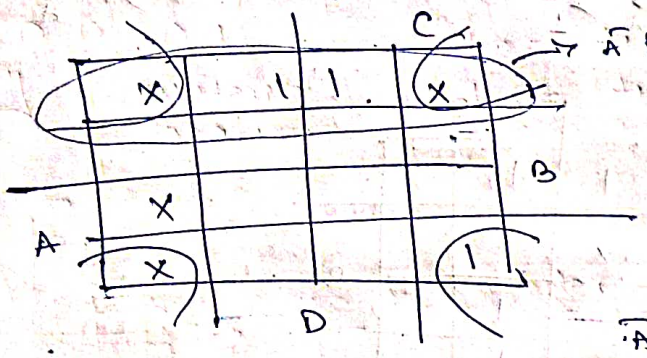
$= \bar{A}B\bar{D}$

Eg. $f = \sum_m (1,3,10) + \sum_d (0,2,8,12)$

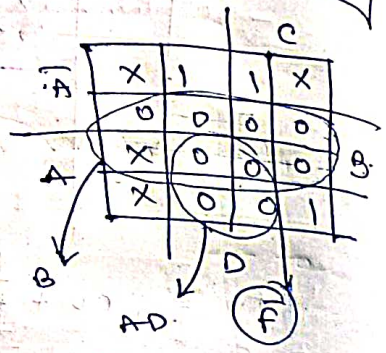
(x) → 0, 1

find $\bar{f}$?

⇒



including 1 necessary



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

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

$\bar{f} = B + AD$

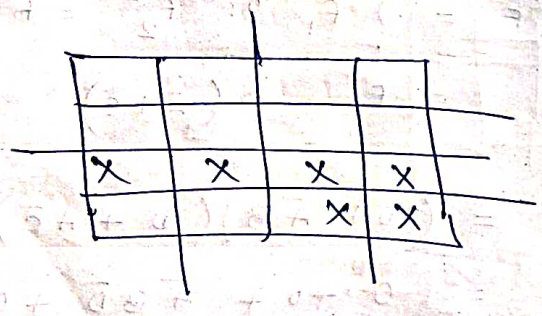
$f\bar{f} = 0$

* BCD from 0000 → 1001

6 symbols are unused :-

(10 → 15)

(A) → (F)

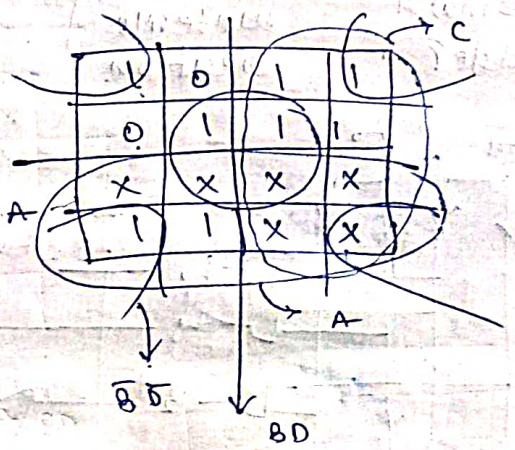


BCD to 7 segment



	a	b	c	d	e	f	g
0 →	1	1	1	1	1	1	0
1 →	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

solve a :-



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

b, → g (solve using Internet) / K-map.

Digital logic - 11 (03/09/2024)

* Quine Mc clusky method (QM)

$$f(w,x,y,z) = \sum_m (0,5,7,8,9,10,11,14,15)$$

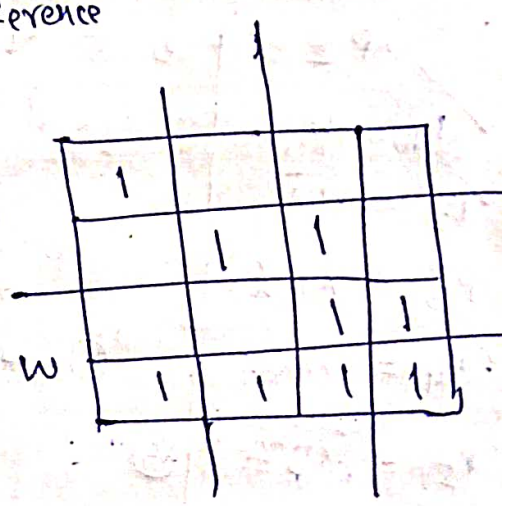
	Decimal no.	Binary
Index 0 →	0	0000
Index 1	8.	1000
Index 2	5	0101
	9	1001
	10	1010
Index 3	7	0111
	11	1011
	14	1110
Index 4	15	1111

14
8 4 2 1
1 1 1 0

we have probability of grouping adjacent variables

Index No. is = no. of 1 in 4 bits

Decimal group	1st reduction	
(0, 8)	- 000	→ P_5
(8, 9)	100 - ✓	1 bit difference
(8, 10)	10 - 0 ✓	
(5, 7)	01 - 1	→ P_4
(9, 11)	10 - 1 ✓	
(10, 11)	101 - ✓	
(10, 14)	1 - 10 ✓	
(7, 15)	- 111	→ P_3
(11, 15)	1 - 11 ✓	
(14, 15)	111 - ✓	



2nd deinal group — 2nd reduction

(8,9, 10, 11) — 10 — → $w\bar{x}$ → P_2

~~8,9,10,11~~

(10, 11, 14, 15) — 11 — → wy → P_1

~~10,11,14,15~~

you can't make grouping in 2nd reduction, so there will be no more reduction

⇒ then you have to check bigger grouping

then check all the numbers are included or not

w 0 → 15

⇒ $f = P_1 + P_2 + \cancel{P_3} + P_4 + P_5$ → Redundant

} PI

⇒ we have to find EPI.

PI	0	5	7	8	9	10	11	14	15
$P_1 = wy$				*	*		⊛		*
$P_2 = w\bar{x}$				*	⊛	*	*		
$P_3 = xyz$		*						*	Redundant
$P_4 = \bar{w}xz$	⊛		*						
$P_5 = \bar{x}\bar{y}\bar{z}$	⊛								

$f = P_1 + P_2 + P_4 + P_5$

1st find out EPI

that is present once

that will be EPI

that 'P' will be essential.

otherwise Redundant

e.g.

		0 ✓	1	2 ✓	8	9	15 ✓	17 ✓	21 ✓	24	25	27 ✓	31 ✓
A	P_1				x	x				x	x		
B	P_2		x			x		x			x		
C	P_3	x	x		x	x							
D	P_4											x	x
*E	P_5						⊛					x	⊛ x
F	P_6									x			
G	P_7								x	x			
*H	P_8							x	⊛				
I	P_9		x					x					
*J	P_{10}	x		⊛									
K	P_{11}	x	x										

incomplete!

$P_1 + P_2 + P_5 + P_8 + P_{10} = f_1$

$P_1 + P_3 + P_5 + P_8 + P_{10} = f_2$

max range of subtraction

1000 → −8

0111 → +7

e.g. 1010 → A

0100 → B

0111

1010 → A

1011 → $\bar{B}$

*0110

421

5−7

0101

1000

01110 = −2

e.g. x = 1010100

y = 1000011

1111 1

1010100

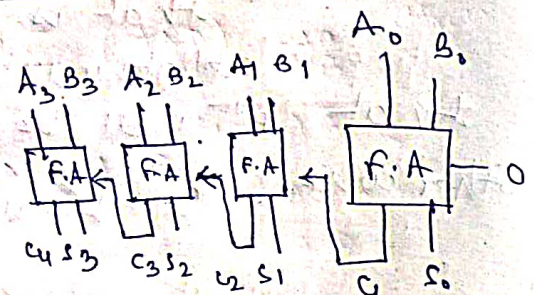
0111100

10001001 (Ans)

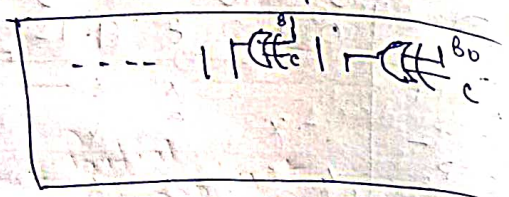
discard

* Binary Addition :-

Half adder
Full adder
4 bit parallel adder



$C_3\ C_2\ C_1\ 0$
$A_3\ A_2\ A_1\ A_0$
$B_3\ B_2\ B_1\ B_0$
$C_4\ S_3\ S_2\ S_1\ S_0$



e.g.
0000
1010
1001
10011

e.g.
1100
0111 → 7
1110 → 14
10101 → 21

* Adder | Subtractor :-

↓
2's complement Method

Addition
↓
A + B + 0

$A_3A_2A_1A_0$
$-\ B_3\ B_2\ B_1\ B_0$

$A - B \Rightarrow A + 2's(B)$
↓
$1's(B) + 1$
↓
$C_3\ C_2\ C_1\ 1$
$A_3\ A_2\ A_1\ A_0$
$\bar{B}_3\ \bar{B}_2\ \bar{B}_1\ \bar{B}_0$
Bi 0 | $S_3\ S_2\ S_1\ S_0$

Bi = 1 → Subtraction
Bi = 0 → Addition

subtraction → $A + \bar{B} + 1$

e.g.
1001
− 0001
1001
1111
1000

		XOR	
0	0	0	B
0	1	1	
1	0	1	$\bar{B}$
1	1	0	

B $\bar{B}$

$-5 - 7 = -9$

4 2 1
0000 0101 → +5
0000 0111 → +7

1's
+1
2's

1111 1010 (−5)
11111000
111110100 → −12

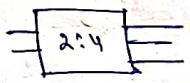
$F = \sum_m$ (list) | ~~$F = \prod_m$ (list)~~

$\bar{F}$ → F = SOP [grp 1's] ✓
F = POS
$\bar{F}$ = SOP
$\bar{F}$ = POS
$\bar{F} = F$ [grp 0's] ✓

$F = \prod_M$ (list)
$F = \sum_m$ (ex list)
then similar technique as previous.

Exam

Design 2:4 decoder:



Truth table ?
Boolean eqn ?
circuit ?
→ Kmap.

Full adder,
Full substractor
half + half adder.

Digital Logic ...

↳ Internal Hackathon

* 1 bit Multiplier :- (AND)

A	B	M
0	0	0
0	1	0
1	0	0
1	1	1

$M = A \cdot B$

* 2 bit Multiplier :-

A_1 A_0
B_1 B_0

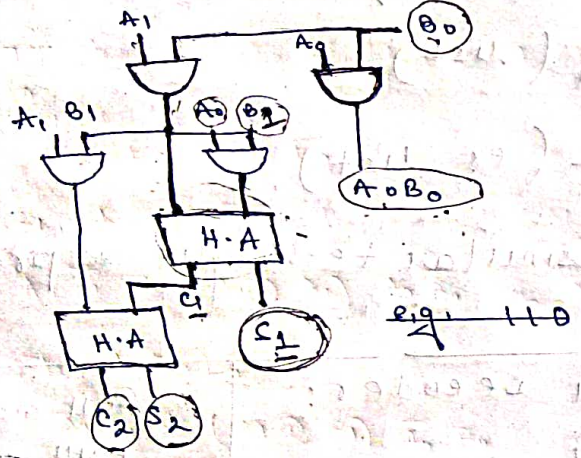
A_1B_0 A_0B_0 ← PP_1 (Partial product)

A_1B_1 A_0B_1 ← PP_2

C_1 C_2 S_1 A_0B_0

C_2 C_2 S_1 A_0B_0 — Adder

⇒



e.g. 11 × 11

1011

1001

21
11
= 3
× 3
9

84-21
1001
= 9.

* 3 bit Multiplier

A_2 A_1 A_0
B_2 B_1 B_0

A_2B_0 A_1B_0 A_0B_0 ← PP_1

A_2B_1 A_1B_1 A_0B_1 ← PP_2

PS_{14} PS_{13} PS_{12} PS_{11} PS_{10} ← PS_1

A_2B_2 A_1B_2 A_0B_2 — — ← PP_3

PS_{25} PS_{24} PS_{23} PS_{22} PS_{21} PS_{20} ← PS_2

$PS_2 = PS_1 + PP_2$

$PS_1 = PP_1 + PS_0$

$PS_0 = 0$

$PS_3 = PS_2 + PP_3$

e.g.

$$\begin{array}{r} 110 \\ 111 \\ \hline 110 \\ 110 \\ \hline 10010 \\ 110 \\ \hline 101010 \end{array}$$

42

110 → 6
111 → 7
42

* 4 bit Multiplier :-

PP_1
PP_2
A_1
$+ PP_3$
A_2
PP_4
A_3

Accumulator

* 1 bit comparator :-

A	B	>	=	<
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$L = \bar{A}B$

$G = A\bar{B}$

$E = A \odot B = \bar{A}\bar{B} + AB$ → XNOR

$= \overline{A \oplus B}$ → XOR

* 2 bit comparator :-

$A_1 A_0$ vs $B_1 B_0$

$E = (A_1 = B_1)$ and $(A_0 = B_0)$

$= E_1 \cdot E_0$

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

$\Rightarrow E = \prod_0^n E_i$

$= E_3 \cdot E_2 \cdot E_1 \cdot E_0$

$\Rightarrow G = G_1 + E_1 G_0$

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

$\Rightarrow L = L_1 + E_1 L_0$

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

$E_i = A_i \odot B_i$

$\Rightarrow E = E_2 E_1 E_0$

$G = G_2 + E_2 G_1 + E_2 E_1 G_0$

$L = L_2 + E_2 L_1 + E_2 E_1 L_0$

Bitwise: $E_i = A_i \odot B_i$, $G_i = A_i \bar{B}_i$, $L_i = \bar{A}_i \cdot B_i$

$G = G_3 + E_3 G_2 + E_3 E_2 G_1 + E_3 E_2 E_1 G_0$

$E = \prod_0^n E_i$

$= E_3 \cdot E_2 \cdot E_1 \cdot E_0$

$L = L_3 + E_3 L_2 + E_3 E_2 L_1 + E_3 E_2 E_1 L_1$

$G = G_1 + E_1 G_0$

$L = L_1 + E_1 L_0$

$E = E_1 \cdot E_0$

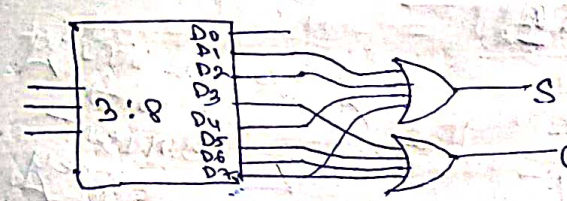
* How to implement circuit of 2 bit comparator
→ see the pic (Imp study material).

* Full adder using decoder & subtractor " "

$S = \sum_m (1,2,4,7)$

$C = \sum_m (3,5,6,7)$

C	S	B	D
0	0	0	0
0	1	1	1
0	1	1	1
1	0	1	0
0	1	0	1
1	0	0	0
1	1	1	1



Full subtractor

$\begin{array}{r} x \\ -\ y \\ -\ z \\ \hline B\ D \end{array}$

B → Borrow

D → difference

Full Adder

$\begin{array}{r} x \\ +\ y \\ +\ z \\ \hline C\ S \end{array}$

C → Carry

S → sum

* Arbitrary function :-

$$F_1 = \sum_m (0, 2, 5, 7)$$

$$F_2 = \sum_m (1, 4, 6, 7)$$

* **Binary to Gray**

$B_3 \oplus B_2 \oplus B_1 \oplus B_0$

e.g. $(0101)_2 \longrightarrow$ Decimal $\longrightarrow (5)$

MSB

$(0111)_{Gr} \longrightarrow (7)$

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

XOR: 0, 1, 1, 1

By taking X-OR between each

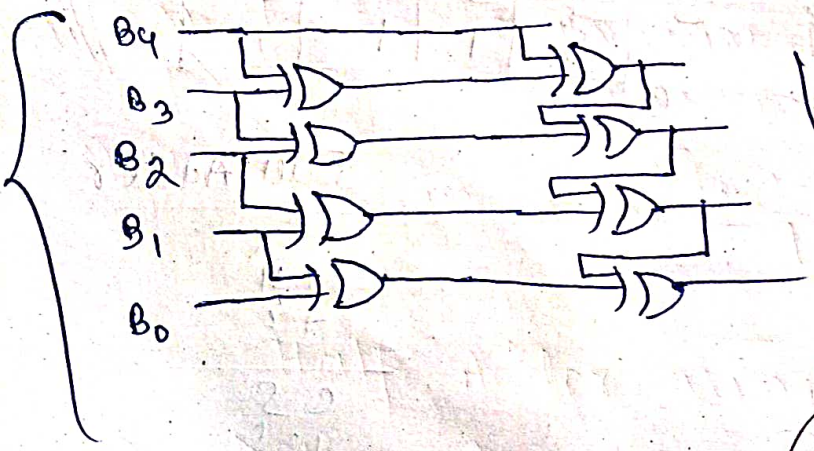
$0 \oplus 1$, $1 \oplus 0$, $0 \oplus 1$

Binary to grey

e.g. $(0111)_{Gr} \longrightarrow (0100)_2$ → grey → binary

MSB

e.g. 1010 → 1111



Previous binary bit with current grey

Similar like this

$(0110)_{Gr} \rightarrow 0100$

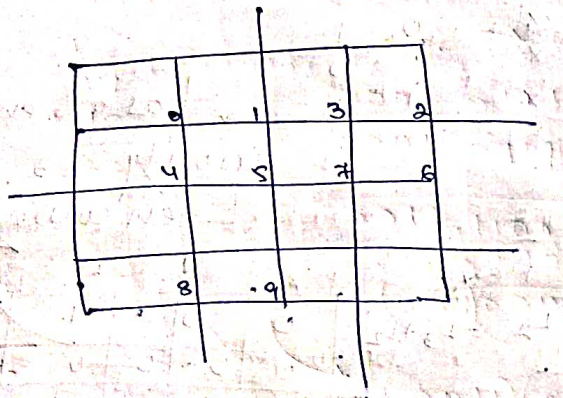
(Revedial class) Digital logic - 13 (17/9/2024)

Doubt Obtain product of sum:-

$$f(A,B,C,D) = \sum_m (3,4,5,7,9,13,14,15)$$

↓ solve using K-map

⇒



Q) $f = \prod_M (3,5,7,8,10,11,12,13)$

$= \sum_m (0,1,2,4,6,9,14,15)$

$F = \sum_m$ (list) = sum of product

F in

different: $\overline{F} = \overline{F \text{ in sum of pr}}$ = product of sum

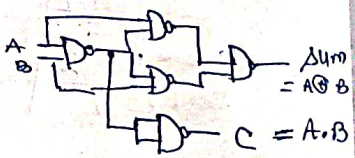
NOW, $\overline{F}$ in sum of product

---

* Half Adder using NAND Gate

→ Design XOR using NAND

→ Truth table, boolean algebra, CKT | what is half adder



1) Number system
2) Boolean algebra
3) KMap, QM method, don't come
4) combinational ckt

} 20 marks

Digital logic - 14 (17/9/2024)

BCD Addition :-

e.g.

~~1010~~ ~~0110~~ existing ✓ 0110 + 0010 = 0001 0010

10 : 1010 + 0110 = 10000 (0001)(0000)

16 : 1011 + 0110 = (0001)(0001)

1101 + 0110 = 10011 (0001)(0011)

1101 → 13, +0110 → 6

8 + 9 = 17 : 1000 + 1001 = (0001)(0111) + 0110 = (0001)(0111)

7 + 9 = 16 : 0111 + 1001

If more than 9 then add 6 ↓ 0110

0000
1111

e.g. 1101 + 0110 ✗

12 + 15 = 27

0 0001 0010 + 0001 0101 = 0010 0111

- BCD 2 bit addition -

e.g. 17 + 15 = 32

1 carry

0001 0111 + 0001 0101

0011 1100 + 0110 = (0001)(0010)

(0011)(0010)

17 + 19 = 36

1 111 0001 0111 + 0001 1001

(0011) 1(0000) + 0110 = 0110

0011 0110

* 92 + 58 = 150

1 1001 0010 + 0101 1000

1111 1010 + 0110 0110

(0001)(0101)(0000) Ans.

---

e.g. Previous year

1 293 + 786 = 1079

8421

① ① ①

0010 1001 0011 + 0111 1000 0110

1010 1(0001) 1001 + 0110 0110 ~~0110~~

Ans. → (0001)(0000)(0111)(1001) ✓

add 6 only when the no. is > 9

e.g. 1 1 472 + 429 = 901

8421

① ①

0100 0111 0010 + 0100 0010 1001

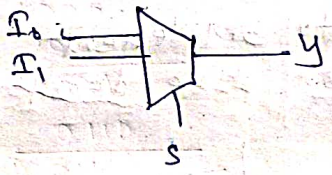
(1001) 1010 1011 + 0110 + 0110

1(0000) 1(0001) ✓ ✓

1001 0000 0001

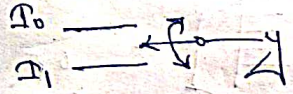
---

* 2:1 Multiplexor :- (~~se~~parallel to serial)

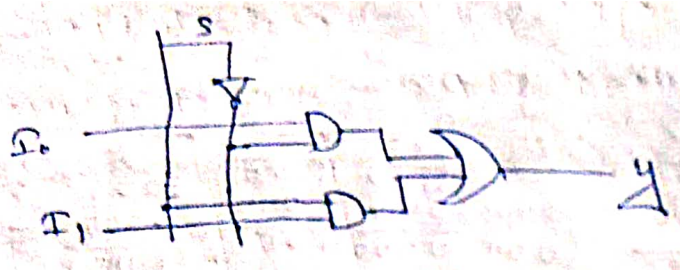


select line

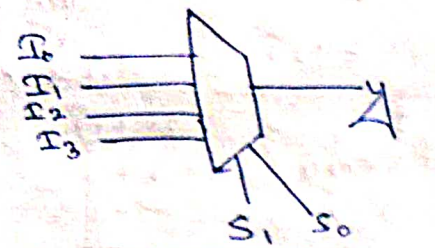
s	y
0	I0
1	I1



$y = \bar{s} I_0 + s I_1$

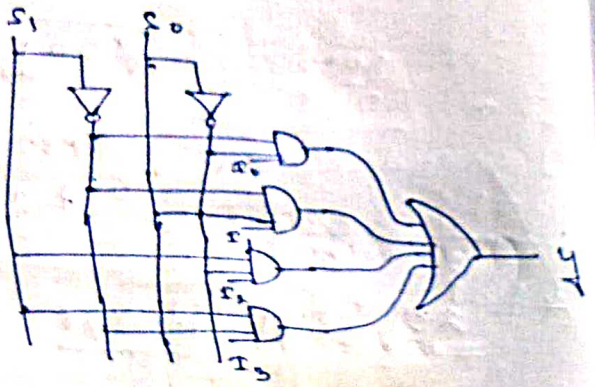


* 4:1 Multiplexor :-

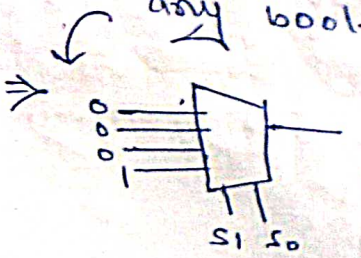


$$Y = I_0 \bar{S_1} \bar{S_0} + I_1 \bar{S_1} S_0 + I_2 S_1 \bar{S_0} + I_3 S_1 S_0$$

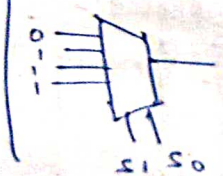
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



- Multiplexor can used to implement any boolean function like decoder -



AND gate

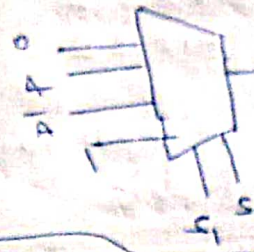


OR gate

* Implement a function using multiplexor

⇒ 01) Implement full adder & full substractor using multiplexors.

⇒



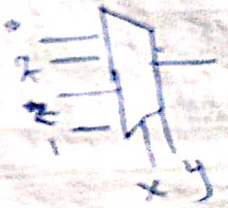
$$Y = D(\bar{S_1}\bar{S_0}) + A(\bar{S_1} S_0) + \bar{A}(S_1 \bar{S_0}) + 1(S_1 S_0)$$

if $S = A$ (var)
$S_0 = B$ (")

⇒

A	B	C	C_0	S	
0	0	0	0	0	z
0	0	1	0	1	
0	1	0	0	1	z
0	1	1	1	0	
1	0	0	0	1	z
1	0	1	1	0	
1	1	0	1	0	z
1	1	1	1	1	

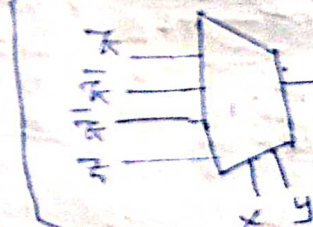
full adder



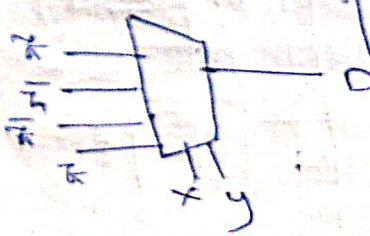
C (carry)

for full substractor

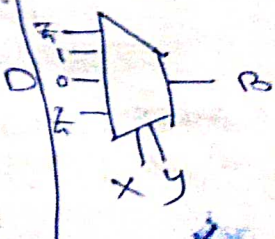
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



S (sum)



D

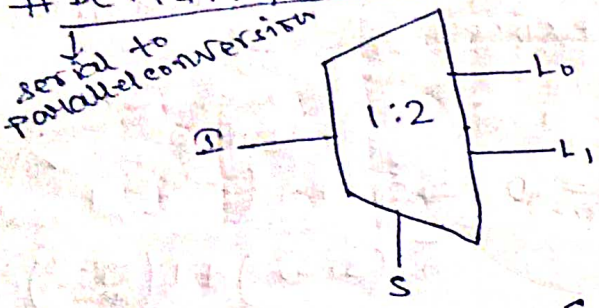


B

Digital logic- 14 (14/04/2024)

DeMultiplexor :- (1:2 DeMultiplexor)

↓ serial to parallel conversion

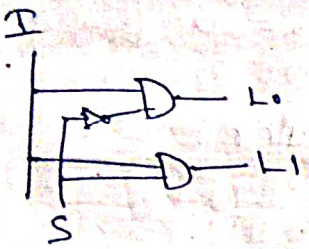


$L_0 = \bar{S} I$

$L_1 = S I$

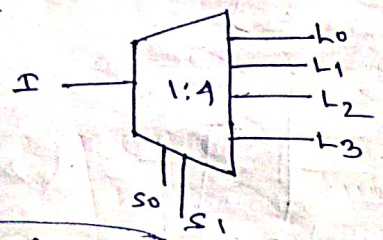
S2P

S	L_0	L_1
0	I	0
1	0	I

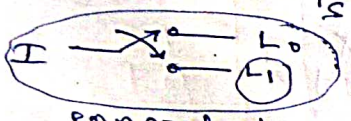


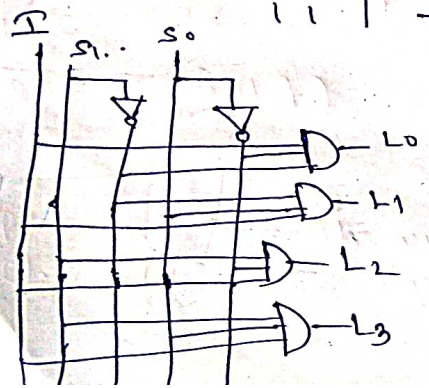
* 1:4 DeMultiplexor :-

(1 source 4 parallel lines)

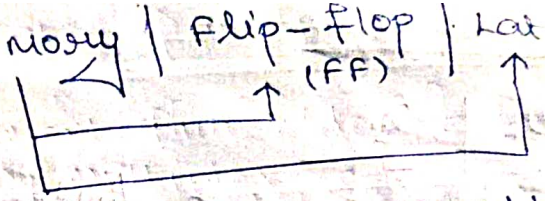


S_1 S_0	L_0	L_1	L_2	L_3
0 0	I	–	–	–
0 1	–	I	–	–
1 0	–	–	I	–
1 1	–	–	–	I





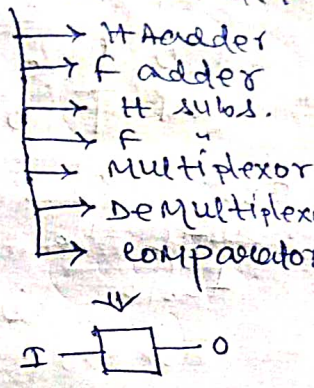
* Memory | Flip-flop (FF) | Latch

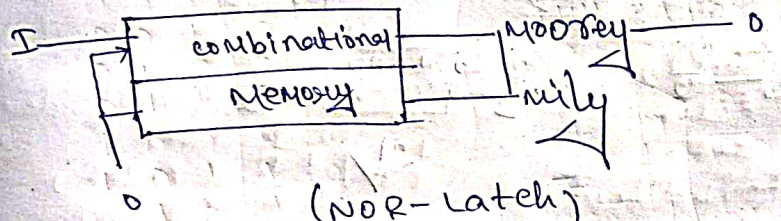


- sequential circuit

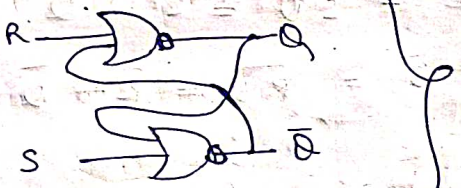


- combinational ckt





(NOR-Latch)



$Q = \overline{R + \bar{Q}} = \bar{R} Q$

$\bar{Q} = \overline{S + Q} = \bar{S} \cdot \bar{Q}$

Latch



you can assume

$Q = Q_R$

$\bar{Q} = Q_S$

$Q_R = \bar{Q}_S$

Truth table

R	S	Q_R	Q_S
0	0	Q	$\bar{Q}$
0	1	1	0
1	0	0	1
1	1	0	0

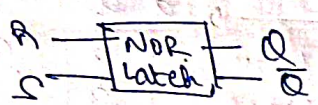
→ holds the previous state

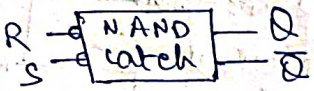
→ invalid we have to avoid this

To verify the table

01 → 00 → 10 → 00

(hold state)







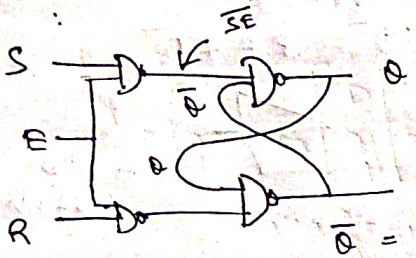
S	R	Q	$\bar{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	Q	$\bar{Q}$ (hold)

Invalid → forced to 1

NAND latch	NOR latch

} both are inverted operation

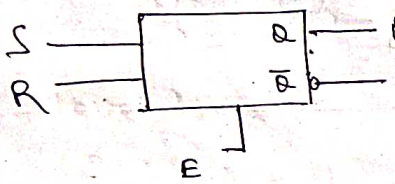
Digital logic-15 (26/09/2024)



$Q = \overline{S\bar{E}\,\bar{Q}} = S\bar{E} + Q$

$\bar{Q} = R\bar{E}\,Q = RE + \bar{Q}$

Enabled SR latch

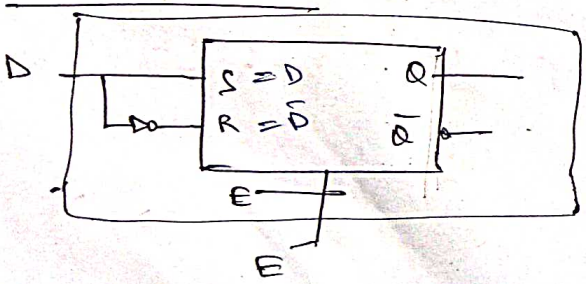


Function Table :-

E	S	R	Q	$\bar{Q}$	
0	x	x	Q	$\bar{Q}$	hold
1	0	0	Q	$\bar{Q}$	hold
1	0	1	0	1	reset
1	1	0	1	0	set
1	1	1	↑	↑	invalid

read / write

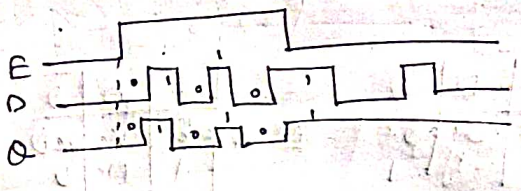
D-latch



Function table

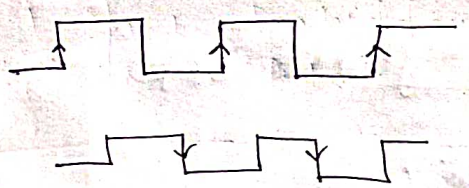
E	D	Q	$\bar{Q}$	
read	0X	Q	$\bar{Q}$	hold
write	10	0	1	reset
	11	1	0	set

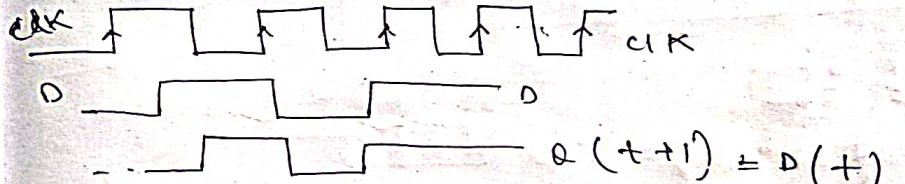
Level triggered

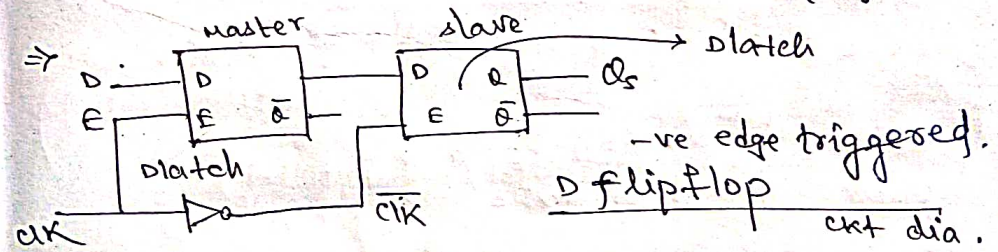


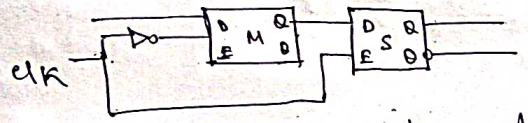
Edge triggered

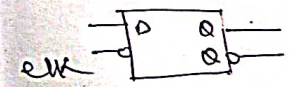
D flip-flop (FF)











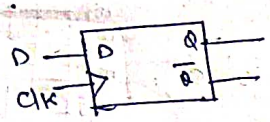
Digital logic - 16 (1/10)

D-latch



E	D	Q (next state)	Q̄ (next state)
0	X	Q	Q̄ (hold)
1	0	0	1 (reset)
1	1	1	0 (set)

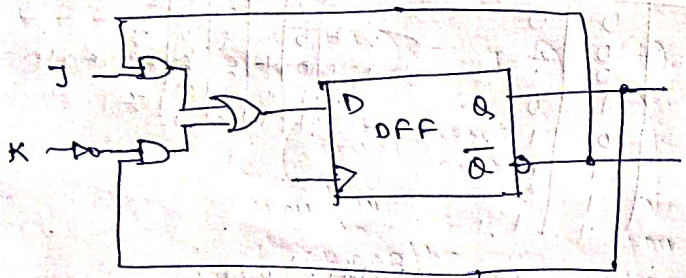
DFF



D	Q(t+1) (next state)
0	0 (R)
1	1 (S)

$Q(t+1) = D(t)$

JK-flipflop using D-flip-flop



$Q(t+1) = D(t) = J\bar{Q} + \bar{K}Q$

$= f/n$ (J, K inputs & present-table $(Q, \bar{Q})$)

$Q(t+1) = J\overline{Q(t)} + \bar{K}\,Q(t)$

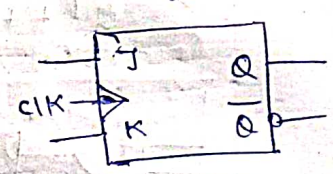
↳ state eqn / char. eqn

next/output state char. eqn

char. table

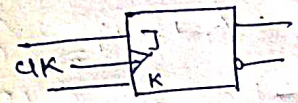
J	K	Q(t+1)	
0	0	Q(t)	hold
0	1	0	reset
1	0	1	set
1	1	$\overline{Q(t)}$	toggle

symbol



JK FF

When both JK are set to 1, 1 → output $\overline{Q(t)}$





toggle mode

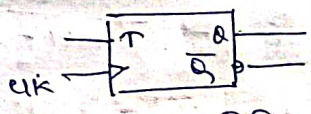
f/2 or 2T

freq. Time period

T. flip flop :-

TFF using JK





T-FF

T	Q(t+1)	
0	Q(t)	hold
1	$\overline{Q(t)}$	toggle.

char. table.

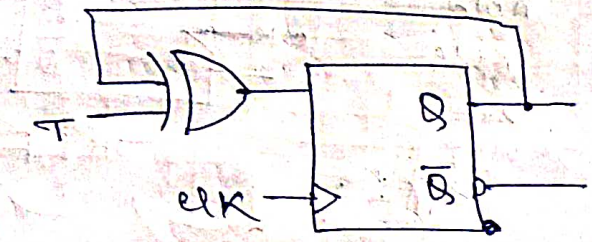
$Q(t+1) = T\overline{Q(t)} + \bar{T}\,Q(t)$

$= T \oplus Q(t)$

] char. eqn

TFF using DFF

→ $Q(t+1) = T \oplus Q(t)$ - [Implement this eqn]



DFF : char. eqn → fn of only input

$Q(t+1) = D(t)$

JKFF : $Q(t+1) = J\overline{Q(t)} + \bar{K}Q(t)$

TFF : $Q(t+1) = T \oplus Q(t)$

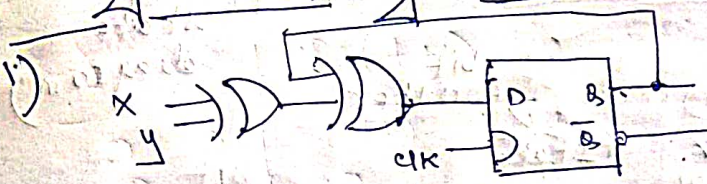
- $Q(t+1)$: next state / output state → you have to keep the (t+1) next state
- $Q(t)$: present state → you can keep the T or remove it (present state)
- fn of input & present state

Using these 3 FF you can implement any state machine :

Next state = f(inputs, present state)

Any Arbitrary state machine :-

1)



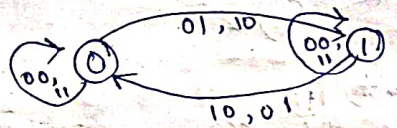
⇒ state eqn : $Q(t+1) = x \oplus y \oplus Q$

⇒ state table :-

x	y	Q(t)	Q(t+1)
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

odd no. of 1 → 1

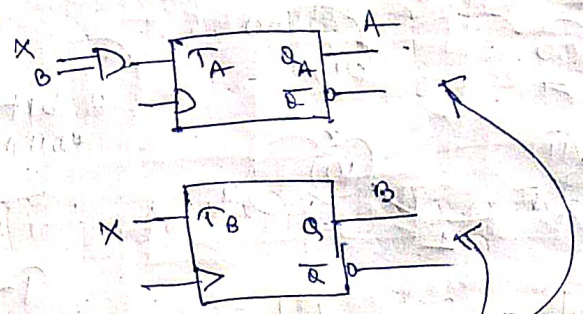
⇒ state diagram :-



↳ represn of state table using directional graph.

2 bit state machine

Any Arbitrary



state machine ckt of

present state

state equ:-

$T_A = BX$

$T_B = X$

$A(t+1) = Q_A(t+1) = T_A \oplus Q_A$

$B(t+1) = Q_B(t+1) = T_B \oplus Q_B$

Next state

$T_A \to BX$, $T_B \to X$

state table:-

BX	X	Q_A (A)	Q_B (B)	A(t+1)	B(t+1)
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
1	1	0	1	1	0
0	1	1	0	1	1
1	1	1	1	0	0

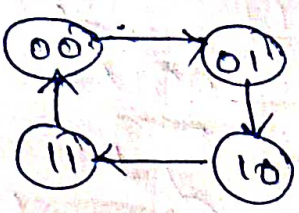
$A(t+1) = BX \oplus Q_A \to A = BX \oplus A$

$B(t+1) = X \oplus Q_B \to B = X \oplus B$

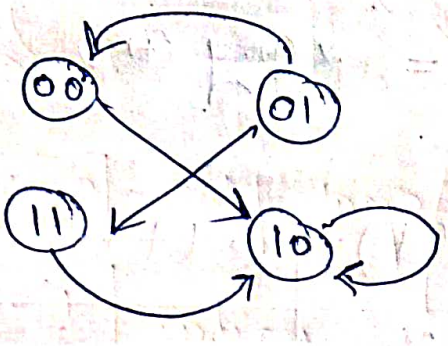
Actual state eqn.

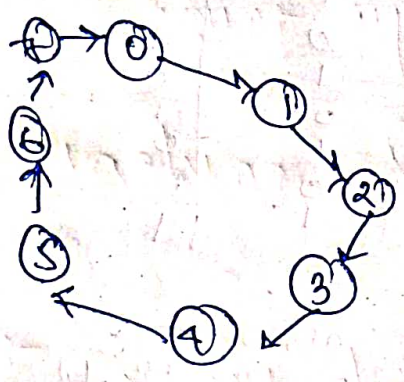
4 state (types of state diagram)

2 bit counter

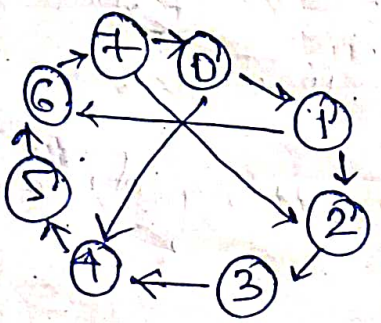
1) 

State dia.

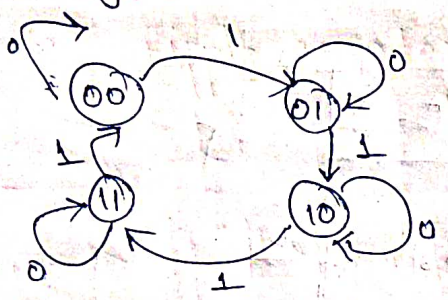
2) 

3) 

3 bit counter (8 state)

4) 

State diagram :



2 bit counter using T-flipflop.

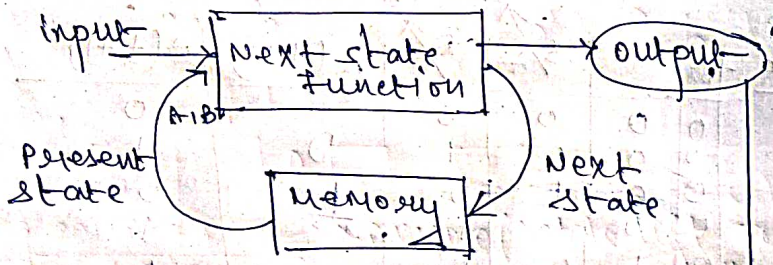
when x=0 self loop / hold the state

when x=1 when count up.

---

* Moore machine & Mealy machine

⇒



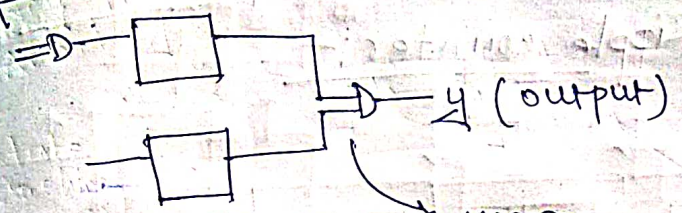
⇒

Moore machine, if : output = f (only state)

Mealy machine, if : output = f (input & state)

output :-

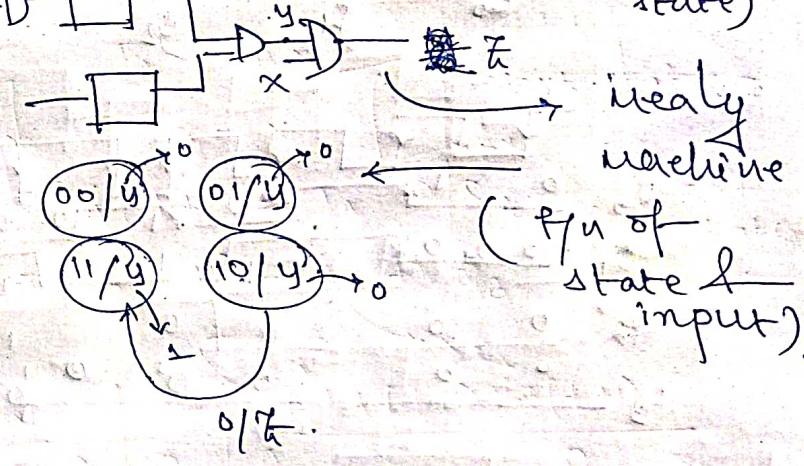
if



Moore machine ← 00 01 11 10

(f/n of only state)

if.



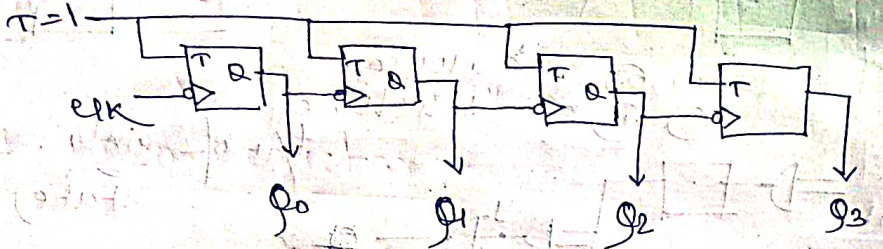
Mealy machine

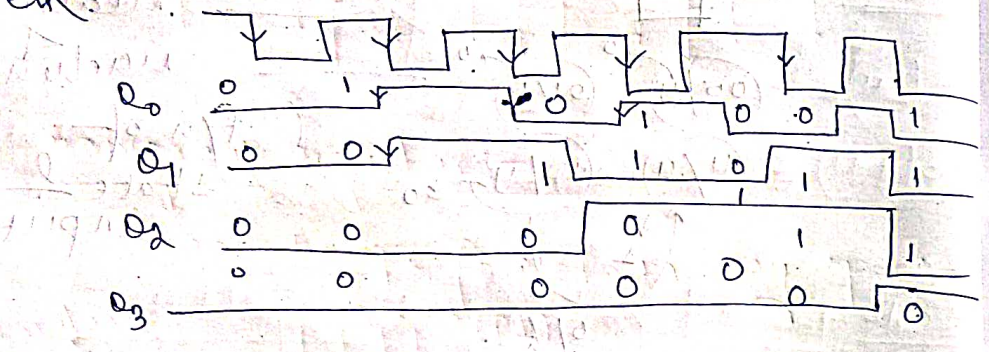
(f/n of state & input).

x	A	B	A (t+1)	B (t+1)	y	z
					0	0
					0	0
					1	0
					0	0
					0	1

Digital logic-17 (3/10/2024)

4-bit Ripple counter:-

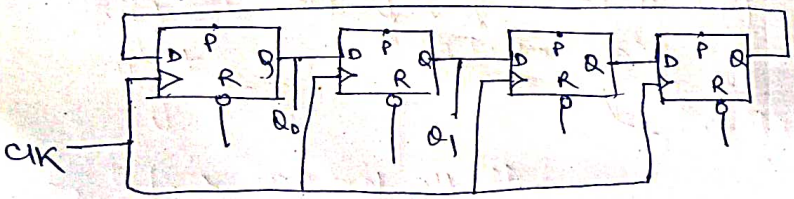


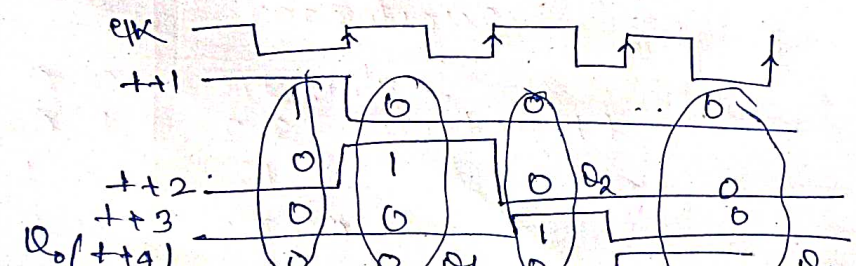


Actually:

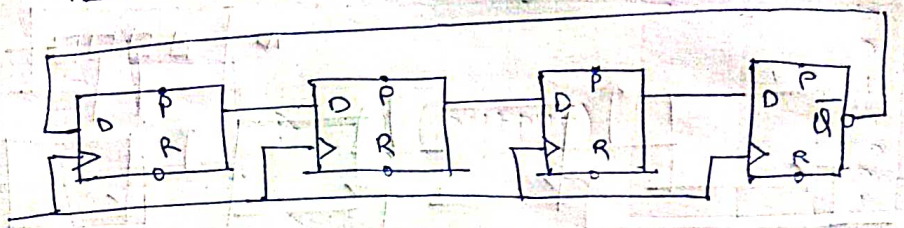


Ring counter:- → purpose (rotate left or rotate right)





Johnson counter:-



If the no. of flipflop = 4

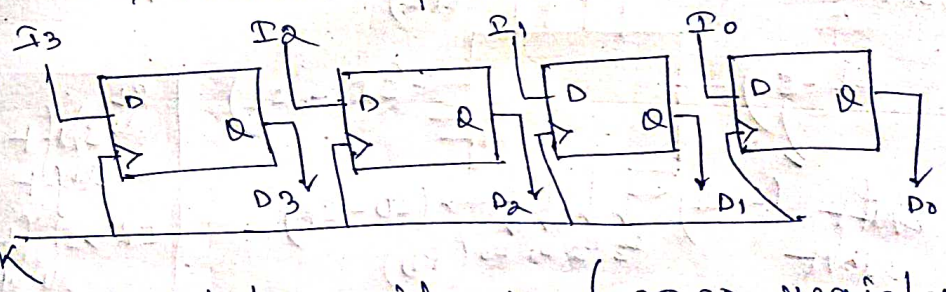
2 × 4 = 8 sequence will be there.

1 0 0 0 (++1)	0 0 1 1 (++6)
1 1 0 0 (++2)	0 0 0 1 (++7)
1 1 1 0 (++3)	0 0 0 0 (++8)
1 1 1 1 (++4)	1 0 0 0 (++9)
0 1 1 1 (++5)	

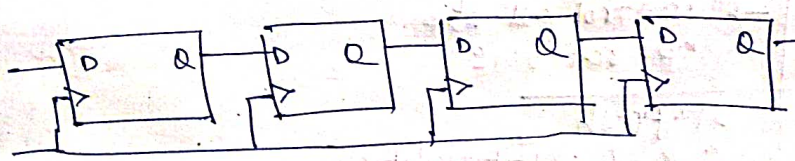
→ you can start anywhere.

PIPO register:-

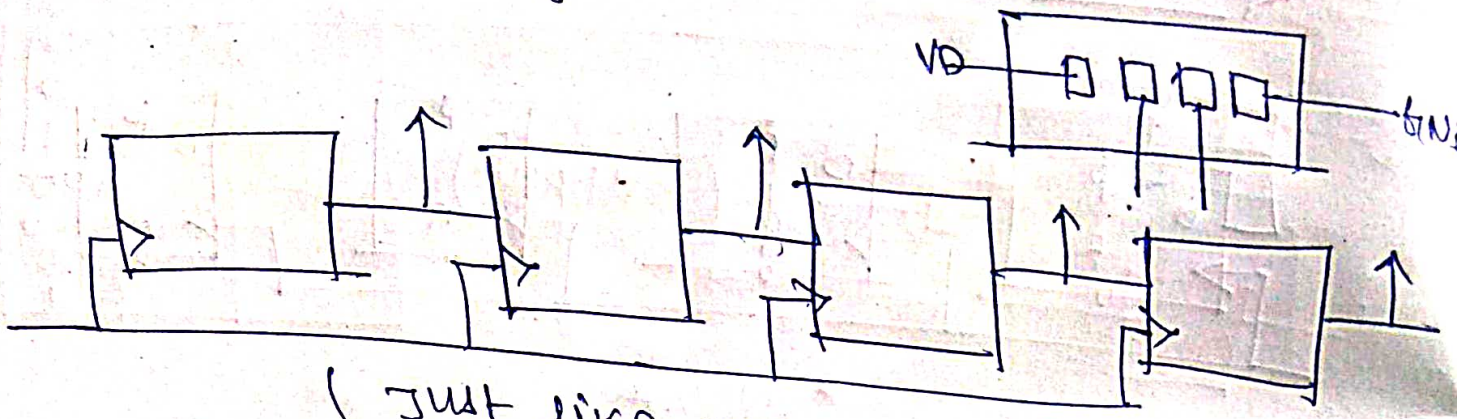
parallel in parallel out.



Serial in serial out:- (SISO register)

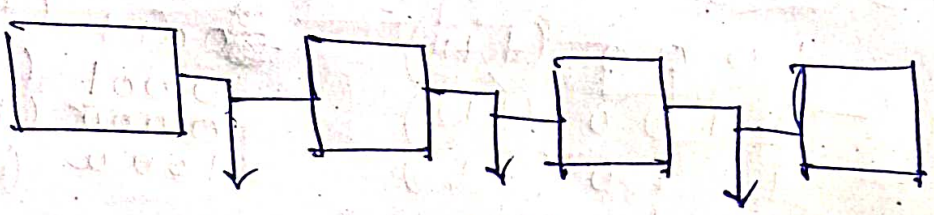


SIPO Register



(Just like prev.

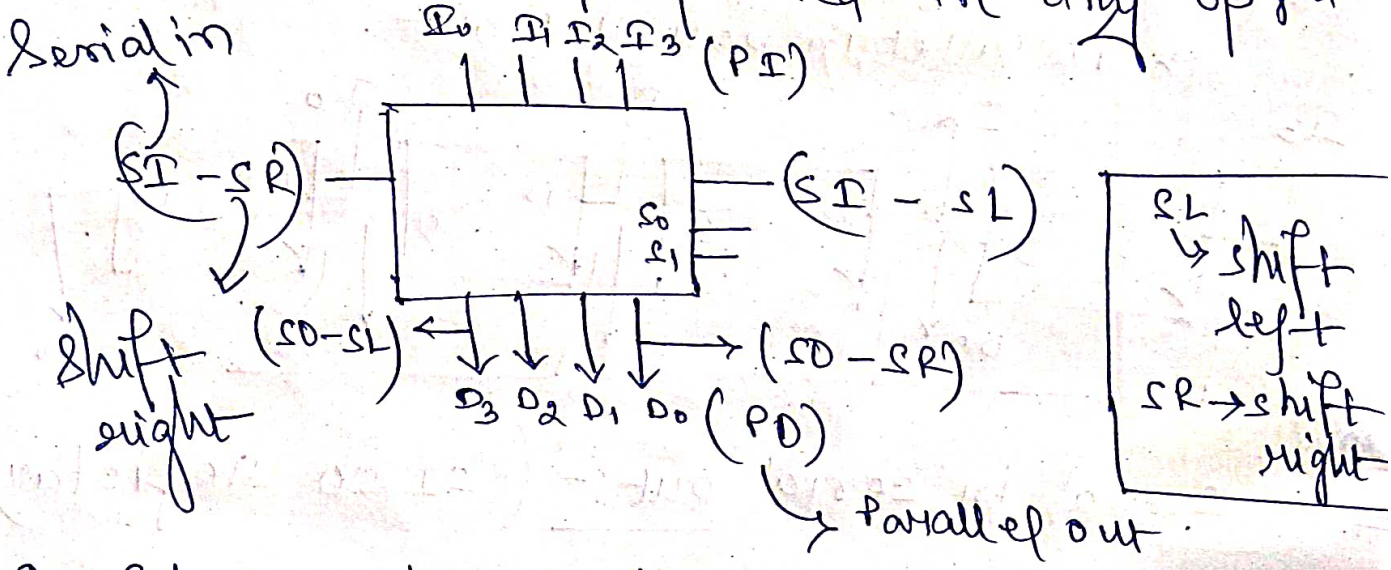
PISO Register:-



Universal Register:-

↓

it can be performed in any opr



S_0 ; S_1	→ control all the operations
0 0	Refresh
0 1	SR (shift right)
1 0	SL
1 1	PIPO