Page No. 432
Date

Digital Electronics

Syllabus -

@No: (414)

→ No. system - (414)
→ Boolean algebra | Kmap = 455 (venn diagram)
→ Switching circuit's = 501
→ Combinational circuit = 544

(logic funcs and minimisation)

→ HA | FA
HS | FS
mux | demux
encoder | decoder

→ sequential circuit = 590
- Latch | FF
- SR
- JK
- T, D

→ Conversion
→ Race round Condition
→ m/s FF

→ Register
counter

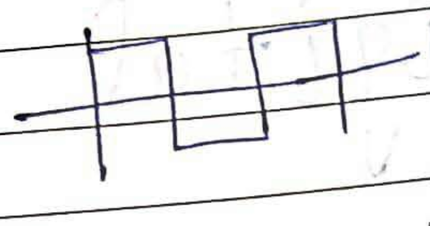
→ state machine
(FSM) Finite state machine

→ A/D and D/A converter

Page No. 422
Date

1.) Basic of digital logic

Digital:



measuring quantities that deals with discrete sets.

In digital parameter acquire discrete values are dealt with

Analog:

Analog to real signal

Page No.
Date

Properties of digital signal
or
Advantage of digital signal

(1) Error immune
Digital signals are affected lesser by noise, and hence probability of error is smaller

(2) Easy to Process

(3) Cheaper (component cost is less)

(4) Power efficient

(5) Error detection and correction is easier

(6) storage of Information

1.1 = Number System

(1) - Decimal number system - (10)

only two state (ON/off)

10 states (Not easy to manage)

(2) Binary number system (2)

(3) Octal number system (8)

(4) Hexa decimal (16)

Hexa = 6, decimal = 10

Toulroes seaicr T July, ekegi

Binary number system

No. of symbole → 2 → 0, 1

Base of Binary no system → 2 (✓)

Decimal	Binary
0	1
1	1
2	10
3	11 → 2^1 2^0
4	100
5	101
6	110
7	111

Page No. 415
Date

8 1000
9 1001
10 1010
!
1

(Bika)

0	0	0	0

0 0 0 1
0 0 1 2
!
0 0 0 9
0 0 1 0
0 0 1 1
0 0 1 2
1
0 0 1 9
0 0 2 0
1
0 0 9 9
0 1 0 0

0 - 0000
1 - 0001
2 - 0010
3 - 0011
4 - 0100

all the number system are developed copying the set/Reset pattern of decimal number system.

Conversion from Decimal to Binary
(Double Double method)

$(24)_{10} = (?)_2$

Note:

Conversion from Decimal to any number system:-

$(x_1 x_2 \text{---} x_n)_{10} = (?)_n$ n = base

n	$x_1 x_2 \text{---} x_n$	r_0
	$q_1 q_2 \text{---} q_n$	r_1
	$q_{11} q_{21} \text{---} q_{n1}$	r_2
	$q_{111} q_{211} \text{---} q_{n11}$	r_3
	⋮	⋮
	K_n where $K_n < n$	$r_n = K_n$

V.V.I
$K_n < n$
$r_n = K_n$

Now, write it in reverse order

$(?)_n = \{(r_n)(r_{n-1}) \text{---} (r_0)\}_n$

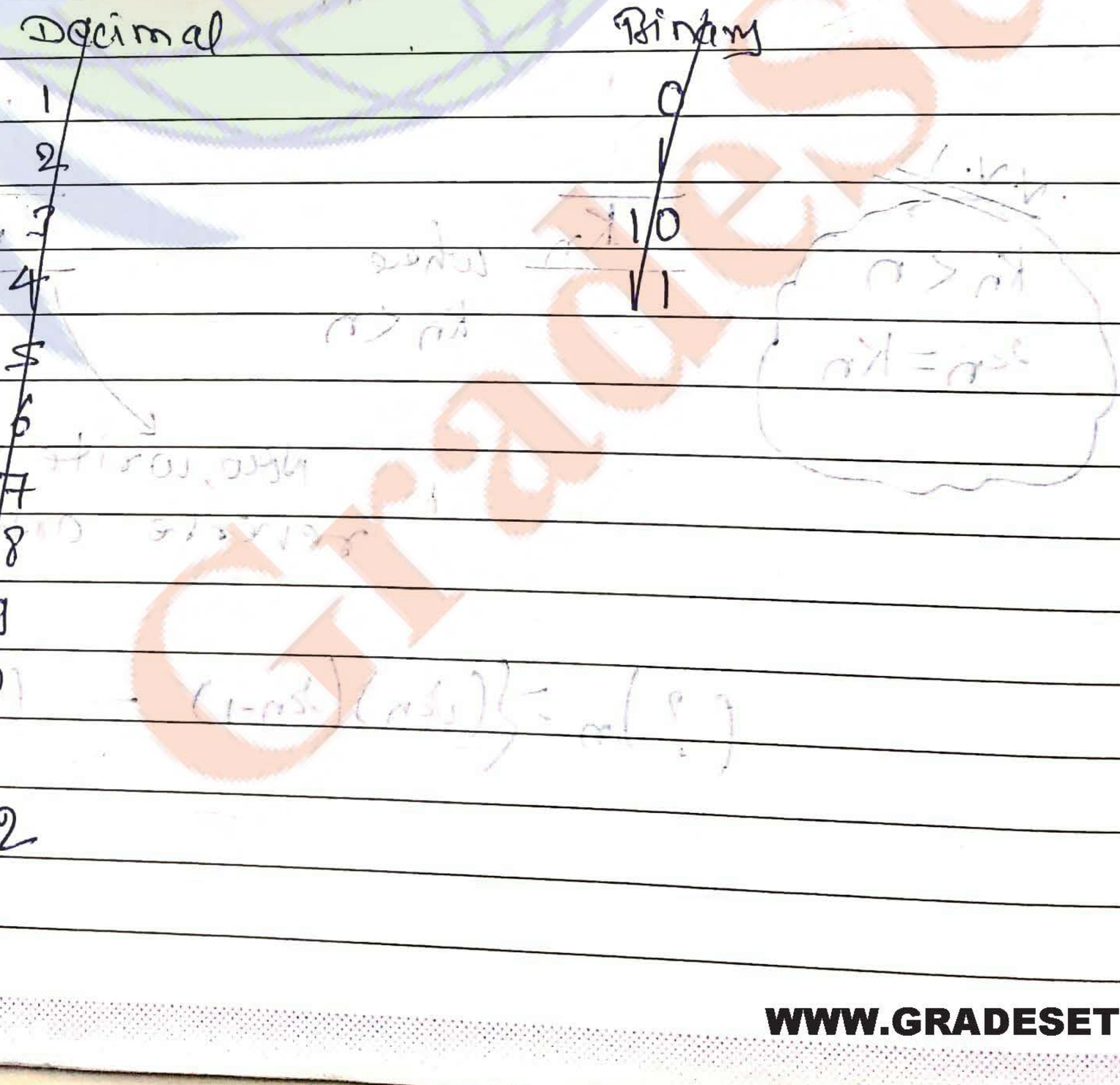
2	24	0
2	12	0
2	6	0
2	3	1
	1	

$(24)_{10} = (11000)_2$

2	96	0
2	48	0
2	24	0
2	12	0
2	6	0
2	3	1
	1	

$(96)_{10} = (1100000)_2$

* Short trick method!-



Short trick methods:

Decimal	Binary
0	0
→1	1
2	10
→3	11
4	100
5	101
6	110
→7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	

Rule 1st:-

Decimal	Binary
$1 \rightarrow 2^0$	1
$2 \rightarrow 2^1$	10
$4 \rightarrow 2^2$	100
$8 \rightarrow 2^3$	1000
$16 \rightarrow 2^4$	10000

$$(2^n)_{10} = (1\,000 \cdots n \text{ times})_2$$

Rule 2:-

Decimal	Binary
$1 \rightarrow (2^1-1)$	1
$3 \rightarrow (2^2-1)$	11
$7 \rightarrow (2^3-1)$	111

$(2^n-1)_{10} = (111 \text{ -- n-times})_2$

Rule 3:-

Decimal	Binary
$1 \rightarrow 2^0$	1
$2 \rightarrow 2^1$	10
$4 \rightarrow 2^2$	100
$8 \rightarrow 2^3$	1000
$16 \rightarrow 2^4$	10000

$\times 2^2$, $\times 2^3$, $\times 2$, $\times 2^2$

$2X$

	1	0	1	0

1	0	0	1	0

When ever a binary number is multiplies by 2^n the result will be same binary number shifted to left by n-time

Page No. 420
Date

Similarly dividing a number 2^n, shifty the binary number to right by n-times.

$24 = 3 \times 8$

$= 3 \times 2^3$

$= 11 \times 000$

$= 11000$

$96 = 3 * 2 * 3$

$= 11$

$= 3 * 32$

$= 11 \quad 2^5$

$= 1100000$

Attention: But we will not be able to convert all the number's.

Rule: 4

$520 =$ → closet power of 2 वाला No. लिखते

$520 = 512 + 8$

$= 2^9 \quad 1000$

$1000000000 + 1000 = 1000001000$

$(147)_{10} = (128 + 19)$

$= 2^7 + 19$

$= 2^7 + 16 + 2 + 1$

$= 10000000 + 10000 + 10 + 1$

$= 10010011$

$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$

$(269)_{10} = 256 + 13$

$= 2^8 + 13$

$8 + 5 \rightarrow 2 + 1 = 10 + 1$

$\rightarrow 2^3 = 1000$

$= 100001101$

$(254)_{10} =$

$\rightarrow 2^8 \rightarrow 100000000$

$(255)_{10} \Rightarrow (2^8 - 1) = 11111111$

-1

$= 11111110$

88
↓↓
symbolic
positional
value

★ Binary to Decimal number System:-

From any no. system to decimal

$$(x_n\ x_{n-1}\ \cdots\ x_2\ x_1\ x_0)_n = (?)_{10}$$

$x^n\ x^{n-1}\ \cdots\ x^2\ x^1\ x^0$

88
symbolic = same
positional = not same
value

$$(88)_{10} = 8\times10 + 8\times1$$

(8 = symbol, 10 = position; 8 = symbol, 1 = position)

$$? = (x^n \times x_n + x^{n-1} \times x_{n-1} + \cdots + x^2 \times x_2 + x^1 \times x_1 + x^0 \times x_0)_n$$

For Binary $x = 2$

↑ Base of the number system

$$(1\ 1\ 1\ 0\ 0\ 1)_2 = (?)_{10}$$

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

$2^7 \times 2^3 + 1$

$$= 2^5\times1 + 2^4\times1 + 2^3\times1 + 2^2\times0 + 2^1\times0 + 2^0\times1$$

$$= 32 + 16 + 8 + 0 + 0 + 1$$

$$= (57)_{10}$$

Note: $111001 = 111000 + 1$

$= 7\times2^3 + 1 = 56 + 1 = 57$

eg. 1101|0 11111 (first four digits 1101 bracketed: 13)

1101 000000 + 11111

$= 13 \times 2^6 + 31$

eg. $(10111100011101)_2 = (?)_{10}$

~~10 1111 000 000000~~ + 11101

~~10 1111~~

eg.

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

$2^{13} + 15 \times 2^8 + 7 \times 2^2 + 1$

★ Octal No. System

Base = 8

Symbol = 0, 1, 2, 3, 4, 5, 6, 7

Decimal to octal

$(29)_{10} = (?)_8$

```
8 | 29   5  ↑
  |----
  | 3
```

$(29)_{10} = (35)_8$

$(37)_8 = (?)_{10}$

$(74)_8 = (7 \times 8^1 + 7 \times 8^0)_{10}$

$= (24 + 7)_{10}$

$= (71)_{10}$

Octal to Binary:-

$(1\ 8\ 4\ 6\ 0\ 0\ 0\ 7)_8 = (?)_2$

↓

every symbol will be written in three bits of binary

1 ↓ 0001 | 8 ↓ 010 | 4 ↓ 100 | 6 ↓ 110 | 0 ↓ 000 | 0 ↓ 000 | 0 ↓ 000 | 7 ↓ 111

$(1010\ 100\ 110\ 000\ 000\ 000\ 111)_2$

Binary to Octal

$(111\ 0\ 101\ 111\ 001)_2 = (?)_8$

(we will start making group from right side) (⟵)

$= (001\ 110\ 101\ 111\ 001) = (16571)_8$

while converting from binary to octal convert every octal digit into three make group of three from right and then write corresponding octal for every group.

☆ Hexa decimal Number System

- Base-16
- Symbols

Decimal	Hexa
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Decimal to Hex9

$(32)_{10} = (?)_{16}$

$2 \times 16 = 20$

16	16	0
	2	

$(32)_{10} = (20)_{16}$

Note

* $(2^1)_{10} = (10)_2$

$(8^1)_{10} = (10)_8$

$(16^1)_{10} = (10)_{16}$

$(40)_{10} = (?)_{16}$

$(2 \times 16) \neq 20$

$3 \times 16 \rightarrow 30$

Note:

$(40)_{10} = (?)_5$

$(42)_{10} = (?)_{16}$

16	42	$\rightarrow 10 \rightarrow A$
	2	

$(2A)_{16}$

In any number system symbols used will always be less than base of the number system.

* Hexa to decimal

$$(BAD)_{16} = (?)_{10}$$

16^2 16^1 16^0

$$= B \times 16^2 + A \times 16^1 + A \times 16^0$$
$$= 16 \times 10^2 + 10 \times 16 + 10$$

* Binary - Computer Understandable.
Hexa -] memory address
Octa -]

* Hexa to Binary

$$(BAD)_{16} = (?)_2$$

B A D
↓ ↓ ↓
11 10 1101
↓ ↓
(1011 1010 1101)$_2$

{ four bit is sufficient to represent $(16)_{10}$

Binary to Hexa

$(\underline{0011}\ \underline{1010}\ \underline{1101})_2$

3 A D

$(3AD)_{16}$

* Hexa to Octal and vice versa

$(BAD)_{16} = (?)_8$

Hexa to octal
Hexa → Binary → Octal

octal to hexa:
octal → Binary → Hexa

$(BAD)_{16}$

$(1011\ 1010\ 1101)_2$

5 6 5 5

$(5655)_8$

$(576)_8 = (?)_{16}$

000 101 111 100

1 7 5

$(175)_{16}$ → Hexa

Q) In a number system with base x, following is true

$(100)_x = (81)_{10}$

what is x ?

Ans. 9

Note $(x_1 x_2 . x_3 x_4)_{10} = (?)_{10}$

$(32.05)_{10} = (?)_2$

$(0.05)_{10} \rightarrow 0.05 \times 2 = 0.10 \rightarrow 0$
$0.10 \times 2 = 0.2 \rightarrow 0$
$0.2 \times 2 = 0.4 \rightarrow 0$
$0.4 \times 2 = 0.8 \rightarrow 0$
$0.8 \times 2 = 1.6 \rightarrow 1$
$0.6 \times 2 = 1.2 \rightarrow 1$

$(0.05)_{10} = (0.000011)_2$

$(32.05)_{10} = (100000.000011)_2$ Ans

$(0.5)_{10} = (?)_2$

$0.5 \times 2 = 1.0$

$(0.5)_{10} = (10.1)_2$

(9) $(10.1)_2 = (?)_{10}$

$x_n = x_0 \; r^{-1} \; r^{-2}$

$= x_n x_{n-1} - x_0 \cdot r^{-1} \cdot r^{-2}$

$= (10)_{10} \rightarrow (2)_2$

$= (0.1) = 1 \times 2^{-1} = 0.5$

* $(0.101)_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = (0.625)_{10}$

~~$\therefore 10(1.0.1) = 1 \times 2^1 + 0 \times 2^0 +$~~

~~$= (2.5)_{10}$~~

(10) $(3A)_5 = (A3)_4$

A is

a) A = 4 (b) A = 5 (c) A = 3 (d) none of these

Ans

Note: किसी भी base वाले number को decimal में convert करके equal करते है॥

Notes

$(100)_x = (1010001)_2$ //

$x^2 \times 1 + x^1 \times 0 + 0 = 2^6 + 2^4 + 1$

$x^2 = 81$

$x = \pm 9$

$x = 9$

Sol)

$3 \times 5 + A \times 1 = A \times 4 + 3$

$15 + A = 4A + 3$

$3A = 12$

$A = 4$ → Base is also 4, so this is not possible

(d) none of these.

- Creating an opportunity
- execution ✓

9th

GATE 2014:

Consider the eqn

$(123)_5 = (x8)_y$

with x and y unknown.

No of possible solutions =

Ans

123

$5^2 \times 1 + 5^1 \times 2 + 5^0 \times 3 = xy^1 + 8xy^0$

$25 + 10 + 3 = xy + 8$

$38 = xy$

$xy = 30$

$30 = xy$

① $\begin{cases} 2 \times 15 \checkmark \\ 1 \times 30 \\ 3 \times 10 \checkmark \end{cases}$

② ~~5 × 6~~

$x < y$

$y > 8$

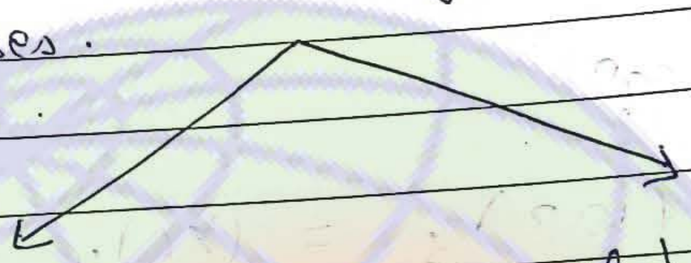
$(80)_9$

Q.)

2

1.2. Codes

Binary codes:
Codes are designed for specific purposes.



(2)

① BCD code
- Binary coded decimal
- numeric code

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

9 — 1001 (Binary)
9 — 1001 (BCD)

10 — 1010 (Binary)
10 — 0001 0000 (BCD)

(toreet every ~~bit~~ seperate)

- easy ~~for~~ making calculator
-

(2) ASCII codes:
(American Standard code for Information Interchanges)

- (Its a alphanumeric codes)
- It's a seven bit code, uses to represent alphabets and special characters and numerals.

$2^7 \rightarrow 128$

- ASCII is modifies ~~to~~ to represent greater symbols and EBCDIC codes.

{ keyboard I (o/p of 4-bit) ↓ A → [ ] ↓ corresponding ASCII generate }

(extended binary coded decimal interchange code)

Its a 8 bit code, make's no. of symbol double.

[illegible]
0 to 9: 48 → 54
A to Z: 65 → 90
a to z: 97 → 122

(4) Excess 3-code

	Binary	(+ve 3)	X 3 - 3
0	0000	→	0011
1	0001	→	0100
2	0010	→	0101
3	0011	→	0110
4	0100	→	0111
5	0101	→	1000
6	0110	→	1001
7	0111	→	1010
8	1000	→	1011
9	1001	→	1100

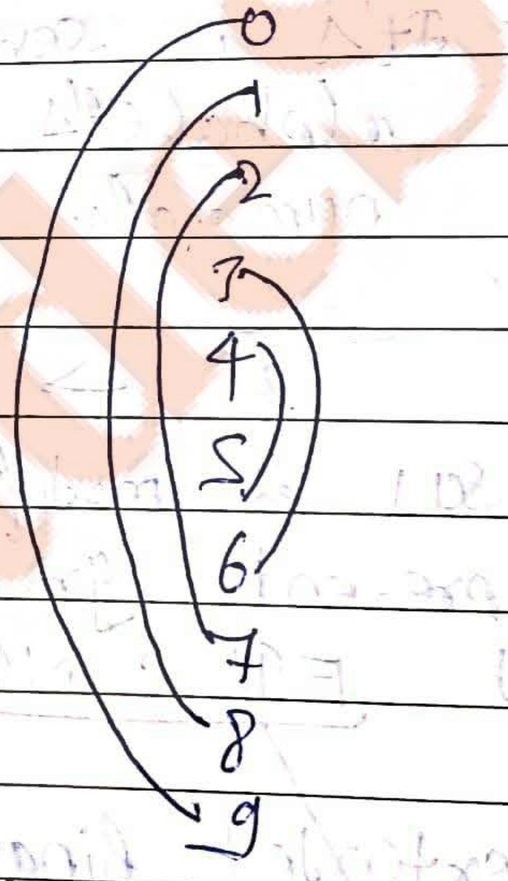
excess 3-code $= (N0 + 3)_2$

Note

9 − 0 = 9 (complement of zero is nine)

9 − 1 = 8 (complement of 1 is 8)

0 ↔ 1
1 ↔ 0



(Complement)

(Self-Complementation)

Page No.
Date

* excess 3 is a self complementary code
It means complementation is true in both decimal and excess 3 notations.

* A code is said to be self complementary if the code word of 9's complement of N, i.e. N' can be obtained from code word of N. by interchanging all zero's by one's and all one's by zero.

* Complements are used to represent negative numbers.

Page No.
Date

1.3 Arithmetic Operations:

Page No.
Date

1.8.4 Signed Number Representation

	Sign Bit	1's complement	2's Comple
-8		1000	1000
-7	1111	1001	1001
-6	1110	1010	1010
-5	1101	1011	1011
-4	1100	1100	1100
-3	1011	1101	1101
-2	1010	1110	1110
-1	1001	1111	1111
0	0000	0000	0000
+1	0001	0001	0001
+2	0010	0010	0010
+3	0011	0011	0011
+4	0100	0100	0100
+5	0101	0101	0101
+6	0110	0110	0110
+7	0111	0111	0111
	-7 to +7	-7 to +7	-8 to +7

Concept of 1's and 2's complement :-

-2
+2

Note: एक बात ध्यान रखें, the value तो same ही रहेगा, लेकिन sign वाले bit change हो जाएगा, अर्थात् -ve दर्शाने वाले हो complement का importance है।

Problem 1) - solved

+x
-x
0

Problem 2 nd. solves

$2^4 = 16$ representation

Note: So 2's complement method ए leaf of all //

-ve = 1 //
+ve = 0 //

Range:-
4 bit number: Signed = -7 to +7 ✓
1's compl = -7 to +7 ✓
2's compl = -8 to +7 ✓

-15	11111	10000
	1000	
	1001	
	1010	
	1011	
	1100	
	1101	
	1110	
	1111	
+0	00000	
+1	00001	
+2		
+15	01111	

Note: In sign bit method and 1's complement method's following drawbacks exist's.
(a) +0 and -0 are represented by different value.
(b) Adding +x and -x does not give null value.

(c) efficiency of representation is not best.

Note:

	3 bit's		
	Sign	ist	2's
-4			100
-3	111		101
-2	110		110
-1	101		111
-0	100		000
+0			000
+1			001
+2		010	010
+3		011	011
	(-3 to 3)		-4 to 3

* Range of Representation using n-bit's

(a) In signed bit method and it's complement method,

$$\text{Range} = -(2^{n-1}-1) \text{ to } (2^{n-1}-1)$$

(b) In 2's Complement method

$$\text{Range} = -2^{n-1} + 0 + (2^{n-1}-1)$$

(c)

Q> wh

So 5

Gra

Q) what is the range of representation using 6 bits. in 2's compliment

Sol^n

$= -32$ to 31

$$\text{Range} = -2^{n}$$

Grate:- ✱ 1997

1111 1111 1111 1111

1

FFFF

1111 1111 1111 1111

1001 → -ve

0001 → +ve

-8 1001 0001

= - (2's com of 1001)

= - (0111)

= -7

Note:
while converting 2's complement of a number copy the bits starting from right till we get 1's one after copying first one complement rest of the bits to the left

F F F F

1111 1111 1111 1111

$-($ 2's complement of FFFF $)$

$-($ 0000 0000 0000 0001 $)$

$= -(1)$

$= -1$

In 2's complent 10 bit

11 0000 1101

$-$ 00 1111 0011

$-(15 \times 2^4 + 3)$

$-(243)$

In 2's complent method 10 bits

0000 0010 00

$\overline{0000}$

$\rightarrow +0$

true, so simply +0 //

Q3> Grate 1997

N

↓

1 Byte → 16 bit

	N

Grate Q

-4 100 ⟷ 1100 11100

-3 101
-2 110
-1 111
0 000
0 000
+1 001
+2 010
+3 011
+4 ~~3 bit~~ 100

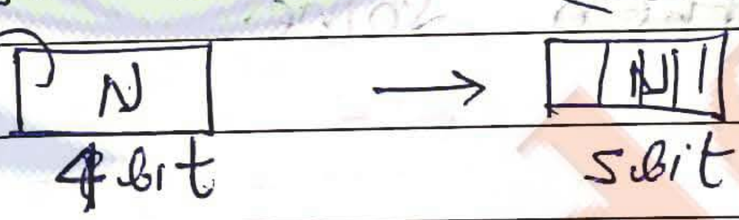
0100 (4 bit) 00100 (5 bit)

msp (most significant will be copied)

(c) A

Gate

1's complement



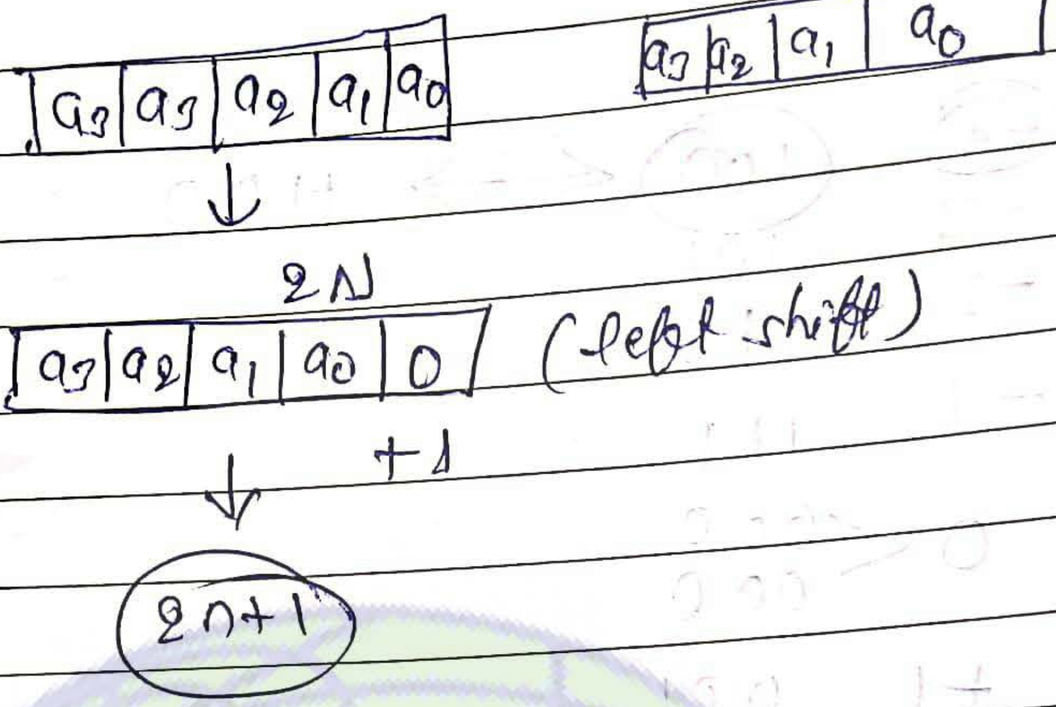
A number N is stored in 4 bit register and it is designated as N. The no. is in 2's complement formate.

Number is now copied in a 5 bit register and after some shifting operation, the result is as shown [a3 | a2 | a1 | a0 | 1]

The result is

(a) N (b) 2N (c) N+1 (d) 2N+1

Page No.
Date



Q4. Gate GATE 1998
mov will do copies

Q5.

+17 16 8 4 2 1
0 1 0 0 0 1
1 0 1 1 1 1

-17 ⟹ 6 bit
↓
including sign bits

steps:- In these type of questions
1st step's is to identify no. of bit's requires to represent the number, then solve.

Q6. ~ -8/

Q7/ -31 to +31 16 8 4 2 1

Q8

1
-5

01
0 1 0
1 0 1

11001 → 2's comp ⇒ -(00111) → -7
1001 ⇒ 2's com ⇒ -(0111) → -7
11001 ⇒ -(00111) ⇒ -7

Date

16 + 2 1

Q9.

47

0100 0011

$(2 \times m)_{10} = (20)_m$

$(7 \times m)_{10} = 70$

47
32
11

16	47	11
	2	(2 T)

T_1 (47) → 32 + 11 → 20 → 11 → 2 T

47

0100 0011

Q10 BCD → base - 5

$(24)_5$

010 100

1000 100 1 00 1

4 2 0 1

Q11 (using 6 bits)

+ve no. → 01110 → +14

-ve no. → 1100 1 → -(00111) = -7

14 - 7 = 7

00 0111

1.5 | Overflow Concept! -

Note (overflow)

In 4 bit

-6 ⇒ 1010
-3 ⇒ 1101
(-9) 0111 → 7

(-9) should be represent in 5-bit number//

⇒ overflow
└ (i) ~~if~~ carry generate
(ii) 2's ~~comp~~ comple

x 1010
y 1101
0111 → z

0110
0011
1001

overflow
└ So, to solve

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

→ This is the condition of overflow in 2's complement.//

V.V.I

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

Now to solve it ~~we~~ requires one extra bit

-6 → 11010
-3 → 11101
(1) 10111 — (01001) → -9

+6 → 0 1 1 0
-7 → 1 0 0 1
1 1 1 1

↓

= (0001)

(-1) (Correct) //

Conditions:-
1st & 2nd: ⇒ $d = x \cdot y \cdot \bar{z} + \bar{x} \cdot \bar{y} \cdot z$

0 0 → 1
1 1 → 0
Overflow

01) Gate 1987

+6 → 0110
-7 → 1001
1111

= (0001)

(-1) //

Note: When a the and the numbers are added using 2's complem. method, If result is not outside the range, then
(a) If overflow occurs result is the
(b) If overflow does not occur result is the and is in 2's complem. form

T₁

Gate (15) $\sqrt{(224)_r} = (13)_r$

$\sqrt{2\times r^2 + 2\times r^1 + 4\times r^0} = 1\times r + 3\times r^0$

$\sqrt{2r^2 + 2r + 4} = r + 3$

~~$2r^2 + 6 = (r+3)^2$~~

~~$2r^2 + 6 = r^2 + 9 + 6r$~~

~~$2r^2 - r^2 - 6r + 6 - 9 = 0$~~

~~$r^2 - 6r - 3 = 0$~~

$2r^2 + 2r + 4 = r^2 + 6r + 9$

$2r^2 - r^2 + 2r - 6r + 4 - 9 = 0$

$r = (5), -1$

Ans

(16) 4 3 ⇒ 32 + 11

0010 1011 } T₁

(22) 11111011 ⇒

−(00000101)

↳ (−5)

(9) ⇒ 11100111

−(00011001)

(−25) (a) Ans

(10) $(-539)_{10}$

512 + 27

↳ −(1 0000 11011) ↳ 9's

−(0 10000 11011)

(1) 10 1111 001 00

D E F

2.) Boolean Algebra & minimization! -

Boolean Algebra

→ George boolean
(1854)

ON/OF

f = A → 10 am < 1 / 0
- → class will start
- → class will not start

2.1) (Boolean) operations! -
Logic

Three operation :-

① NOT A / $\overline{A}$ (negation)

$f = \overline{A}$

② Join (And)

AND:

f = A and B

③ Join (OR)

OR:

f = A or B

NOT, AND, OR

where A, B are boolean functions.

NOT, AND, OR → Operator

a) Boolean algebra deals with statements/ variables having only two possible values.

b) Boolean variables can only have two values true or false
0 or 1
Yes or NO

c) There are three basic operations in boolean
(i) NOT
(ii) AND
(iii) OR

NOTE Postulates :-
eg. Is light always travel in straight line.
(Postulates are more fundamental than theorem)

Postulates are prerequisit

★(1) NOT

$\bar{A}$ = NOT A

(9) $\bar{\bar{A}} = A$ Involution

$\bar{0} = 1$
$\bar{1} = 0$

(2) AND (·)

A	B	f
0 · 0		= 0
0 · 1		= 0
1 · 0		= 0
1 · 1		= 1 ✓

Postulates:-

①

A	B	f
0 · 0		= 0
0 · 1		= 0

$A \cdot 0 = 0$ ✓

②

$0 \cdot 1 = 0$

$1 \cdot 1 = 1$

$A \cdot 1 = A$ ✓

③ $A \cdot \bar{A} = 0$ ✓

④ $A \cdot A = A$ ✓

(3) OR (+)

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 1$

Postulate:-

(6) $A + 1 = 1$ ✓ ... किसी भी एक जोड़ीगे तो एकही

(7) $A + 0 = A$ ✓

(8) $A + \bar{A} = 1$ ✓ ...

(9) $A + A = A$ ✓

2.2) Law of boolean algebra:-

Theorems
absorb, distr, conjun

☆ Theorems:-

10) Distribution Theorem

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

Note: (only on boolean)

$$\boxed{x + y \cdot z = (x + y) \cdot (x + z)}$$ ✓

(**) eg. $f = \bar{A} + ABC + \bar{A}BC$

$= \bar{A}(1 + BC) + ABC$

$= \bar{A} + ABC$

$\bar{x} \quad x \; \bar{y} \; \bar{z}$

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

$= 1 \cdot (\bar{A} + BC)$

$= \bar{A} + BC$

$1 + A = 1$

Date

11) Absorption Theorem :

$$x + \bar{x}y = (x+\bar{x})(x+y)$$
$$= 1 \cdot (x+y)$$
$$= x+y$$

So,

$$\boxed{x + \cancel{\bar{x}}\, y = x+y}$$

complement will be absorbed. (बाद वाला complent absorb होगा)

$$\boxed{\bar{x} + \cancel{x}\, y = \bar{x}+y}$$

complent of 1st will absorbes

eg. $f = \bar{x} + xy$

$= \bar{x} + 1$

$f = \bar{A} + ABC$

$= \bar{A} + BC$

2.3) Boolean algebric theorem:-

De-morgan's Theorem

$$\overline{X+Y+Z} = \overline{X} \cdot \overline{Y} \cdot \overline{Z}$$

$$\overline{X \cdot Y \cdot Z} = \overline{X} + \overline{Y} + \overline{Z}$$

"Break the bar, change the sign"

✓☆ Three ways of simplication of Boolean algebra

(i) Using Boolean algebra postulates and theorem

(ii) K-map (Graphical method of solving)
Note: till five to six variable.

(iii) Tabulation method
Note: 7 or more variable.

eg:) $f = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

$= (\overline{A}+A)BC + A\overline{B}C + AB\overline{C}$

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

$= BC +$

$A = A + A + A + \cdots$

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

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

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

$= BC + AC + AB$

$= AB + BC + AC$

or

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

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

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

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

$= \bar{A}BC + AB + CA$

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

$= B(A + C) + CA$

$= AB + BC + CA$

eg 2.

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

$= \bar{A} \cdot \overline{\bar{A}CD} \cdot \overline{\bar{A}\bar{B}} \cdot \bar{C}$

$= (\bar{A} \cdot A)CD\left(\bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}}\right)$

$\bar{A}A$

0

$= 0$ Ans

eg:> $f = \overline{\overline{ABC} + \overline{\overline{AB+CD}}}$

Ans $\overline{\overline{ABC}} \cdot (\overline{\overline{AB+CD}})$

$\overline{ABC} \cdot (AB + CD)$

$\overline{ABC}\, AB + \overline{ABC}\, CD$

$= (\overline{A}+\overline{B}+\overline{C}) AB + (\overline{A}+\overline{B}+\overline{C}) \cdot CD$

$= AB\overline{C} + \overline{A} CD + \overline{B} CD$ Ans

~~$= AB\overline{C} + C\overline{A}D + \overline{B}CD$~~

(12) Consensess Theorem

$f = xy + yz + \overline{x}z$

~~$= (xy + yz + \overline{x}z)\, xyz$~~

~~$= xyxyz$~~

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

$= xy + xyz + \overline{x}yz + \overline{x}z$

$= xy(1+z) + \overline{x}z(y+1)$

$= xy \cdot 1 + \overline{x}z \cdot 1$

$= xy + \overline{x}z$

① only one variable should be complement
② three variable
③ each repeat only two time,

Imp

$xy + yz + \overline{x}z = xy + \overline{x}z$

इन दोनो के aggrement जानता है और उसे को out कर देते हैं

अगर कोई एक complement हो तो भी complement होगा बस वही वाला term होगा।

T1

eg. $\bar{A}\,BC + A\,DE + BC\,DE = \bar{A}\,BC + A\,DE$

eg.> $\bar{x}\,\bar{y} + \bar{y}\bar{z} + x\bar{z} = \bar{x}\,\bar{y} + x\bar{z}$

eg2.> $A\bar{B} + B\bar{C} + \bar{A}\,C = A\bar{B} + \bar{A}\,C$

Condition:

* To apply consensus theorem, check following conditions
  (i) the number of variables are three
  (2) Each variable is used two times
  (3) only one variable is complement

  If above three conditions are satisfies then those product term will contained complementes variables are retained.//

* Boolean functions:-

variables can also be made with multiple alphabets and if the combination of alphabet come always in bunch then it is just one variable.

T1 $\bar{A}\,BC + A\,DE + BC\,DE = \bar{A}\,BC + A\,DE$

after shows to same

2.5) Representation of boolean functions :-

Boolean function

- minimal
- Canonical (the actual form)
  - SOP (Sum of Product) → made of min term
  - POS → made of max term
- SOP (Product of sum)
- POS

Note :- with n-variables maximum possible boolean functions are

2^{2^n}

for $n=2$

total function $= 2^{2^2} = 2^4 = 16$

$f(A, B, C, D)$ → order matters here

A	B	C	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

→ $\bar{A} \cdot B \cdot C \longrightarrow m_3$

→ $A \cdot \bar{B} \cdot C \longrightarrow m_5$

→ $AB\bar{C} \longrightarrow m_6$

→ $ABC \longrightarrow m_7$

If $n=1$, total $f = 4$

$f(A) = f=A,\ f=\bar{A},\ f=1,\ f=0$

A	AB
$\bar{A}$	$A\bar{B}$
B	$\bar{A}B$
$\bar{B}$	$\bar{A}\bar{B}$
$A+B$	$\bar{A}B + A\bar{B}$
$A+\bar{B}$	$AB + \bar{A}\bar{B}$
$\bar{A}+B$	0
$\bar{A}+\bar{B}$	1

total entries

Note: minimum के case में 0 Prime लेंगे क्योंकि मतलब Zero (0) निकाले में bar(-) लगेगा।
Date
क्योंकि maximum के case में '1' Prime है, इसका मतलब 'i' के case में bar(-) लगेगा॥

min term:-

min terms are ~~ANDed terms of~~ ANDed terms of n-variables for which functions value is '1'.

T_i — (min term = ANDed term)

- variable may be primed or Unprimed

$X \rightarrow$ Unprimed (1)
$\overline{X} \rightarrow$ Primed (0) (Note: Prime का मतलब bar(-) लगा हुआ होगा)

- In min-terms input variable having value '0' are considered Prime. and ~~value having~~ variable having value '1' are considered Un-Prime.

$m_{n'}$

where

$n \rightarrow$ Decimal equivalent of $(XYZ)_{10}$

m_0

☆ • Canonical SOP operation is ORed ~~terms~~ expression of all min terms.

$$f = \Sigma (m_3, m_5, m_6, m_7)$$

or

=

BC
AC
Not min term

Page No. 460
Date

$$f = \Sigma m(3, 5, 6, 7)$$

or

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

Trade off:

* Study_hard
  - Space is also character.

Note: 2's complement:

No of bits required
$n-1$ $n-1$
-2 to $+(2-1)$

* Number system & Code
  - Universal
  - like BCD is a code

* 00 7.33
  - 00 00 00 00 00

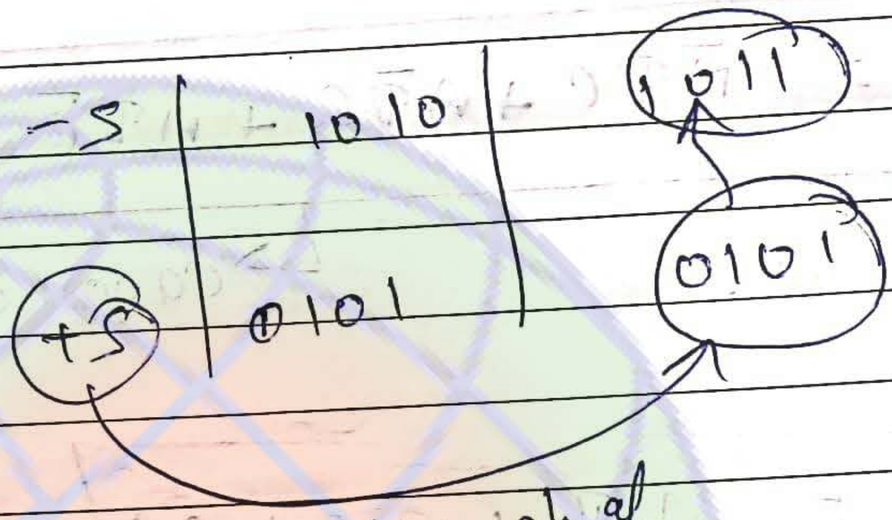
* किसी भी number को convert करने के लिए इसी पहले decimal number system में

T1

Note

X → 2's complement of P = −P

X → 2's complement representation of P = P



f → fn(canonical) → minimal → circuit drawing

A	B	C	f	minterm		maxterm
0	0	0	0			A + B + C — M_0
0	0	1	0			A + B + $\bar{C}$ — M_1
0	1	0	0			A + $\bar{B}$ + C — M_2
0	1	1	1	$\bar{A} \cdot B \cdot C$	m_3	
1	0	0	0			$\bar{A}$ + B + C — M_4
1	0	1	1	$A \cdot \bar{B} \cdot C$	m_5	
1	1	0	1	$A \cdot B \cdot \bar{C}$	m_6	
1	1	1	1	$A \cdot B \cdot C$	m_7	

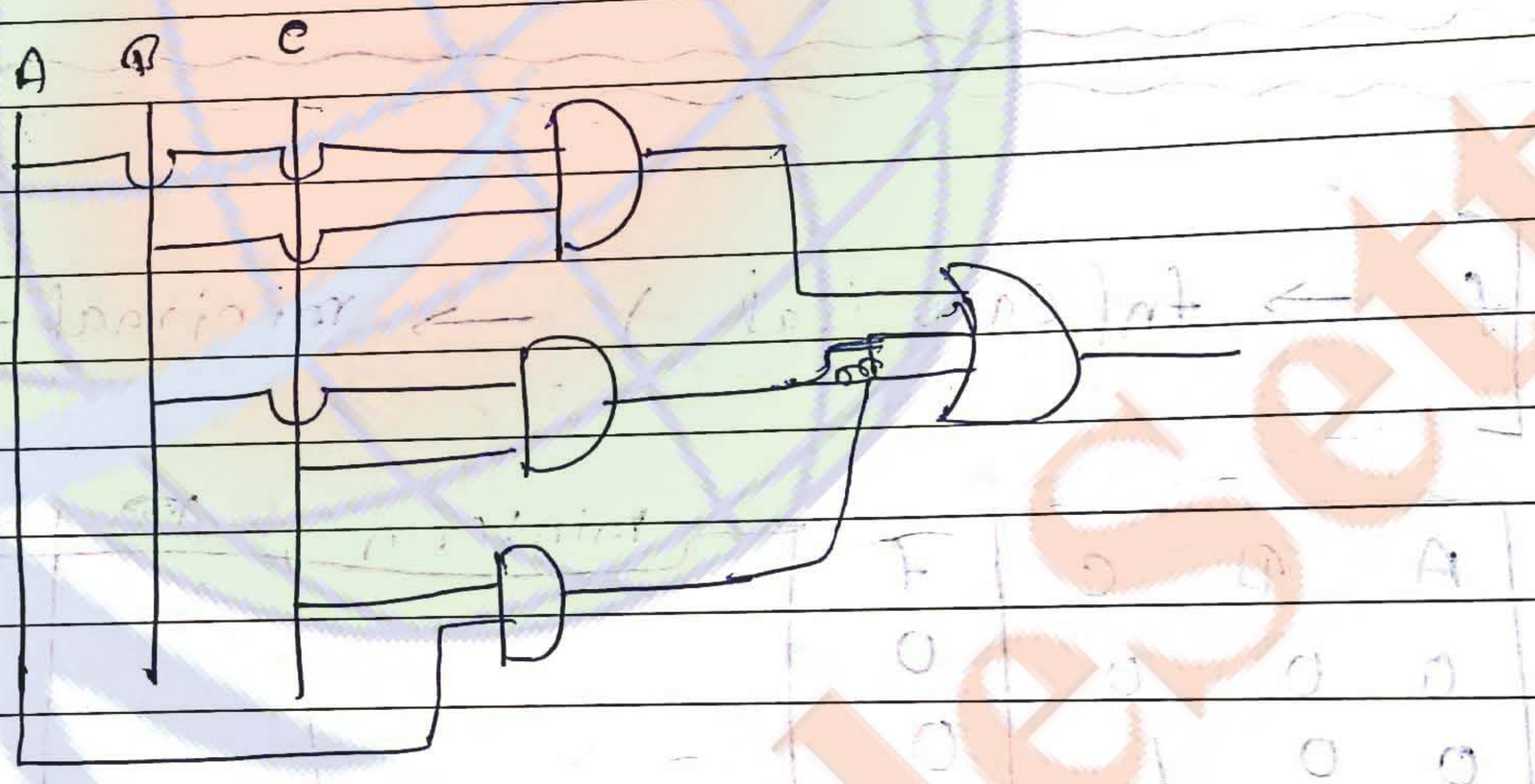
$f = \Sigma\{m_3, m_5, m_6, m_7\}$

$= \Sigma m\{3, 5, 6, 7\}$

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

↳ canonical/SOP

$f = A\cdot B + B\cdot C + CA$



Max-terms =

It is ORed term of n-variable for which function value is zero

eg. In a function f of [x, y, z]

$\bar{x}+\bar{y}+\bar{z}$, $x+\bar{y}+z$ etc are max-term

while

$\bar{x}+\bar{y}$ is not a max term.

* max-term are represented by M_n

where n is decimal equivalent of xyz

Note:

In the writing of max-term, variables which are low/zero in value are considered unprime and variables having value are represented by prime variable. ✓

↳ 0 → unprime → $\bar{A}$

1 → Prime → A

$\therefore f = \pi\{m_0, m_1, m_2, m_4\}$

$= \pi m\{0, 1, 2, 4\}$

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

Canonical POS expression is boolean function which is ANDed expression of all the max-terms.

Hence boolean function's in canonical form's may acquire two format's:-

(i) It is ORed expression of all min-terms (SOP form)

(ii) It is ANDed expression of all the max terms (POS form)

Note min-term में work करने में familier हूँ
⇓
पर Question solve करने के लिए होगा
"max-term" ही use करें।।

always try to prefer max-term to solve the question.

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

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

$\rightarrow AB + BC + CA$

distribut absorption

$\therefore x+yz = (x+y)(x+z)$

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

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

$= AC + AB + BC + AB$

$= AB + BC + CA$ ← minimal form में (SOP) है

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

$= (A+B)\ (B+C)\ (C+A)$ ← minimal form in (POS)

⇒ No. of literals for minimal expression is smallest.

2.4) Minimization of boolean functions

Minimization of ~~expressions~~ functions

(1) K-map (Karnaugh map)

Not Canonical	Canonical
$ABC + AD$	$ABC + \bar{A}BC$
$AD(C+1)$	$\downarrow$
$= AD$	BC

Note: minimisation ≥ minimisation is the extreme simplification
↳ Diff b/w

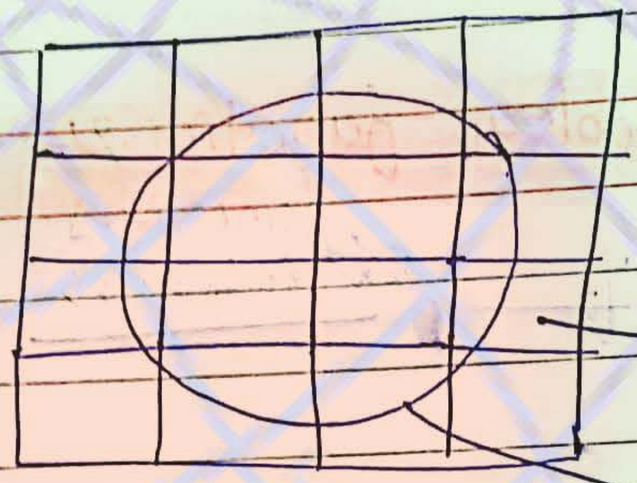
- Graphical method to minimize boolean function.
- K-map is a tool only for minimize

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

we have to group together

$AB(C+\bar{C})$ } ⇒ Group बनाने के तरीका "Grouping"

K-map:



2 → ✓
4 → ✓
8 → ✓
16 → ✓

Properties of K-map:

(I) cell $\rightarrow 2^n$

variable = 4

No of cell = $2^4 = 16$

(II) cell → Rows and Columns

4 rows and 4 columns

(iii) Rows and colums are numberd in gray code.
↓
Unit distance code
cycle code

Not Unit distance code.

3 → 011
4 → 100 ✗
↓
because all three are changes here

we requires only one bit difference like in Robot,
↓
in gray code

If
3 → 010
then
4 → 0110

(iv) f = {A, B, C, D}
A → MSB, D → LSB

Rows will always constitute, with MSB
&
colums " " " LSB

Binary		Gray
0 → 00	→	00
1 → 01	→	01
2 → 10	→	11
3 → 11	→	10

max: $A+B$
min: $\bar{C}\bar{D}$ ... AB

max	min	AB \ CD	00	01	11	10
$A+B$	$\bar{A}\bar{B}$	00	0 (0000)	1 (0001)	3 (0011)	2 (0010)
$A+\bar{B}$	$\bar{A}B$	01	4 (0100)	5 (0101)	7 (0111)	6 (0110)
$\bar{A}+\bar{B}$	AB	11	12 (1100)	13 (1101)	15 (1111)	14 (1110)
$\bar{A}+B$	$A\bar{B}$	10	8 (1000)	9 (1001)	11 (1011)	10 (1010)

- Rows and colums are in Gray code
- But cell's are in Binary.

→ Learn the number which gives easy//

if (0) max-term

$0101 = (A+\bar{B})\cdot(C+\bar{D})$
$= M_5$

if (1) min-term is a product term.
$0101 = \bar{A}\cdot B\cdot\bar{C}\cdot D = m_5$

A	B	C	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$2^n = 2^3 = 8$ cells k-map.

⇒

A \ BC	00	01	11	10
0	0 (0)	0 (1)	1 (3)	0 (2)
1	0 (4)	1 (5)	1 (7)	1 (6)

(Graphical truth table)

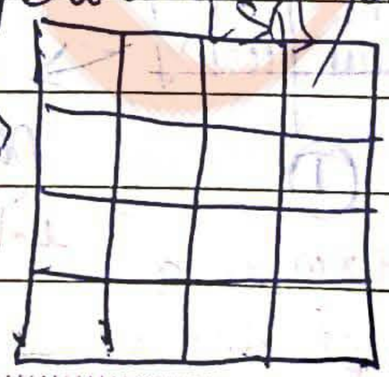
In k-map rows and columns are numbered in Gray code.
but
cells are numberd as decimal equivalent of binary.

* when no. of variables is even, k-map is in square shape.

* when no. of variable is odd has we will keep no. of columns more then no. of rows.
(Not has any fast rule)

* MSB/higher Precedence bits are in rows and
LSB/lower bits is in columns.

*

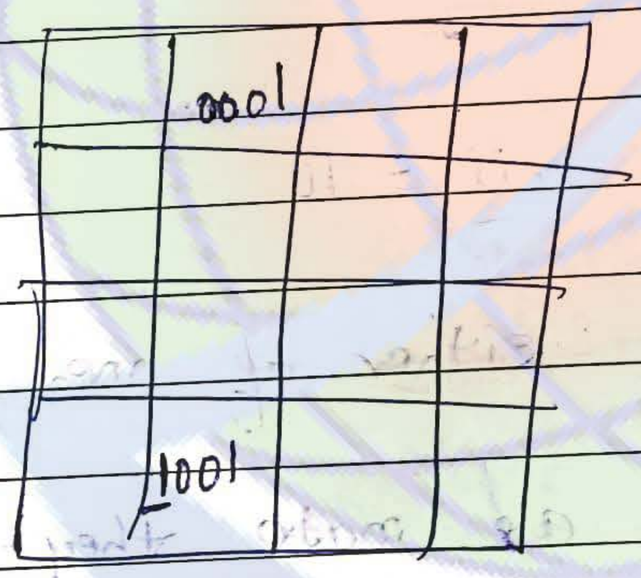


Page No.
Date

#> Df assumptions are made in opposite way that is
higher/MSD in columns and lower/LSD in rows
cell number will change accordingly.

→ for minimization
① we group :-

n-variable ⇒ Group of adjacent cell's are made.
+



cells correspondy binary numbers are differs by one

2) No. of cells in a group can be 2^r.
where r = 1, 2, 3, 4 ~

$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$

3) Adjacent cell meaning:- In K-map two cells are said to be adjacent if their binary representation differs only by one bit.

Ans

(1) n-variable
k-Group

Every group provide, a Product term or a sum term.

(2) No. of literals provides by any group is equal to $n-r$ → No. of literals.

where
n → is no. of variable
r → Group contains 2^r cell.

Eg. Let $n=4$, No. of cell $= 16$

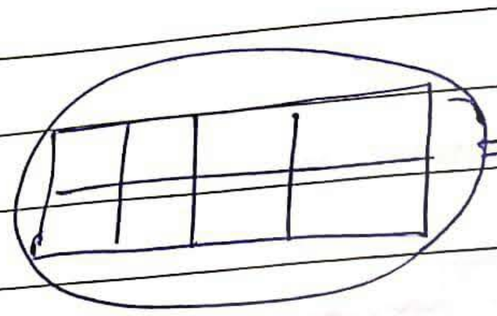
Groups are made either of ones or of zeros.

If groups of ones are made they corresponds to min terms
and
If grop of zero are made they corresponds to max terms.

No. of Literal $= n-r$

	Group (No. of cells) 2^r	No of literals $(n-r)$
$r=0$	1	4
$r=1$	2 (Pair)	3
$r=2$	4 (Quad)	2
$r=3$	8 (octate)	1
$r=4$	16	0

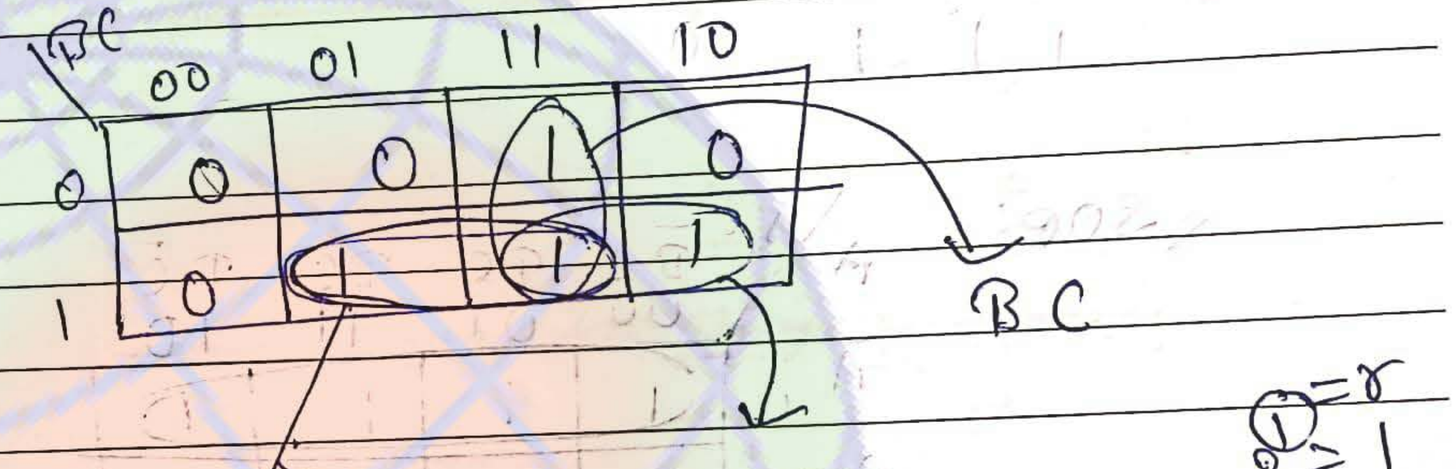
Number of literals $= n-r$
$= 4-4$
$= 0$



No of literals = 4

जादू की छड़ी

* To get SOP expression group of 1's is made and to get POS expression grouping of 0's is made.



SOP :⇒
↳ Group of 1's

Change की possibility है

only write common part

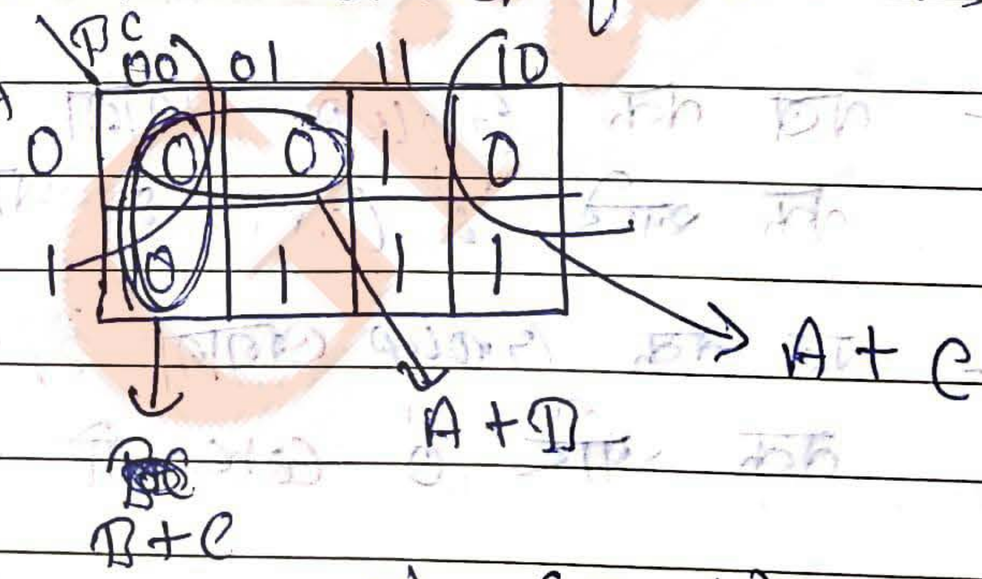
so function of SOP is = AB + BC + CA

verify the formula// n−r = 3−1 = 2

* During grouping always start with largest, although there are few exception

* POS:

↳ For POS Group of Zero's (0) will be made.



so, $f = (A+D)\cdot(B+C)\cdot(C+A) = ADB + DC + CA$

Eg.>

A	B	C	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

SOP:-

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	1	1	1	1
A 1	0	0	0	0

$\Rightarrow \bar{A}$

POS =

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
A 0	1	1	1	1
$\bar{A}$ 1	0	0	0	0

$= \bar{A}$

Note: Here, no. of Group = 1 //

SOP:- तब तक Group बनाता है, जब तक सारे '1' Cover हो जाएँ।

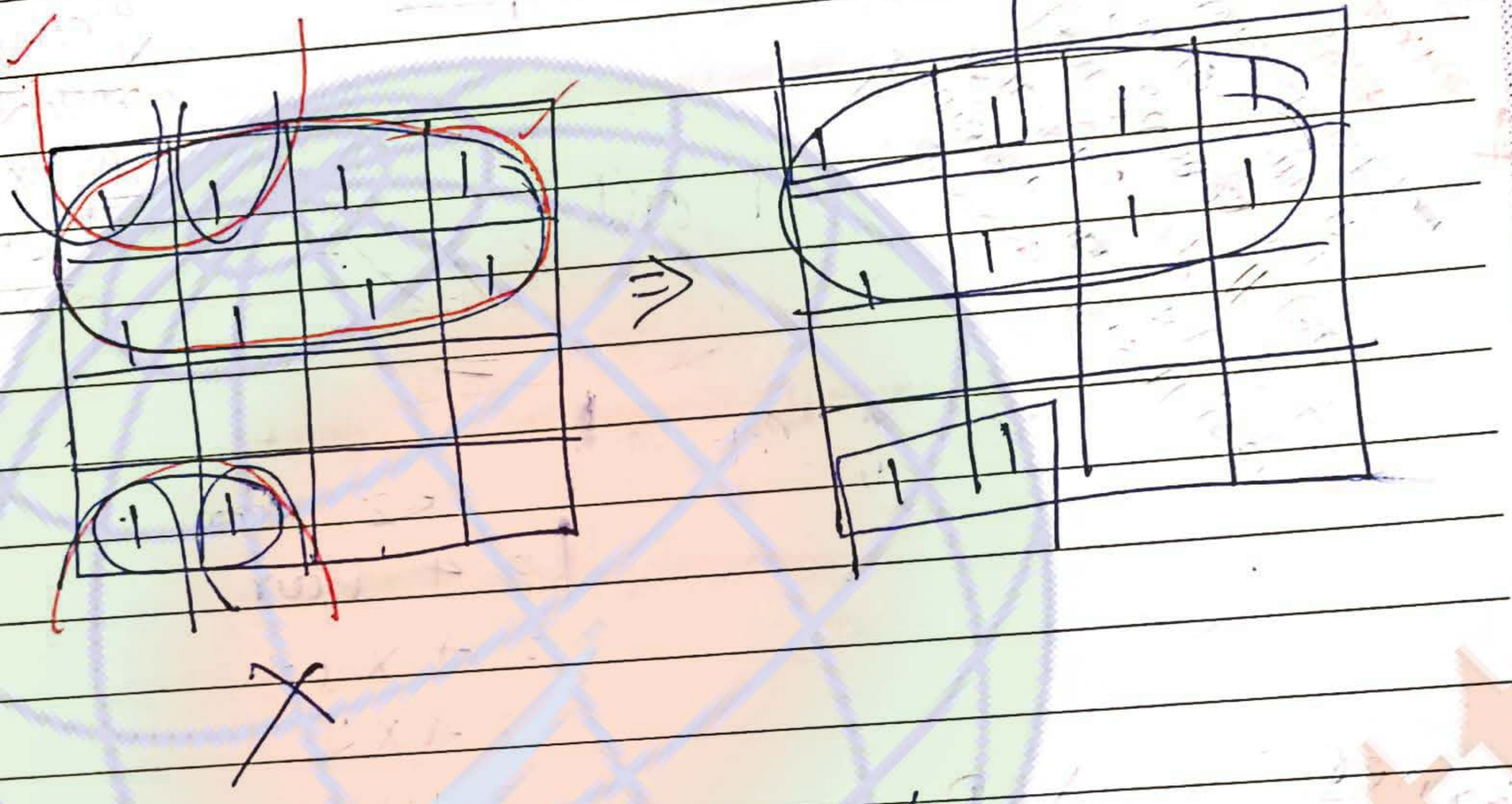
POS:- तब तक Group बनाना है, जब तक सारे '0' Cover हो जाएँ।

Eg.>

Page No.
Date

Note: During grouping; we start with largest possible group.
and grouping continues till all 1's (SOP) or all 0's (POS) are covered.

Eg.>



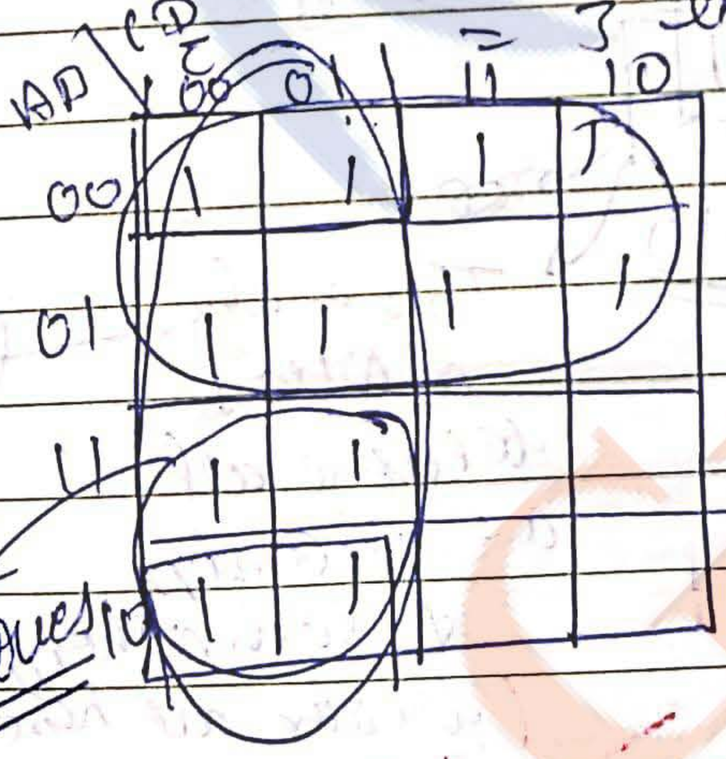
f = 1 octet + 1 Quad

= (4-3) + (4-2)

= 1 + 2

= 3 literals

Eg.>



Not allowed (Start maximum से start करो)

No of literals

= 1 octet + 2 Quad

= 2 octet

= 2 x (4-3)

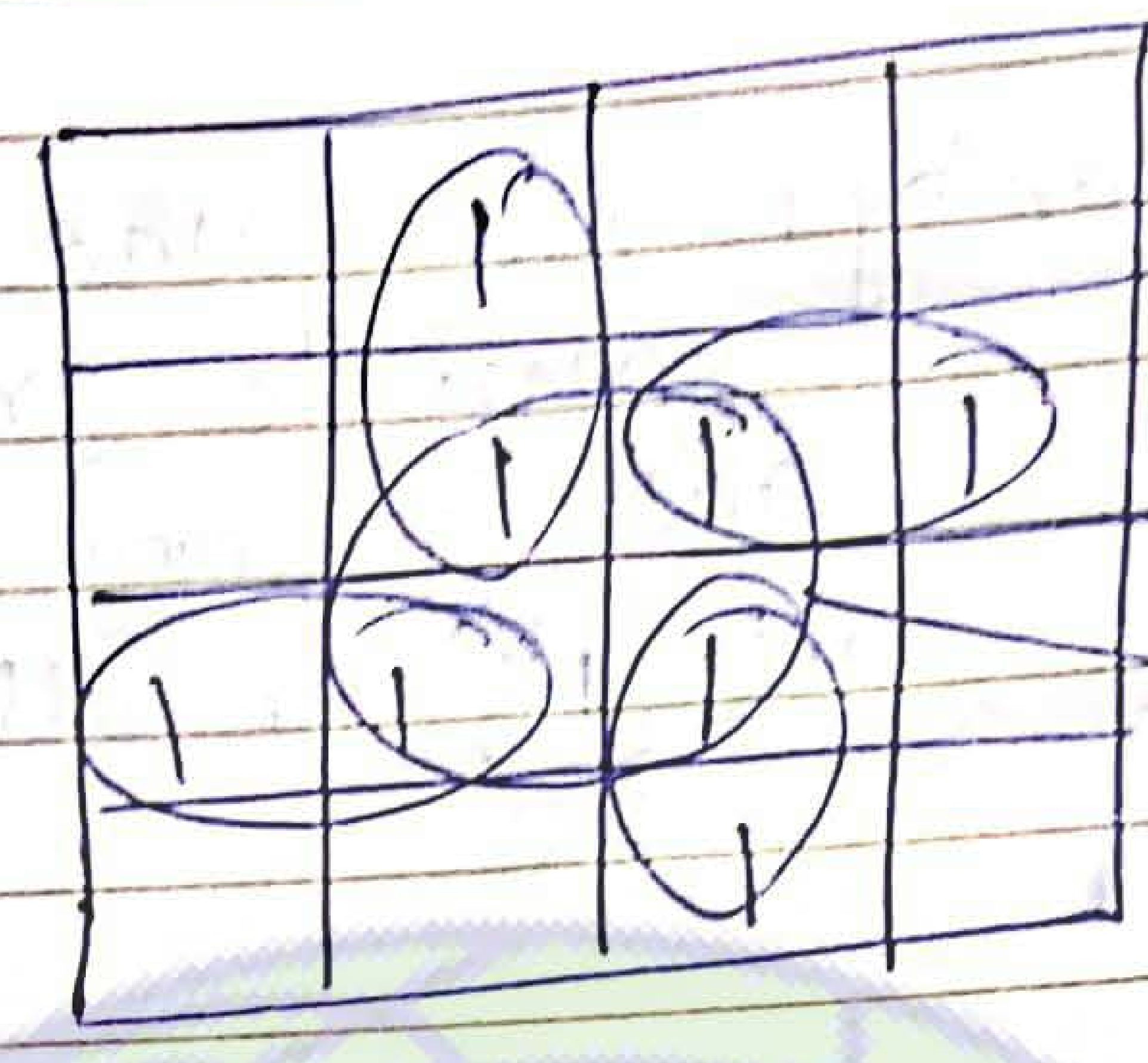
= 2 x 1

= 2

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

R_1, R_2

* Redundant से max remove करें



R_1 → redundant group. (याद रखें max वाले को ही delete करना है॥)

याद रखें: maximum वाला ही delete करना है॥

[illegible]

1 Quad = 4 − 2 = 2

No. of literals = 4 Pair
= 4 × (4 − 1)
= 4 × 3
= 12॥

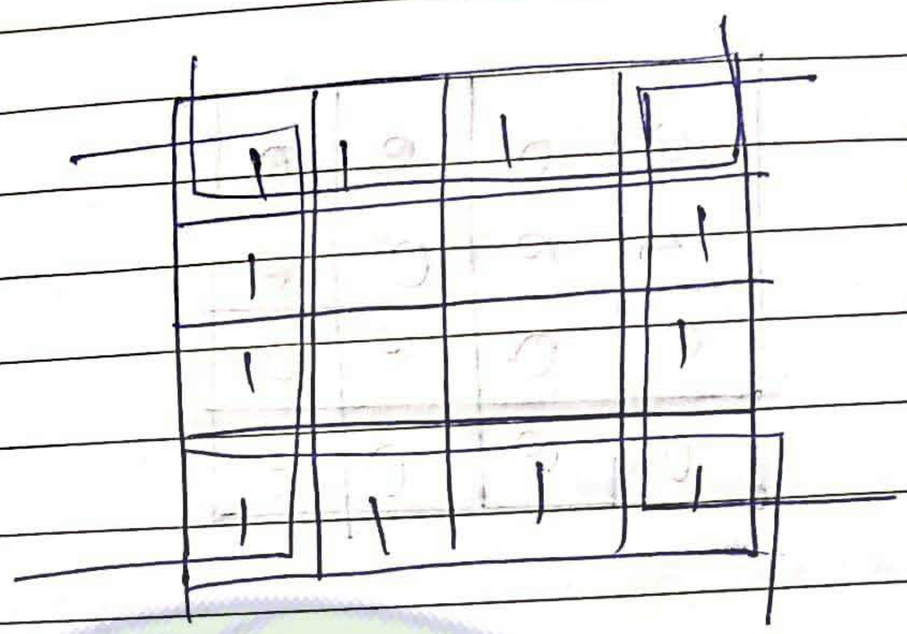
Corner Quad

$\bar{B}\bar{D}$

This is also a Quad because all are adjacent to each other (because all have only one bit difference.) ← R_2

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

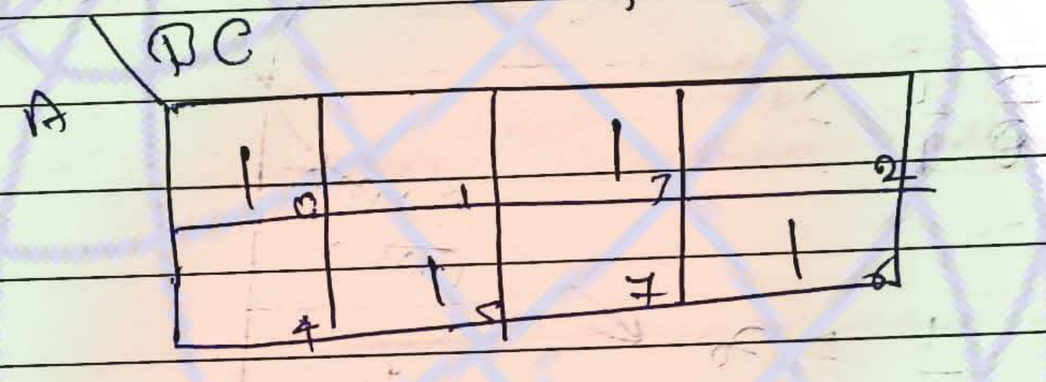
$= B \odot D$



= 2 Octate
= 2 literal

Any K-map set-pattern:

#



$f = [A, B, C]$

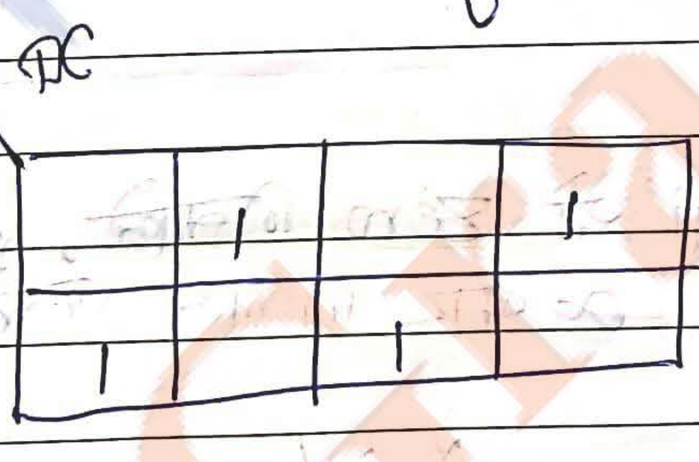
$f = \sum m(0, 3, 5, 6) = A \oplus B \oplus C$

(even detector but not Remainder 2 bit ex-or)

- No. of 1's is even in all the case.

000, 011, 101, 110

#



$f = \sum m\{1, 2, 4, 7\}$

$= 001, 010, 100, 111$ (all odd number of 1's)

$= A \oplus B \oplus C$ (ex-or)

Set Pattern -

e	o	e	o
o	e	o	e
e	o	e	o
o	e	o	e

o → even number
1 → odd number

Note:

$\frac{1}{1} \geq$

$\frac{0}{1} \geq$

$\frac{1}{0} \Rightarrow \infty$

रामानुजन

कितने गुने

Learn it:

⇒ Think
⇒ Contemplate
of
Contemperation ✓

$\frac{10}{10}$ → 10 apple
10 को मिला।
= 1

$\frac{10}{5}$ → 10 apple
5 को दो।
= 2 or
10 apple
5-5 करके
निकालो

$\frac{10}{0}$ 10 में शून्य निकाले, तो ∞ बार निकाल सकते हैं।

$\frac{0}{0} \Rightarrow$ Undefined

$\frac{x}{x} = 1$

$\frac{0}{x} = 0$

$\frac{x}{0} = \infty$

* Interest / hobbies :
  → e.g. should be all knowledge //

IES → IAS → chairman

→ interview carefull chosen
  ↓ करते
  Last
  ↓
  next quest

* Don't care Conditions! -

A	B	e	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x or d
1	1	1	x or d

this is don't care

$$f = \sum m[4,5] + d[6,7]$$

$$f = \pi M(0,1,2,3] + d[6,7]$$

A \ BC	00	01	11	10
0	0	0	0	0
1	1	1	X	X

Don't care conditions corresponds to those values of functions, for which function can be assumes 0 or 1

Don't care conditions are chosen as zero or one which even simplifies the function more.

POS! -

We will assume ✱ cross (X) to be 1.

A \ BC	00	01	11	10
0	0	0	0	0
1	1	1	X	X

$f = A$

(Here $X = 1$ (assumed)

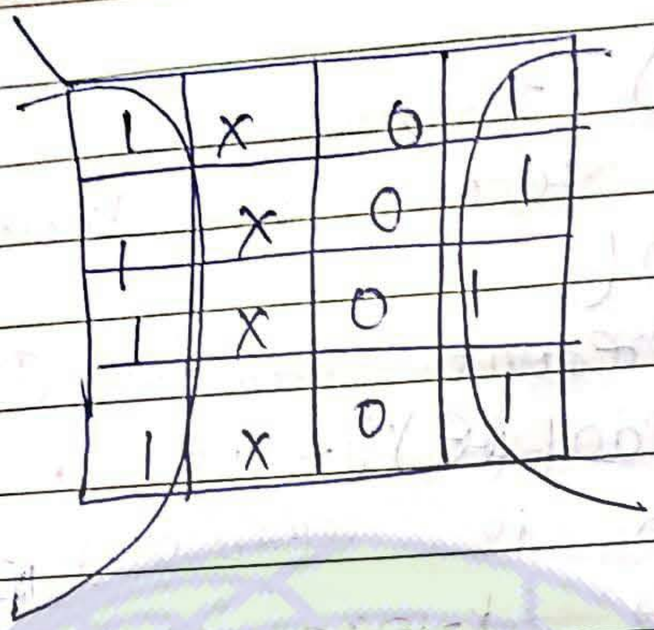
POS! -

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

$f = A$

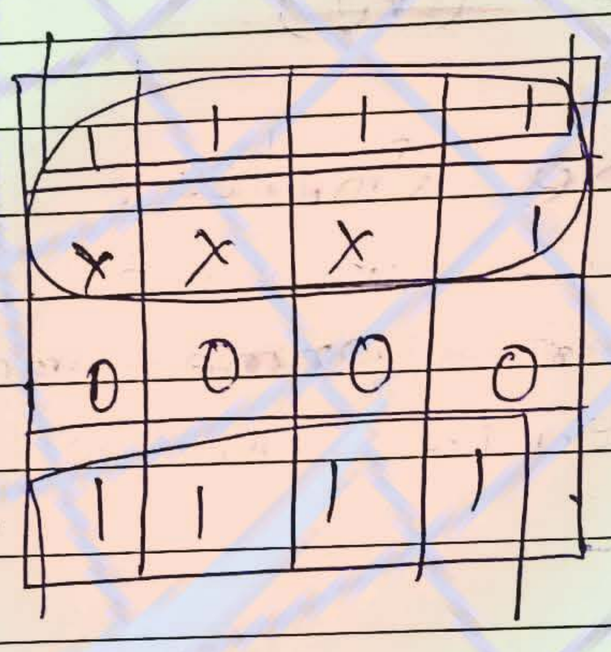
Here also $X = 1$

eg)



Hey take X=0

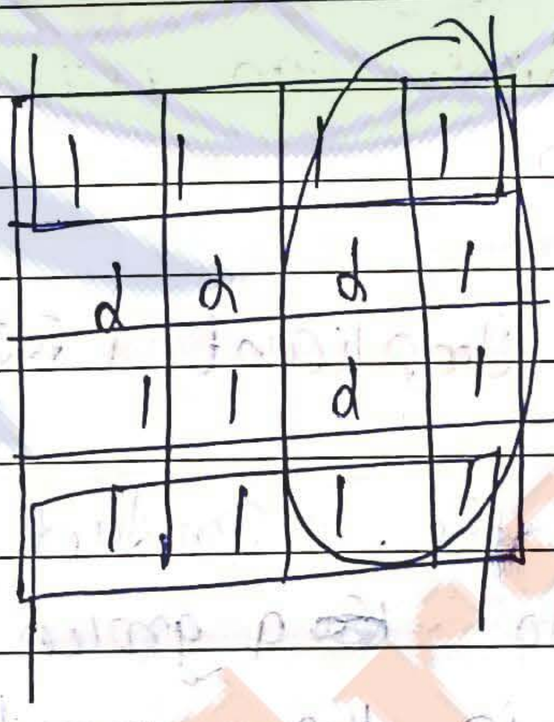
1 octate = 4 literal



Pair → Not prefer because, No. of literal will increase,

Quad →

2 octate = 2 literals



Note: (Don't care may be treates intentionally)

2 octates:

2-6) Implicants, Prime Implicants and essential Prime implicants:-

(1) * Implicants (I):-
each min term is known as Implicant
Represent Capital I.

(2) * Prime Implicants (PI):- It is a Product term which is obtained by combining cell's in the k-map.
Represent (PI)

(3) Essential Prime Implicant (EPI):-
essential prime implicant is the Product term or prime implicant, which always exists in the minimized function
or
EPI correspond to the product term coming from a group, which covers atleast one one(1), which can not be covered by other group.

(4) Redundant Prime Implicant (RPI):-

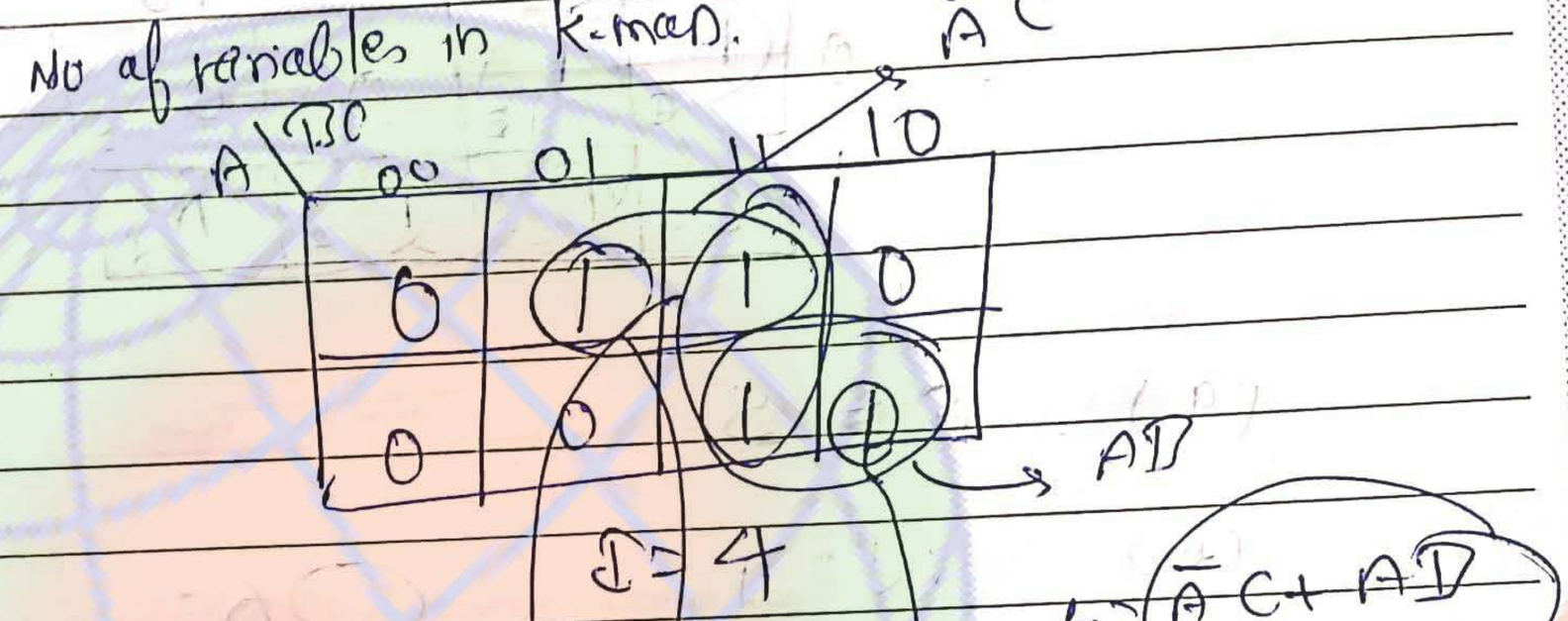
RPI is the Product term, corresponding to a group, which never comes in the most minimised expression of function.
or
RPI is the product term corresponding to group for which all ones of the groups are covered by some other group.

Q1

Ex> $f(A,B,C) = \sum m [1,3,6,7]$

(a) min term = I = 4

No of variables in K-map.



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

(b) Prime - Implicant :-

जितने A बने (no means no. of groups = 3)
(सभी)

PI = 3

(c) essential PI = 2
(actual)
(जितने group बनेगा)

* जो actual वाले में आ रहा है = EPI
जो actual वाले में नहीं आ रहा है = RPI

(d) RPI = 1

Q (e) Non - EPI :- 0

↳ EPI भी नहीं, RPI भी नहीं

eg. minimizes expression for

$f[A,B,C] = \Sigma m[0,1,5,6,7]$

find I, PI, EPI, RPI, ...

Sol^n

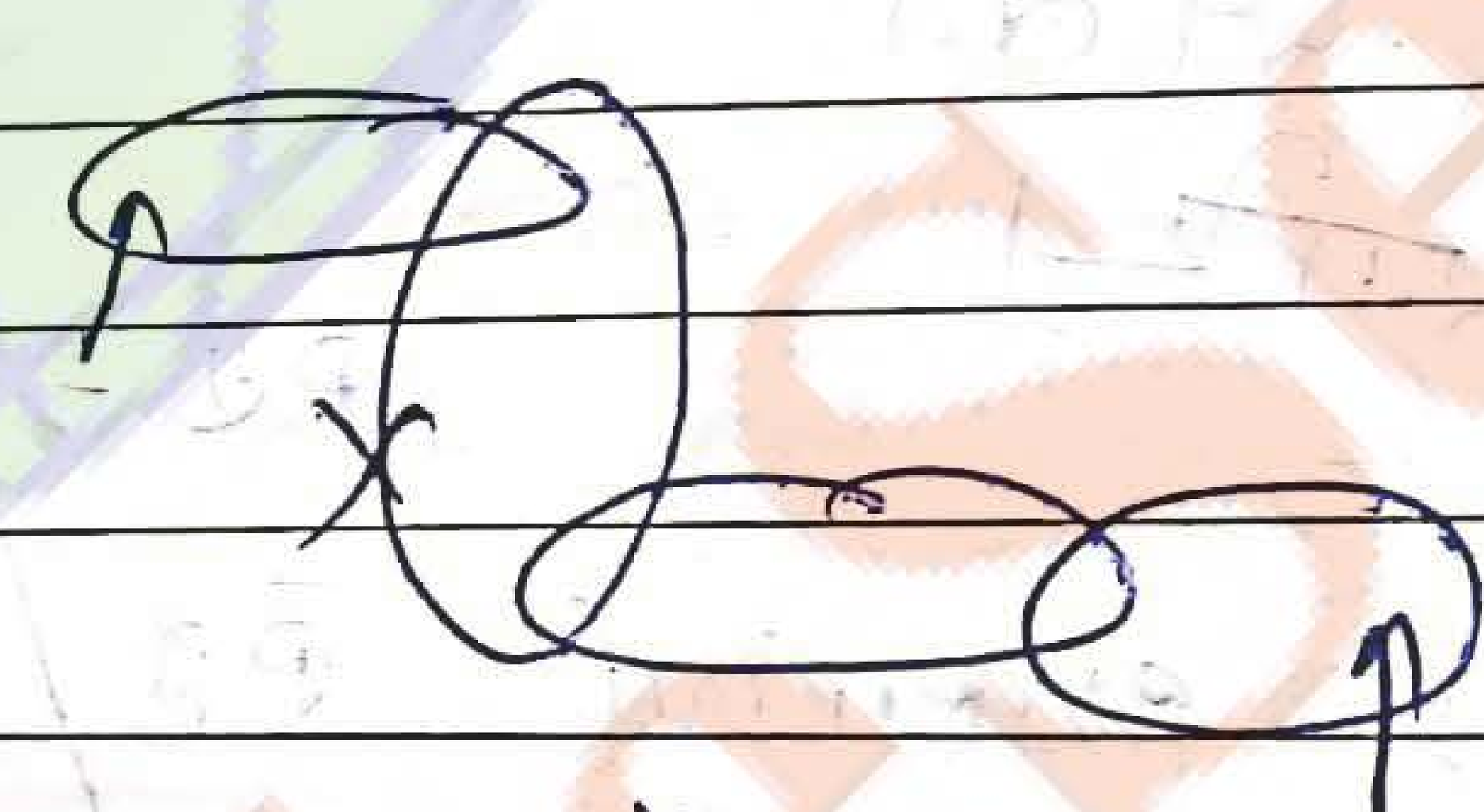
A \ BC	00	01	11	10
0	1 (0)	1 (1)	0 (3)	0 (2)
1	0 (4)	1 (5)	1 (7)	1 (6)

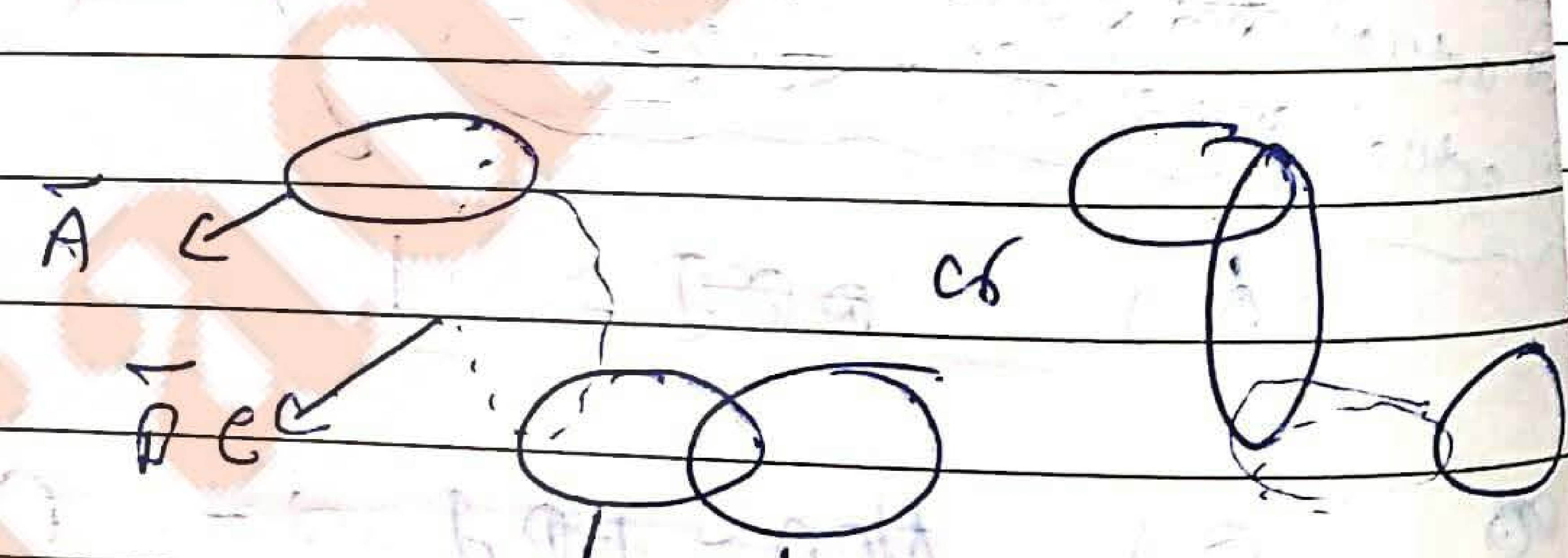
(a) I = 5

(b) PI = 4

(c) EPI = 2

जो final sol^n में आएं ||





Note: अगर अलग-अलग case possible हो तो जो दोनों में essential है अर्थात जो दोनों में common हो, बस उसे ही लेना है || essential = most जरूरी

$f = \bar{A}\bar{B} + AC + AB$

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

EPI = 2

जो essential है

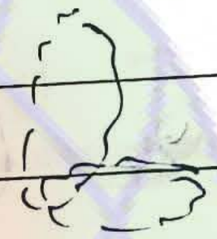
(d) RPI = 0

(जो कभी भी ना आएँ।)

कभी को एक, कभी ना कभी ज्यादा या एक हो

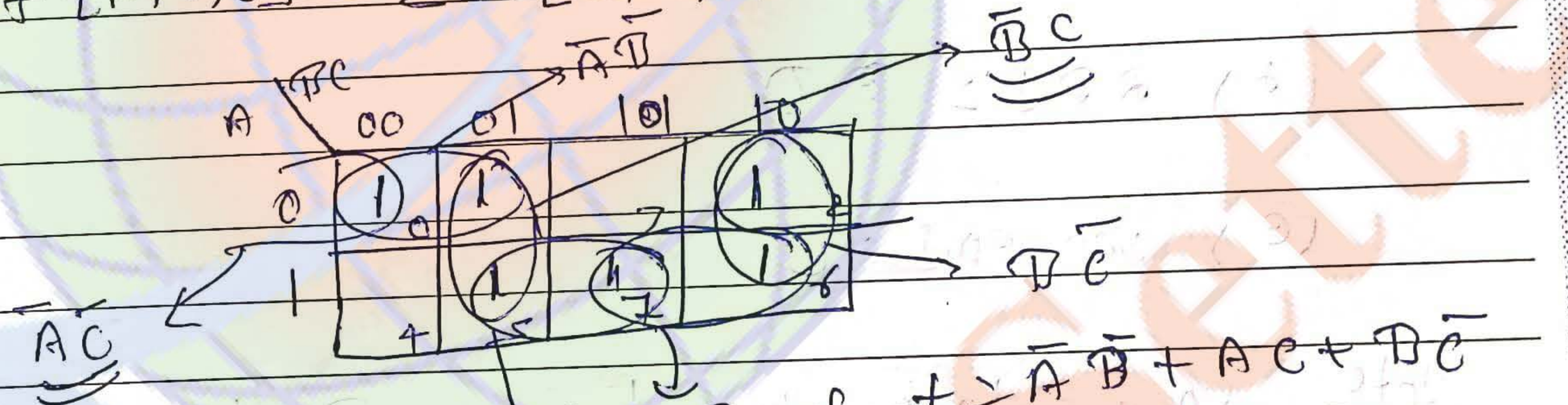
(e) Non-EPI = ?

↳ जो EPI में नहीं, जो RPI में नहीं



Q.) $f[A,B,C] = \sum m[0,1,2,5,6,7]$

Sol



(a) EPI = 6

(b) PI > 6

(c) EPI = 0 (1 1)

(that always appear)

(d) RPI = 0 (1 1)

(जो कभी भी ना आए)

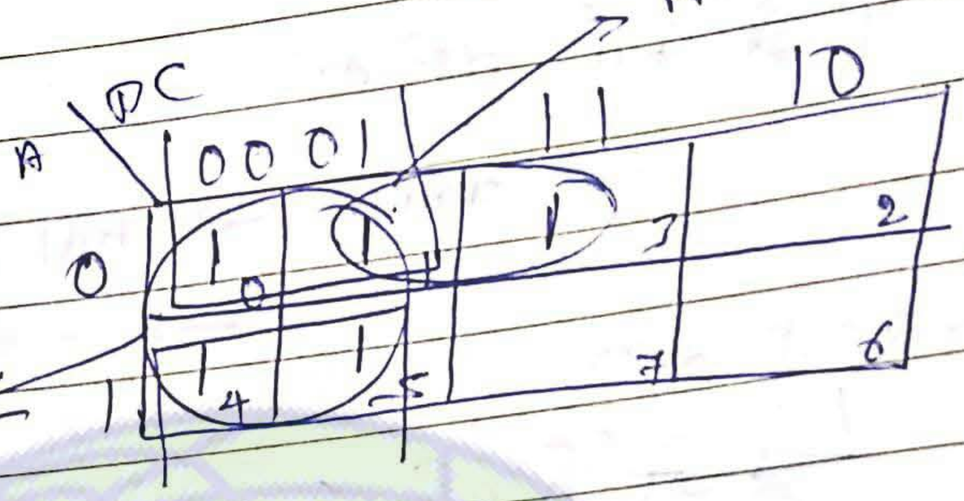
(e) Non-EPI = 6

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

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

(1 1)

Q) $f(A,B,C) = \sum m(0,1,2,4,5)$



(a) EPI = 2

(b) PI = 2 (Group Prime implicant)

(c) EPI = 2

(d) RPI = 0

(e) Non-EPI = 0

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

Note short-trick to solve the Problems:—

(1) EPI:- सबसे पहले $\sum m($ ↑ $)$
जितने number होंगे वही EPI होगा।

(2) PI: K-map बनाएँ, और इसके बाद maximum ~~group~~ ~~of~~ number से group बनाएँ और ~~for~~ possible group बनाएँ, साथ और जितने number of group होंगे वो PI होंगे।

I, ← अगर 666 हो तब =)

(3) EPI :- Actual/exact में जितने group होंगे, अर्थात जो final solution में आएंगे वो EPI होंगे।

↳ इसका मतलब जो हमेशा आएगा सभी solution में आएगा वही EPI होगा।

(4) RPI :- जो Group कभी भी ना आएँ।

(5) Non-EPI :- जो कभी आएँ, और कभी नहीं आएँ।

eg.

A \ B	0	1
0	1	0
1	1	$\bar{C}$ → can complement

Find f[A, B, C] => ?

What is f — directly पता कर लो।

Ans: So expand it into full K-map having eight cells.

A	B	C	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

$$f = \bar{B} + \overline{AC}$$ Ans.

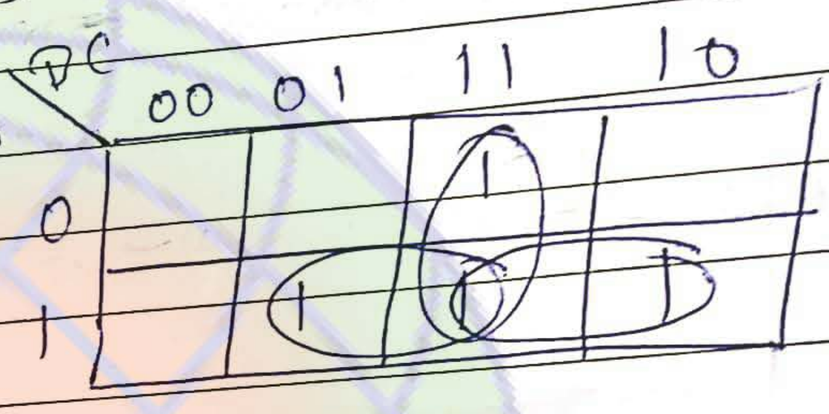
Page No. 490
Date

Q) f is a function of three variables A, B, C such that f is high when majority number of inputs are high. find f.

Sol:

$f(A, B, C) =$

A	B	C	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

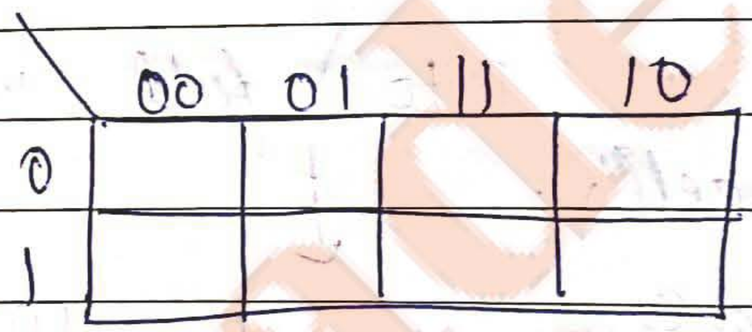


$f = AC + \bar{B}C + AB$

Q) f is a high when majority numbers of inputs are zero, then what is f.

Sol:-

A	B	C	f
0	0	0	1
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	0



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

or

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

$\because A + \bar{A} = 1$

Page No. 49_7
Date

$f = \overline{AD + DC + CA}$ (using demorgans law)

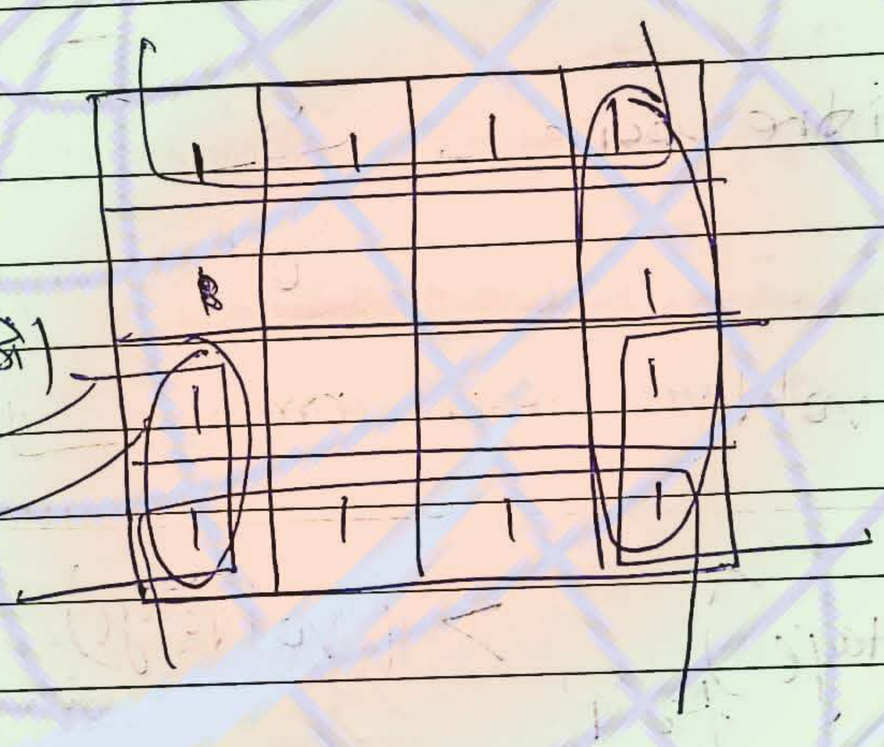
$= \overline{A}\,\overline{D} \cdot \overline{D}\,\overline{C} \cdot \overline{C}\,\overline{A}$

$= (\overline{A} + D) \cdot (\overline{D} + C) \cdot (C + \overline{A})$

$= (\overline{D} + \overline{A}C) \cdot (\overline{A} + C)$

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

Q.)



(quad choose करो)

X

Note

$I = 11$

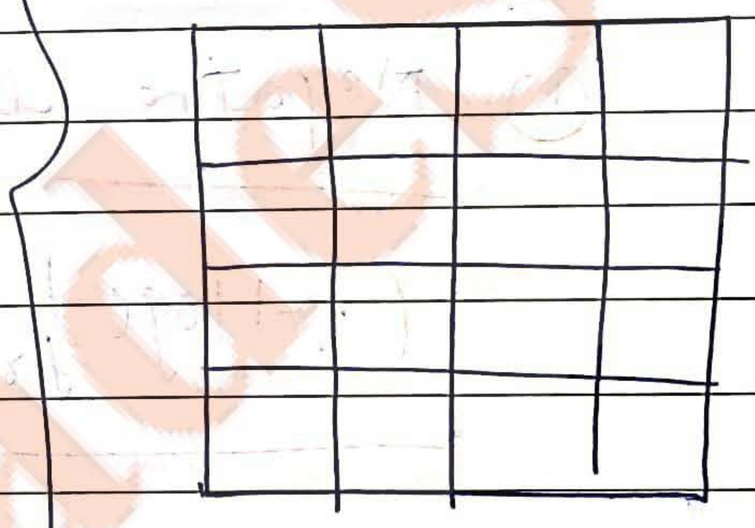
$PI = 3$

$EPI = 3$

$RPI = 0$

$NON-EPI = 0$

Note:



Page No. 4
Date

Imp for gate

2.7.) Duality :-

In Digital electronics we have two kinds of logic

① -ve logic

② +ve logic

② positive logic :- '1', '0' voltage level corresponding

$$(\text{voltage})_{\text{for '1'}} > (\text{voltage})_{\text{for '0'}}$$

① Negative logic :-

$$(\text{voltage})_{\text{for '0'}} > (\text{voltage})_{\text{for '1'}}$$

eg.> +ve logic

$\checkmark$(a) 1 → +5 V
0 → 0 V

$\checkmark$(b) 1 → 0 V
0 → -5 V

} Generally +ve is used

eg: -ve logic

-ve (A) 1 → 0V
0 → +5V
} generally ✓ use it

-ve (B) 1 → −10 V
0 → −5 V

AND

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

OR

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

* Let Any AND gate is designed and working in the logic.

1 → +5V
0 → 0V
} the

0 → 0V
0 → +5V
} -ve

Then truth table of AND gate is

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

Truth table in terms of voltage

A	B	Y
0 V	0 V	0 V
0 V	5 V	5 V
5 V	0 V	5 V
5 V	5 V	5 V

Now Let the AND is working in the logic

0 V → 1

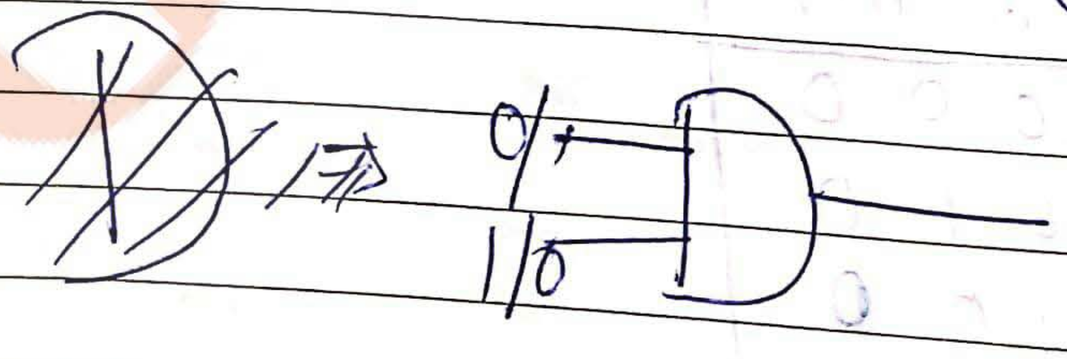
5 V → 0

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

⇒ Rearrangment

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

Thus it is truth table of OR gate.//



R1

Note:

0·0 = 0 (OR: 0, 0 → 0)

1·1 = 1 (OR: 1, 1 → 1)

0·1 = 0 (OR: 0, 1 → 0)

1·0 = 1 (OR: 1, 0 → 1)

1·0 = 0 (OR: 1, 0 → 0)

0·1 = 1 (OR: 0, 1 → 1)

1·1 = 1 (OR: 1, 1 → 1)

0·0 = 0 (OR: 0, 0 → 0)

* Hence the logic AND gate works like an OR gate in the logic, similarly OR gate of the logic becomes AND gate in the logic, this property is called duality.

Duality is the change of behavior of any digital circuit when it's environment is changed from the logic to the and vice versa.

Eg: Duality:

AND ⇌ OR

1 ⇌ 0

R1 { X ⇌ X

$\bar{X} \rightleftharpoons \bar{X}$

R1 R1

R1 { NOT ⇌ NOT

NAND ⇌ NOR

EX-OR ⇌ EX-NOR

* Step's to convert or write Dual of any expression

(1) Change all OR into AND and all AND into OR.

(2) Convert

0 ⇌ 1

(3) keep variable as they are.

eg (i) A + BC ⟶ Dual → A · (B + C)

(ii) A + B + CD ⟶ Dual of it = A · B · (C + D)

R1 (iii) A · 1 ⟶ Dual → A + 0
= A

(iv) X(Y + Z) = XY + XZ } this will be true in both the and

Note: f = X(Y + Z)

↓

this is function

R1

$x(y+z) = xy + xz$ ✓

$x + yz = (x+y)\cdot(x+z)$ ✓ both is true//

If one will true then it's dual will also be true.

* Self Dual expression

If dual of any expression leads to the same expression, this kind of expression is called self dual.

eg.

$AB + BC + CA$

↓

Dual of this is

↓

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

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

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

$= AB + BC + CA$ → self dual.

* For n-variables, maximum number of self dual expression is equal to

R1 → $2^{2^{n-1}}$

eg $n=1$

No. of self dual is $A, \bar{A}$

$A \leftrightarrow \bar{A}$

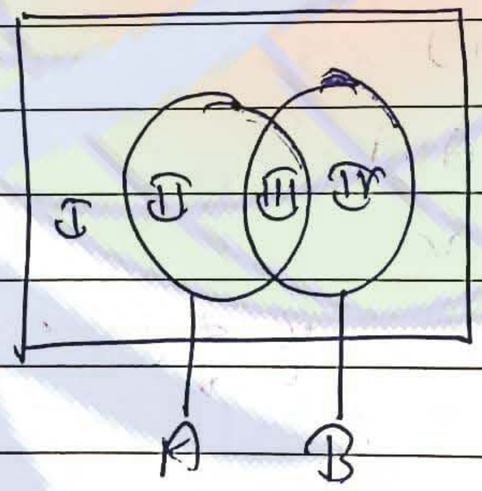
$\bar{A} \leftrightarrow A$

If $n = 2$

$A, \bar{A}, B, \bar{B}$

If we take two times dual it results in same expression

2.8 * Venn Diagram



I → $\bar{A}$ and $\bar{B}$

→ $\bar{A}\,\bar{B}$

II → A and $\bar{B}$

→ $A \cdot \bar{B}$

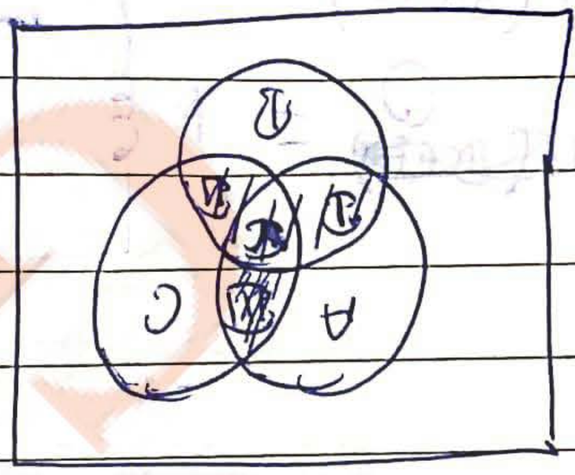
III → A and B

→ $A \cdot B$

IV → $\bar{A}$ and B

→ $\bar{A} \cdot B$

Q.7>



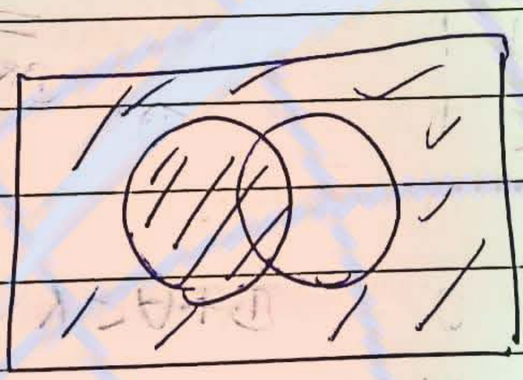
$f = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}C + AB\overline{C}$

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

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

$= A \oplus B \oplus C$

Q.8>



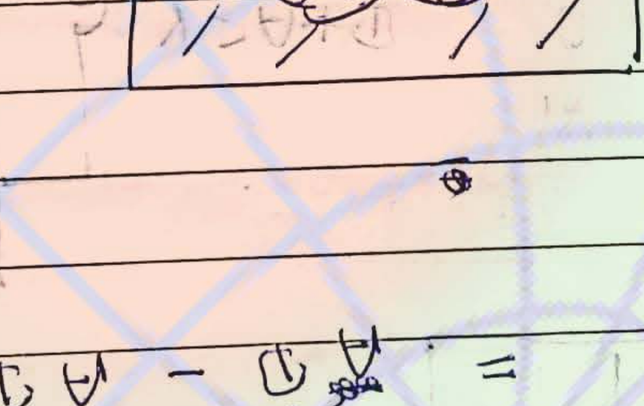
$f = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

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

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

$= A \oplus B \oplus C + ABC$

Q.9>



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

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

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

$= \overline{A \oplus C}$

$= A\overline{B}\overline{C} + \overline{A}BC + AB\overline{C} + ABC$

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

$= A\overline{B} + AB$

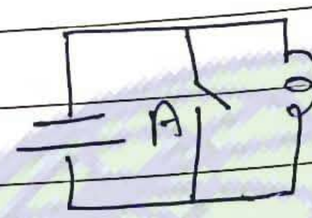
$= A(\overline{B} + B)$

$= A$

3.> Logic gates and switching circuit

3.0 Switching Circuits :-

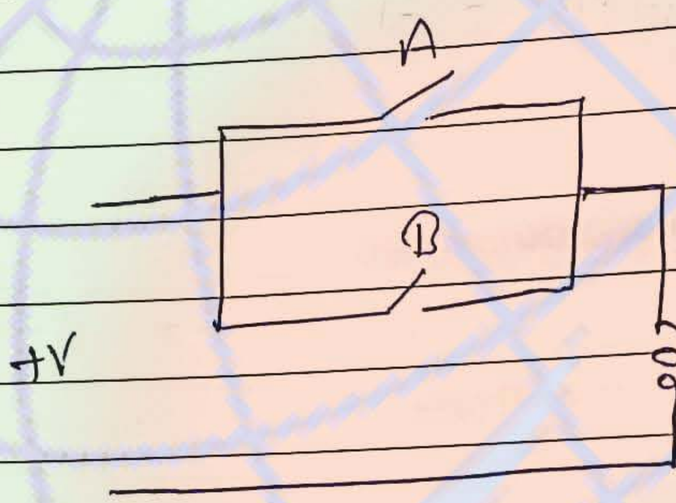
(1)



$y = \bar{A}$ (NOT)

A	y
0	1
1	0

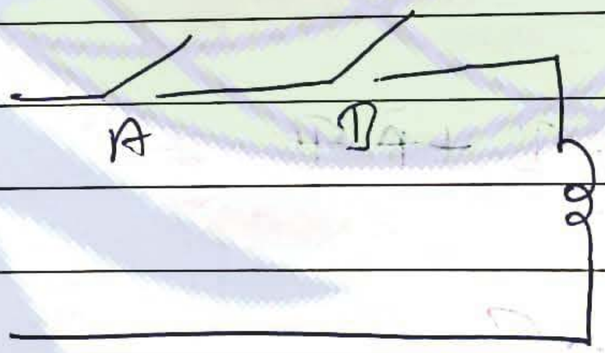
(2)



$y = A + B$

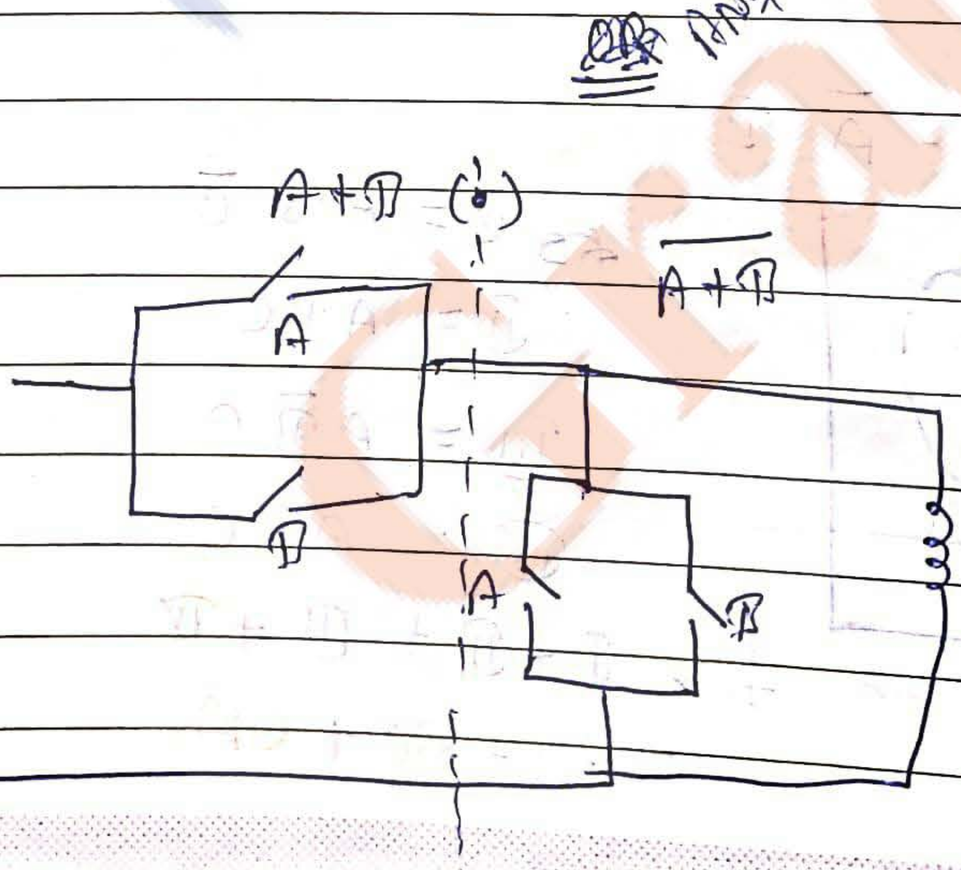
A	B	y
0	0	0
0	1	1
1	0	1
1	1	1

(3)



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

eg.>



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

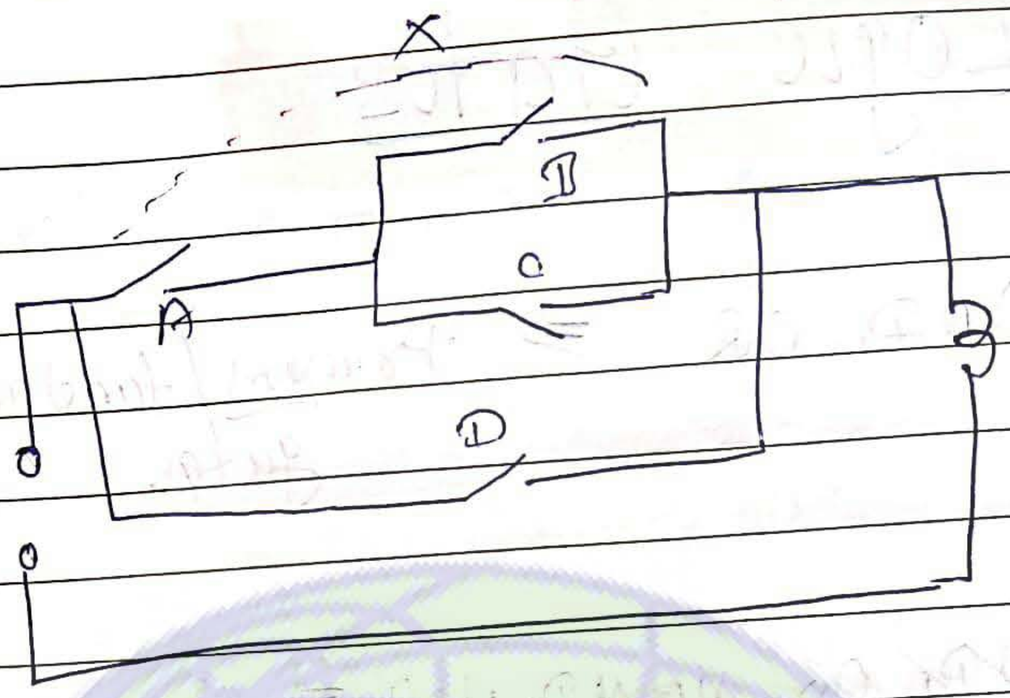
$= (A+B)(\overline{A+B})$

$= x \cdot \bar{x}$

$= 0$

Page No. 502
Date

eg.



$Y = D + X$

$= D + [A \cdot (B + C)]$

* Statements based questions-

(1) A logic circuit has three inputs A, B, C and output D is f, f is logic 1 for following combination.

(a) A and C are true
(b) A and B are false ← अगर simply बोलेंगा तो यह individual के लिए होगा
(c) A, B and C are true
(d) A, B and C are false

Ans. OP ✓

(a) $f \rightarrow A \cdot C$

(b) $f \rightarrow \bar{A} \cdot \bar{B}$

(c) $f \rightarrow A\bar{B}C$

(d) $f \rightarrow \bar{A}\bar{B}\bar{C}$

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

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

Note
$\overline{AB}$ → A and B are true is not true
or
$\bar{A}\bar{B}$ → A and B are true simultaneously is not true } AB

R1 21-5-17

Page No. 502
Date

☆ 3.1 Logic Gates

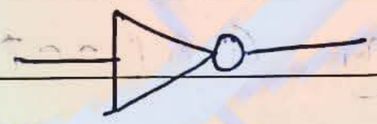
① NOT, AND, OR ⇒ Primary / fundamental gates.

② NOT, AND, OR NAND, NOR ⇒ Universal gates

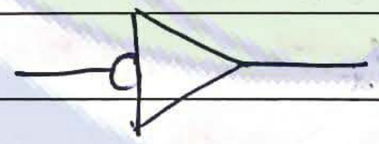
③ EX-OR, EX-NOR ⇒ For mathematical calculation (Parity generation)

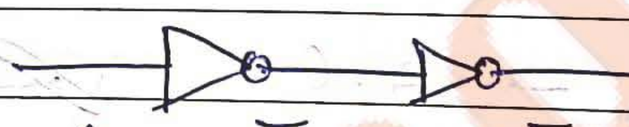
① NOT

A	y
0	1
1	0



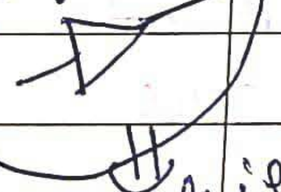
or





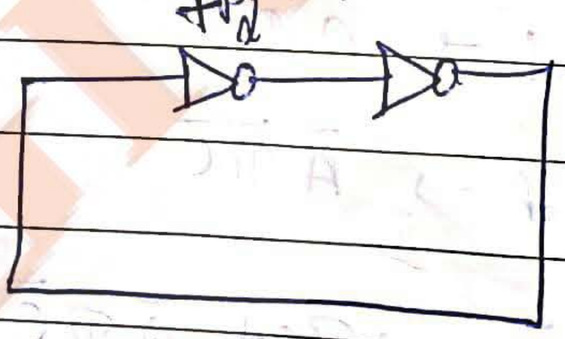
(this is also buffer)

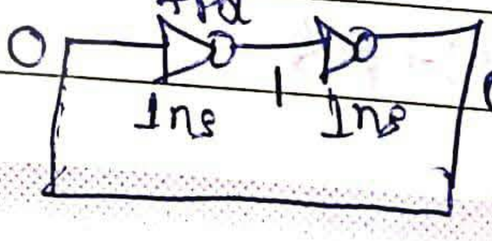
Note
Buffer



output is same as input.
use: Delay
(but after some time)

t_{Pd} → Propogation delay



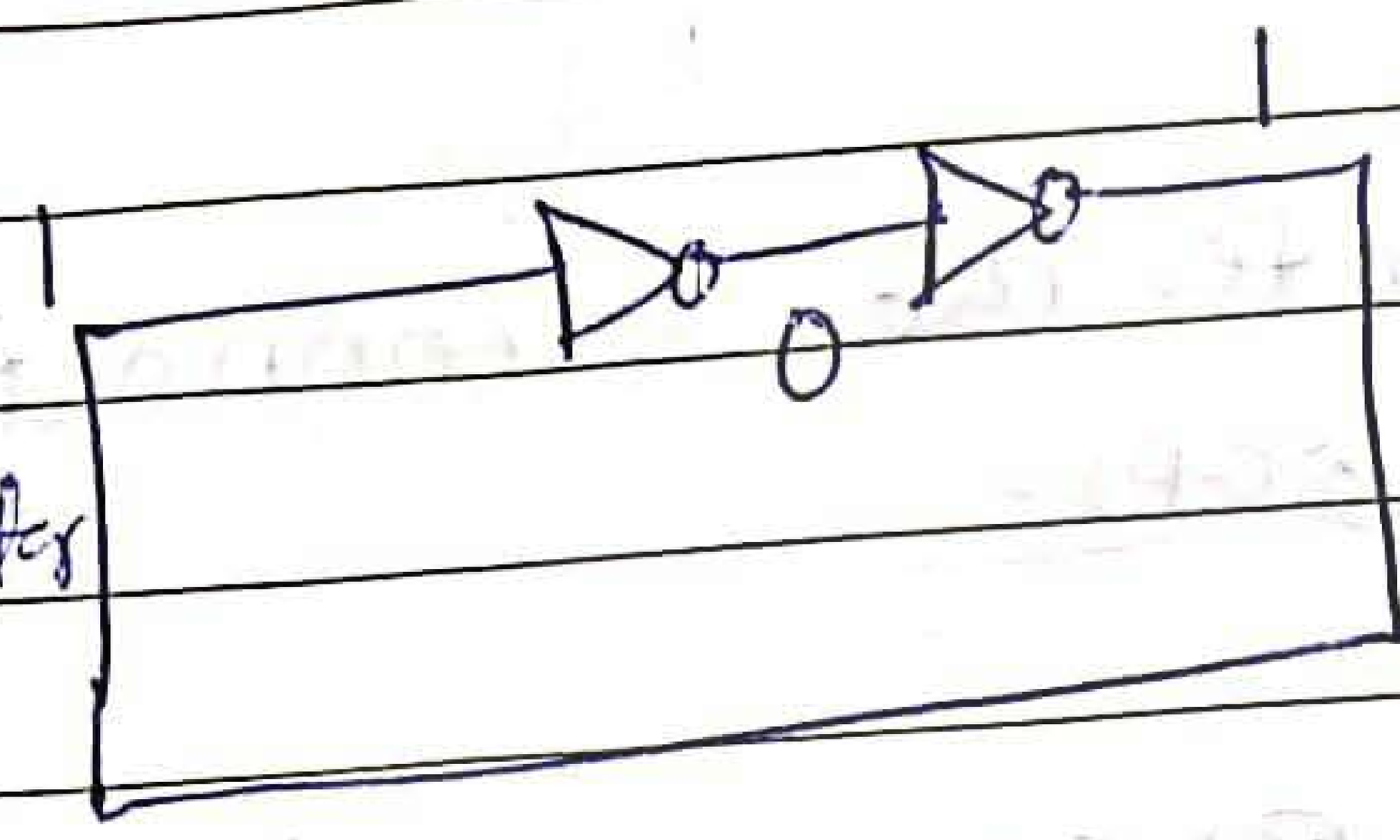


we stored '0' here

R, -21-5-17

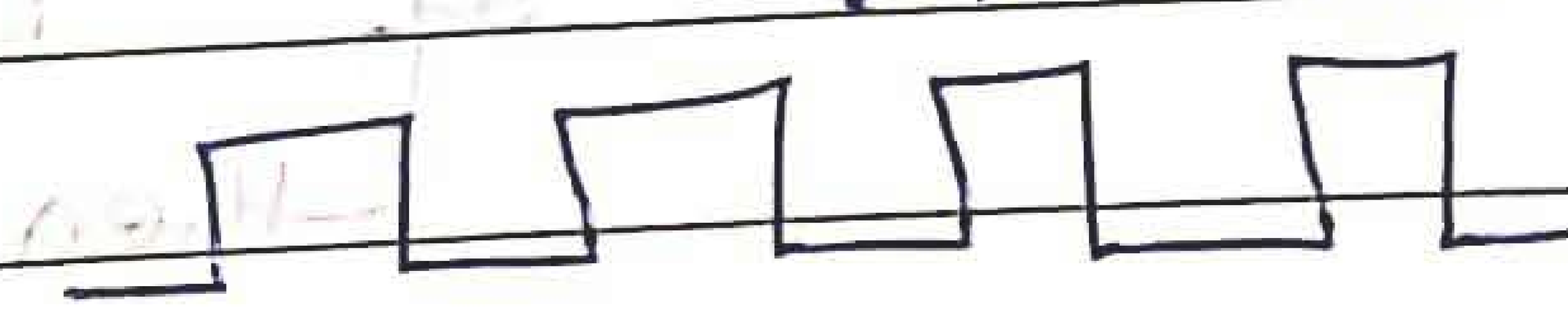
Page No. ____
Date ____

Bistable multivibrator

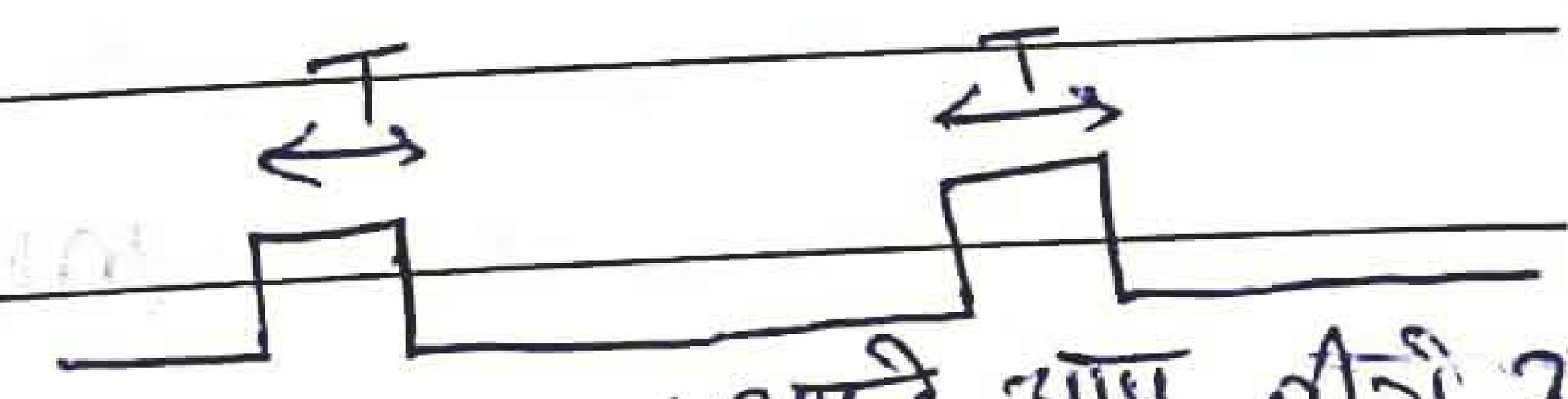


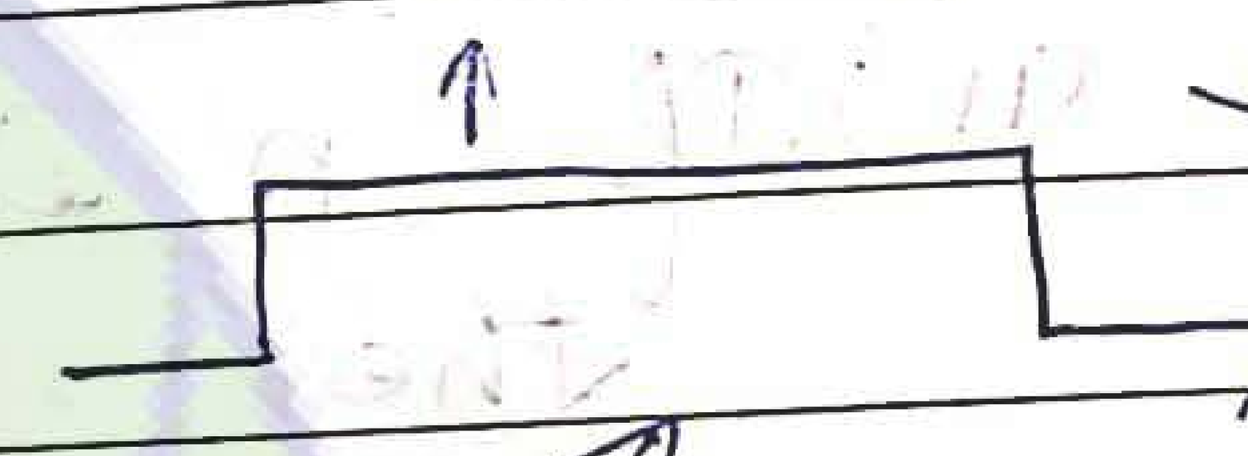
1 — — — — —
— — — —

(we stores 1 here) stable कोई भी नहीं है

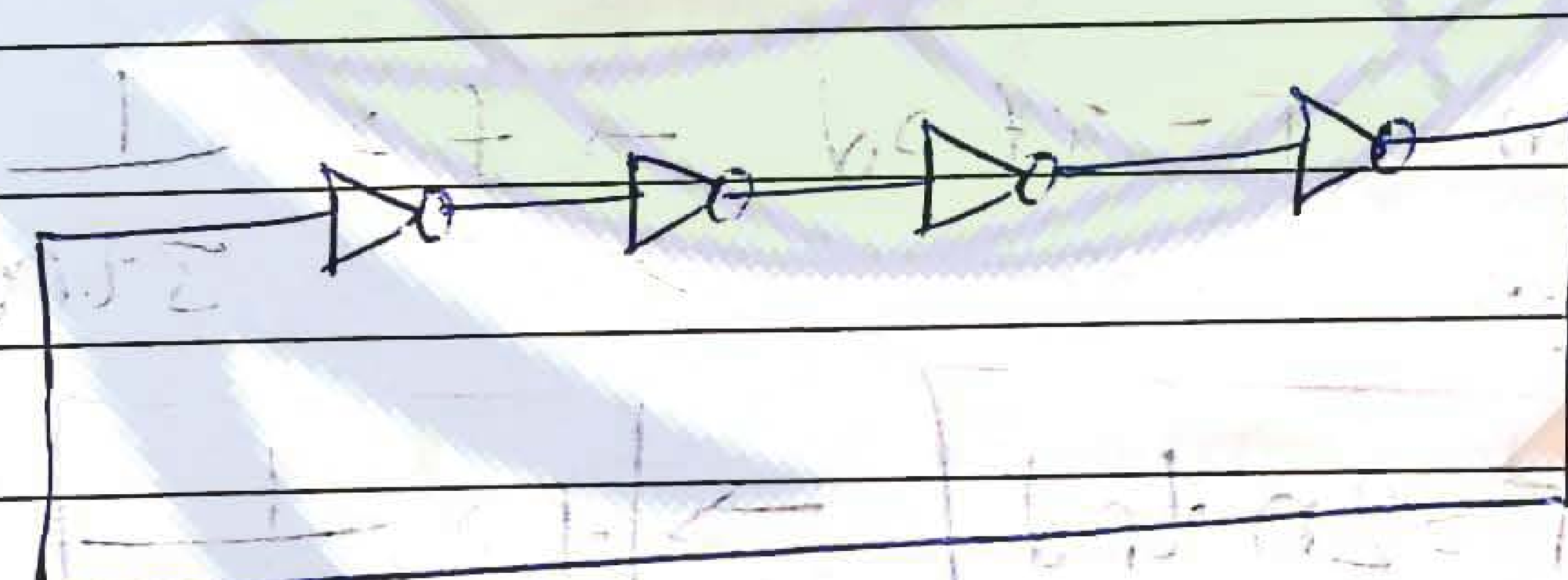


★ multivibrater:
- ASTABLE
- MONOSTABLE — एक state stable है। और आ (अपने आप नीचे आ जाएगा, क्योंकि नीचे वाला stable है।)
- BISTABLE — दोनो state stable है। (दोनो ही state stable है)



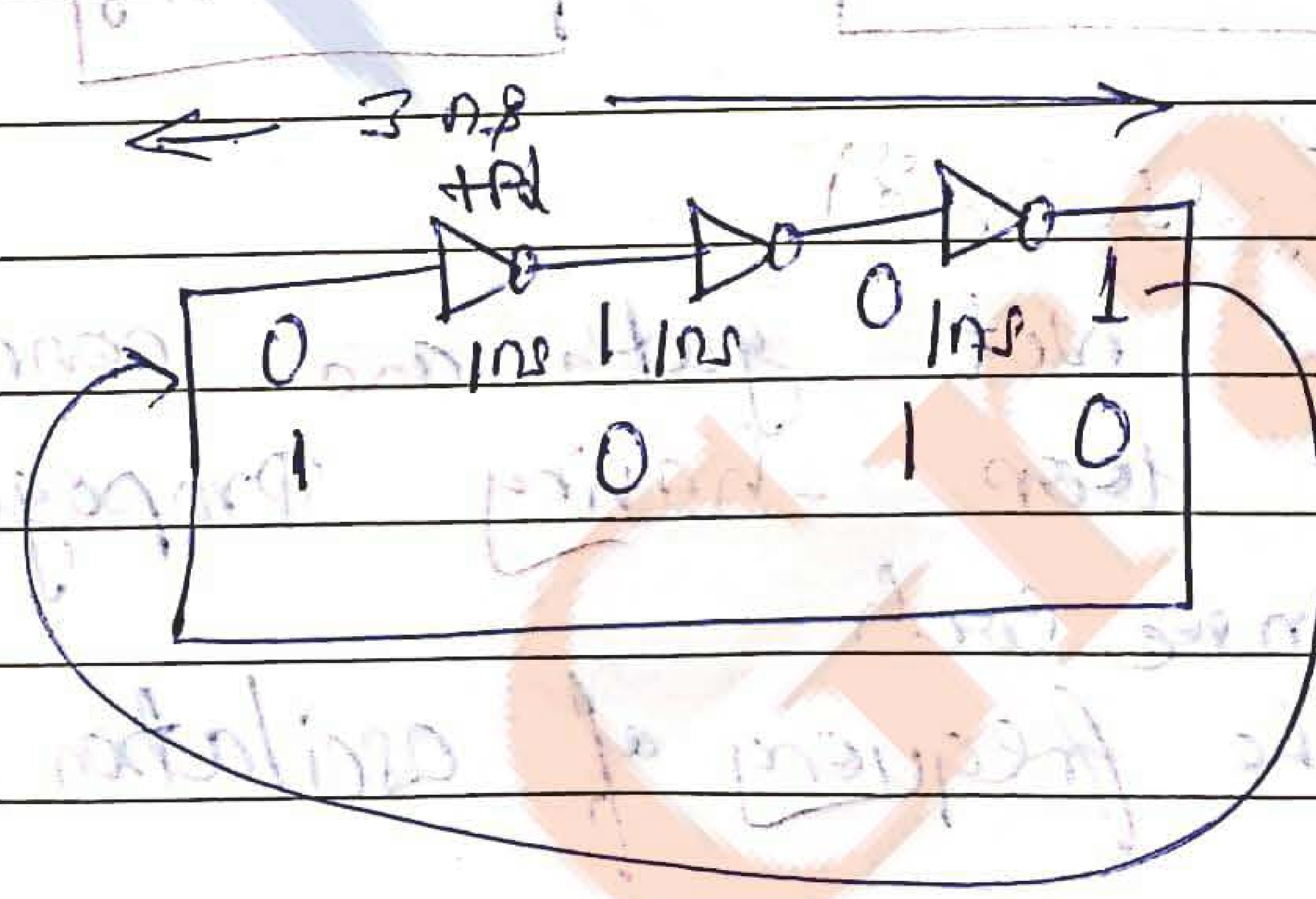


★



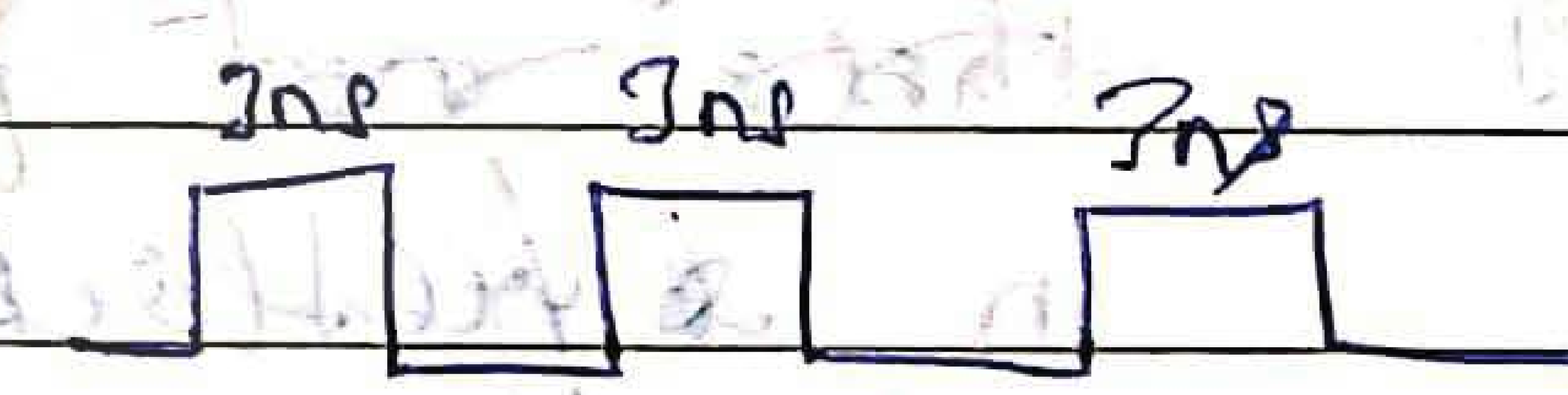
Yes, It is bistable multivibrater.

*

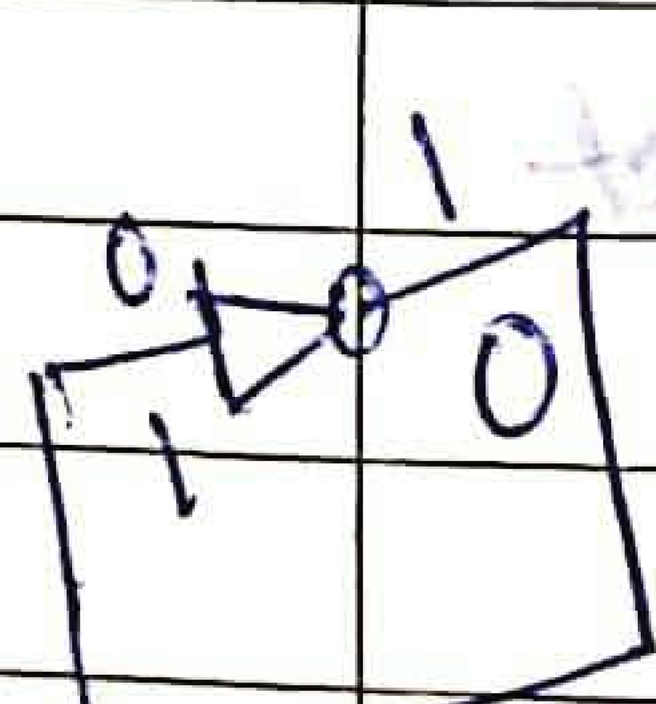


Astable multivibrater

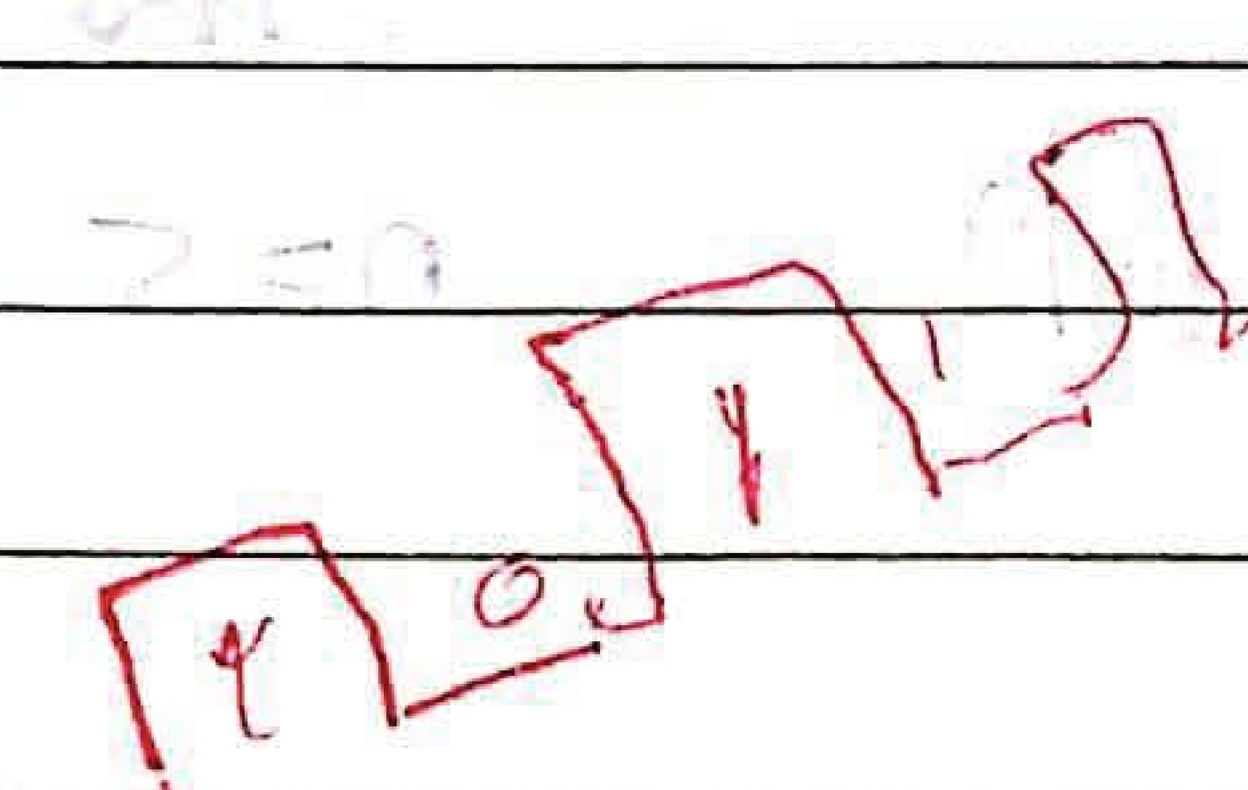
⇒ 3 ns interval



+3Pd



Astable multivi



Page No. ______
Date ______

* If n-not gates are connected in feedback loops—

(i) If n is even number, then
circuit acts like bistable multivibrator

(ii) If n is odd number then
circuit acts like astable multivibrator.

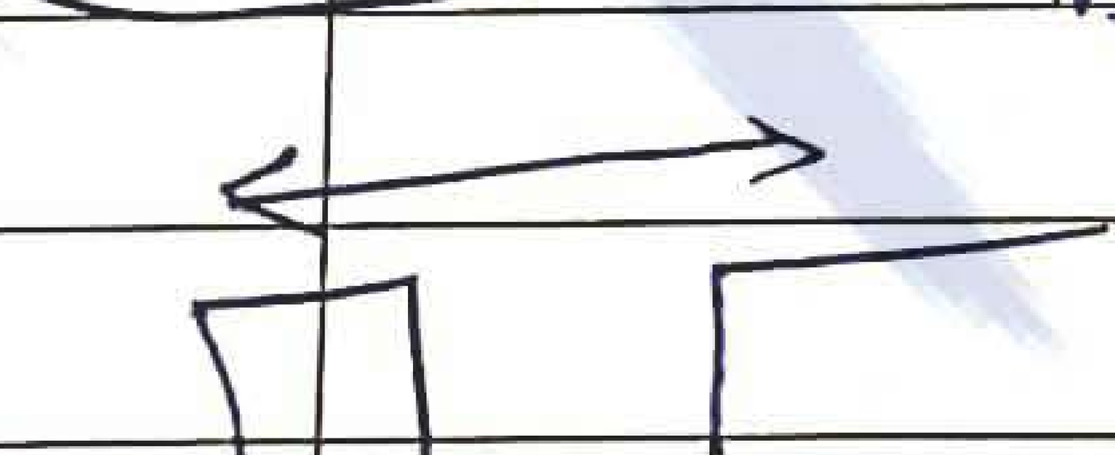
When circuit is Astable multivibrator with n-Not gates having propagation delay of t_{pd} then ~~freq~~ frequency of the output

when $T = 3t_{pd} \Rightarrow f = \frac{1}{3t_{pd}}$

$freq = \frac{1}{time\ period}$

General form:-
V.V.I

$$T = 2n\,t_{pd} \rightarrow f = \frac{1}{2n\,t_{pd}}$$



↖ पूरी होती होता है, cycle complete होता है।

eg. ~~Three not~~ five Not gates are connected in ~~a~~ feedback loop having propagation delay of 5 nsec each
what is the frequency of oscilation at the output.

Sol/n $n = 5$,

Hi, I learn

$T = 2 n t_{pd}$

$= 2 \times 5 \times 5 \times 10^{-9}$

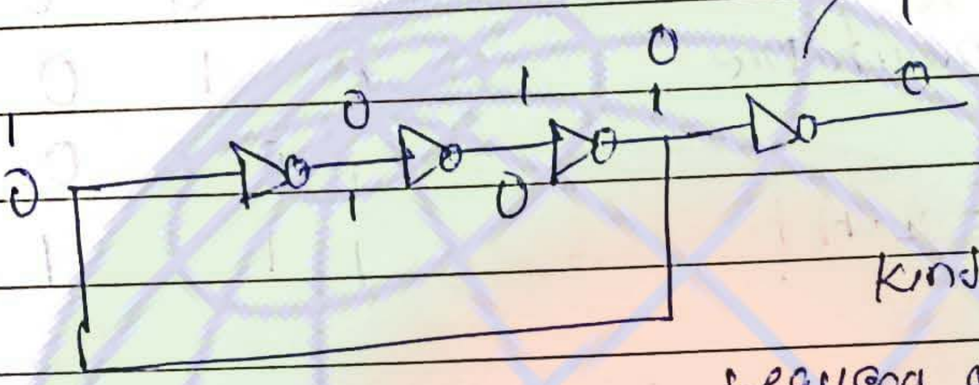
$f = \frac{1}{T} = \frac{1}{2 \times n \times t_{pd}}$

$f = 20 MHz$

$f_1 = 1/n\beta = 10^9 Hz$

→ इसमें signal पर कोई effect नहीं करेगा। frequency में।।

eg.



kind of circuit

frequency of oscilation.

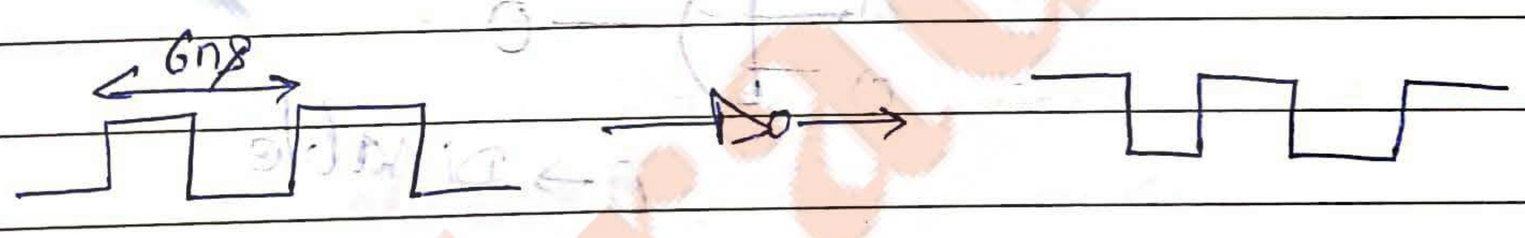
$n = 3$

$T = 2 n t_{pd}$ =

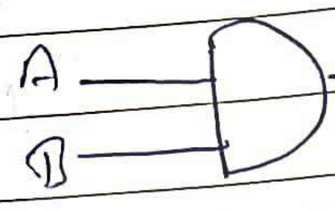
$T = 2 \times 3$

$f = \frac{1}{T} = \frac{1}{6} GHz$

$f = \frac{1}{2 \times 3 \times 1 ns} = \frac{1}{6} GHz$



(2) AND gate:-



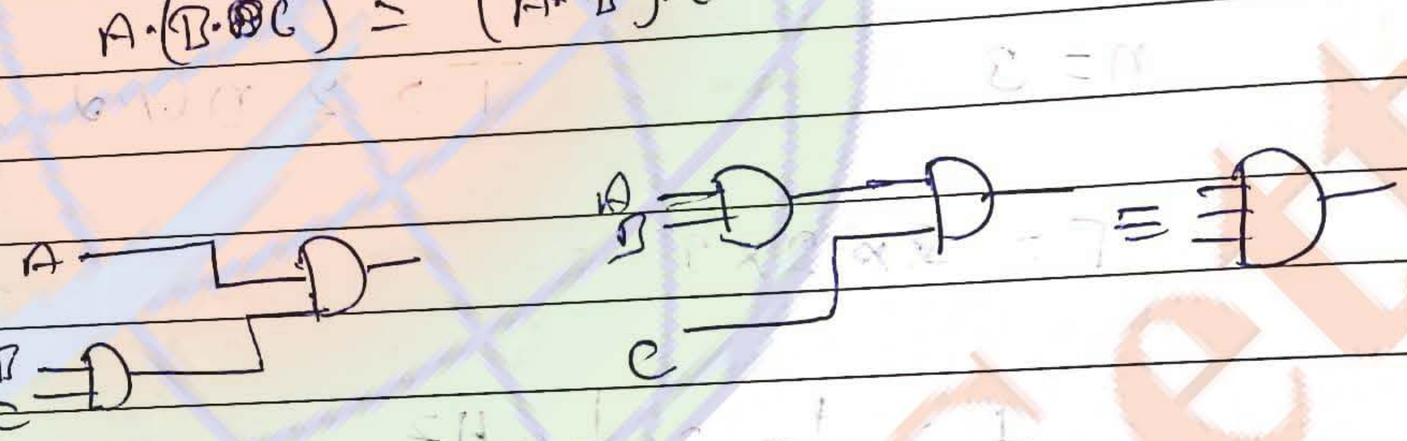
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

- follows commutative law

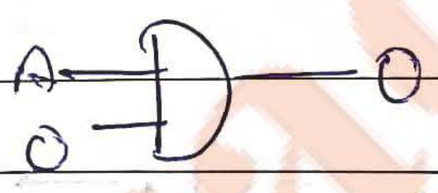
$$A \cdot B = B \cdot A$$

- follows associative law

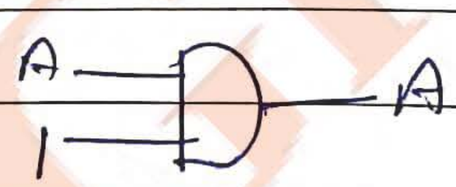
$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$



Note (1) Zero is working like a disable input

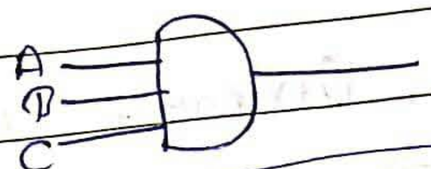


(2)



One (1) works like enable input for One (1) gate.

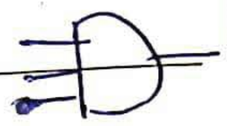
*



$Y = A \cdot B \cdot C$

Unused input = high = 1

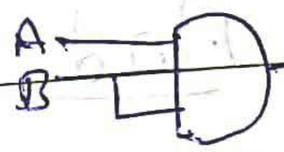
$= A \cdot B$ ✓ , take C = 1 →



Unused input is either A, or B

$y = A \, B \, B$

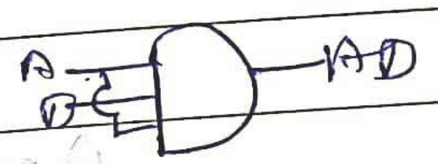
$= A \, B$ ✓



or

$y = A \cdot B \cdot A$

$= A \, B$ ✓



Note:!

① In TTL (transistor transistor logic) open / floting terminal works like high (1) input

where as

in ECL (emiter coupled logic) open / floating terminal works like zero (0).

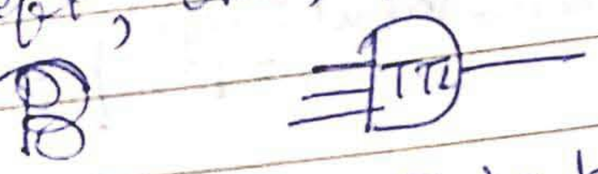
② The unused input or terminal in multi input n-gate may be connected in any one of following manner

(a) May be connected to high value (or 1)

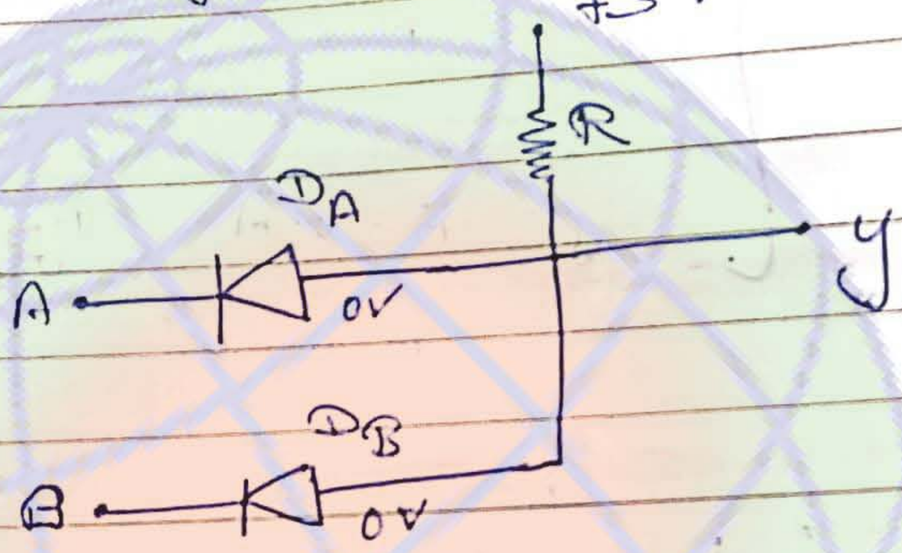
It may be connected to one (1) or high input

(b) It may be connected to one of the useful input.

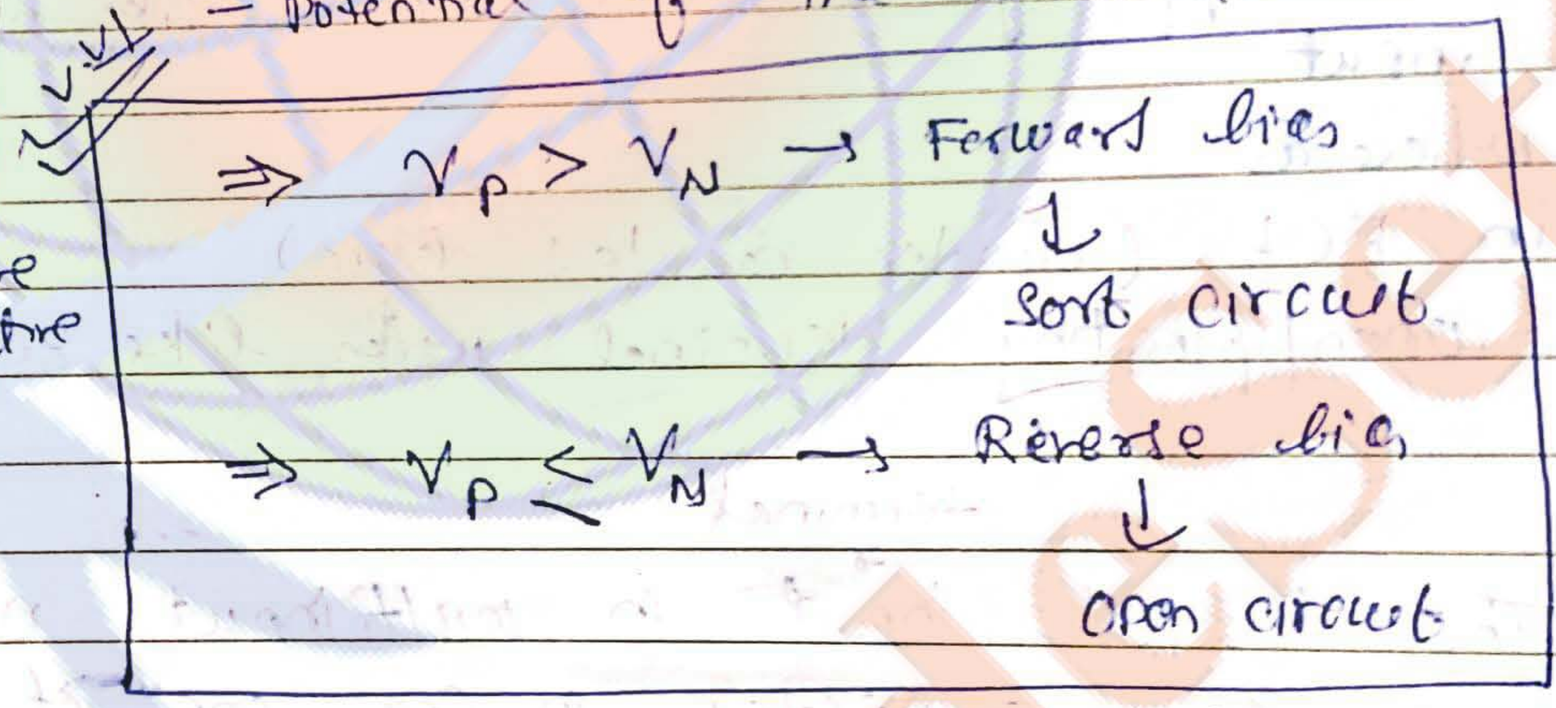
(3) If gate is TTL, Unused input may be left, open, or floating.



(#) And gate Using Diode:-



- Potential of the terminal

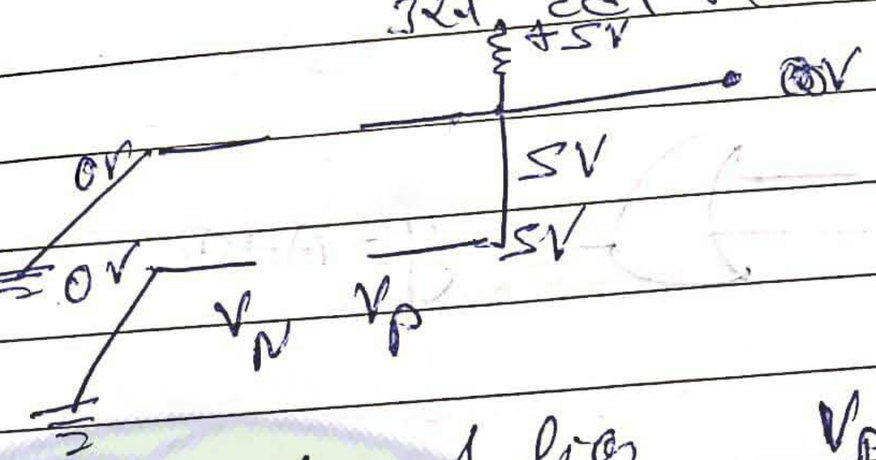


A	B	D_A	D_B	Y
0	0	ON	ON	0
0	1	ON	OFF	0
1	0	OFF	ON	0
1	1	OFF	OFF	1

Page No.
Date

T_1

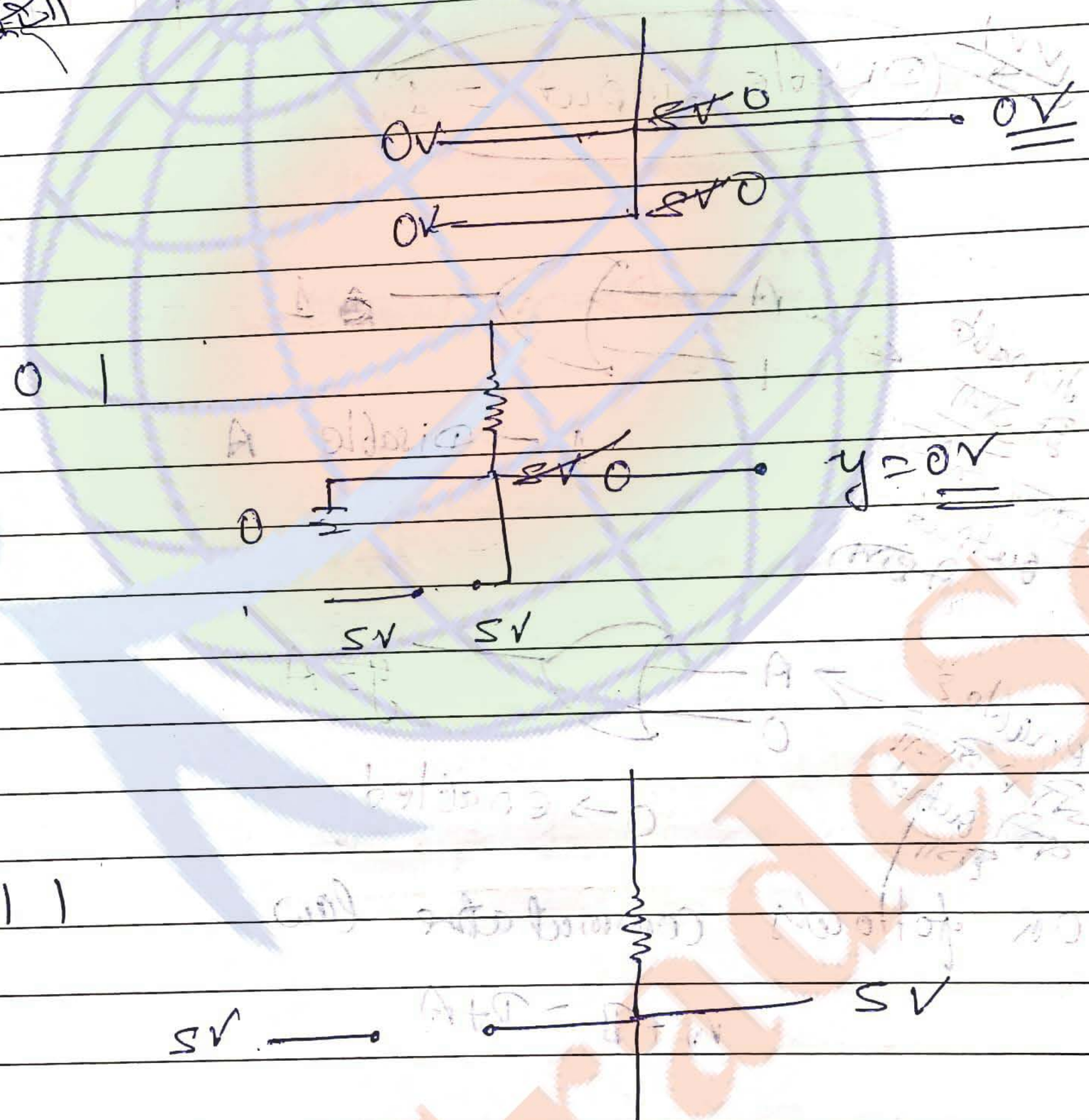
Step 01:- → (1) जिस diode पर काम करना चाहते हैं उसे हटा दीजिये



00

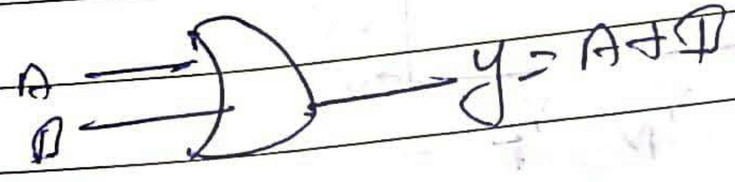
Note
Bias करने के लिए diode को remove करेंगे।

forward bias $V_P > V_N$



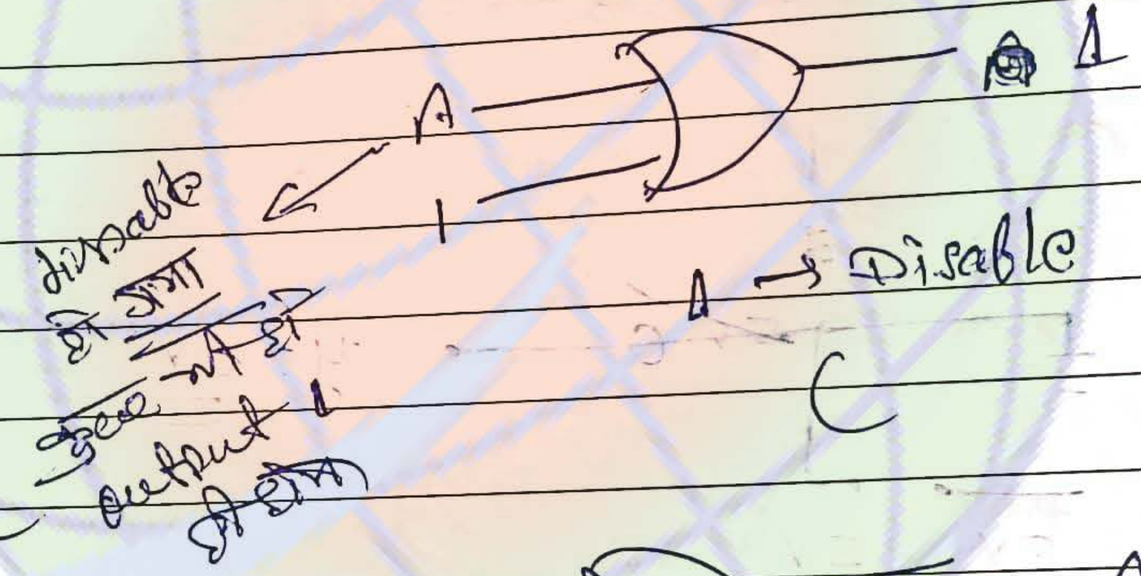
11

(3) OR gate:



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

Disable input = 1

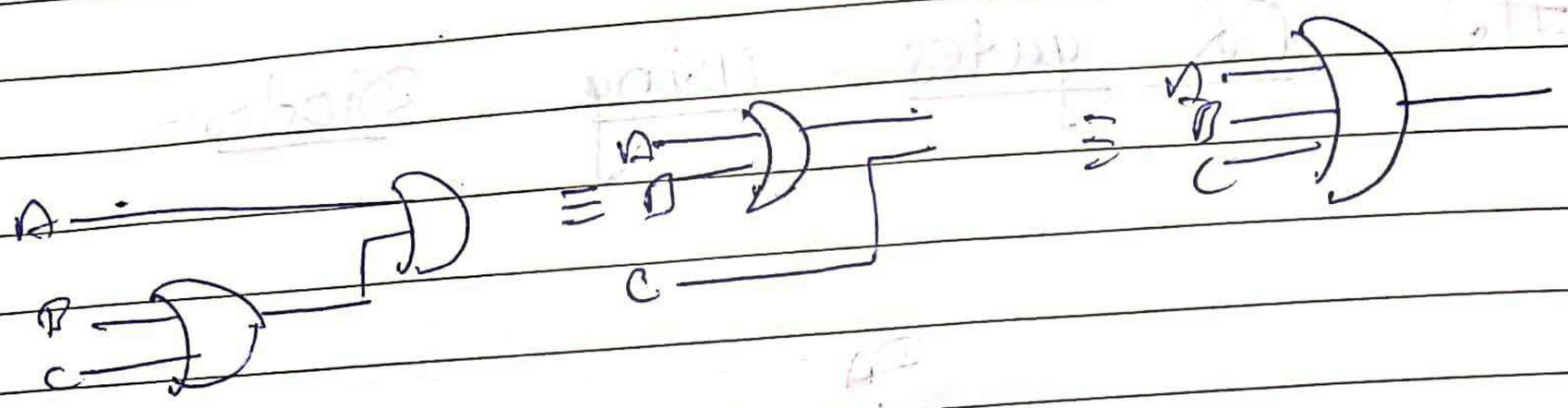


OR follows commutative law

$$A + B = B + A$$

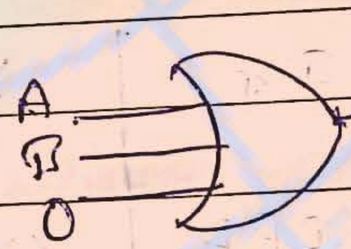
follow associative law

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



The Unused input of multiinput OR gate can be connected in any one of following ways:-

(a) It can be connected to Zero (0)

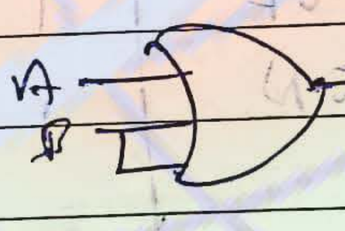


$y = A + B + C$

$= A + B + C_0$

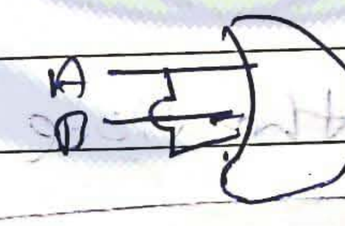
$= A + B$

(b)



$Y = A + B + B$

$= A + B$



$Y = A + B + A$

$= A + B$

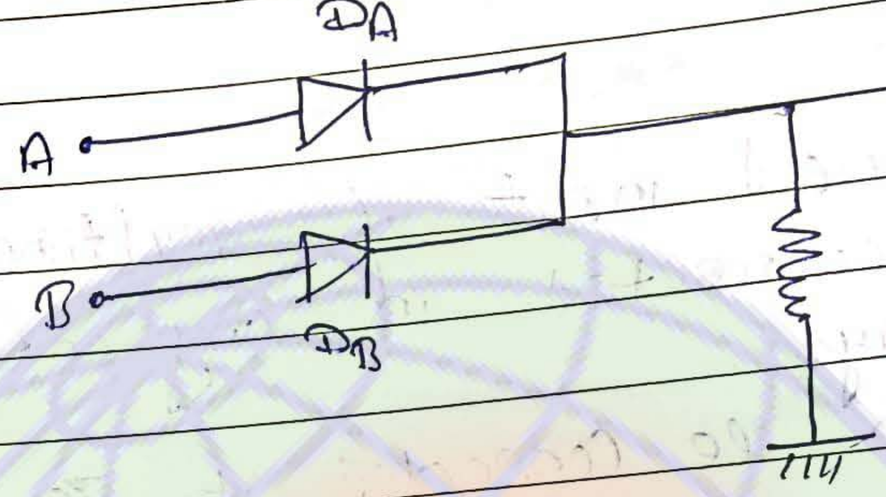
It can be connected to one of the cases

(c) If family is ECL, unused input can be left open or floating.

Page No.
Date

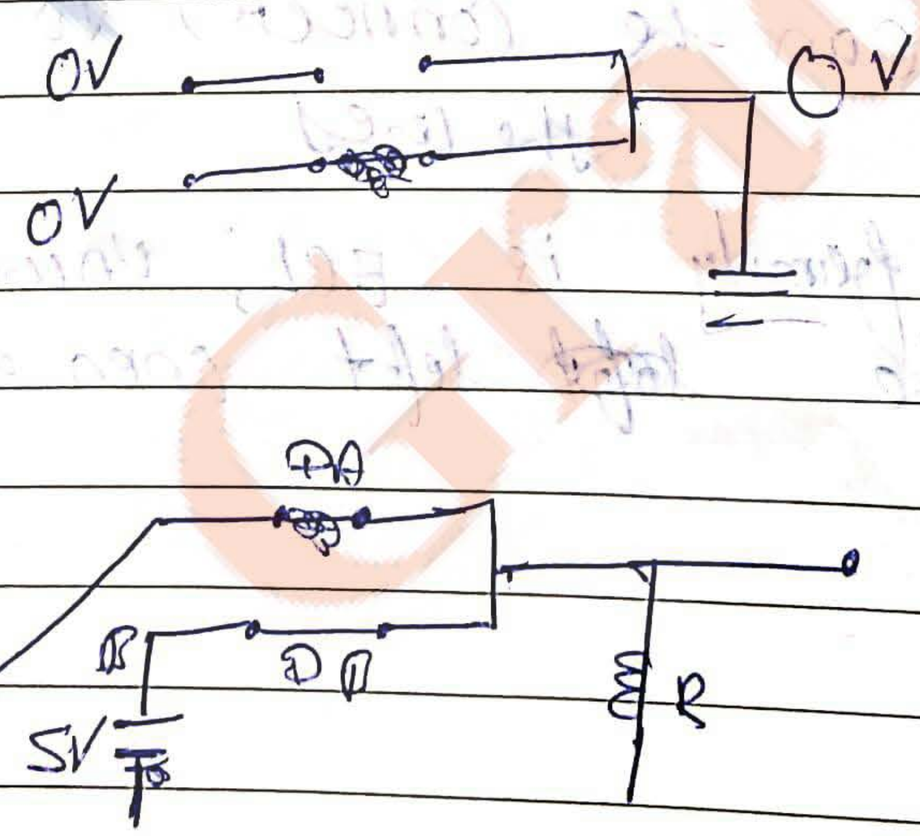
3.2

(#) OR gates using Diodes:-

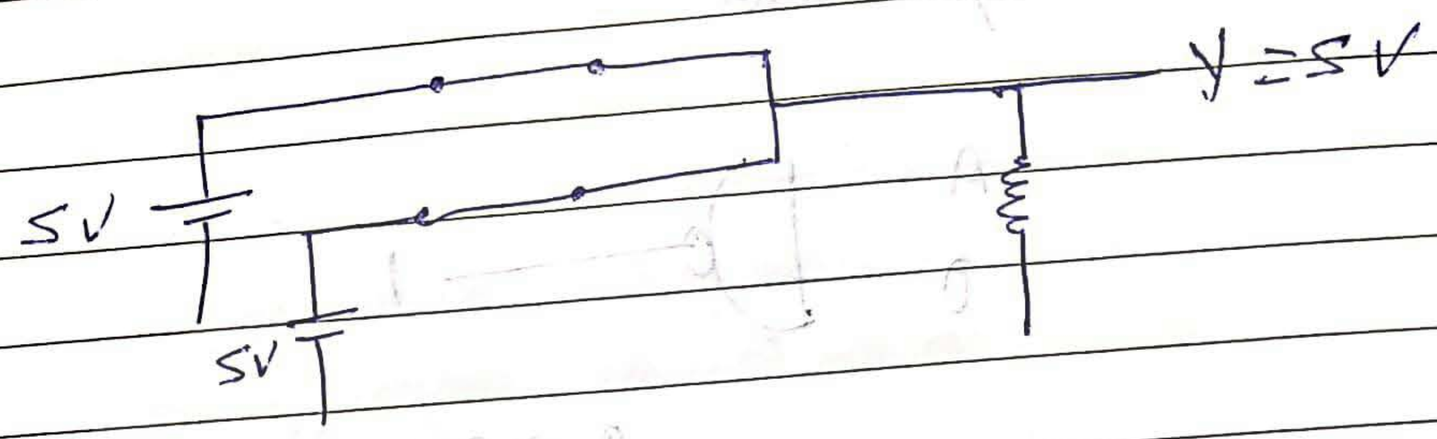


A	B	DA	DB	Y
0	0	OFF	OFF	0
0	1	OFF	ON	1
1	0	ON	OF	1
1	1	ON	ON	1

00 Step 01 to Solve
(To remove the diode



11

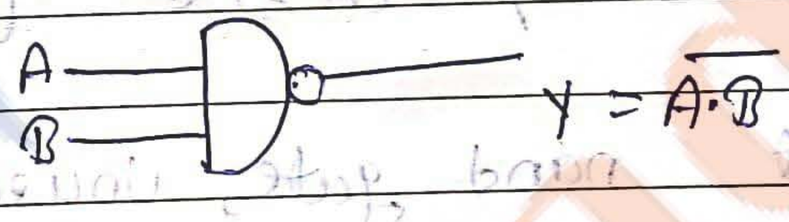


★ When we say xz is implicit of f, it means xz will appear during expansion of f, from minimal form to canonical form.

3.2) Universal gates:-

NAND gate.

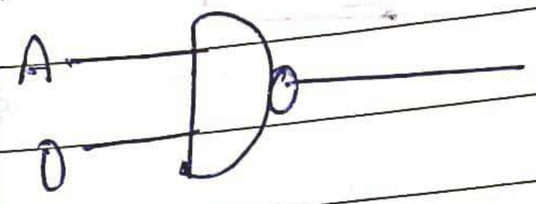
NAND = NOT + AND



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

At NAND gate, any zero at the input make the output 1.

Disable ~~Enable~~



0 → Disable

Enable:

1 → enable.

Note

(a) Nand follows commutative but does not follow associative law.

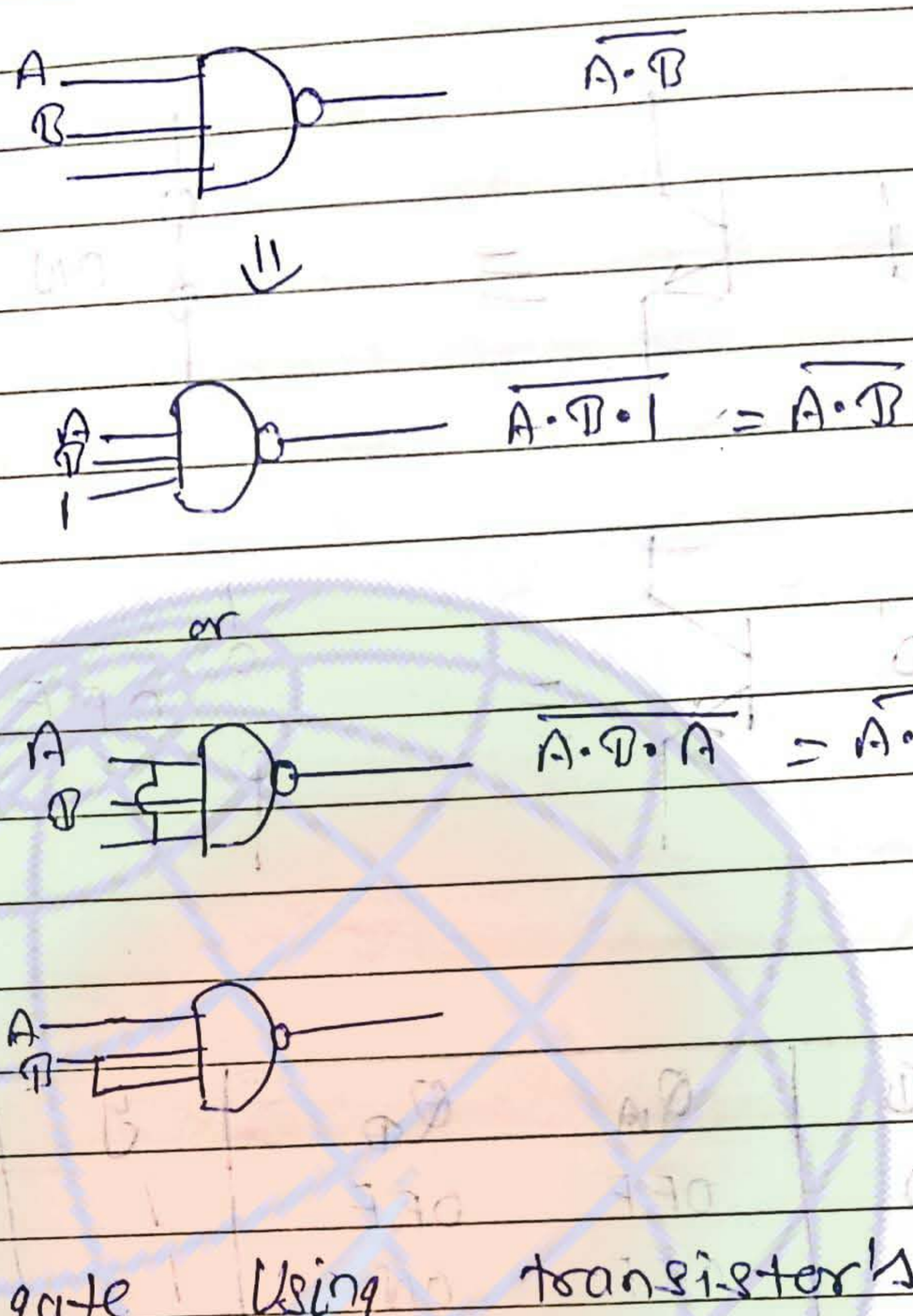
(b) $\overline{AB} = \overline{BA}$ (✓)

$\overline{A \cdot (B \cdot C)} \neq \overline{(\overline{AB}) \cdot C}$ (X)

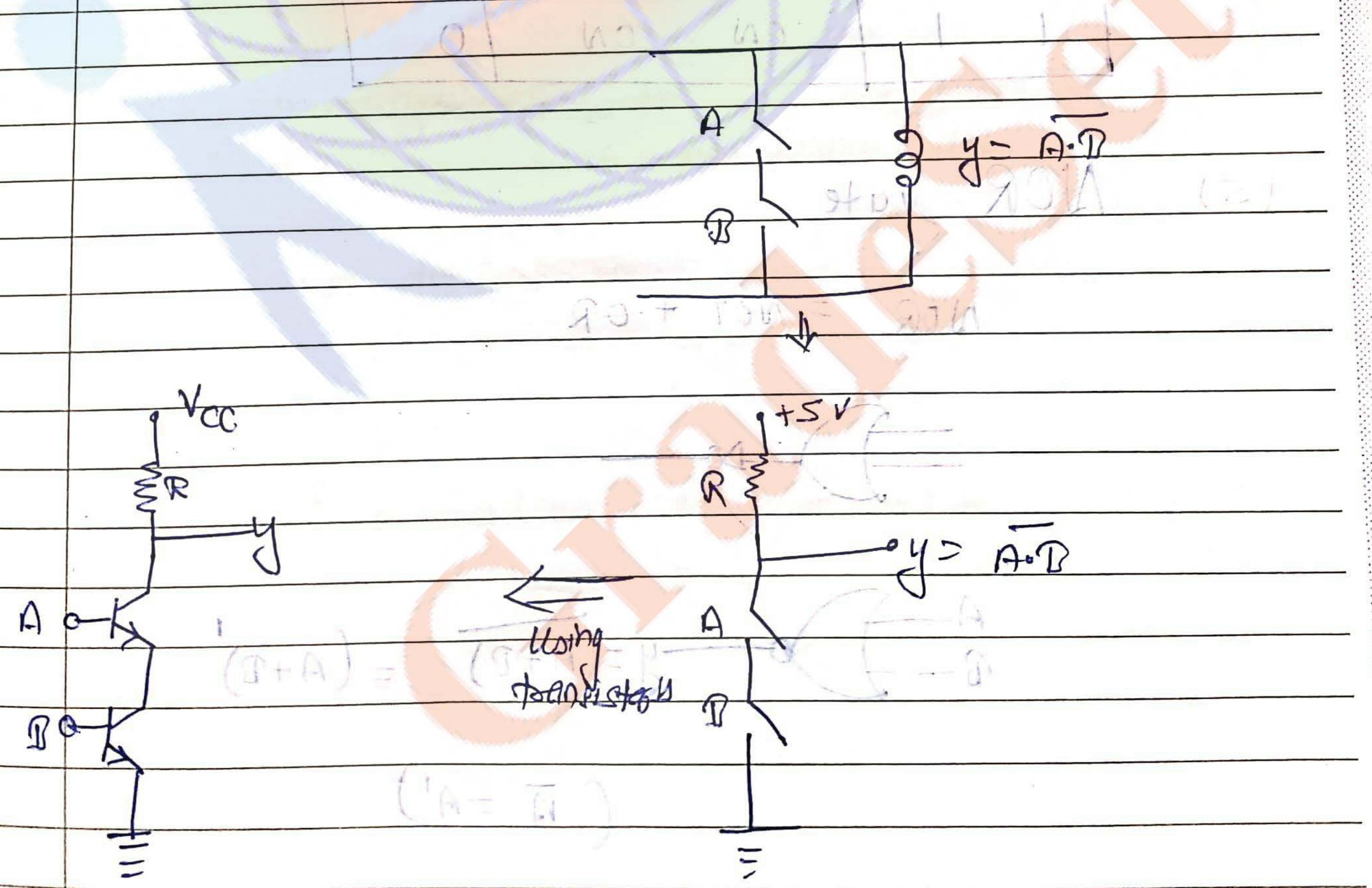
(c) In multi input nand gate, unused input may be connected in the same manner as in ~~and gate~~ AND gate.

In begining of logic circuits NAND is very useful because using NAND only, we can design all kind of logics

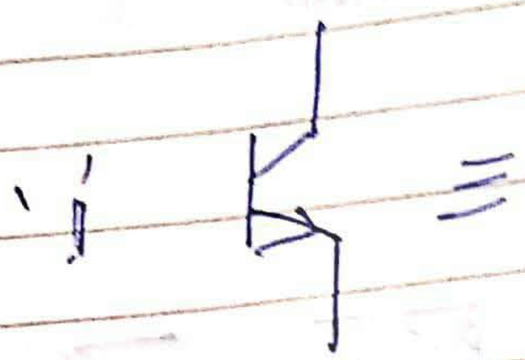
NAND is universal gate and is most widely used in sequential circuits.

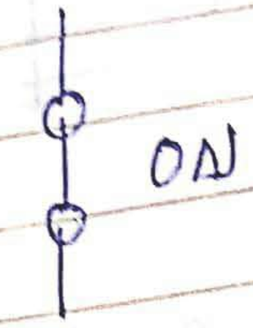


* NAND gate Using transistor's

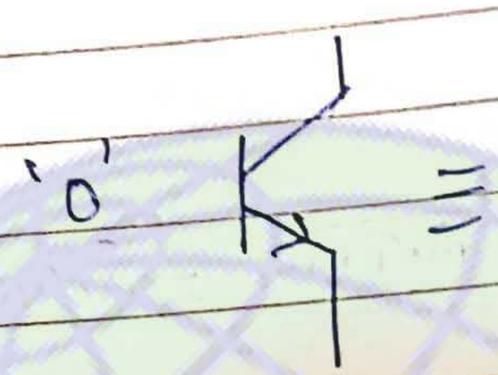


Note:





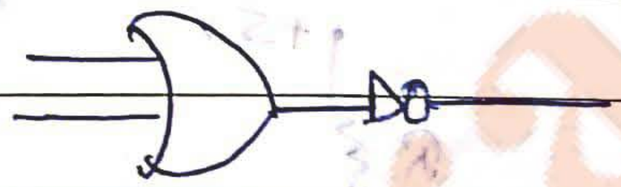
NPN

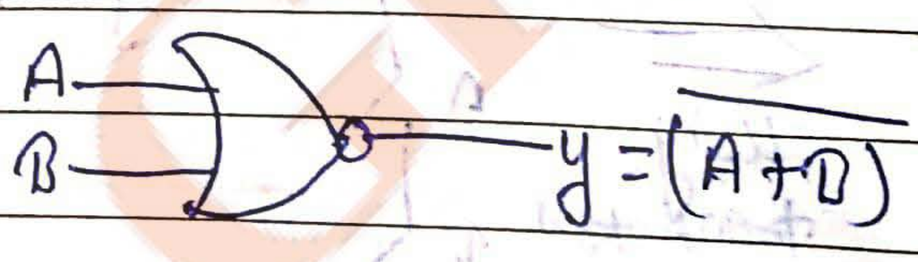


A	B	Q_A	Q_B	y
0	0	OFF	OFF	1
0	1	OFF	ON	1
1	0	ON	OFF	1
1	1	ON	ON	0

(5) NOR gate

NOR = NOT + OR





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

$(\bar{A} = A')$

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

Disable!

'1' → disable

enable!

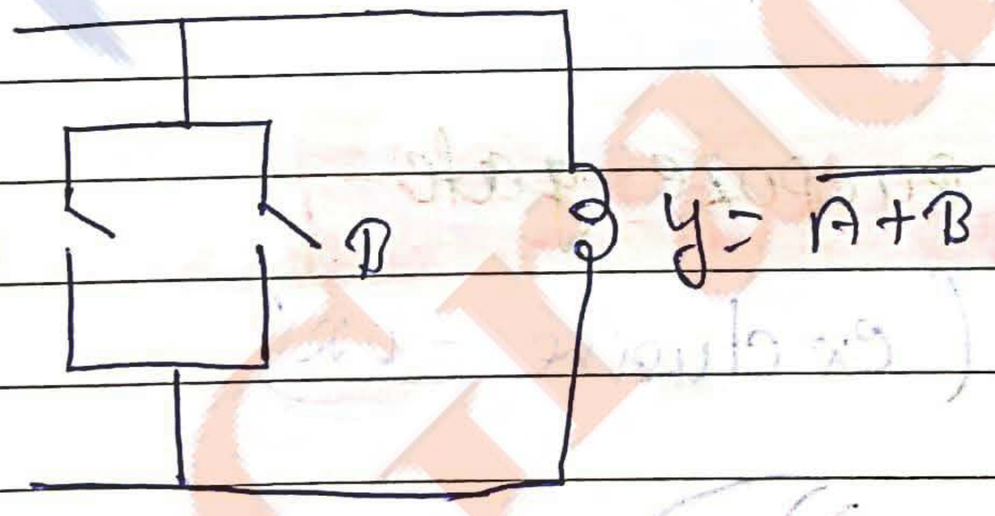
'0' → enable

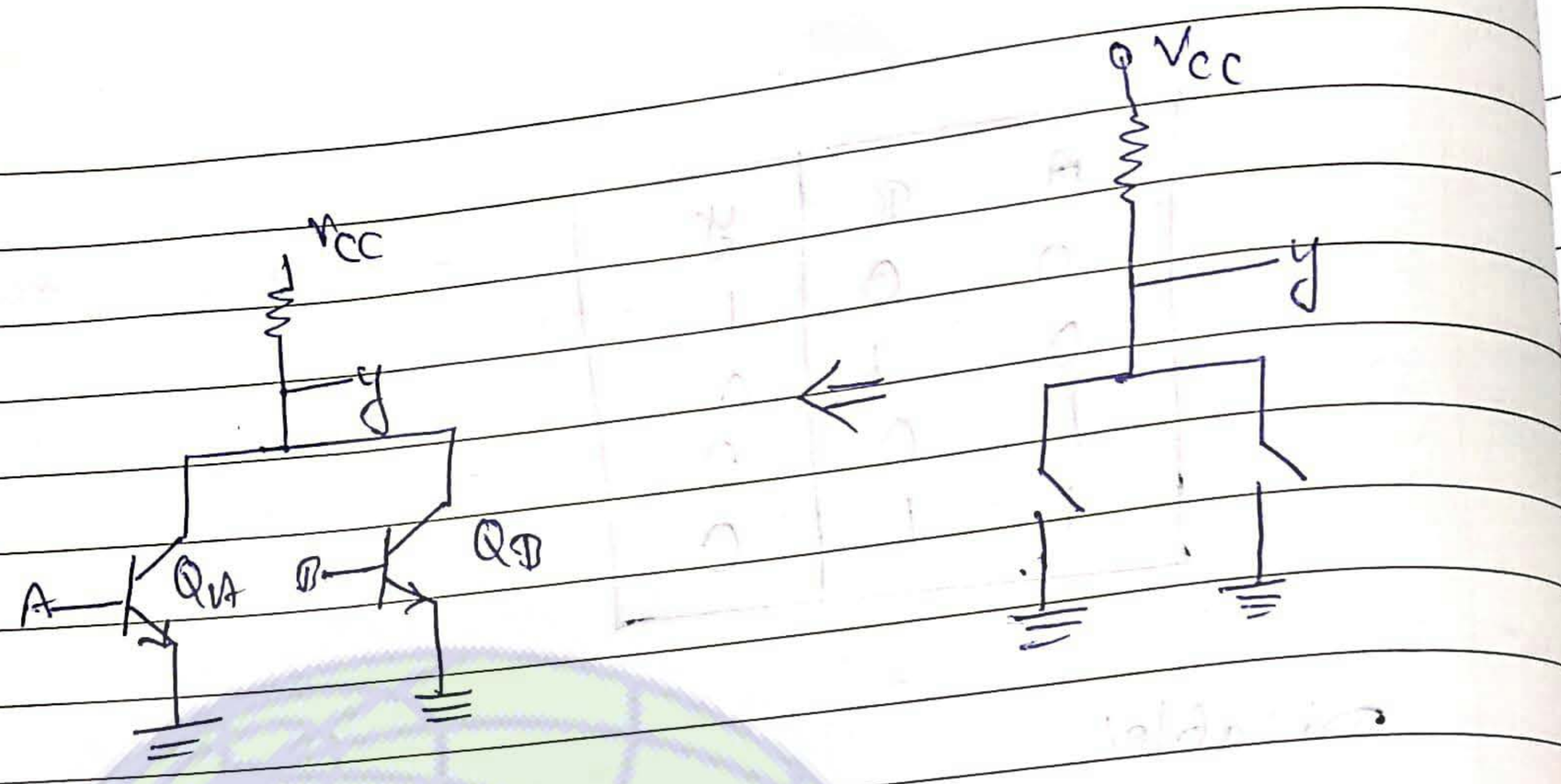
Note:

In NOR gate, if any of the input is high(1) Output is low(0).

(a) ~~Commutative~~ NOR follows Commutative law but doc not follow associative law.

* NOR using transistors!-





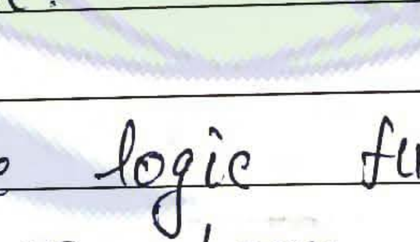
A	B	Q_A	Q_B	y
0	0	OFF	OFF	1
0	1	OFF	ON	0
1	0	ON	OFF	0
1	1	ON	ON	0

- In multiinput NOR gate, Unused input may be connected in the same manner as in OR gate.

All the logic functions may be implemented using NOR, hence NOR is an Universal gate.

3.3) Special purpose gate-

(6) Ex-OR (exclusive - OR)



$y = A \oplus B = \overline{A \odot B}$

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

$\oplus$ → Ex OR operation

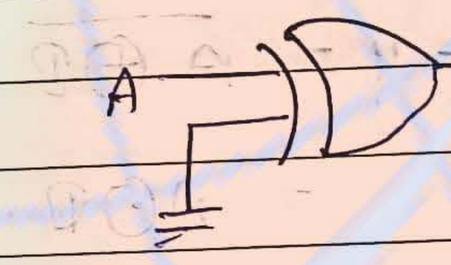
A	B	y	
0	0	0	$A+B$
0	1	1	$\rightarrow \bar{A}B$
1	0	1	$\rightarrow A\bar{B}$
1	1	0	$\bar{A}+\bar{B}$

Then it call it as "OR" (Plus वाला)

* Disable / enable:-
  Not disable or enable input is available.

Note:

(1)



$y = A \oplus 0$

$= \bar{A} \cdot 0 + A \cdot \bar{0}$

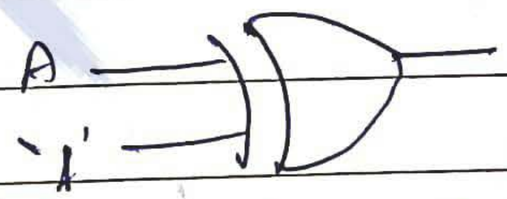
$= 0 + A \cdot 1$

$= A$

(It works like buffer)

[Output is same like input.]

(2)



$y = \bar{A} \cdot 1 + A \cdot \bar{1}$

$= \bar{A}$

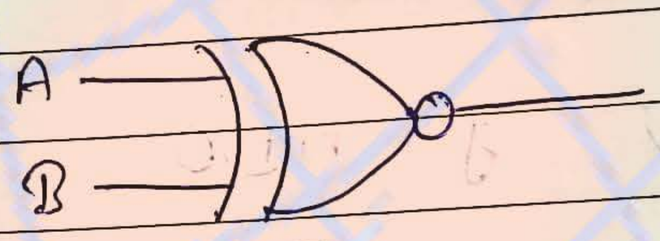
It works like an ["inverter"]

Hence, if one input is made Zero, then EX-OR behaves like a buffer for other input

If one input is high (1), then Ex-OR behaves like an inverter for other input

There is no disable/enable input for Ex-OR.

(7) Ex-NOR



$y = \overline{A \oplus B}$

$= A \odot B$

$= \bar{A} \cdot \bar{B} + AB$

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

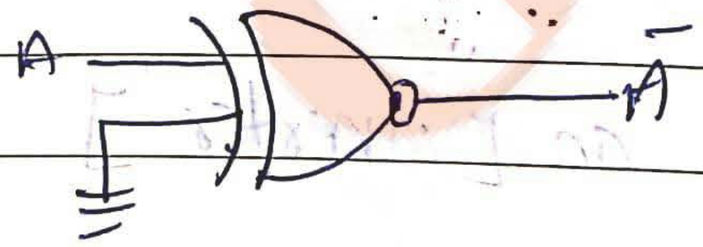
Truth table:

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

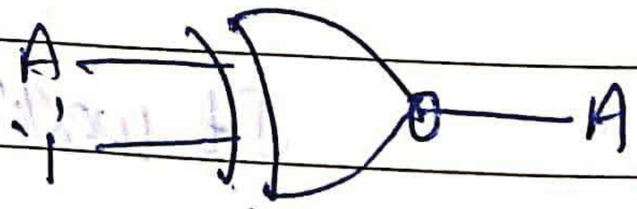
* Disable/enable

There is no disable/enable input

*

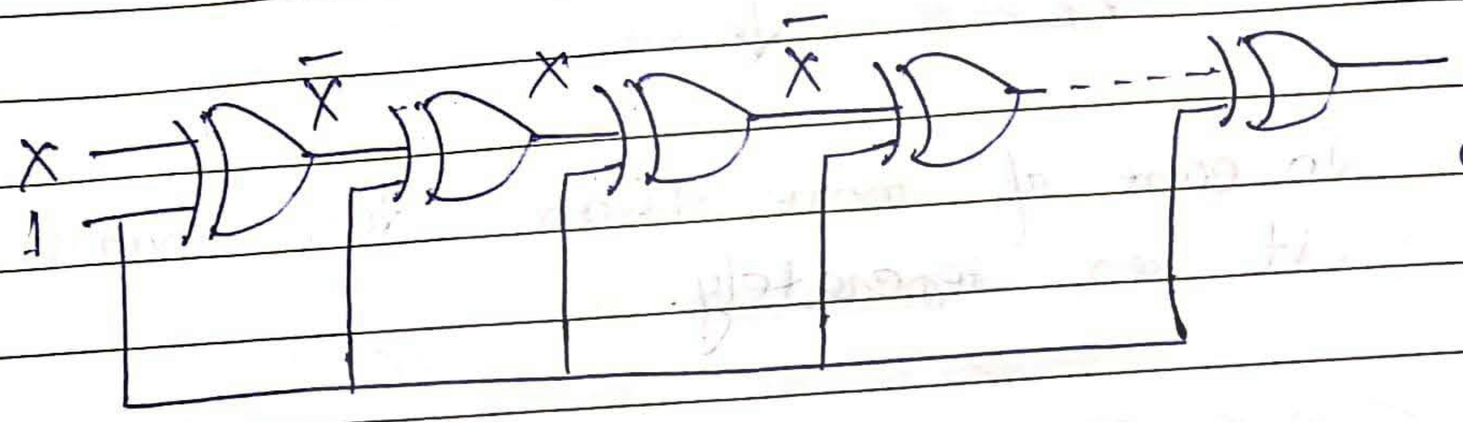


Inverter



Buffer

(1)



No. of gates = 21

Since No. of gate is odd,
Hence

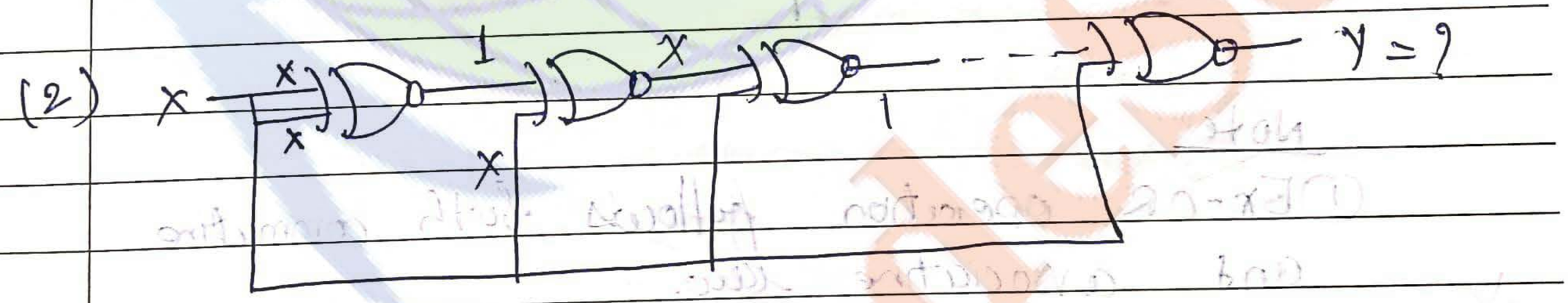
$$\boxed{y = \bar{x}}$$

If No. of gates is even

$$\boxed{y = x}$$

Note:
In case of 1, works like inverter in EX-OR

(2)



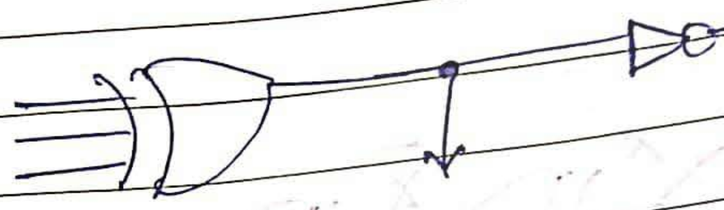
If No. of gates = 20 ⇐
y = ?

$$\boxed{y = x}$$ Ans

Note:
(1) In two input EX-OR gate, when inputs are dissimilar, output is '1'
(2) In two input EX-NOR gate, when inputs are similar, output is '1'.

Note:
(1) Commercially EX-OR and EX-NOR gates are available only with two inputs.

(4)



In case of more than two inputs, treat it as seperately.

(3)

(4)(5) — A — B — C

☆ 3-Input EX-OR :-

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

(पहले दो को आपस में)

$y = A \oplus B \oplus C$

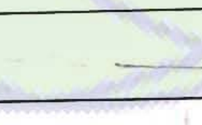
0
1
1
0
1
0
0
1

Number of 1's off in input output = 1

Note:

① Ex-OR operation follows both commutive and associative law.

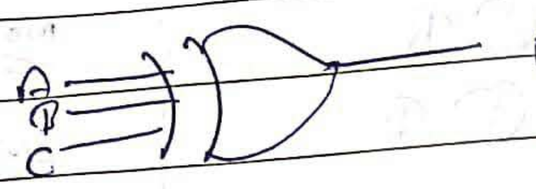
② Ex-OR ⇒ The most generalised definition for Ex-OR operation is

Ex. 

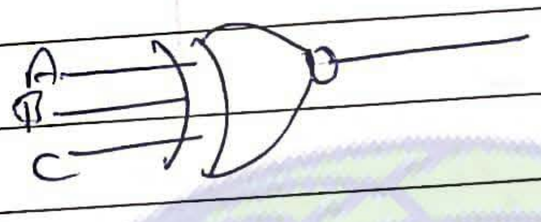
Output of Ex-OR is '1' when number of 1's at the input of Ex-OR is odd.

(6)

(3)



$A \oplus B \oplus C$



$\overline{A \oplus B \oplus C}$

(4) ✱

A,B \ C,D	00	01	11	10
00		1		1
01	1		1	
11		1		
10	1		1	

(1's ones)

(0's ones)

$f = A \oplus B \oplus C \oplus D$

* Properties of Ex-OR and Ex-NOR :-

(1)

$A \oplus A = 0$

$A \oplus \bar{A} = 1$

$A \odot A = 1$

$A \odot \bar{A} = 0$

$A \oplus 1 = \bar{A}$ (inverter)

$A \oplus 0 = A$ (buffer)

$A \odot 1 = A$ (buffer)

$A \odot 0 = \bar{A}$ (inverter)

KV. (2) ***

(2) Relation b/w $\oplus$ and $\odot$

$A \oplus \bar{B} = A \odot B$

$\bar{A} \oplus B = A \odot B$

अगर देखें, कि ये कैसे काम कर रहा है

$A \odot \bar{B} = A \oplus B$

$\bar{A} \odot B = A \oplus B$

(3) $A \odot B = \overline{A \oplus B}$

Proof:

$f = A \oplus \bar{B}$

Let $\bar{B} = y$

$f = A \oplus x$

$= \bar{A}x + A\bar{x}$

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

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

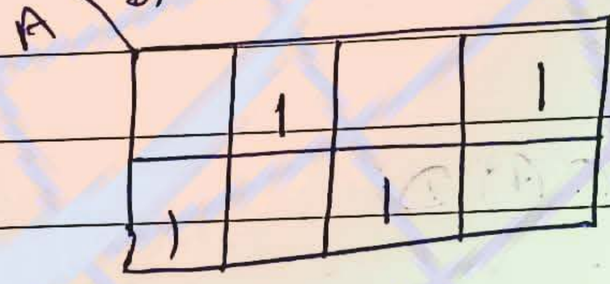
$= A \odot B$

Note-

$A \odot B = \overline{A \oplus B}$

$\overline{A \odot B} = A \oplus B$

(4)



EX-OR

$f = A \oplus B \oplus C$

(a) $A \oplus B + C$

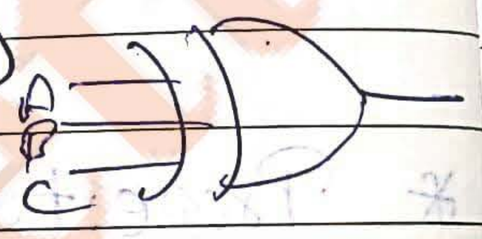
(b) $\overline{A \odot B \oplus C}$

(c) $\overline{A \oplus B \odot C}$

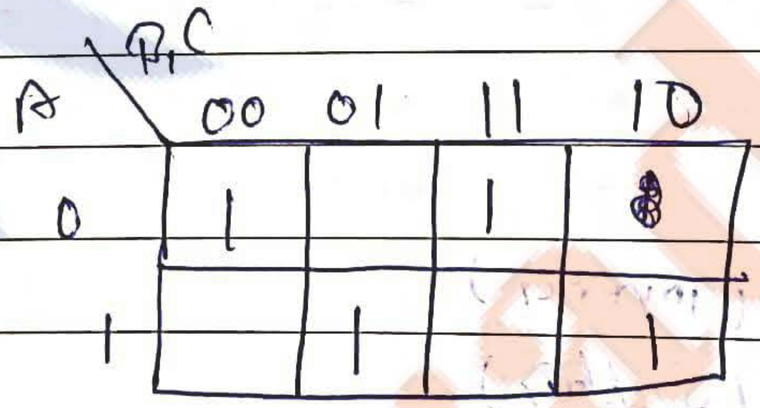
(d) $A \odot B \odot C$

(e) all of the above

(EX-OR)



5)



EX-NOR

(Reverse of EX-OR)

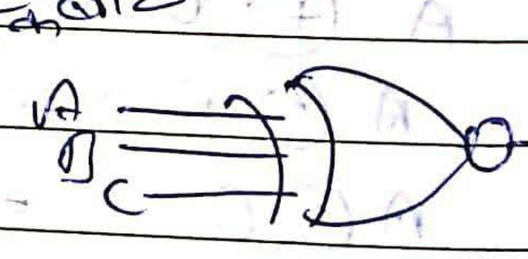
f is (a) $\overline{A \oplus B \oplus C}$

(b) $A \odot B \oplus C$

(c) $A \oplus B \odot C$

(d) $A \odot B \odot C$

(e) all of above



$f = \overline{A \oplus B \oplus C}$

Let

$B \oplus C = X$

$f = \overline{A \oplus X}$

$= A \odot X$

$= A \odot B \oplus C$

$f = A \odot (B \oplus C)$

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

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

Let

$A \odot B = X$

$f = X \odot \bar{C}$

$= \overline{X \oplus C}$

$= \overline{X \odot C}$

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

Note

(a) 1 time $\odot$ $\rightarrow$ $\odot$ (odd no. of times)

(b) 2 times $\odot$ $\rightarrow$ $\oplus$

Note

$A \oplus B \oplus C = A \odot B \odot C$

$A \oplus B \oplus C \oplus D = \overline{A \odot B \odot C \odot D}$

(6) When Ex-NOR operation occurs between variables,
(a) odd number of Ex-nor operation is same as one-time ex-nor.
(b) Even number of Ex-nor results in Ex-or operation.

(7)

A,B \ C,D	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1			

f is
(a) $A \odot B \oplus C \oplus D$
(b) $A \odot B \odot C \odot D$
(c) $A \oplus B \odot C \oplus D$
(d) $A \oplus B \oplus C \odot D$

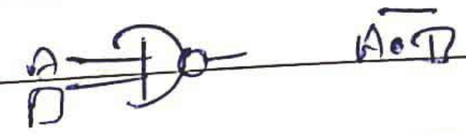
(8) $\overline{A \oplus B \oplus C \oplus D} = A \oplus B \oplus C \odot D$
$= A \oplus B \odot C \oplus D$
$= A \odot B \oplus C \oplus D$
$= A \odot B \odot C \odot D$
$= \overline{A \odot B \odot C \oplus D}$
$= A \odot B \odot C \oplus \overline{D}$
=

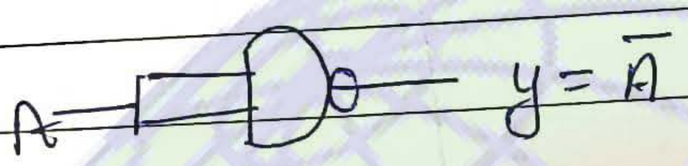
(3.4) 3.4) Relization of logic gates Using Universal gates:-

NAND as an Universal gate:

(1) NOT:

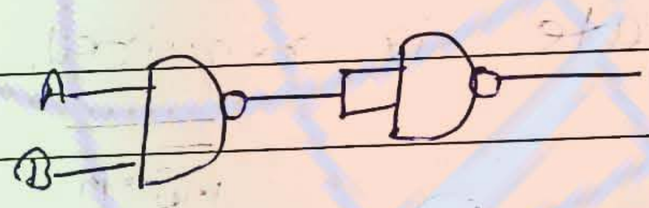
$y = \bar{A} = \overline{A \cdot A}$





Hence, '1' NAND is required.

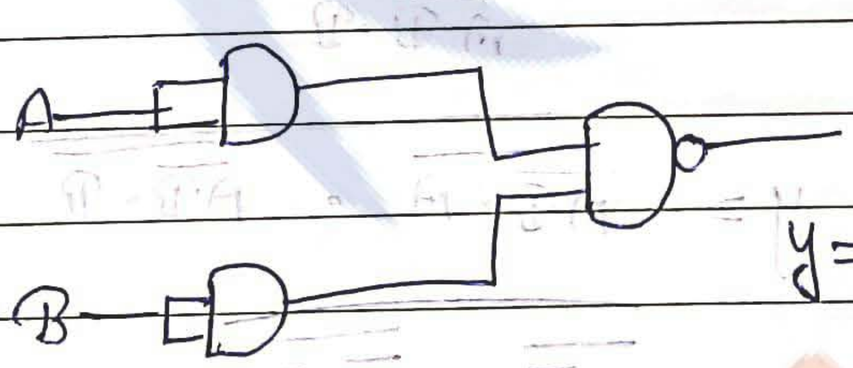
(2) AND



$y = A \cdot B = \overline{A \cdot B}$

Hence, '2' NAND is required

(3) OR



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

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

Hence, '3' NAND is required.

(4) EX-OR

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

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

Date

$\overline{A}B = \overline{\overline{A \cdot B} \cdot AB}$

$\overline{A}\,\overline{B} = \overline{\overline{AB} \cdot A}$

$\therefore$

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

or

$= \overline{\overline{A} \cdot B + 0}$

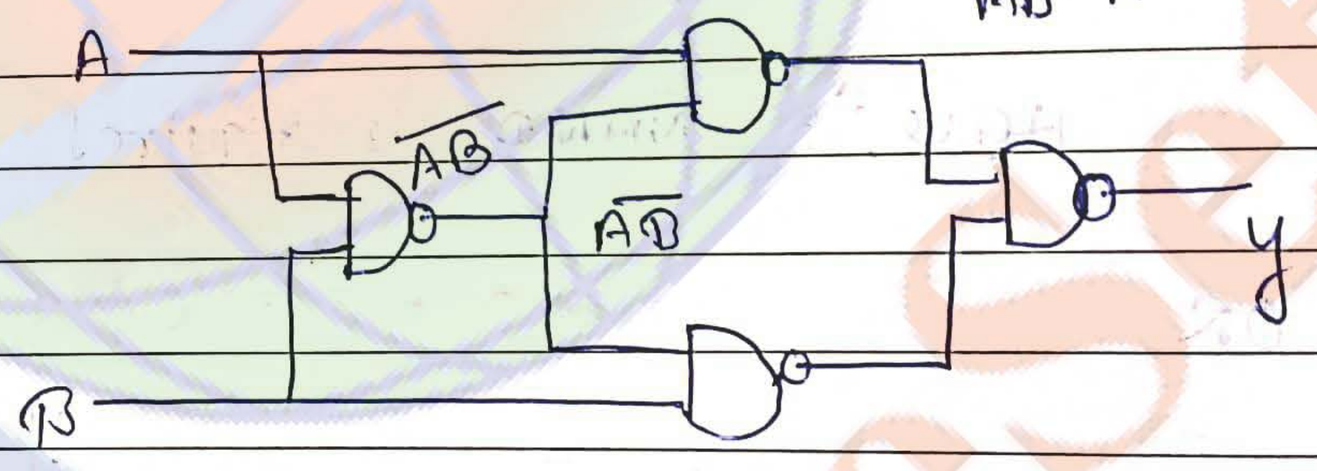
$= \overline{\overline{A} \cdot B + B\overline{B}}$

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

$= \overline{\overline{AB} \cdot B}$

Hence, 4 NAND gate is required.

$\overline{\overline{AB} \cdot A}$



$\overline{A\overline{B} \cdot B}$

$y = \overline{\overline{A\overline{B}} \cdot A} \cdot \overline{\overline{A\overline{B}} \cdot B}$

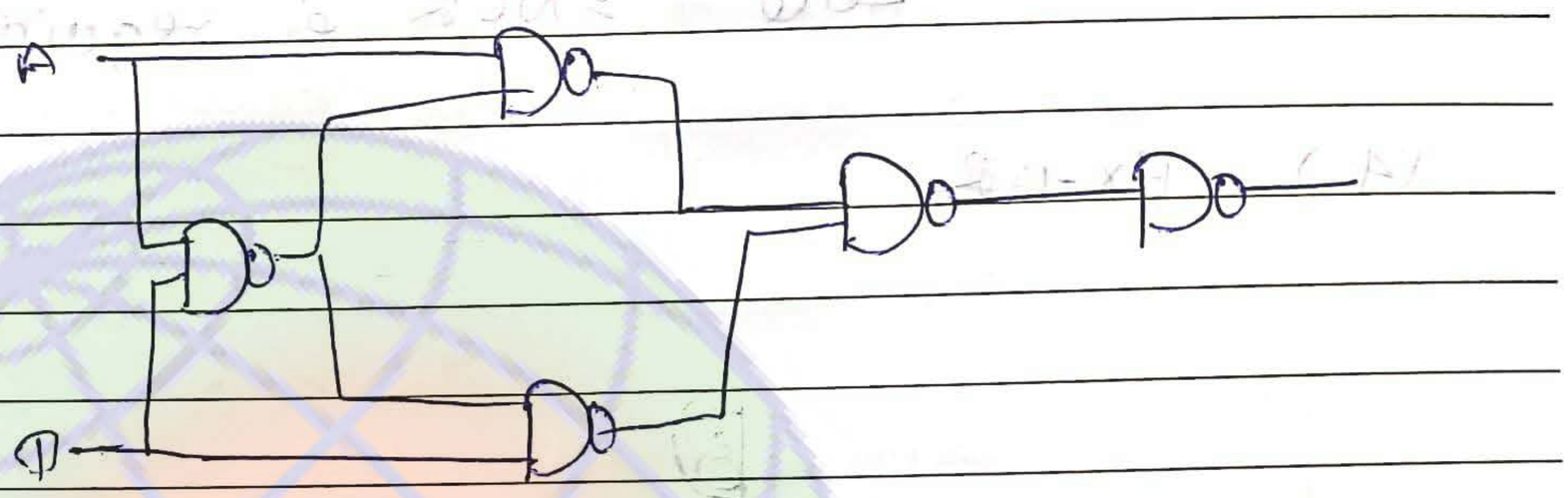
$= \overline{A\overline{B}} \cdot \overline{\overline{A} \cdot B}$

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

$= A \oplus B$

(5) EX-NOR

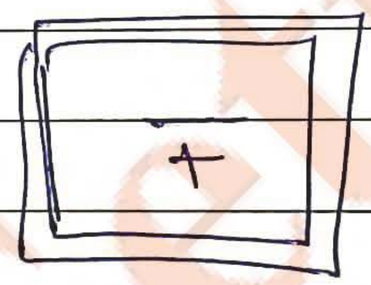
minimum number of NAND requires ie 5

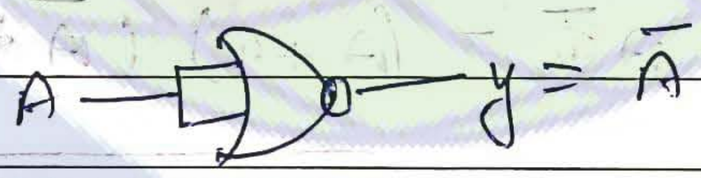


☆ NOR as Universal gate

(1) NOT

$$y = \bar{A} = \overline{A + A}$$





Hence 1 NOR is required.

(2) AND

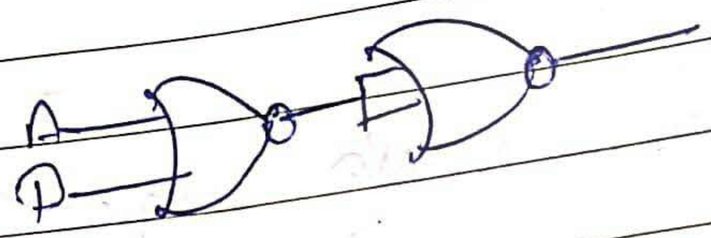
$$y = A \cdot B = \overline{\overline{A \cdot B}} = \overline{\bar{A} + \bar{B}}$$

$$y = A \cdot B$$

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

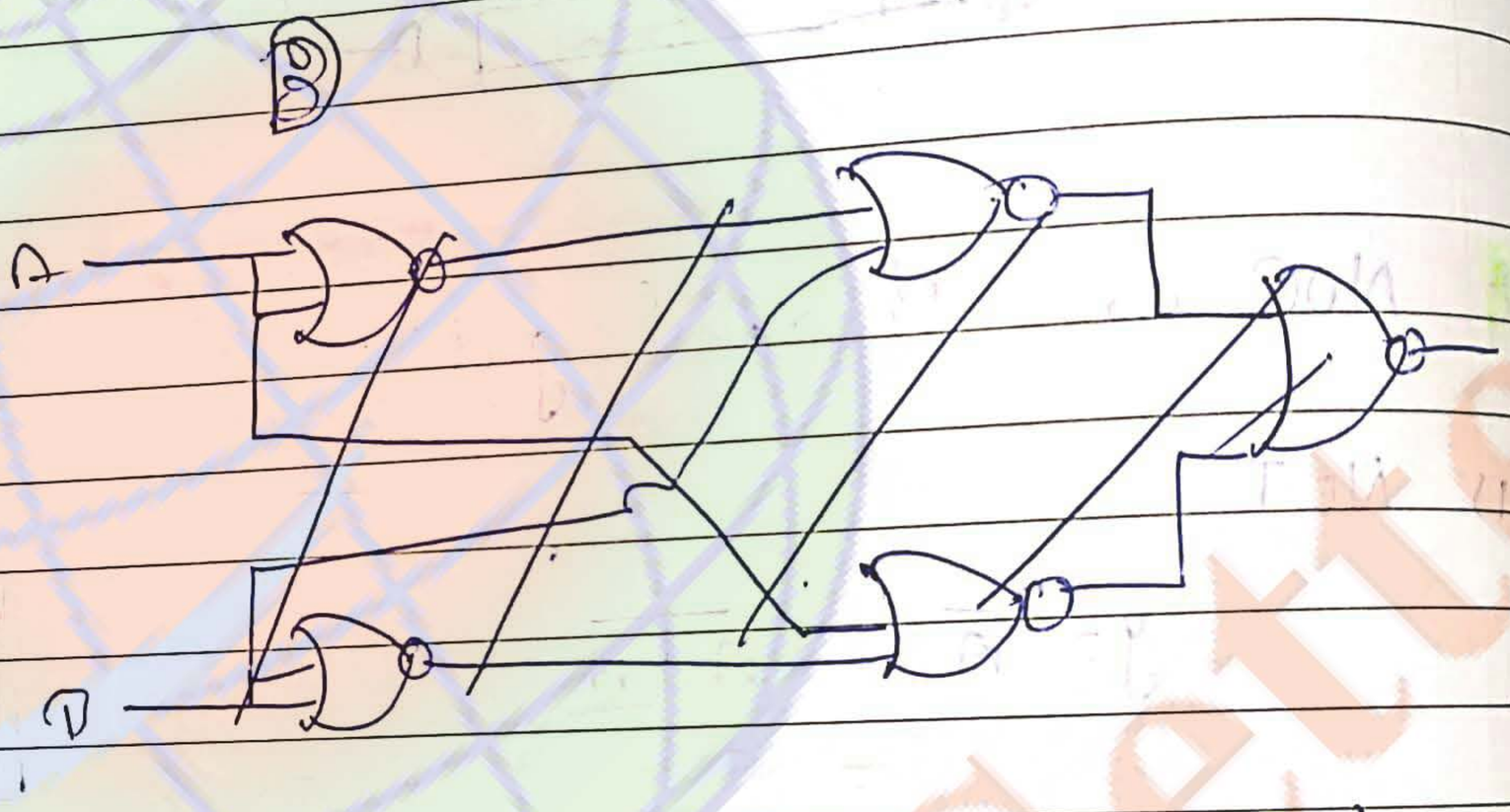
Here 3 NOR is requires

(3) OR



Here 2 NOR is required.

(4) Ex-OR

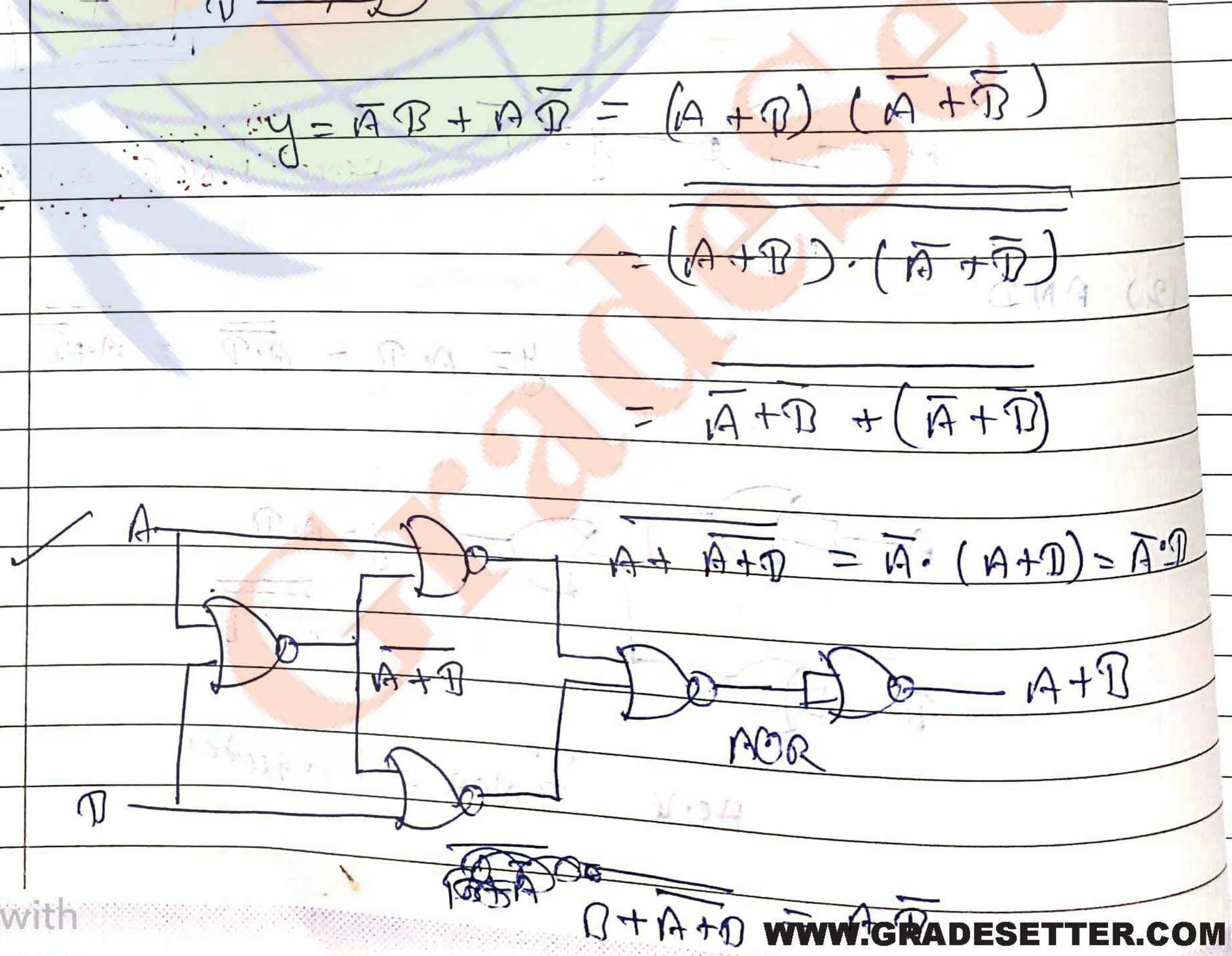


(5)

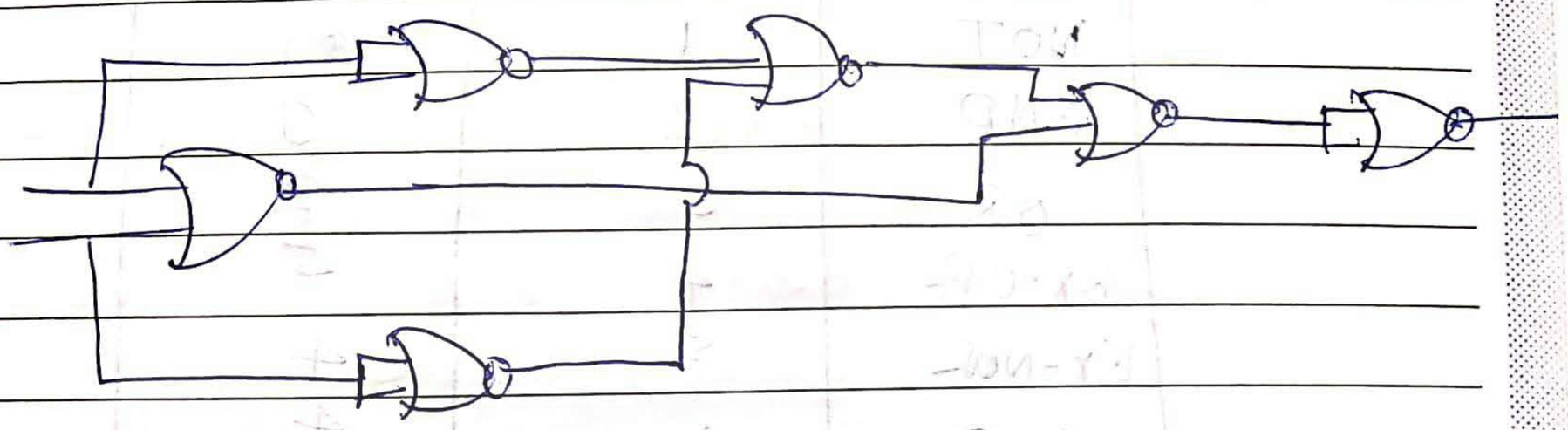
$$y = \bar{A}B + A\bar{B} = (A+B)(\bar{A}+\bar{B})$$

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

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



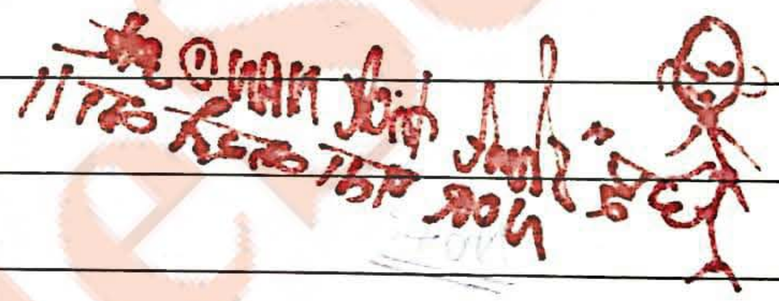
or



5 NOR gate are required.

(5) Ex-NOR

minimum 4 NOR gate are requires.



Page No. 323
Date

	NAND	NOR
		0
		1
NOT	1	1
AND	2	2
OR	3	3
EX-OR	4	4
EX-NOR	5	4
NAND	1	1
NOR	4	

Q) How many NAND (Two input) gate are requires to realise $f = x\bar{y}z$

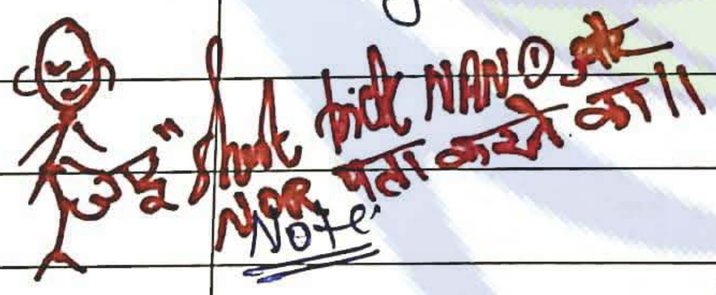
(a) 3 (b) 4 (c) 5 (d) 7

Ans. NAND $= \overline{\cdot}$

$x\bar{y}z = \overline{\overline{x\bar{y}z}} = \overline{\overline{xz \cdot y}}$ ← we not take whole bar

$= \overline{\overline{xz} \cdot y} = \overline{\overline{x \cdot z} \cdot \bar{y}}$

↳ 5 NAND gate are requires.

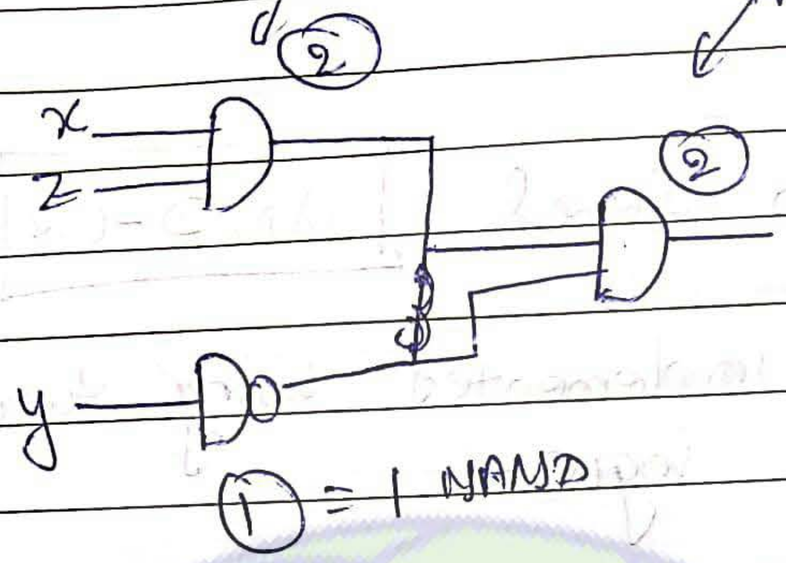


During calculation of minimum number of gates, we change the final formate to the one, which is reliable by available gate.

eg.

to realise using NAND, final expression should be $\overline{\cdot}$ and for NOR +

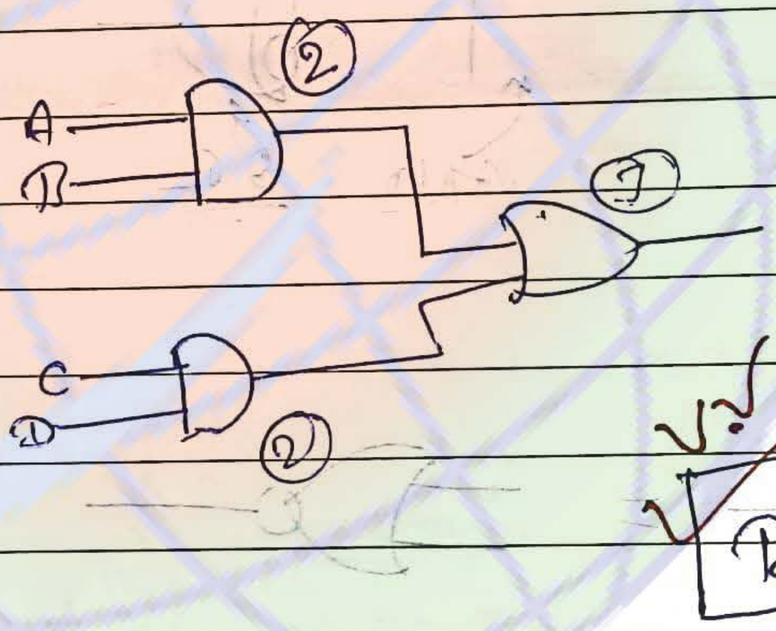
AND = 2 NAND



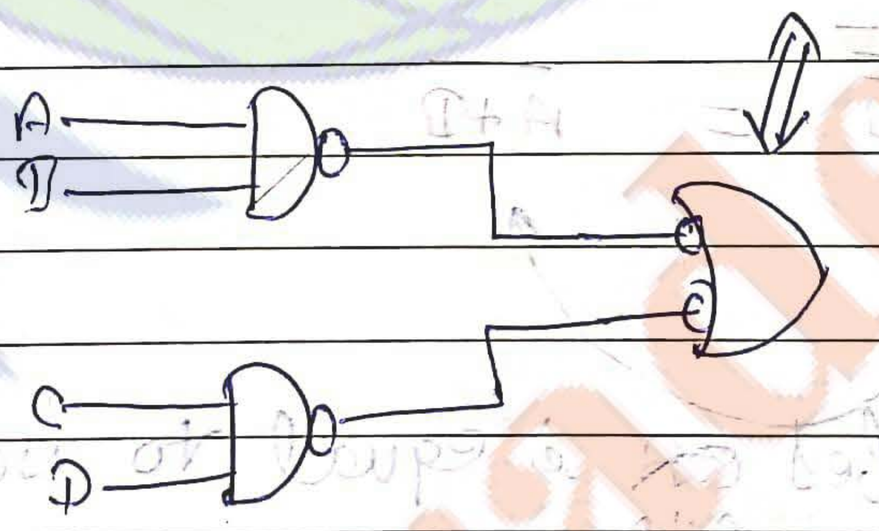
So, minimum 5 NAND is required

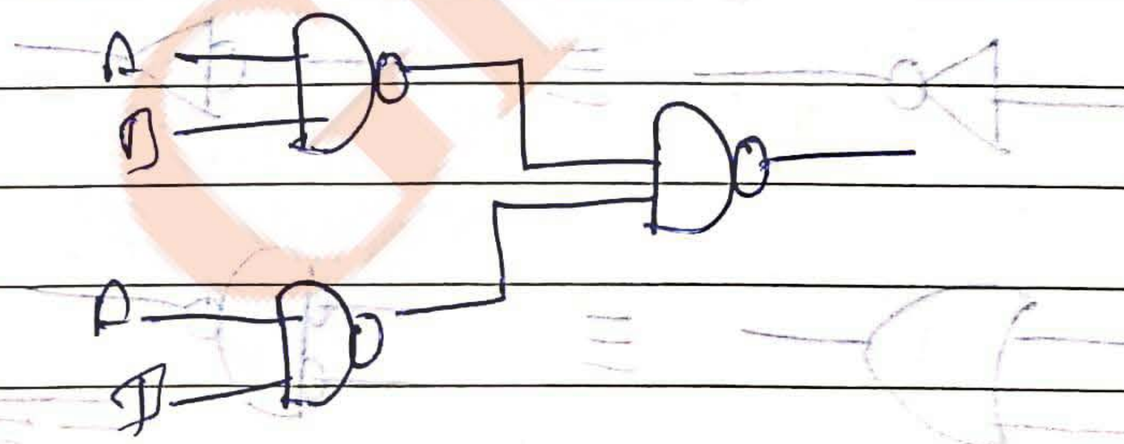
Q) A $f = AB + CD$, what is minimum no. of NAND required to realise the function.

Sol



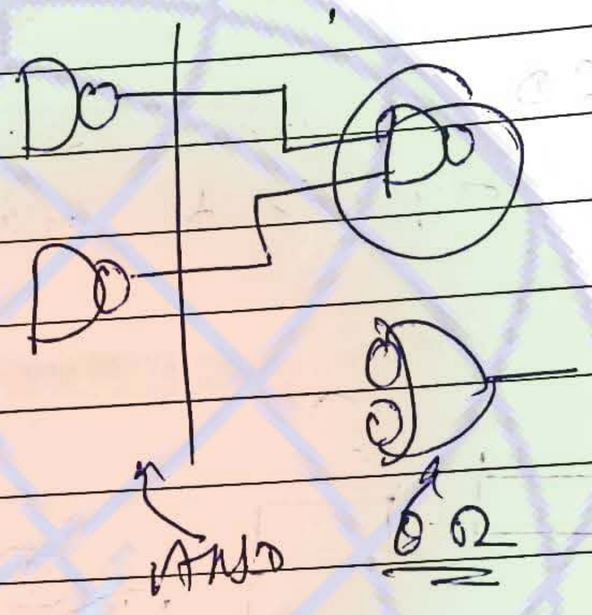
Bubbles OR = NAND gate



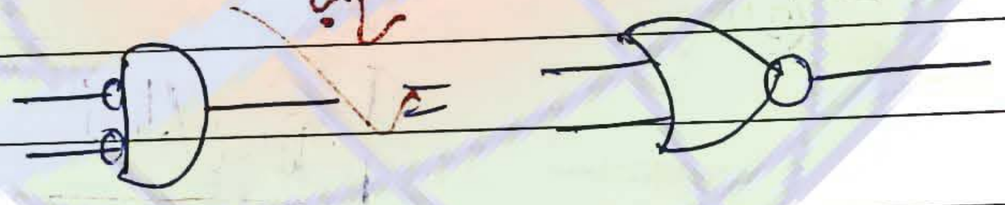


Short-fact

(a) Any two level AND-OR logic can be best implemented using two level NAND-NAND logic



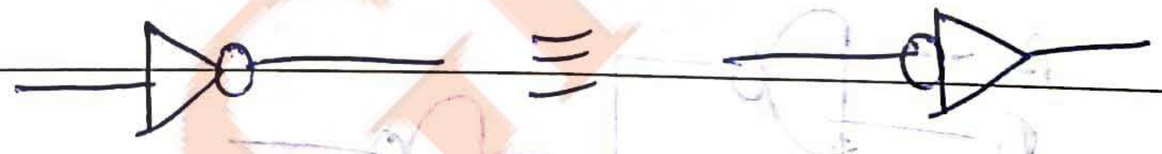
(b)



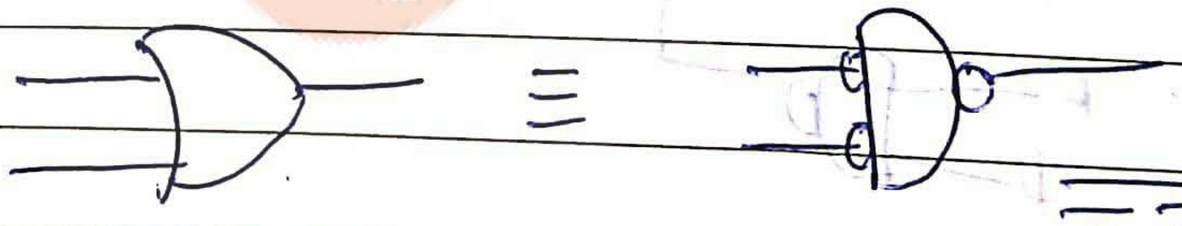
$\overline{\overline{A} + \overline{B}} = \overline{A + B}$

Bubbles AND is equal to NOR gate//

(c) (i)

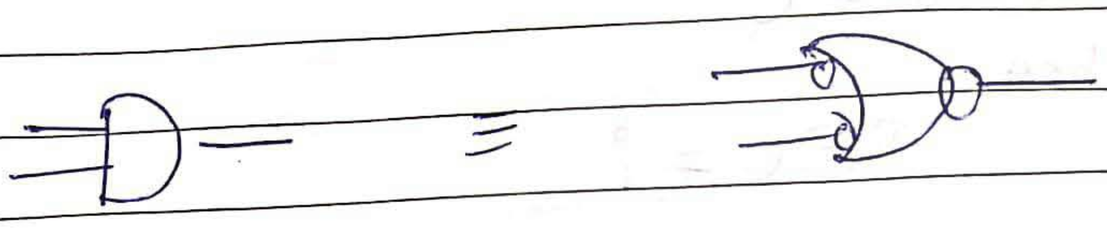


Duality (ii)

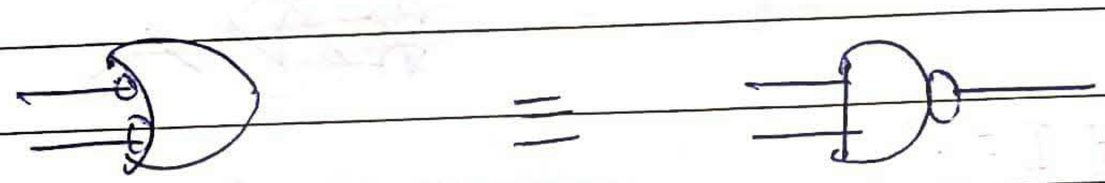


Duality

(iii)

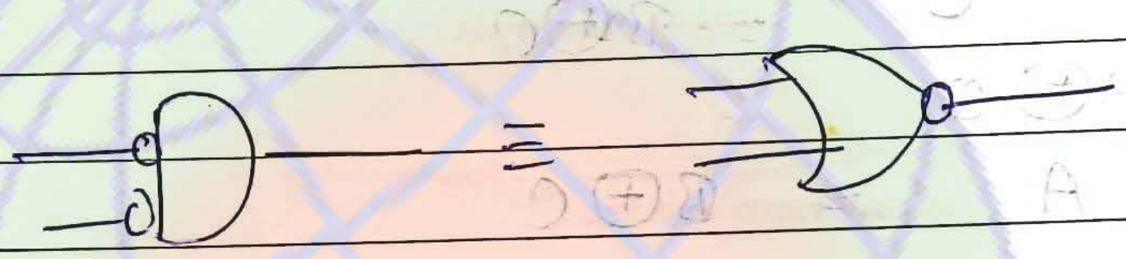


(iv)



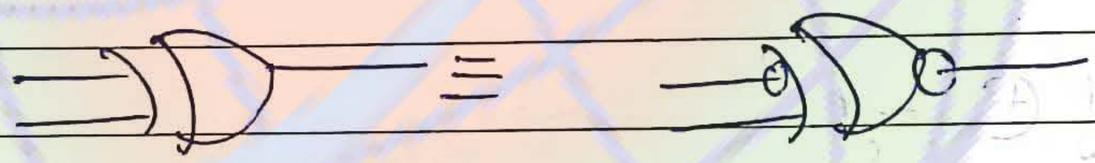
Bubbles OR NAND

(v)

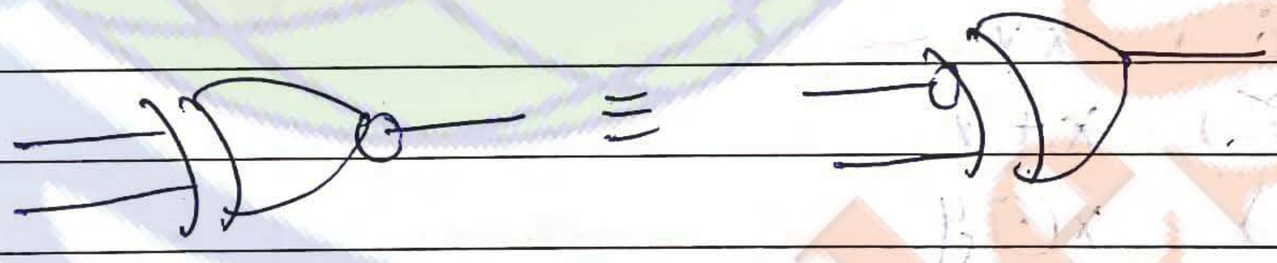


bubbles AND NOR

(vi)



(vii)

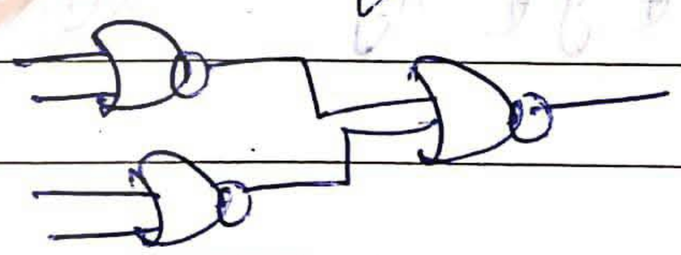


$A \oplus B = \overline{A} \odot B$

Note:

(d) Any two level OR-AND can be best realises using two level NOR-NOR function

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



Q> If $A \oplus B = C$
then
$B \oplus C = ?$

Ans.

Since
$A \oplus B = C$

$A \oplus B \oplus B = C \oplus B$
$0 = B \oplus C$
$A \oplus 0$
$A = B \oplus C$

$\therefore$

$$\boxed{B \oplus C = A}$$

Q> $x \oplus y \oplus xy$

(a) $x \oplus y$
(b) xy
(c) $x + y$
(d) $x \odot y$

Ans:

	x	y	xy	f
	0	0	0	0
	0	1	0	1
	1	0	0	1
	1	1	1	1

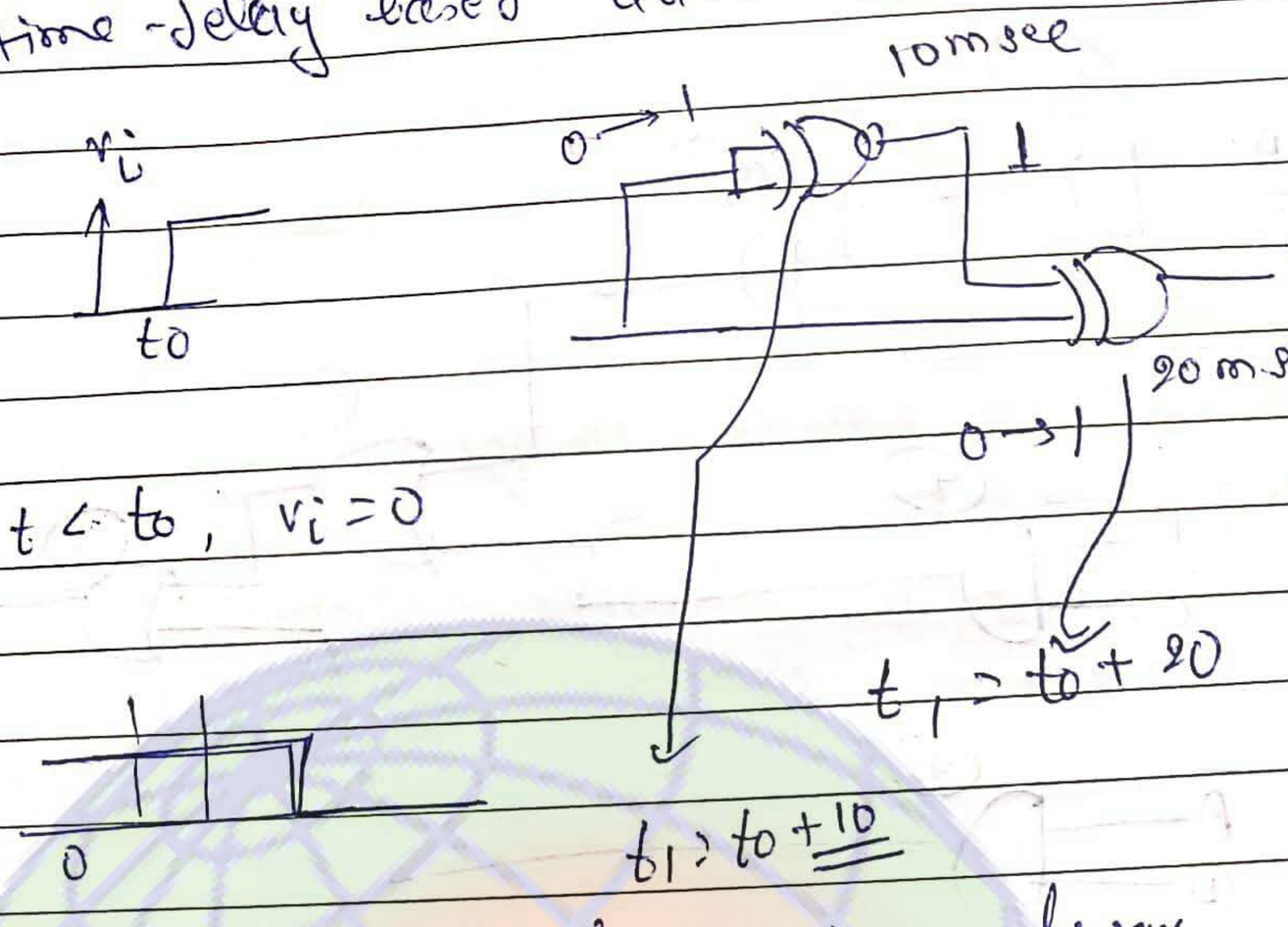
odd ones

$\Rightarrow x \oplus y \oplus xy = x \oplus xy \oplus y = (\bar{x}xy + x\bar{x}\bar{y}) + y$
$= x\bar{y} \oplus y$
$= \overline{x\bar{y}} \cdot y + x\bar{y} \cdot \bar{y}$
$= \bar{x} \cdot y + y + x\bar{y}$
$= x + y$

apply distributive

$x \oplus y \oplus x \cdot y$
$x \oplus$

Gate level (time-delay based Questions):-



In this type of question, always consider the time delay and accordingly see the effect change.

Q) what is the minimum number of two input nand gate requires to realise function

$$y(A,B,C) = \pi(0,5,6,7)$$

Ans. NAND $= \overline{\cdot}$

↳ i.e SOP is better for realise NAND"

Note: NAND does not follow associative law

$$y(A,B,C) = \pi(0,5,6,7)$$

$$= \Sigma(1,2,3,4)$$

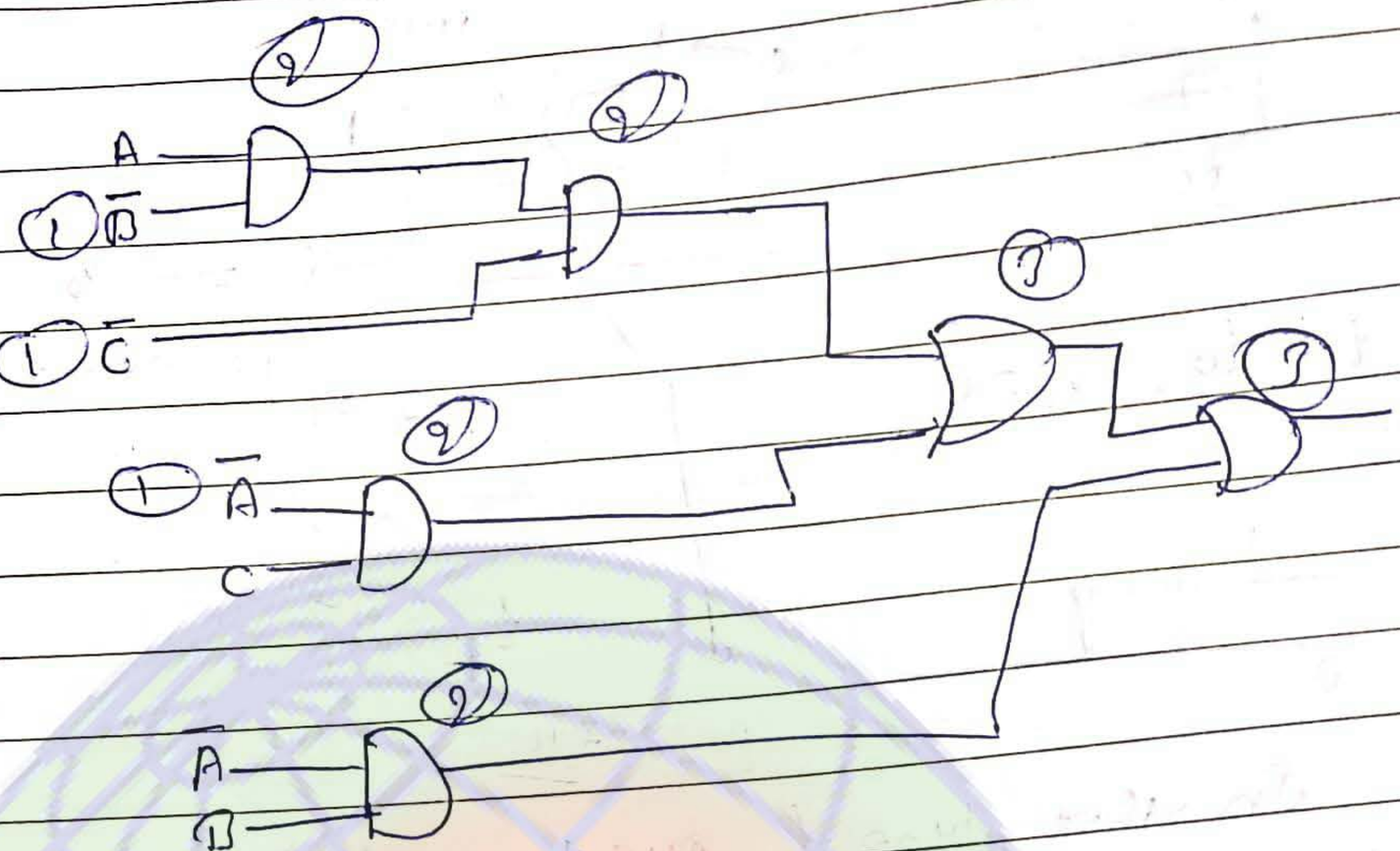
$$= \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C}$$

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

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

$A\bar{B}C$

Page No. 540
Date



17 NAND of two inputs is require (3)

Realisation of Logic

NOTE: To realise functions, using NAND gate, dot(·) can not be realised directly

Hence for any such realisation, two times complements are necessary.

$\overline{A \cdot B}$ → 2 bar

Using Dot! (14)

3.5)

(1)

(2)

Page No. _____
Date _____

3.5) Gray Code

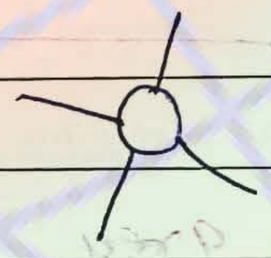
(1) Gray code is a unit distance code, which is also called cyclic code.

(2) Unit distance code has a property, that for successive number's representation, only one bit differs.

for eg.

3 in gray code is 0 1 0
2 in gray code is 0 1 1 → only 1 bit differ

(3) Gray code is used in
(a) K-map
(b) shaft-angle encoding.



hand shaft of robot movement
- requires Precision + speed

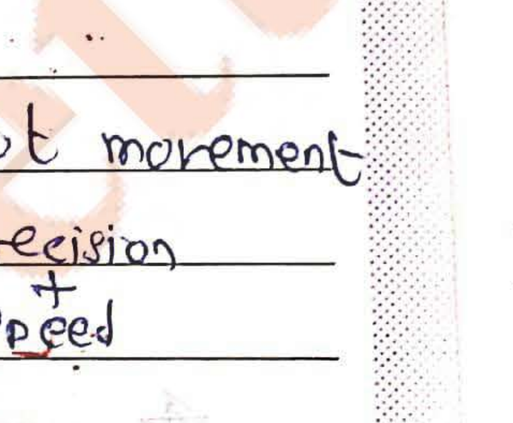
6 → 0 1 1 0
7 → 0 1 1 1
8 → 1 0 0 0 → all 4 bit change

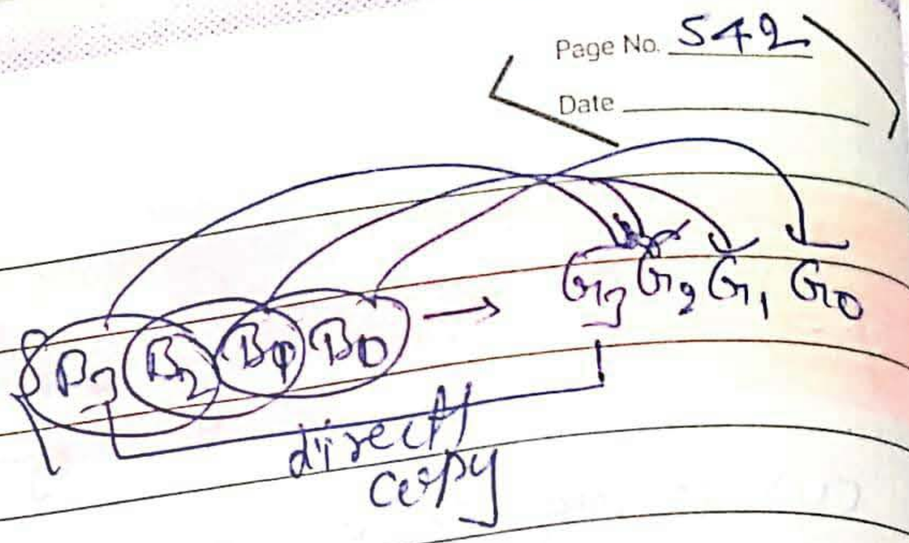
directly not

(4) Binary code → Grey code

Binary code	Grey code
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

ex-or
ex-or
ex-or





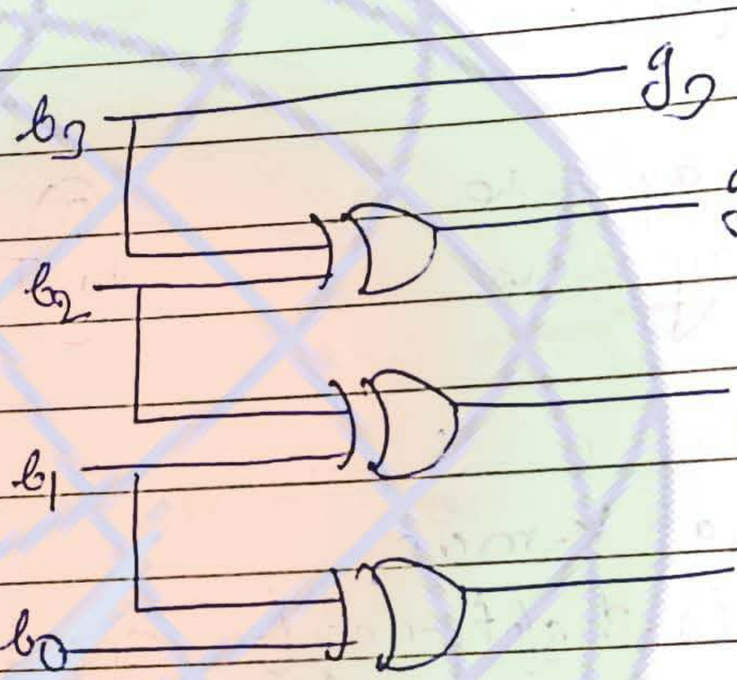
* n = 4 bit

$G_3 = b_3$

$G_2 = b_3 \oplus b_2$

$G_1 = b_2 \oplus b_1$

$G_0 = b_1 \oplus b_0$



Binary ⟶ Grey

(Binary to grey converter)

* Grey to Binary code:-

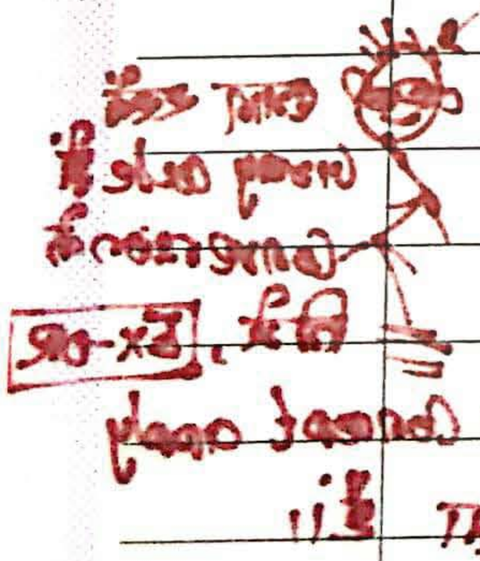
$b_3 = g_3$

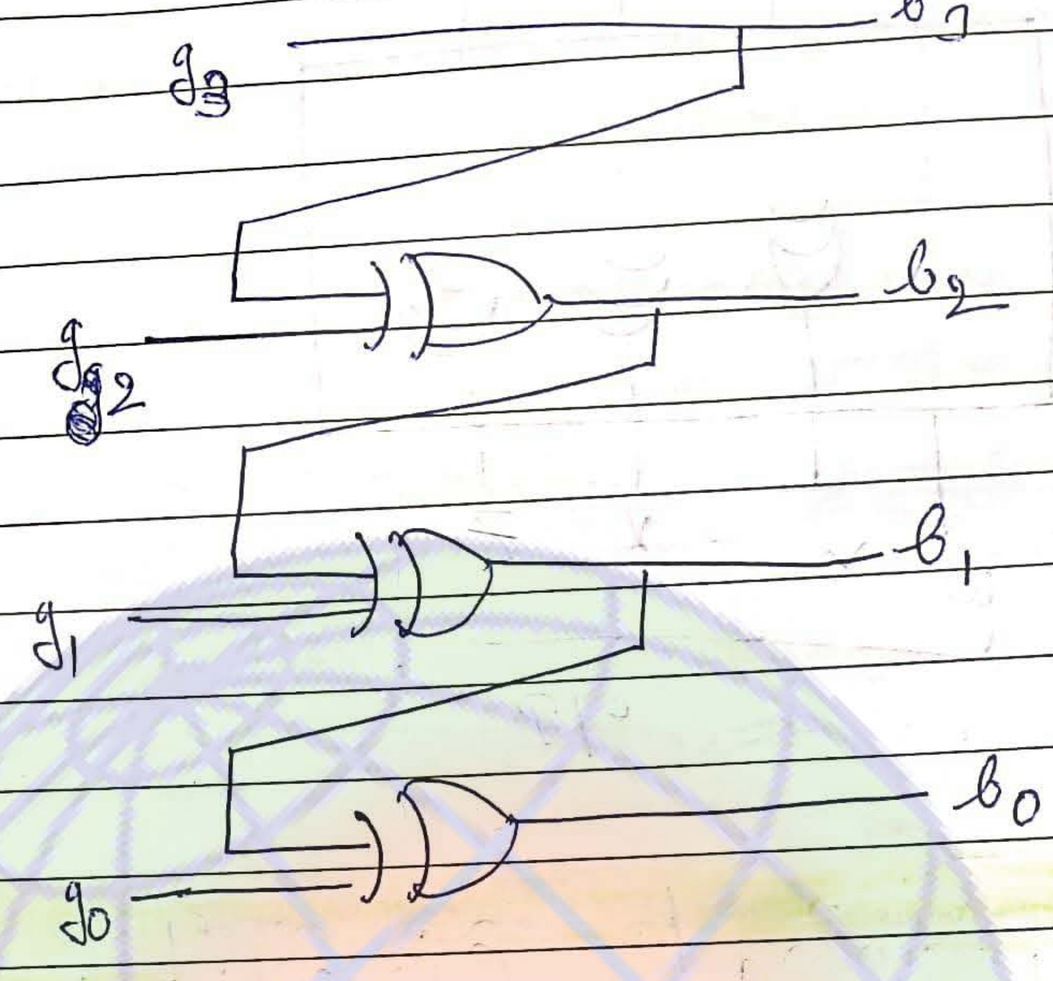
$b_2 = b_3 \oplus g_2$

$b_1 = b_2 \oplus g_1$

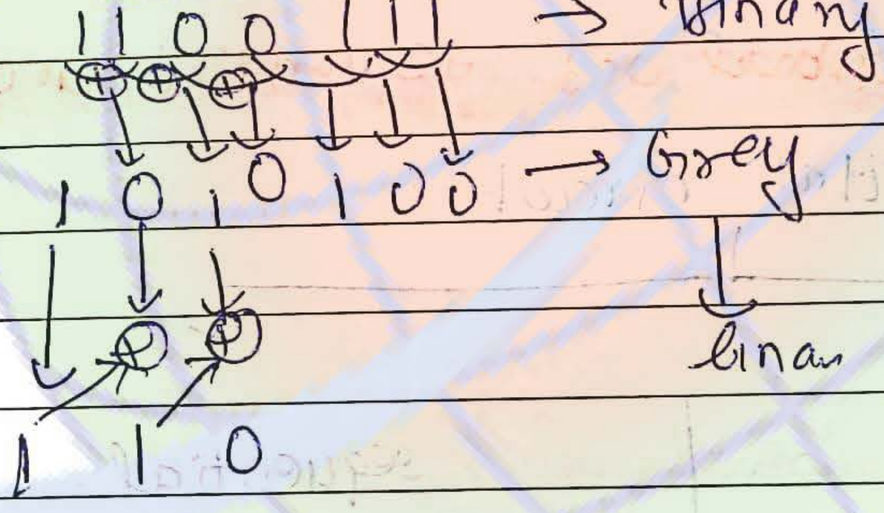
$b_0 = b_1 \oplus g_0$

$G_3 G_2 G_1 G_0 = b_3 b_2 b_1 b_0$





eg.>

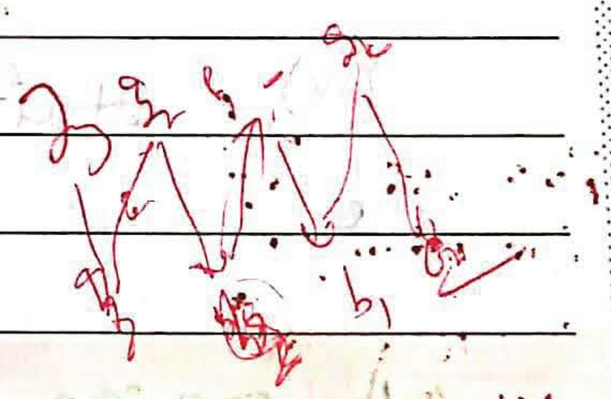


Note:

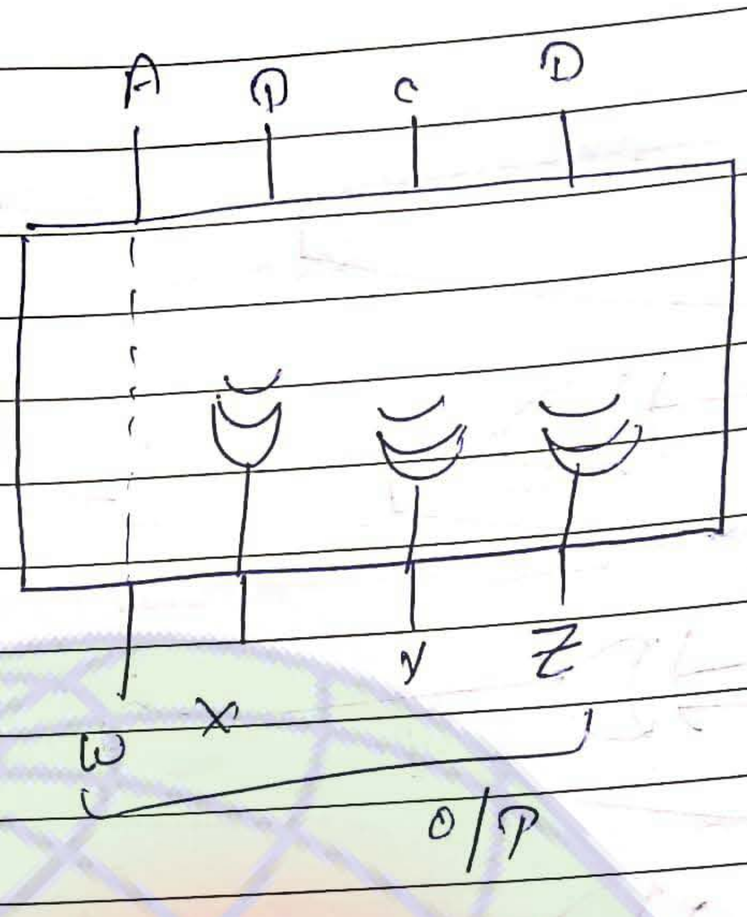
Grey code is used in shafts-angle measurment and ~~transmision~~.

because

Probability of error is reduced.



Q)



4. ☆ Combitional circuits:-

4.0) Diffb b/w Combinational and sequential circuits.

Digital circuit

Combinational	sequential
(i) No memory	(i) memory ✓
(ii) No feedback	(ii) No feedback
(iii) Present o/p depends on present input	(iii) Present output depends past output and Present input.
(iv) : stable	(iv) less stable

4.1) Design procedure for combination circuit.

* Design procedure:-

specification → I/o variables → assign symbol to each variable → truth table → SOP/POS → Implement

Page No.
Date

4.2) Arithmetic Circuits:-

* Combitional circuits may be arthematical and non arithmetical

- Arithmatic → Half adder (HA) / Fall Adder (FA)
  H (HS) (FS)
  comparator
- Non arthematic → mux / Demux
  Encoder / Decoder

~~Arithmetic circuit~~

★ Arthematical Circuits:-

(1) HA (Half Adder):-

Step's to design any combinational ckt:-

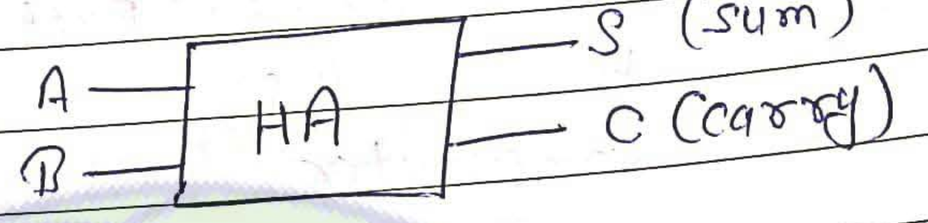
(1) Identify inputs and outputs.
(2) Draw the truth table
(3) write logical expression.
(4) minimize expression
(5) Draw the ckt.

A+B:-

A	0	0	1	1
B	0	1	0	1
S	0	1	1	① 0

carry ↓ ↘ Sum

(I) Half adder adds two numbers, which are one bit number's.



i/p's → A and B
o/p's → S and C

(II) Truth table

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(III) logical expressions

sum =

$S = \Sigma m(1,2)$ — min-term

$S = \pi\{0,3\}$ — max-term

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

$S = A \oplus B$

$S = A \oplus B$

Carry

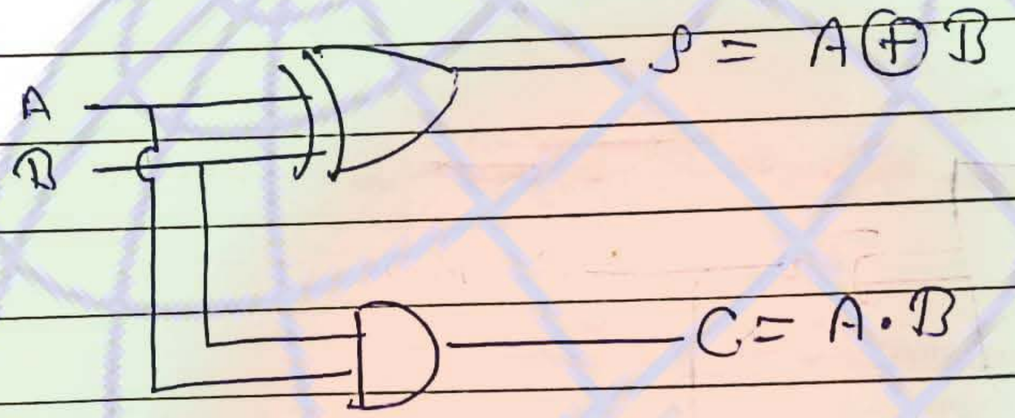
0	0	0
0	1	0
1	0	0
1	1	1

$C = A \cdot B$

$= \Sigma m\{3\}$

$= \pi\{0,1,2\}$

(V)



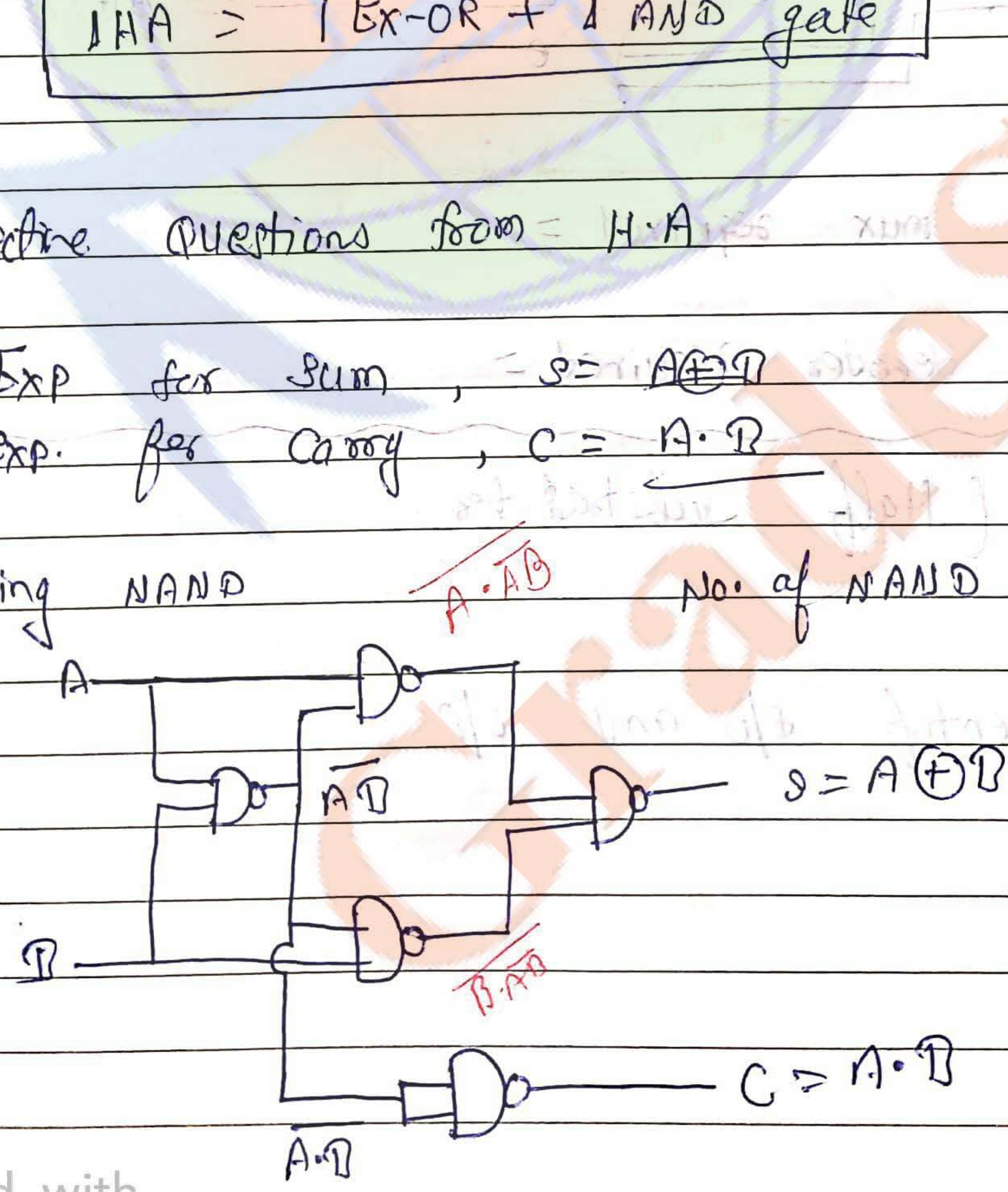
HA = 1 Ex-OR + 1 AND gate

Objective Questions from H.A

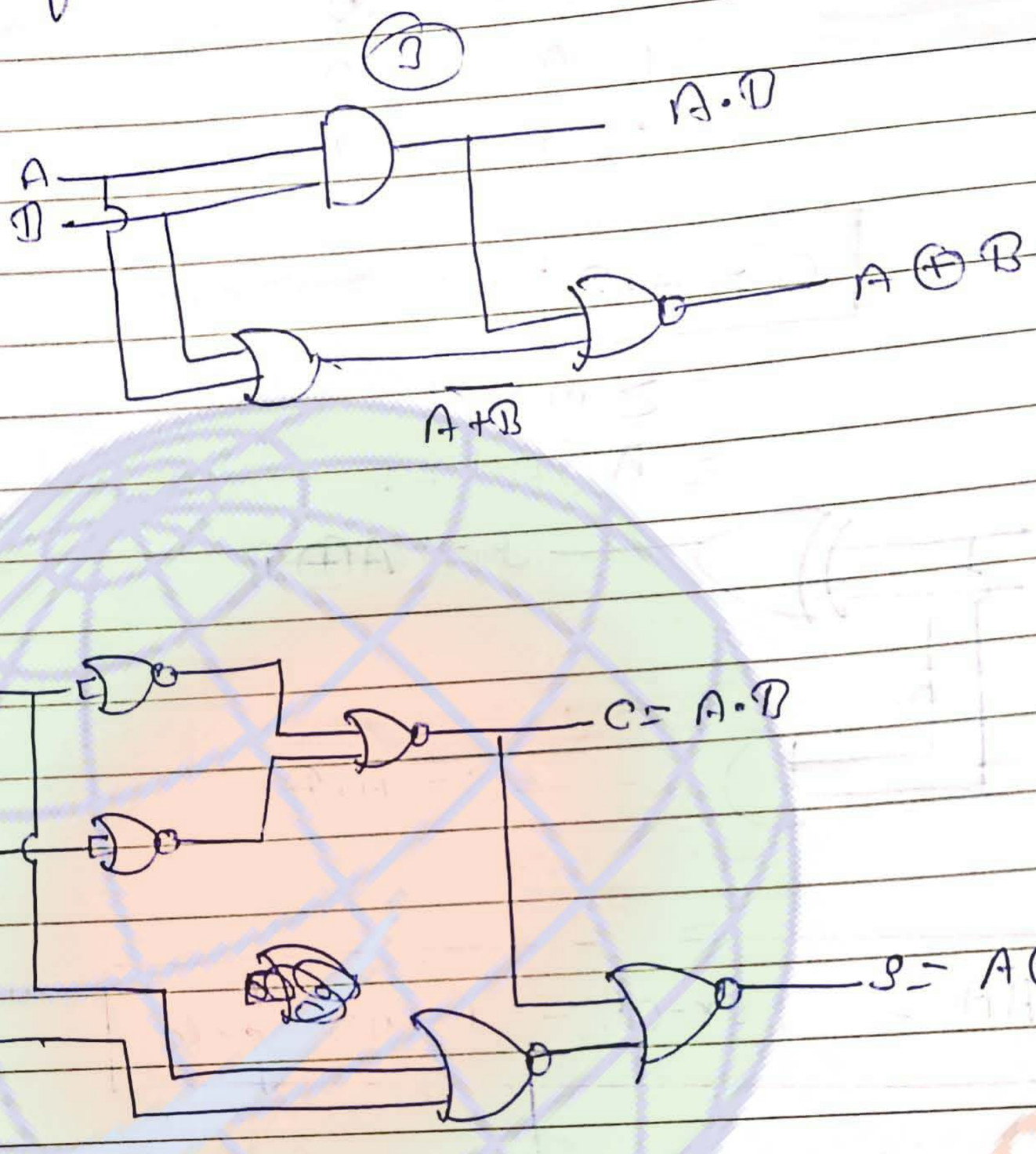
(1) Exp for Sum, $S = A \oplus B$

(2) Exp. for Carry, $C = A \cdot B$

(3) Using NAND — No. of NAND requires = 5



(4) No. of NOR required = 5



(5) No. of mux required =

(6) No. of decoder required =

---

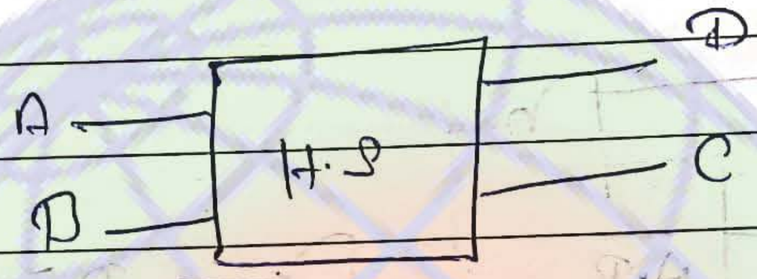
(2) HS (Half subtractor)-

Steps:

(1) Identify i/p and o/p

(i)

	1 0			
A	0	0	1	1
B	0	1	0	1
D	0	1	1	0
Borrow ← C	0	1	0	0



(ii) Truth tables:

A	B	C	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

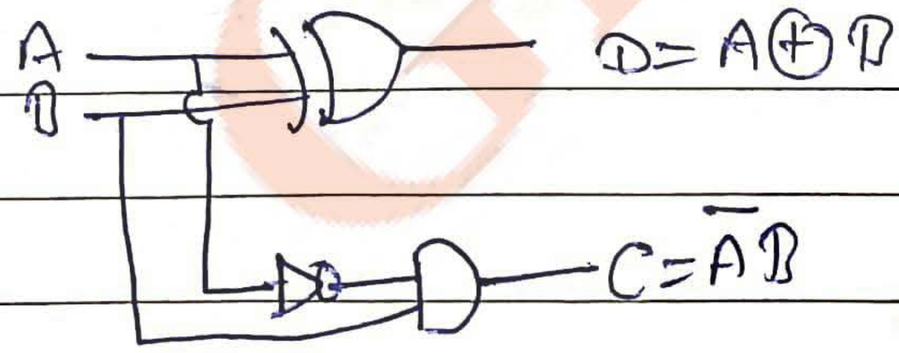
(iii) Expression

$$D = A \oplus B$$

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

(iv) ~~circuits~~ already minimised.

(v) circuit

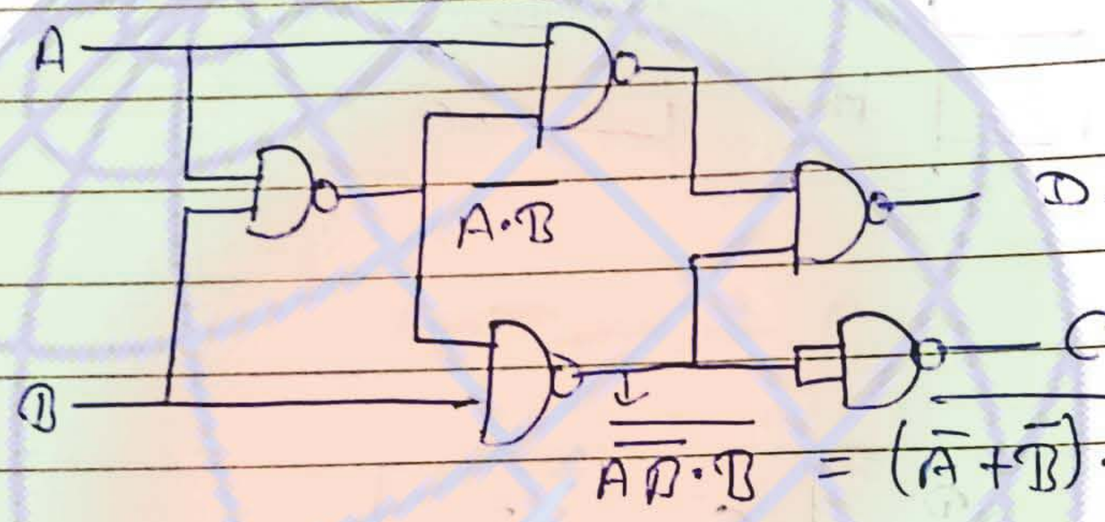


Objectives:

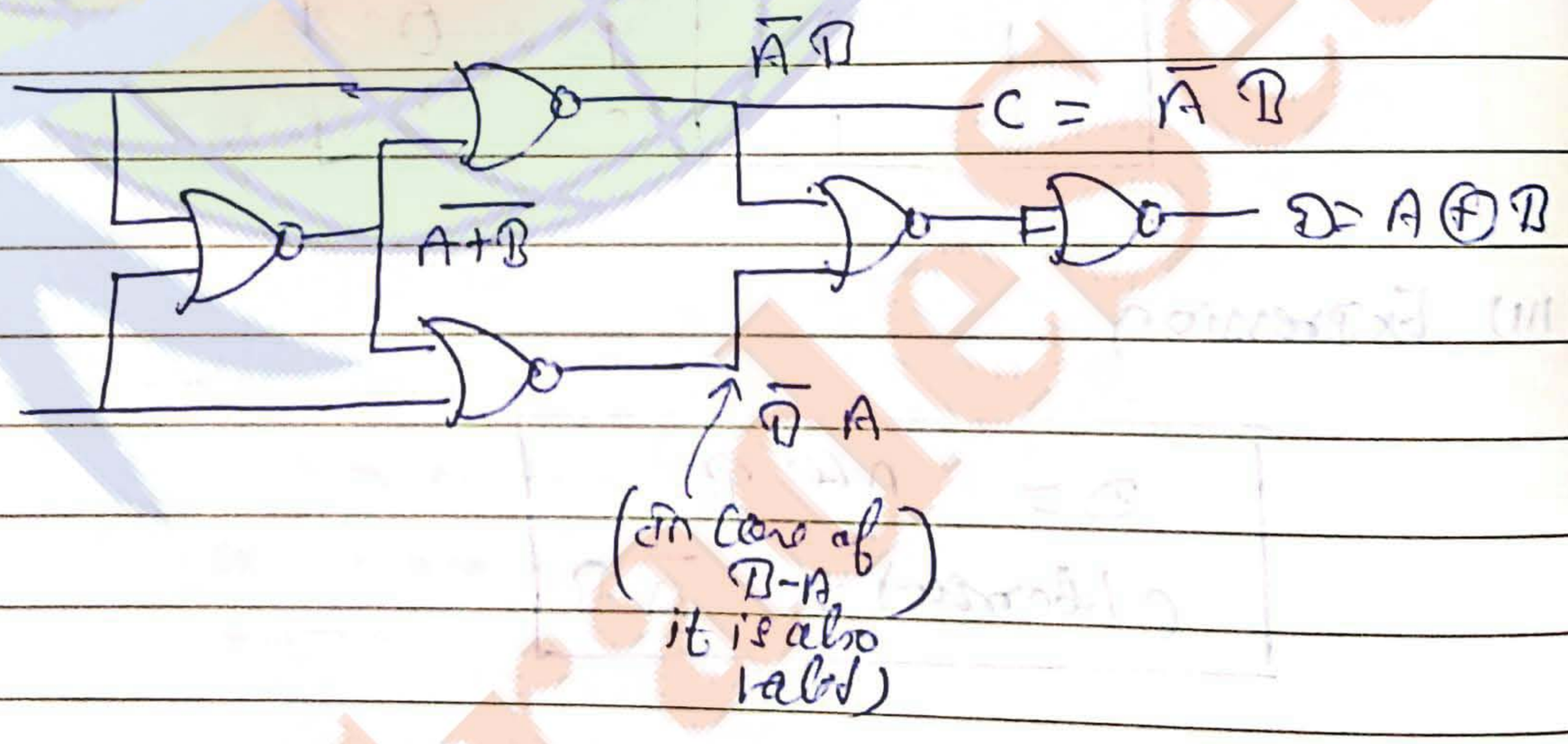
(1) Expression for difference (D): $D = A \oplus B$

(2) Exp. for borrow : $C = \overline{A} \cdot B$

(3) No. of NAND required = 5



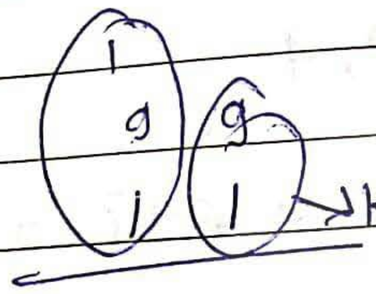
(4) No. of NOR required = 5



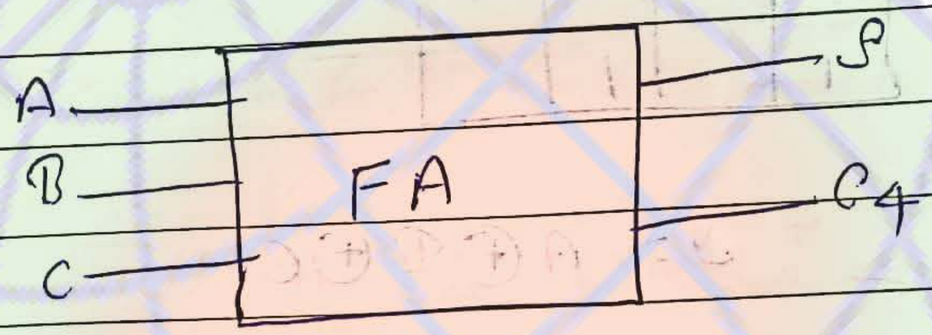
(5)

(6)

(3) Full Adder (FA)



(i) full adder adds three numbers A, B, C, results in S, Cy



(ii) Truth table

A	B	C	S	Cy
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(iii) write the expression for sum and carry

$S = \sum m\{1, 2, 4, 7\}$ min-term

$= \prod m(0, 3, 5, 6)$ max-term

$= A \oplus B \oplus C$

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

Page No. 552
Date

$C_y = \Sigma \{ 3, 5, 6, 7 \}$

$= \pi \{ 0, 1, 2, 4 \}$

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

(iv) To minimise

(sum) S:

	1		1
1		1	

$S = A \oplus B \oplus C$

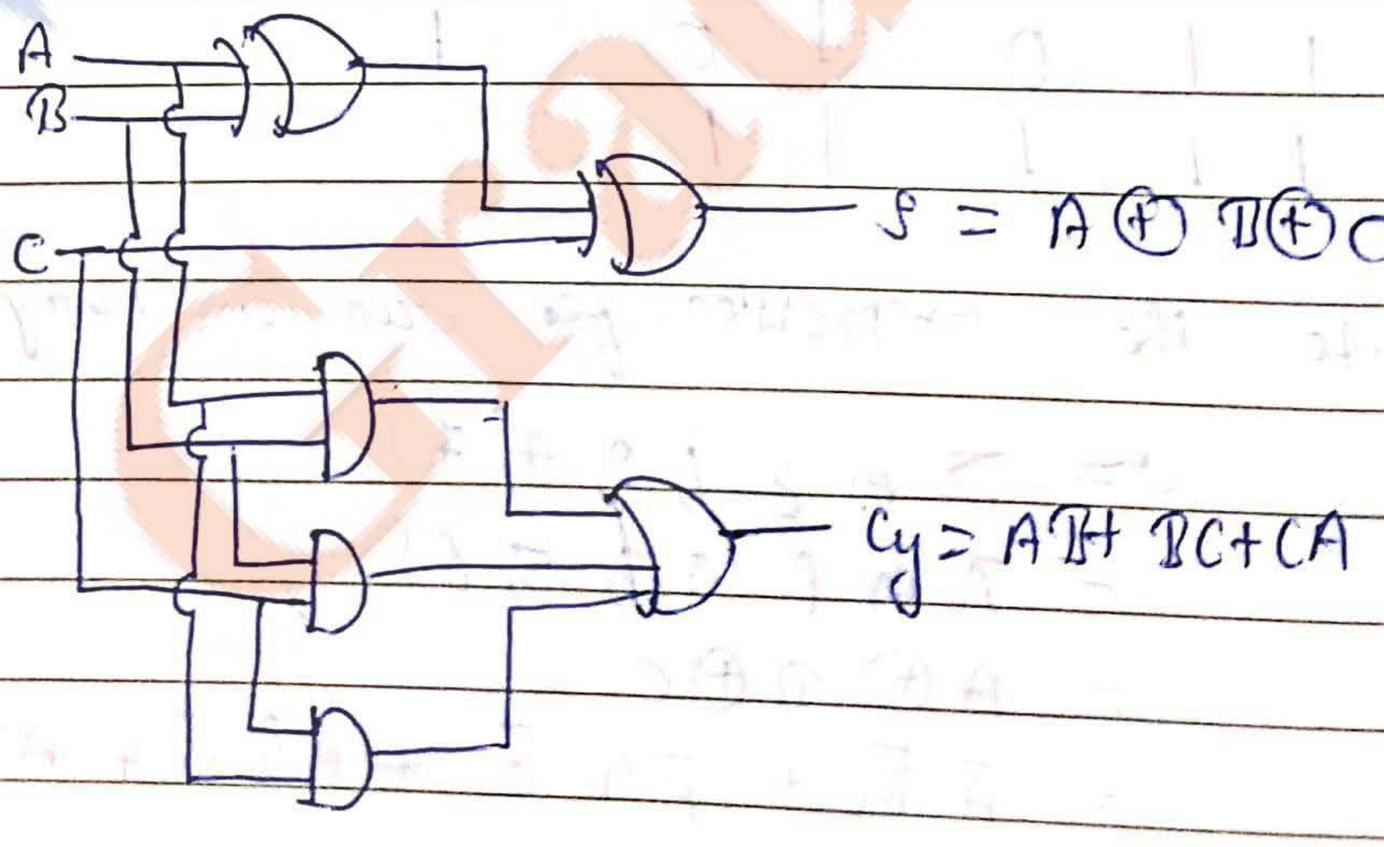
(carry) C:

0	0	1	0
0	1	1	1

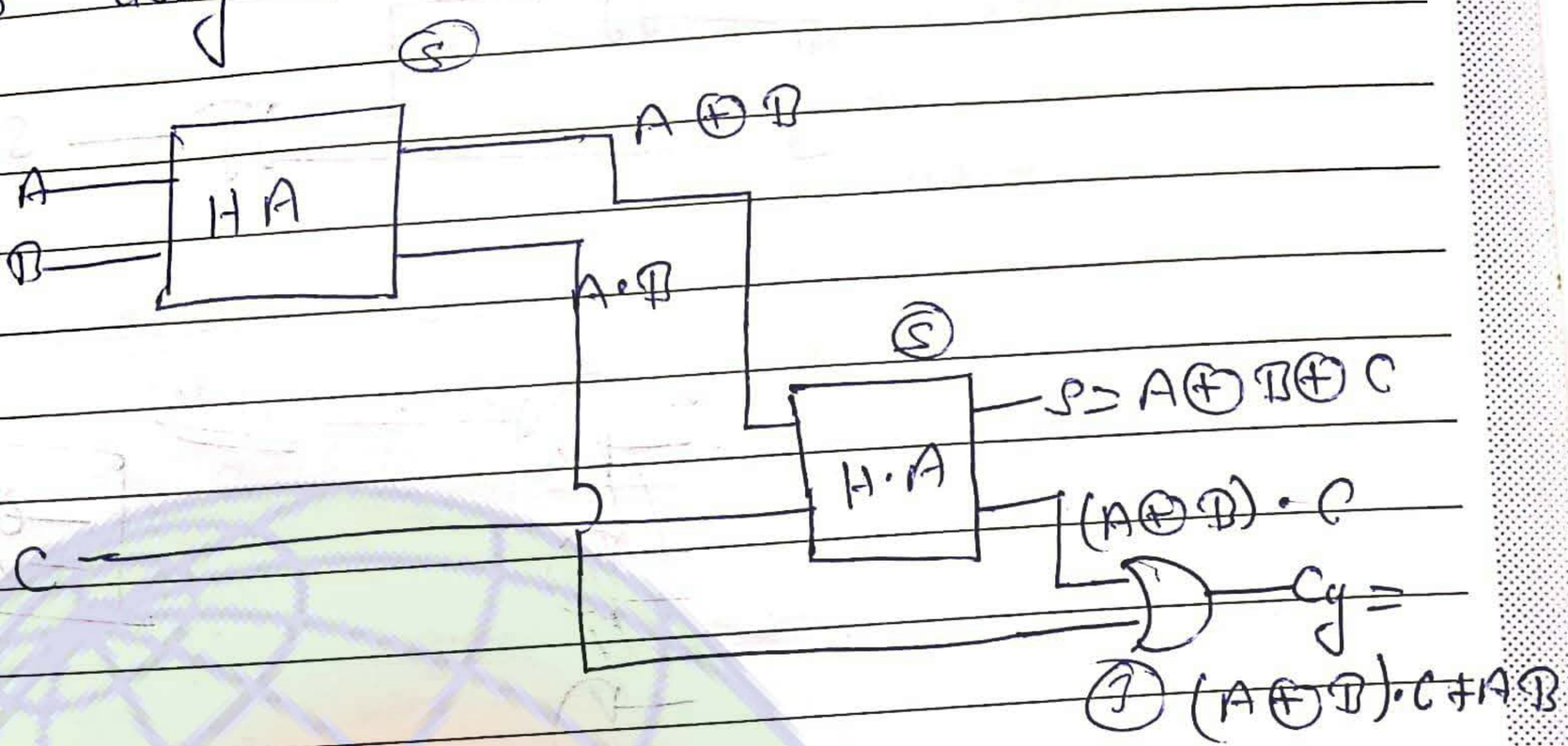
self dual ~~of~~ A ~~by~~

$C_y = AB + BC + CA$

(v) Draw the circuits:



* Full adder using half adder:-



$$C_y = \sum (3, 5, 6, 7)$$

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

$$\boxed{C_y = (A\bar{B} + A\bar{B})\cdot C + AB}$$

↳ Alternate expression for carry

$$\boxed{1\,FA = 2\,HA + 1\,OR\ gate}$$

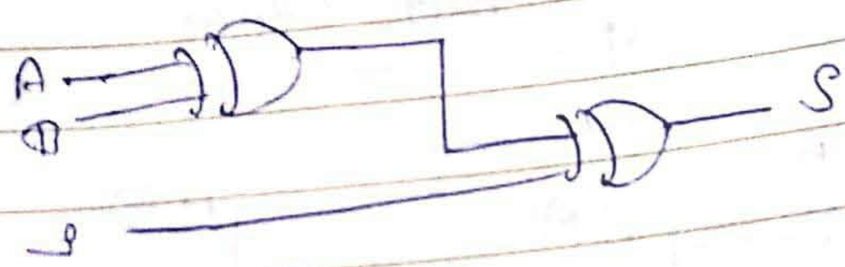
Objective:

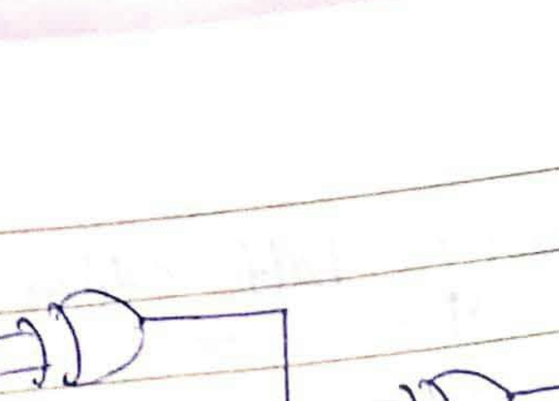
(1) Exp. for Sum, $S = A \oplus B \oplus C$

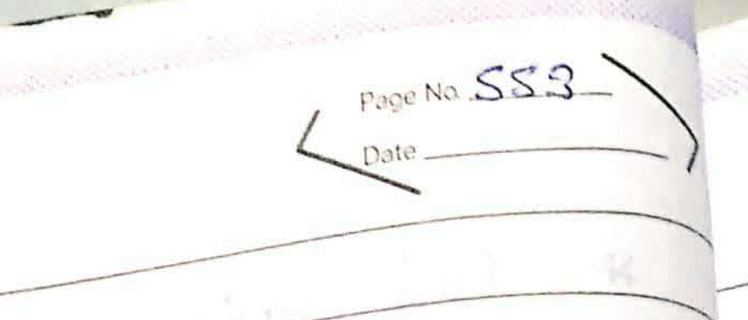
(2) Exp. for carry, $C_y = AB + BC + CA = (A \oplus B)\cdot C + AB$

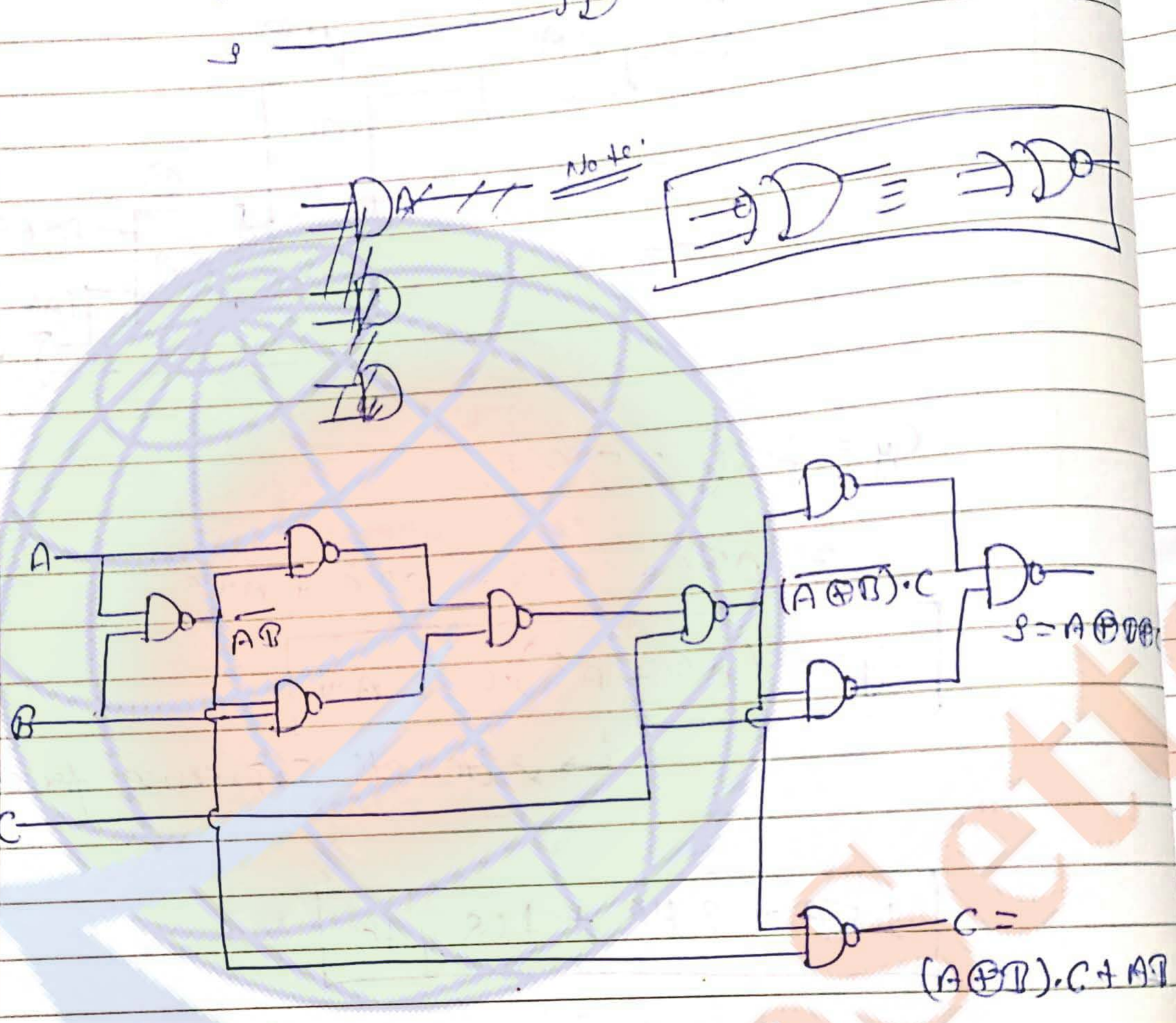
(3) No. of half adder and OR gate requires = 2 HA + 1 OR gate

(4) No. of NAND required = 9









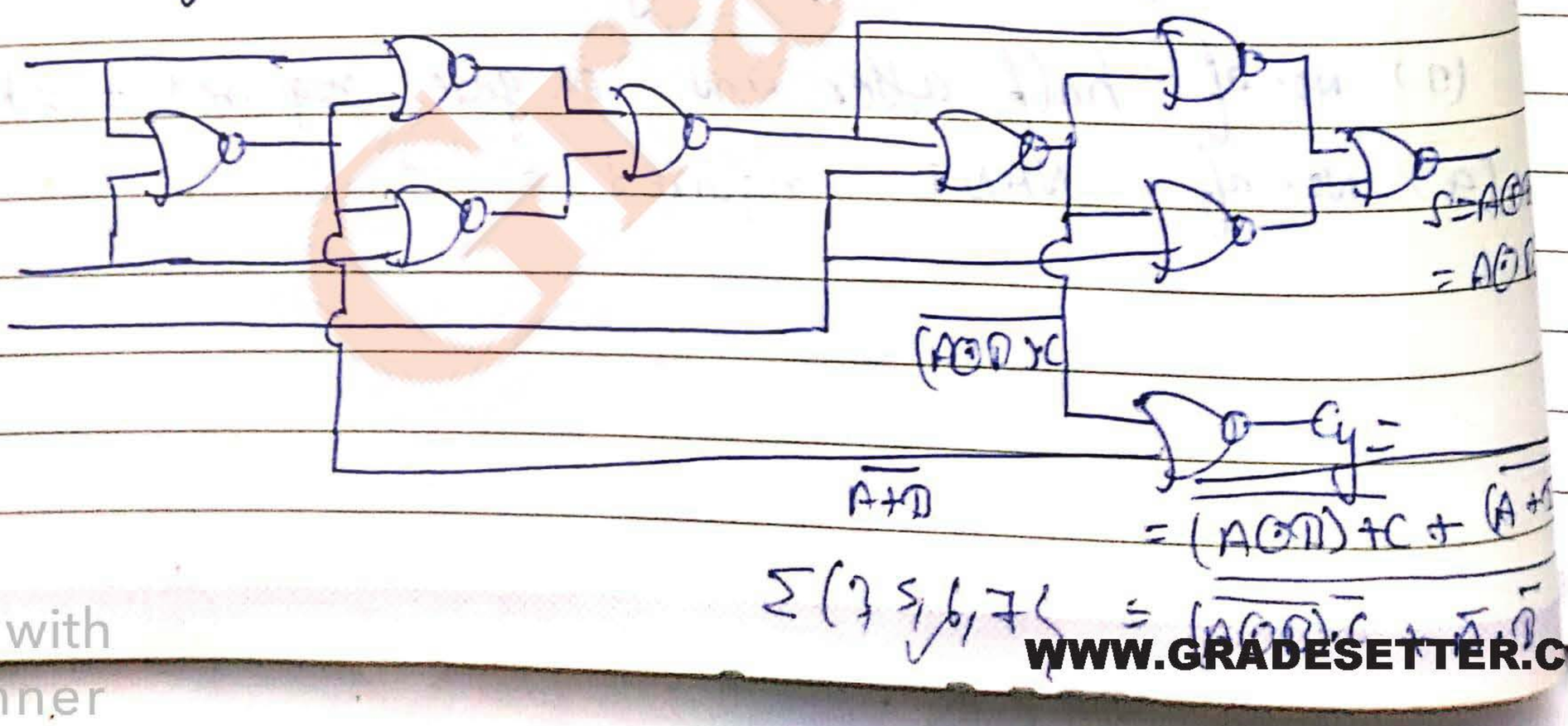
vvi

(6)

No. of NAND required = 09

(5) No. of NOR required = 09

(7)



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

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

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

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

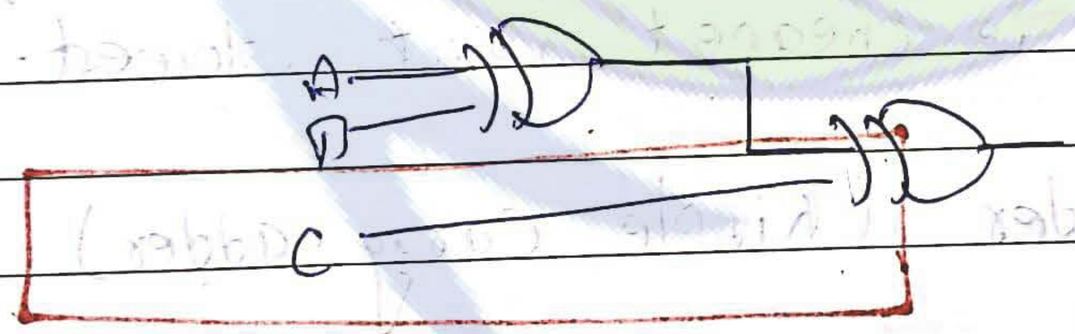
$$= \Sigma(2, 4, 0, 1)$$

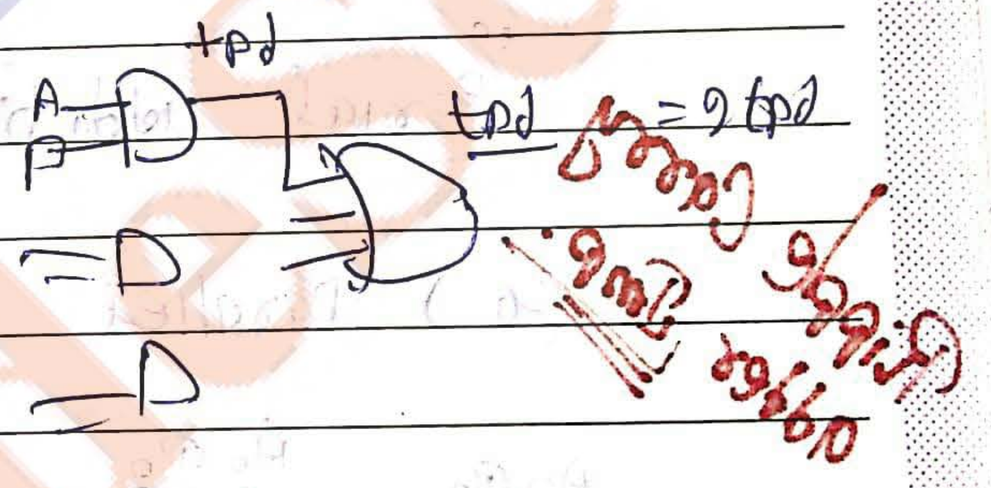
$$= \Sigma\{3, 5, 6, 7\}$$

vvi

(6) ~~2 tpd = time delay.~~

In full adder, if propogation delay of one gate is tpd, then to provide sum or carry as output minimum 2 tpd





(7) ~~Full adder~~ One full adder may add three numbers, but to add group of bits multiple full adder's are required. If only one full adder is used to add group of bits, the process of addition becomes very slow.

(8) Adding Group of bits:

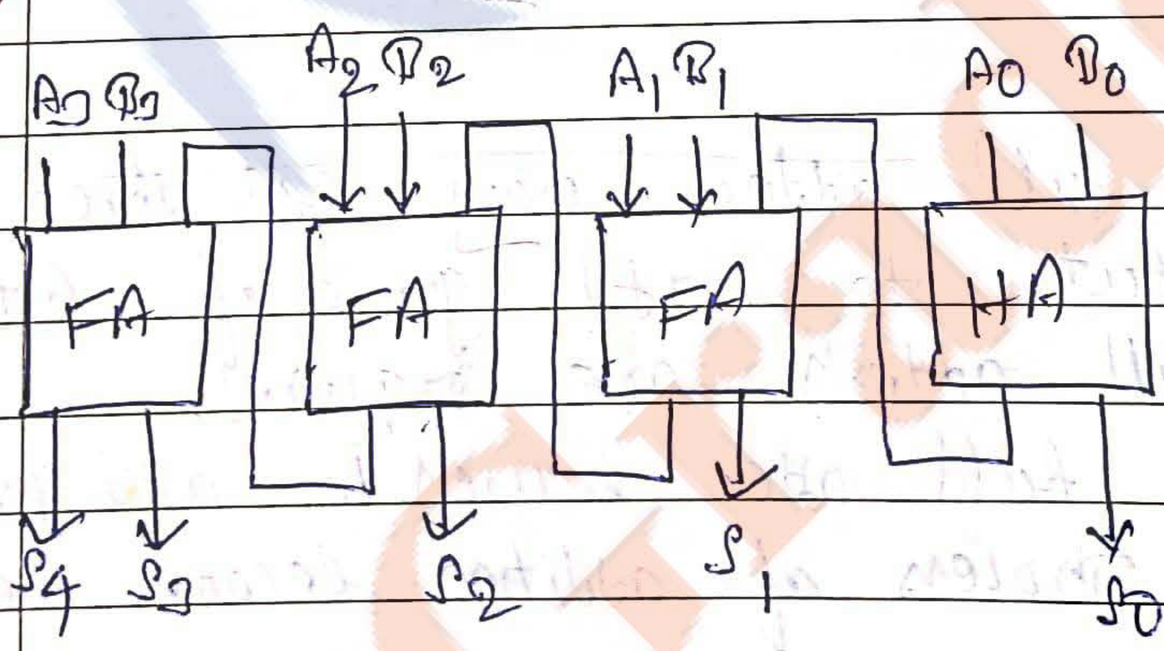
(a) Serial adder (one full adder is required)

C3 C2 C1
A3 A2 A1 A0
B3 B2 B1 B0
S4 S3 S2 S1 S0

"Serial addition is cheapest, but slowest."

Ripple Carry adder Imp.

(b) Parallel Adder (Ripple Carry adder)



C3 C2 C1
A3 A2 A1 A0
B3 B2 B1 B0
S4 S3 S2 S1 S0

Hence, in Paraller addition of 4 bits, we need (i) 3 FA + 1 HA
or
(ii) 4 FA
or
(iii) 1 HA + 3 OR

General formula:-

Hence, for n-bit addition

(n-1) FA + 1 HA

or

'n' FA

or

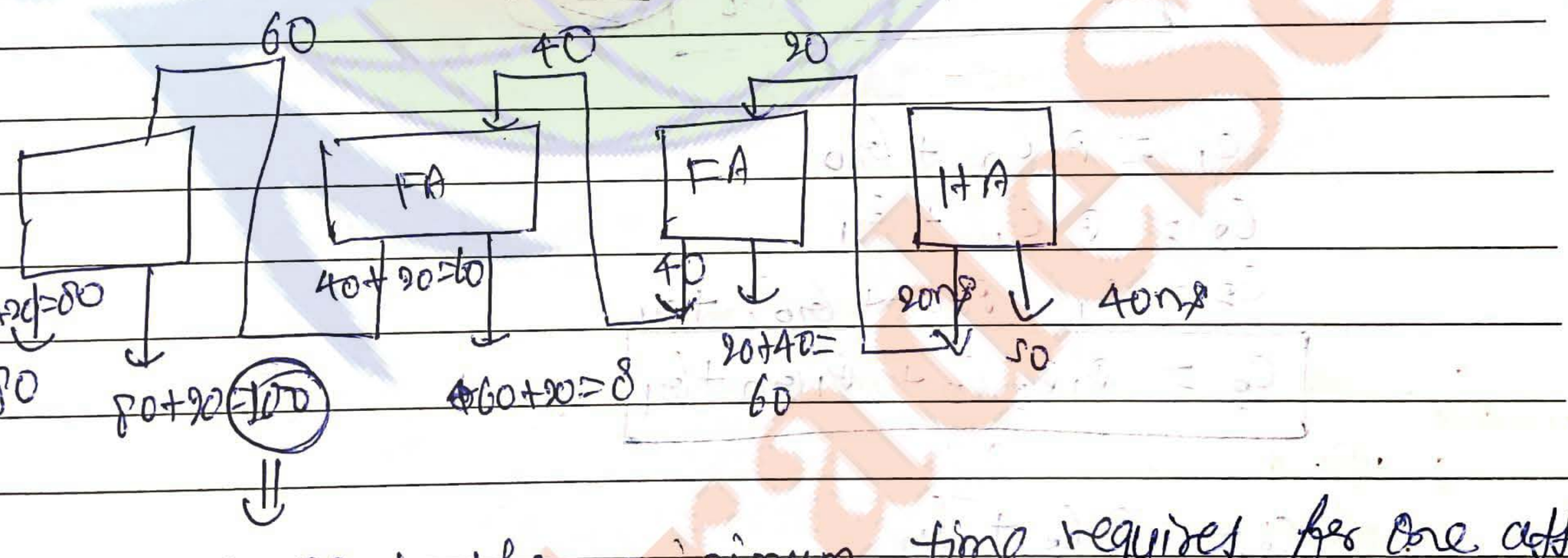
(2n-1) HA and (n-1) OR gate.

Note:

7(2 HA + 1 OR) + 1 HA = 7 HA + 7 OR gate //

Q> In a four bit ripple carry adder, Four full adder's are requires. each full adder Produces 40 nsec for sum and 20 nsec delay for carry. what is maximum rate of addition.

Ans



100ns is the minimum time requires for one addition

So,

1 Addition = 100 ns

∴ Rate of addition = $\frac{1}{100 \times 10^{-9}} = 10^7$ addition/sec.

Page No.
Date

Parallel adder also produces delay because carry process serially

to

overcome this problem,

" look-Aheas carry adder is used "

(iii) look-Aheas carry adder.

look Aheas carry adder is the fastest adder

$$C_1 = (A_0 \oplus B_0) \cdot C_0 + A_0 B_0$$

let

$$P_i = A_i \oplus B_i$$

$$G_i = A_i \cdot B_i$$

$$S_i = P_i \oplus C_i$$

$$\boxed{C_{i+1} = P_i C_i + G_i}$$ note

$$C_1 = P_0 C_0 + G_0$$

$$C_2 = P_1 C_1 + G_1$$

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

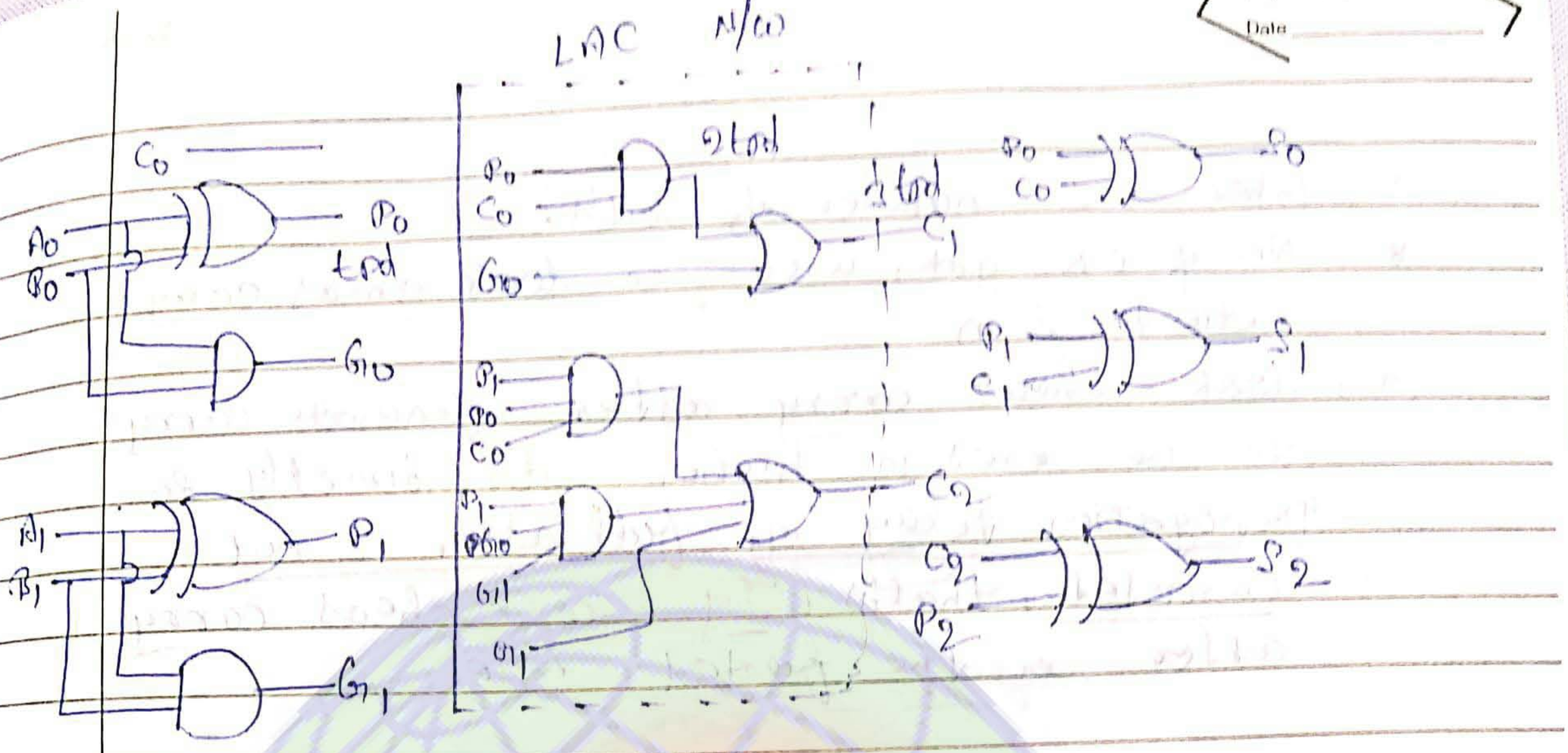
$$\boxed{C_2 = P_1 P_0 C_0 + P_1 G_0 + G_1}$$

$$C_3 = P_2 C_2 + G_2$$

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

$$\boxed{C_3 = P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2}$$

$P_0 = A_0 \oplus B_0$	$G_0 = A_0 B_0$	$S_0 = P_0 \oplus C_0$
$P_1 = A_1 \oplus B_1$	$G_1 = A_1 B_1$	$S_1 = P_1 \oplus C_1$



- In lookAheas carry adder carry is generated by two level, AND-OR gate circuit

* In look ahead carry adder, if all logic gates have same ~~Propa~~ propagation delay, then to provide carry 3 tpd delay è produced & to provide sum, hold 4 tpd delay è produced.

* No. of AND gates requires in look aheas carry network of look ahead carry addor equal

→ $\left\lceil \frac{n(n+1)}{2} \right\rceil$

where 'n' is number of bits.

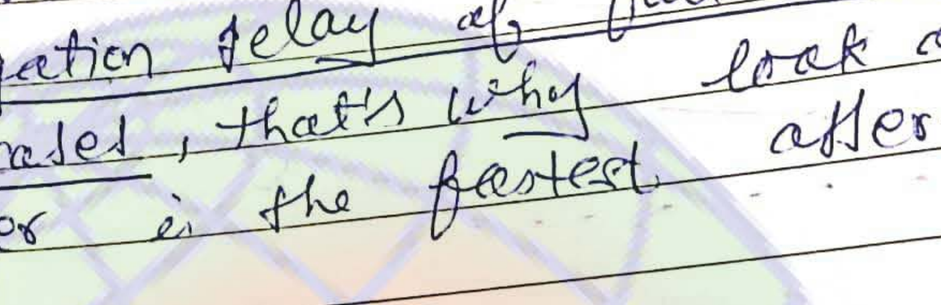
* Where n is number of bit's.
* No. of OR gates used in look aheas carry network is n

(iii) look aheas carry adder is generate carry to be used in higher bit's directly. So Propogation delay of full adder is not Cascaded, that's why adder is the fastest look aheas carry adder.

Full Substracter:-

$D = A - B - C$

(i)



(ii) Full Substracter:-

truth-table:

A	B	C	D	Cy
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

"Substraction के case में पहले वाले bit के साथ Substraction Perform करते फिर last वाले के साथ अर्थात C के साथ वाले के साथ Substraction perform करते है।"

(iii) $D = \Sigma\{1, 2, 4, 7\}$

$= \pi\{0, 3, 5, 6\}$

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

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

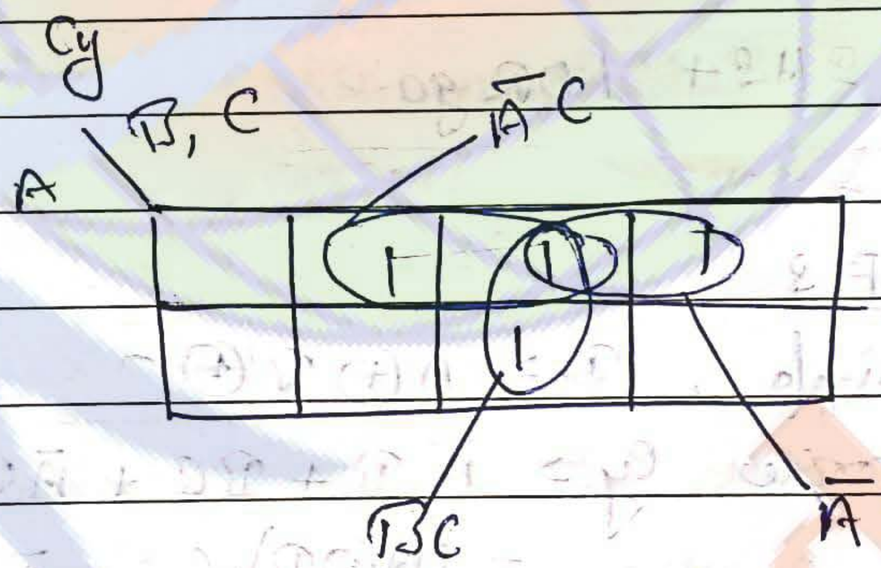
$Cy\,(Borrow) = \Sigma\{1, 2, 3, 7\}$

$= \pi\{0, 4, 5, 6\}$

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

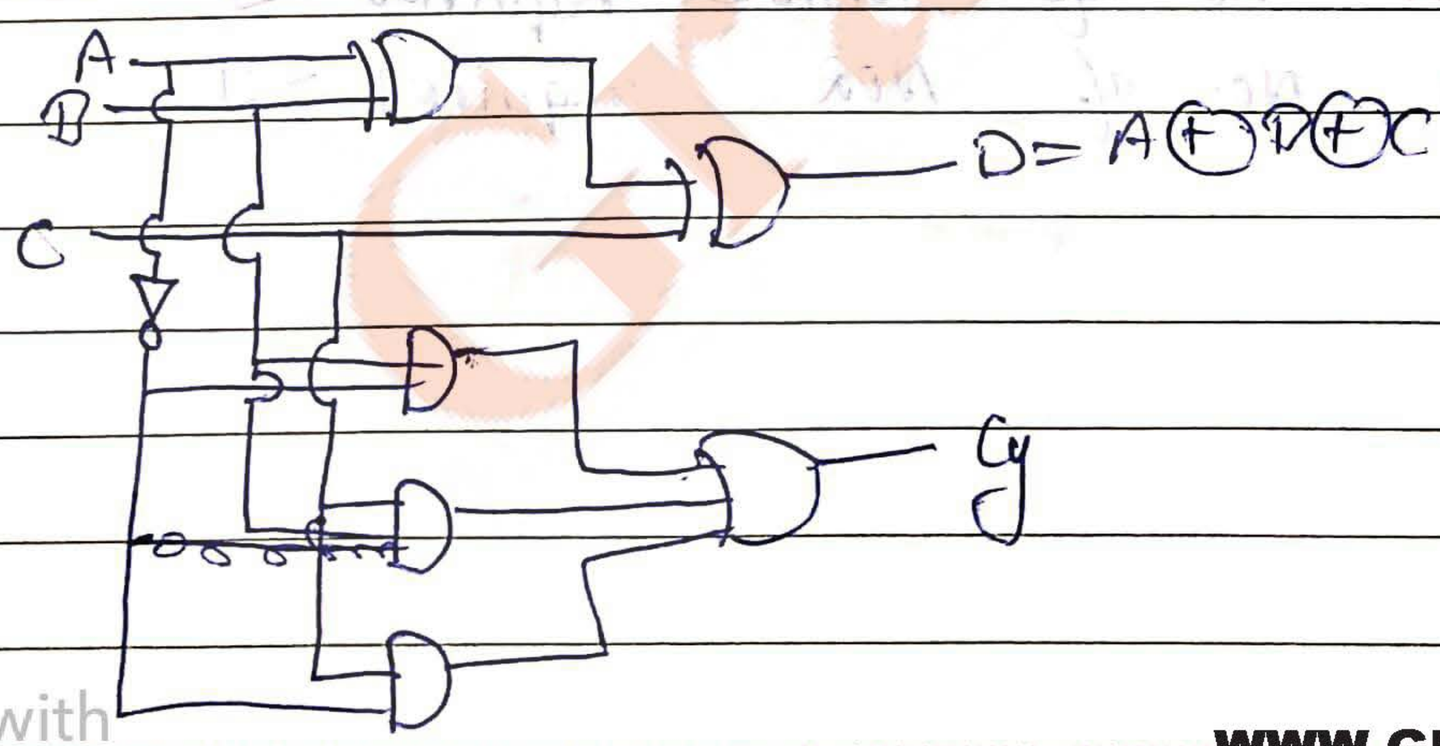
(iv) minimized Expression

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



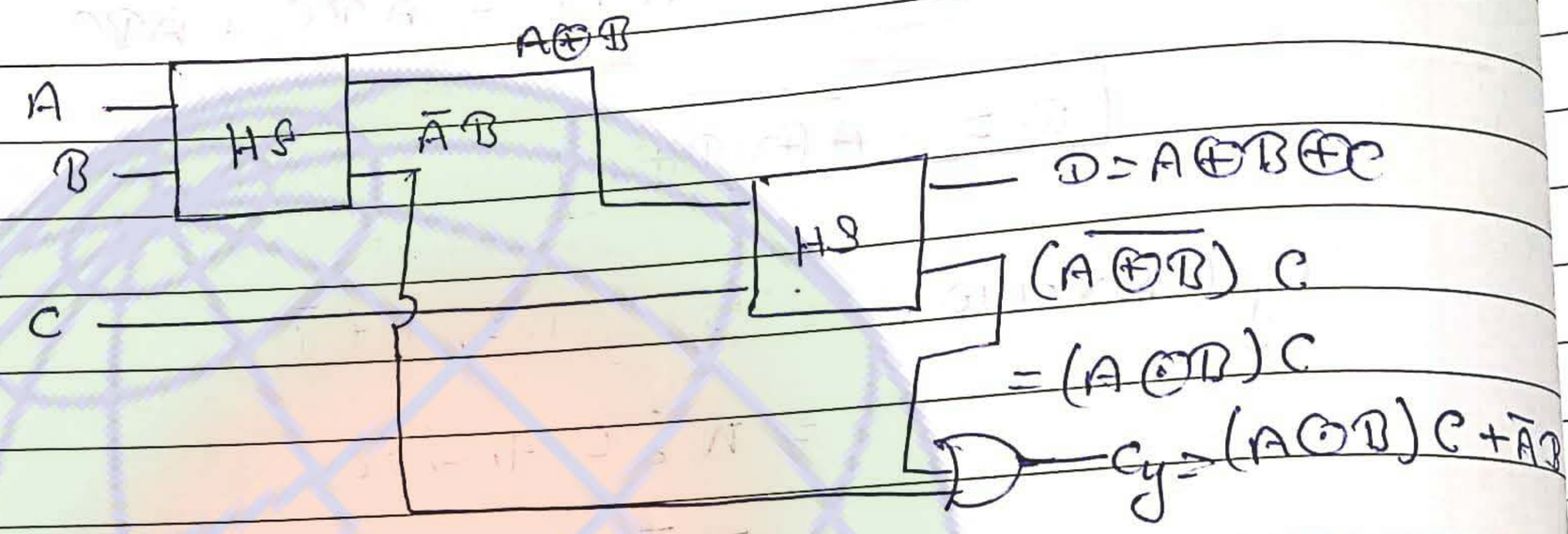
$$\boxed{Cy = \bar{A}B + BC + \bar{A}C}$$

(v) Circuit:



* Full Substractor Using half substractor

H.S $\begin{cases} D = A \oplus B \\ C_y = \bar{A} B \end{cases}$



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

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

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

1 FS = 2 HS + 1 OR gate

Objective of FS

(1) Exp for Diff, $D = A \oplus B \oplus C$

(2) Exp for borrow, $C_y = \bar{A}B + BC + \bar{A}C$

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

(3) No. of HS and OR gate = 2 HS + 1 OR gate.

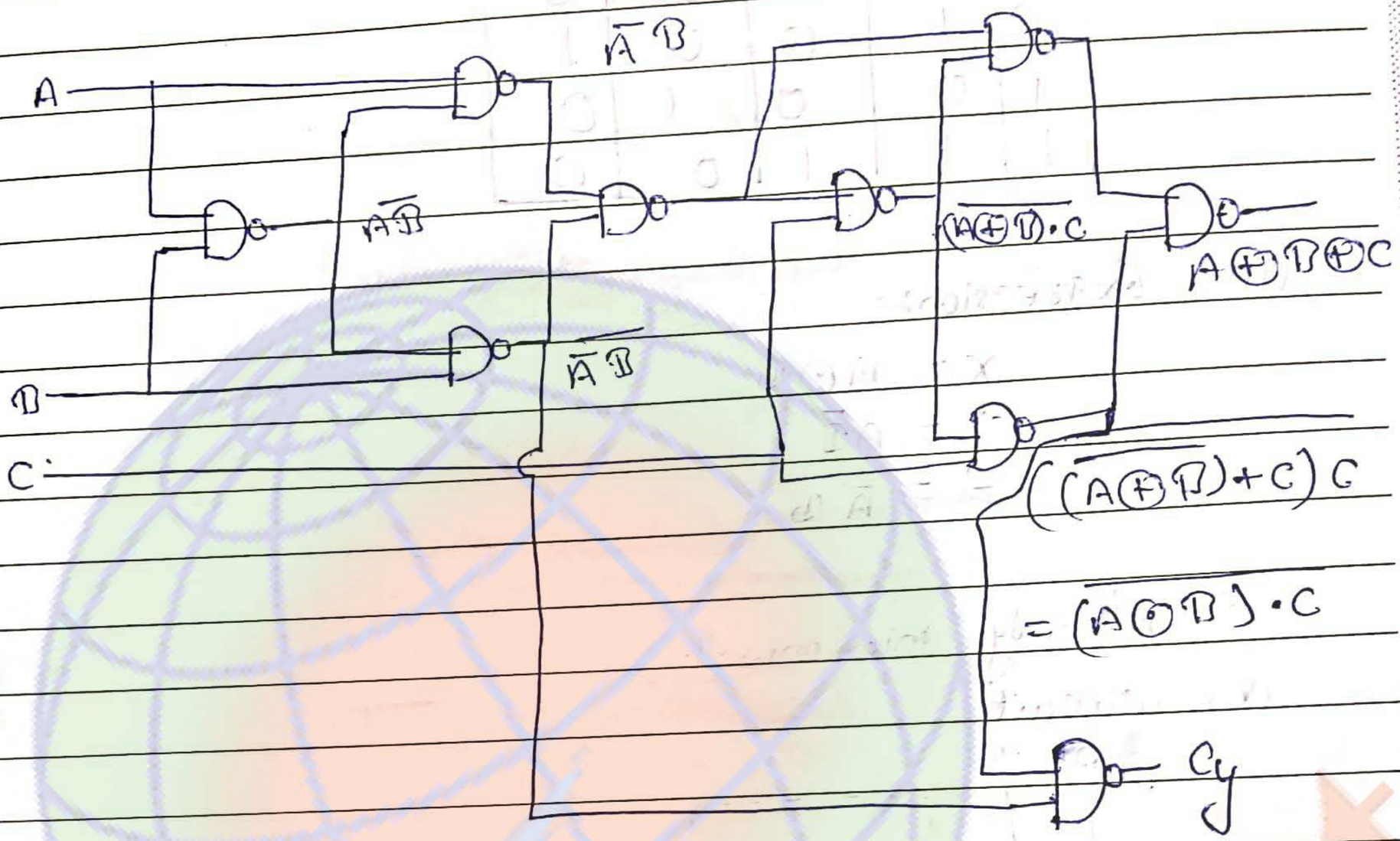
(4) min no. of NAND required = 9

(5) min No. of NOR required = 9

(6)

(7)

Full subtractor using NAND gate -



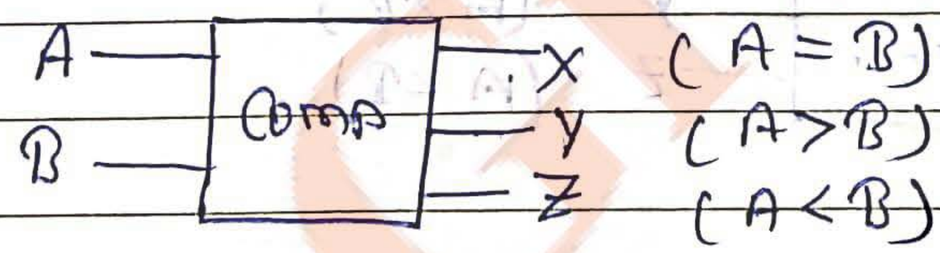
Full Sub-stractor using NOR gate!

Replace NAND with NOR, it will work.

✱ Comparator :- (Comparison b/w objects)

⊢ There should be two input, so that Comparison will be is Possible

(1) Identify input's and Output's



(11) For One bit Comparator,

Truth table

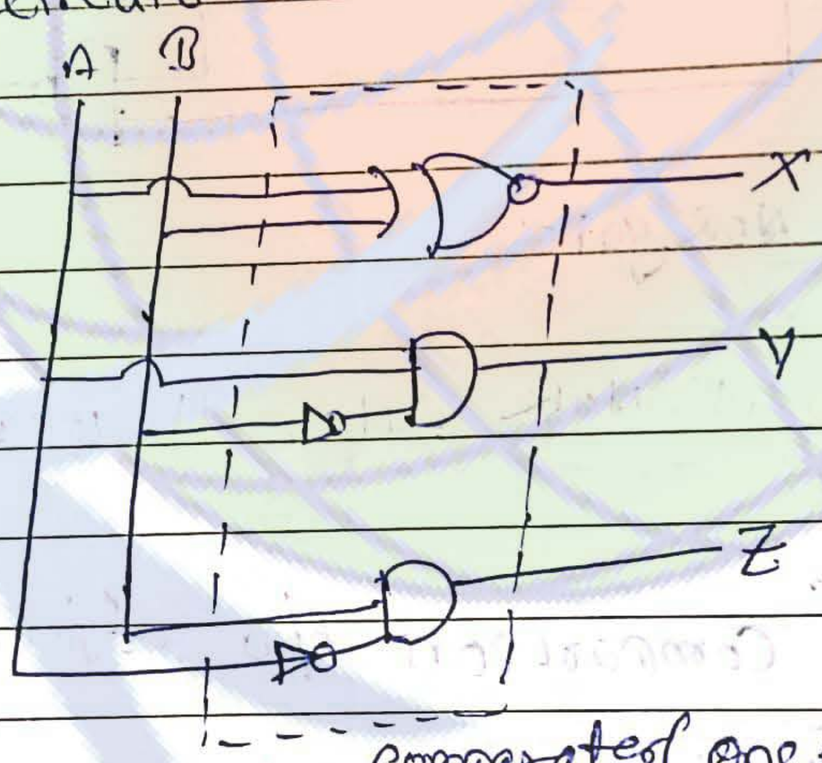
A	B	X	Y	Z
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

(iii) Expression:-

$$X = A \odot B$$
$$Y = A\bar{B}$$
$$Z = \bar{A}B$$

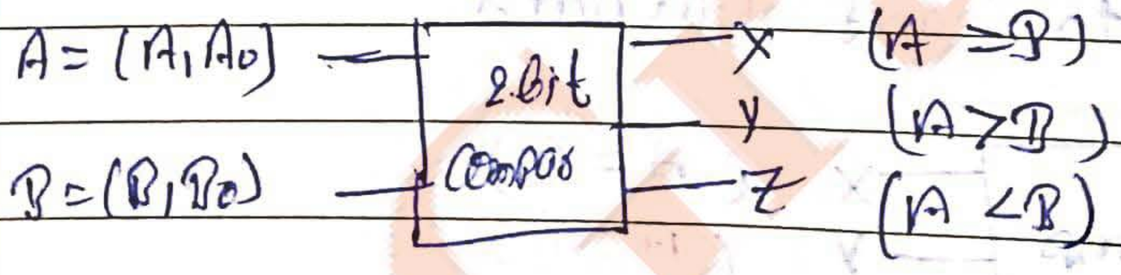
(iv) Already minimised.

(v) Circuit



comparator (one bit comparator)

* 2 bit Comparator:-



X = 1 when A = B

= 1 when $A_1 = B_1$ and $A_0 = B_0$

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

$y_B = 1$ when $A > B$

$y_B = 1$ $A_1 > B_1$ or $(A_1 = B_1)$ and $(A_0 > B_0)$

first highs will compare

20
10

21
25

higher then go next

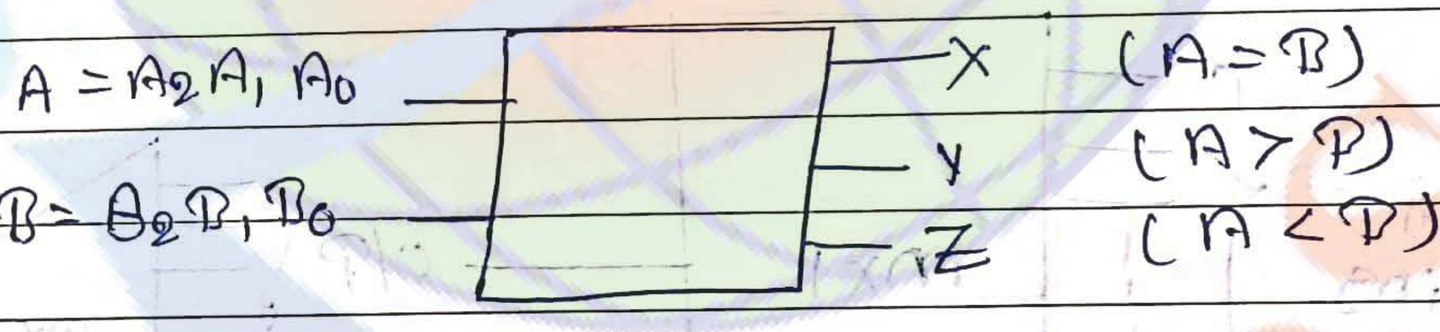
$$y = A_1 \bar{B}_1 + (A_1 \odot B_1) \cdot A_0 \bar{B}_0$$

$Z = 1$ when $A < B$

$Z = 1$ $\bar{A}_1 B_1 + (A_1 \odot B_1) \cdot \bar{A}_0 B_0$

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

* 3-bit Comparator:-



$X = 1$ when $A = B$

$x = 1$ when

$$X = (A_2 \odot B_2) \cdot (A_1 \odot B_1) \cdot (A_0 \odot B_0)$$

$$Y = A_2 \bar{B}_2 + (A_2 \odot B_2)(A_1 \bar{B}_1) + (A_2 \odot B_2)(A_1 \odot B_1) \cdot A_0 \bar{B}_0$$

$$Z = \bar{A}_2 B_2 + (A_2 \odot B_2)(\bar{A}_1 B_1) + (A_2 \odot B_2)(A_1 \odot B_1) \cdot \bar{A}_0 B_0$$

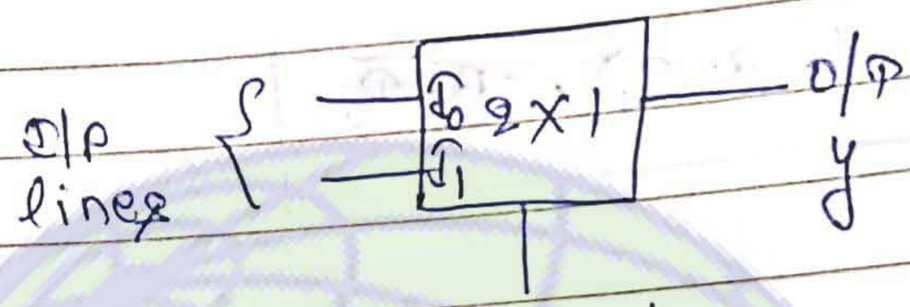
4.3 Non-Arithmetic circuits:-

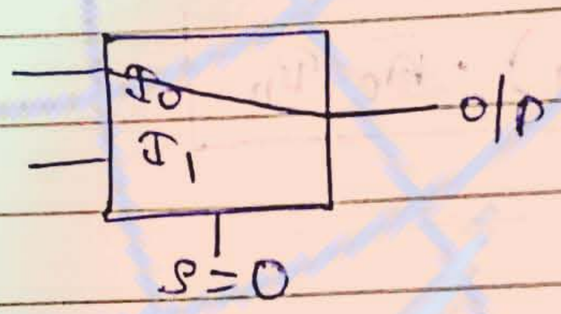
☆ Multiplexer:- (is a Universal logic circuit)

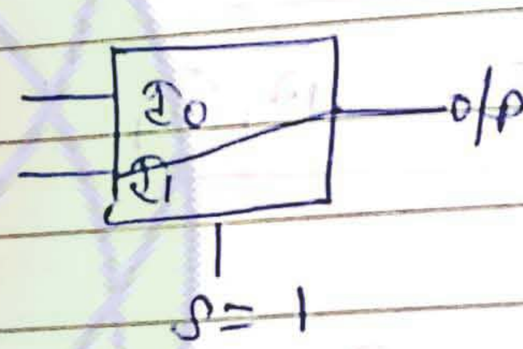
(i) MUX is an many to one circuit.

(ii) It has multiple input and one output.

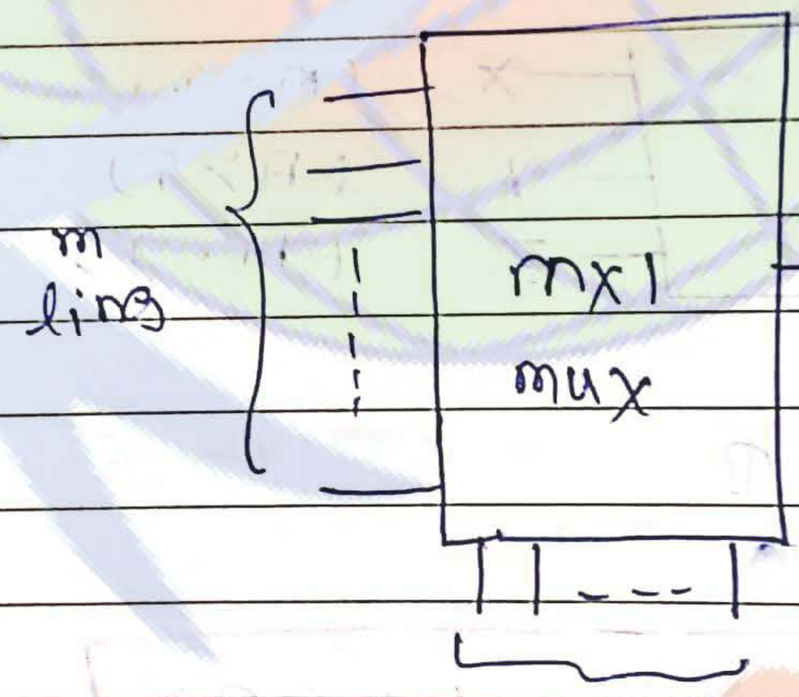
(iii) minimum size of MUX is 2×1 (input, output)

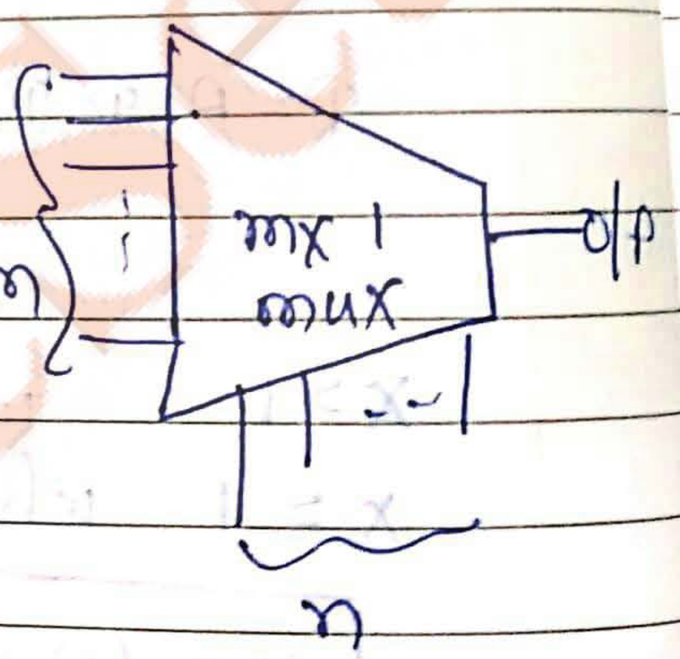






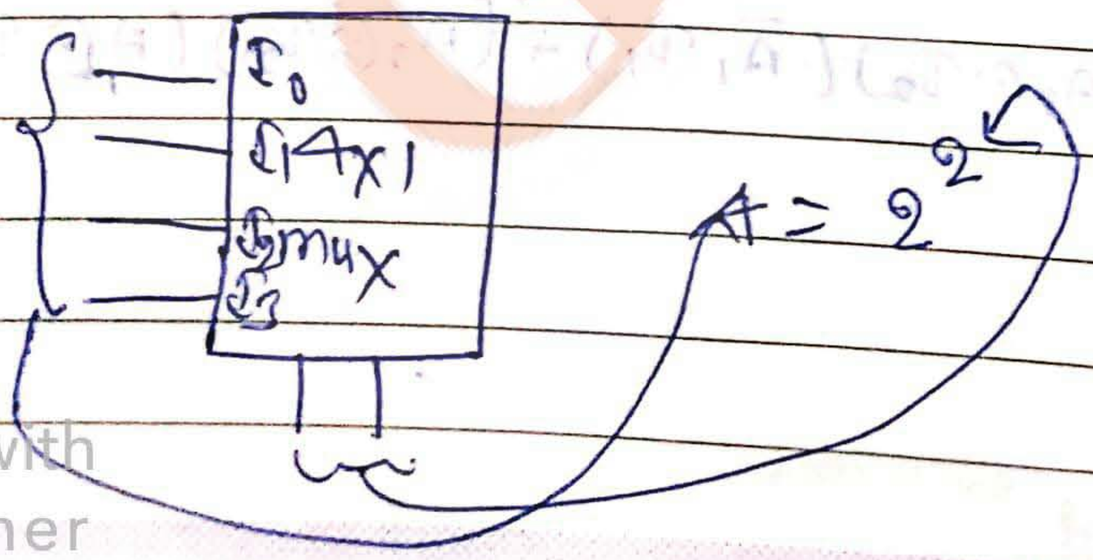
(iv)





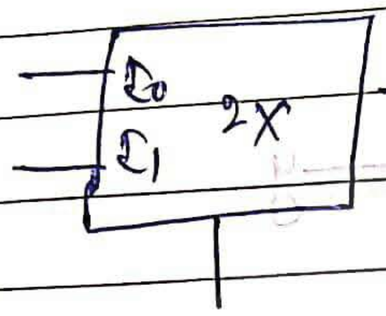
$$m = 2^n$$

eg.



(v) 2×1 mux

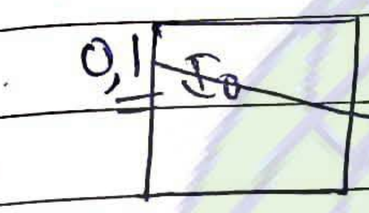
truth table

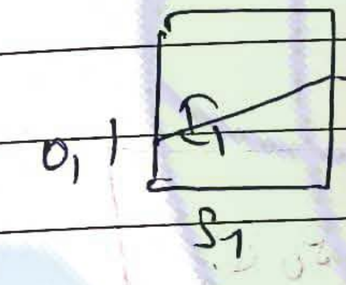


truth table

S	y
0	I_0
1	I_1

Actual truth table, with only 1 & 0.





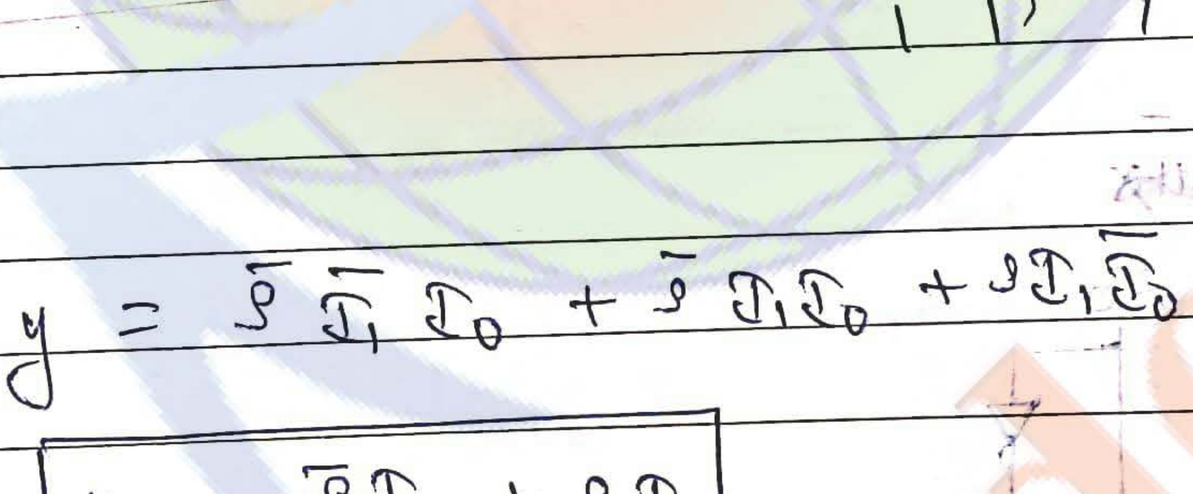
S	I_1	I_0	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(vi) Expression

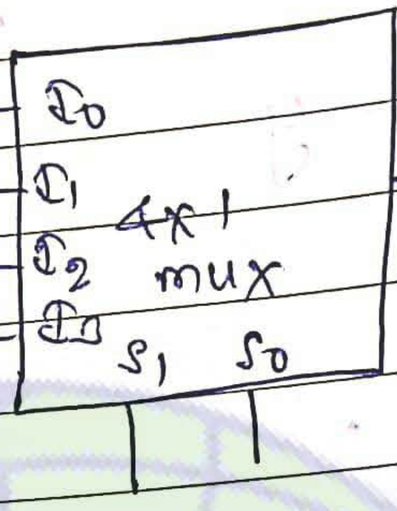
$$y = \bar{S}\,\bar{I_1}\,I_0 + \bar{S}\,I_1 I_0 + S I_1 \bar{I_0} + S I_1 I_0$$

$$\boxed{y = \bar{S} I_0 + S I_1}$$

(vii) circuit



* 4x1 mux



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

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

* Circuit :-

* 8x1 mux

$$y = \bar{S}_2\bar{S}_1\bar{S}_0 I_0 + \bar{S}_2\bar{S}_1 S_0 I_1 + \bar{S}_2 S_1 \bar{S}_0 I_2 + \bar{S}_2 S_1 S_0 I_3$$
$$+ S_2\bar{S}_1\bar{S}_0 I_4 + S_2\bar{S}_1 S_0 I_5 + S_2 S_1 \bar{S}_0 I_6 + S_2 S_1 S_0 I_7$$

Note:

(i) multiplexer ckt involves AND-OR logic

(ii) Any logic ckt can be derived using multiplexer, Therefore mux is called Universal logic circuit

* Model of Question

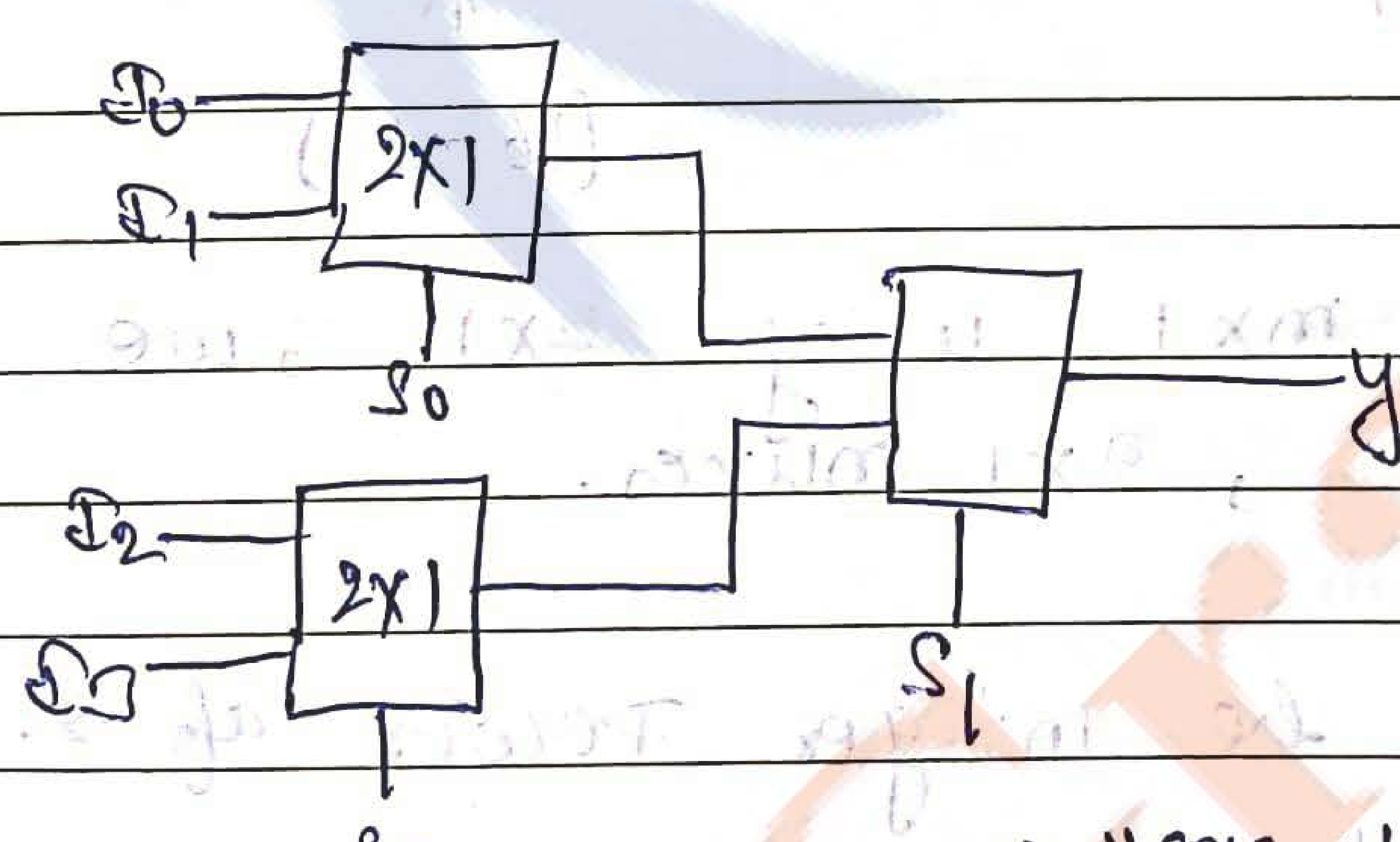
(1) Derivation of higher order mux using lower order mux.

(a) 4X1 using 2X1

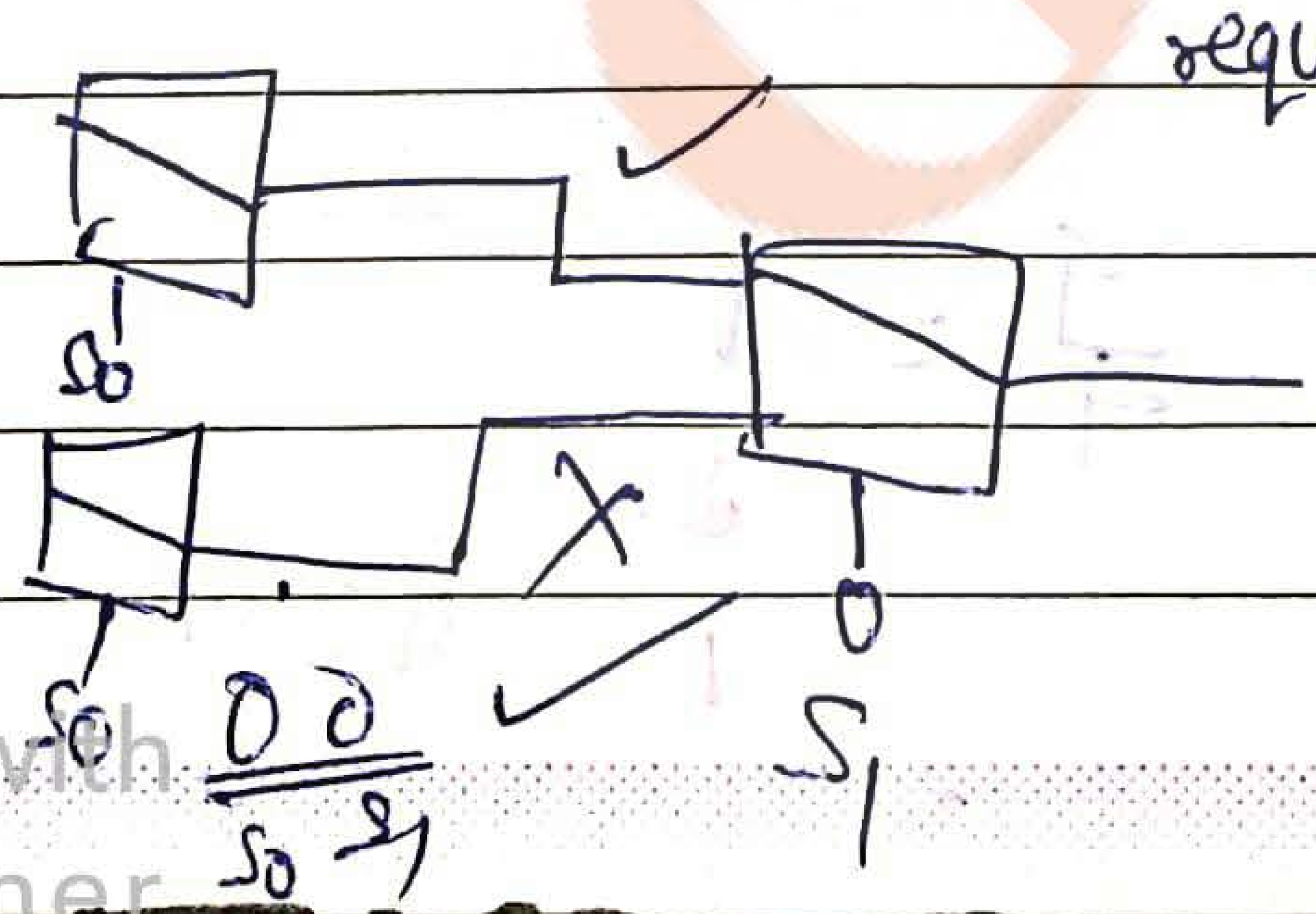
segregation of the Problem ⟺ Best way of solving any Problem

⇓

Problem को छोटे-छोटे parts में बाँटना ||



→ Hence, three 2x1 mux are required

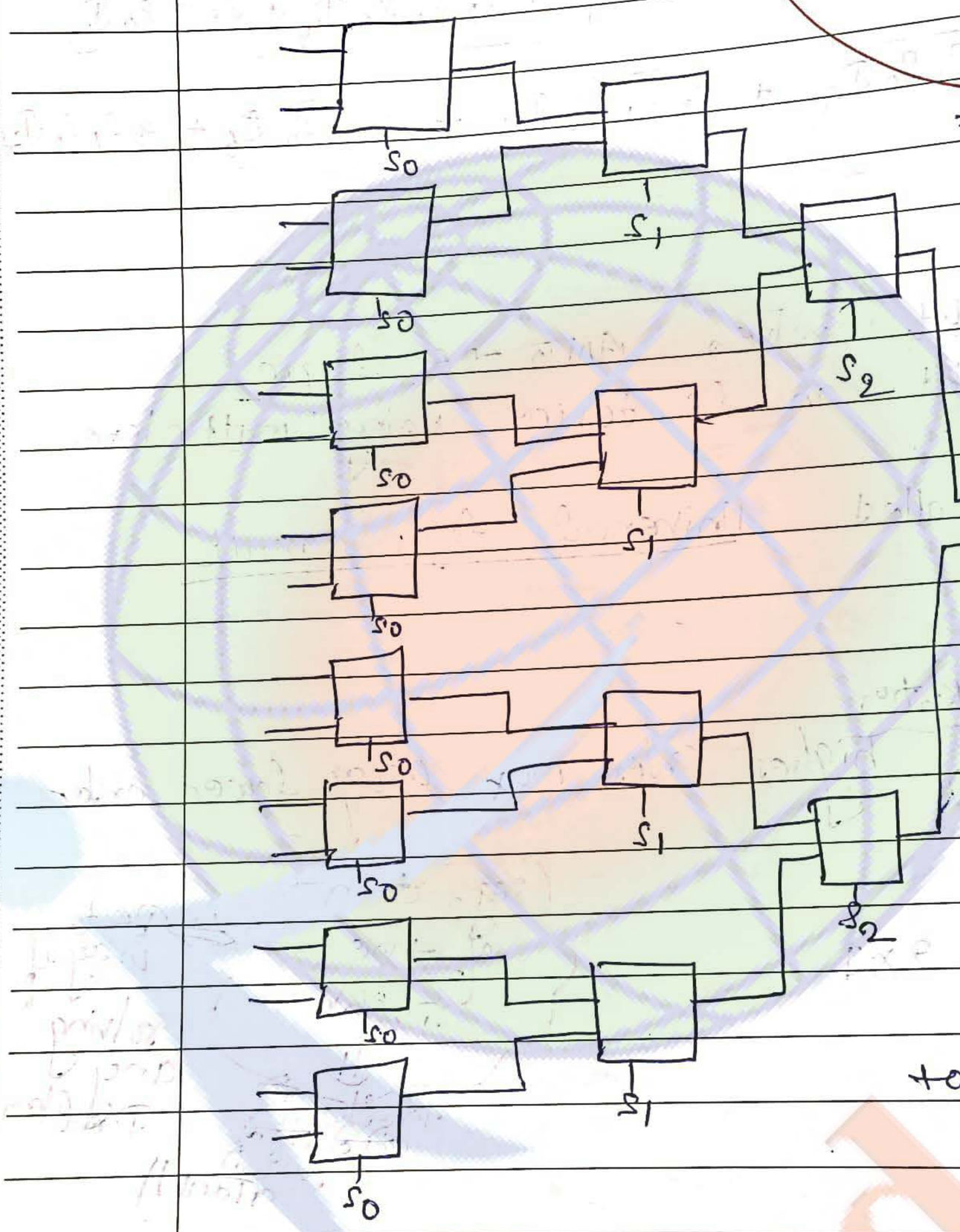


$$\frac{4}{2} = \frac{2}{2} = 1$$

2+1 = 3

पहला वाला छोड़ कर सभी का ऊपर वाला लेना
last का (+1) वाला भी लेना है।

* 16x1 using 2x1



we need four select lines.

$16 = 2^{x}$ → select line.

$\Rightarrow \frac{16}{2} + \frac{8}{2} + \frac{4}{2} + \frac{2}{2} + 1$

$= 8 + 4 + 2 + 1$

$= 15$ ← short trick Method

total 15 mux required.

(16 − 1)

* To construct mx1 using 2x1, we require (m−1), 2x1 muxes.

But,

m should be integer power of 2.

* 16x1 using 4x1 :

$\frac{16}{4} = 4$, $\frac{4}{4} = 1$

4 + 1 = 5 muxes

Page No. 569
Date

Eg

* 256x1 using using 16x1

larger number ← 256/16 (remainder) + 16/16 → One (1) आने पे एक चाहिये।।

16 + 1 = 17 chip's ई requires.

Q> How many chip's are required to construct 16x1 from 8x1

A

$\frac{16^2}{8} + \frac{2}{8}$ (रुकना वहा है जहा पे 1 होगा। रुकेगा नीचे)

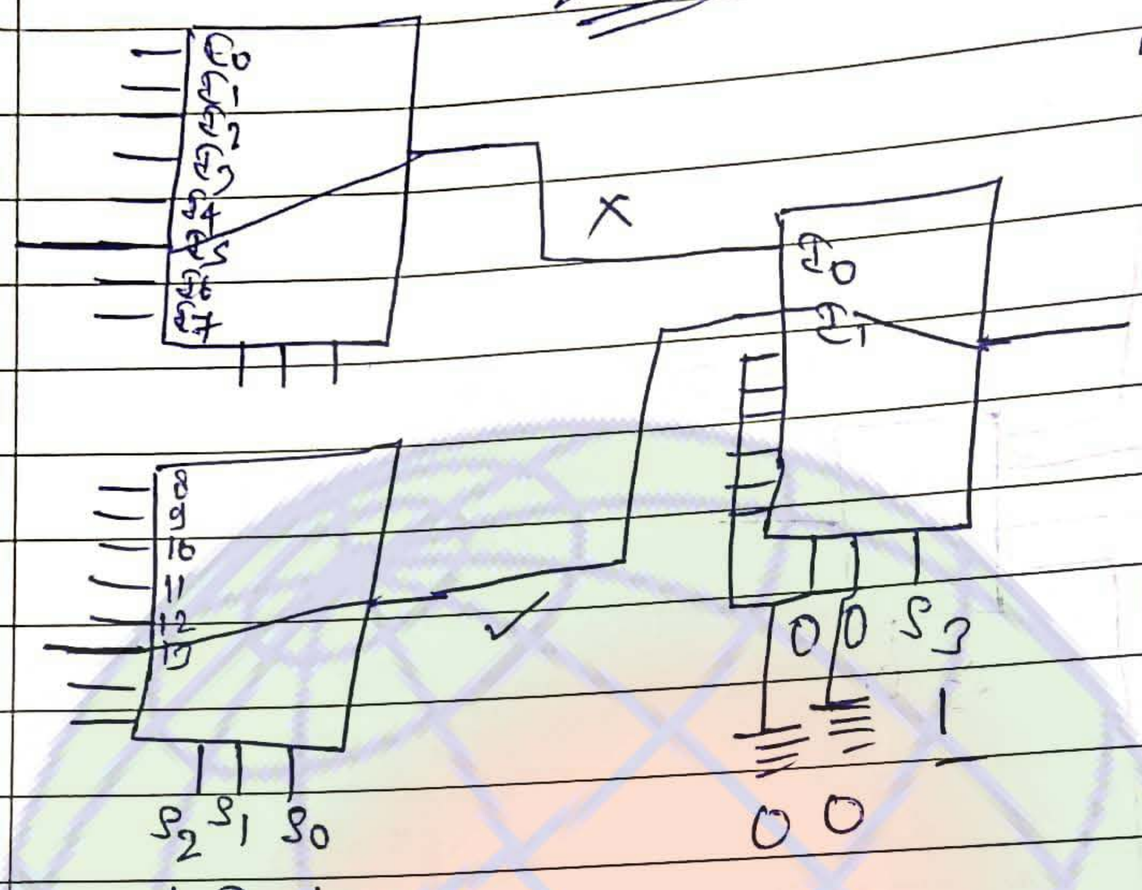
$2 + \frac{1}{4} = 2.5$

2 full chips — $\frac{1}{4}$ we required only $\frac{1}{4}$ of the third chips.

$16 = 2^4$ → four select lines,

2 chips

1/4



0101 → 5

1101 → 13

⇒ So total 3 chips are requ

Q> How many chips are requires 128x1 using 8x1

A $\frac{128}{8} + \frac{16}{8} + \frac{2}{8}$

$16 + 2 + \frac{1}{4}$

18 + 1 = 19 chip's //

model = 02

☆ Mux as a universal circuit:-

(a) Using 2x1 mux

(1) NOT :-

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

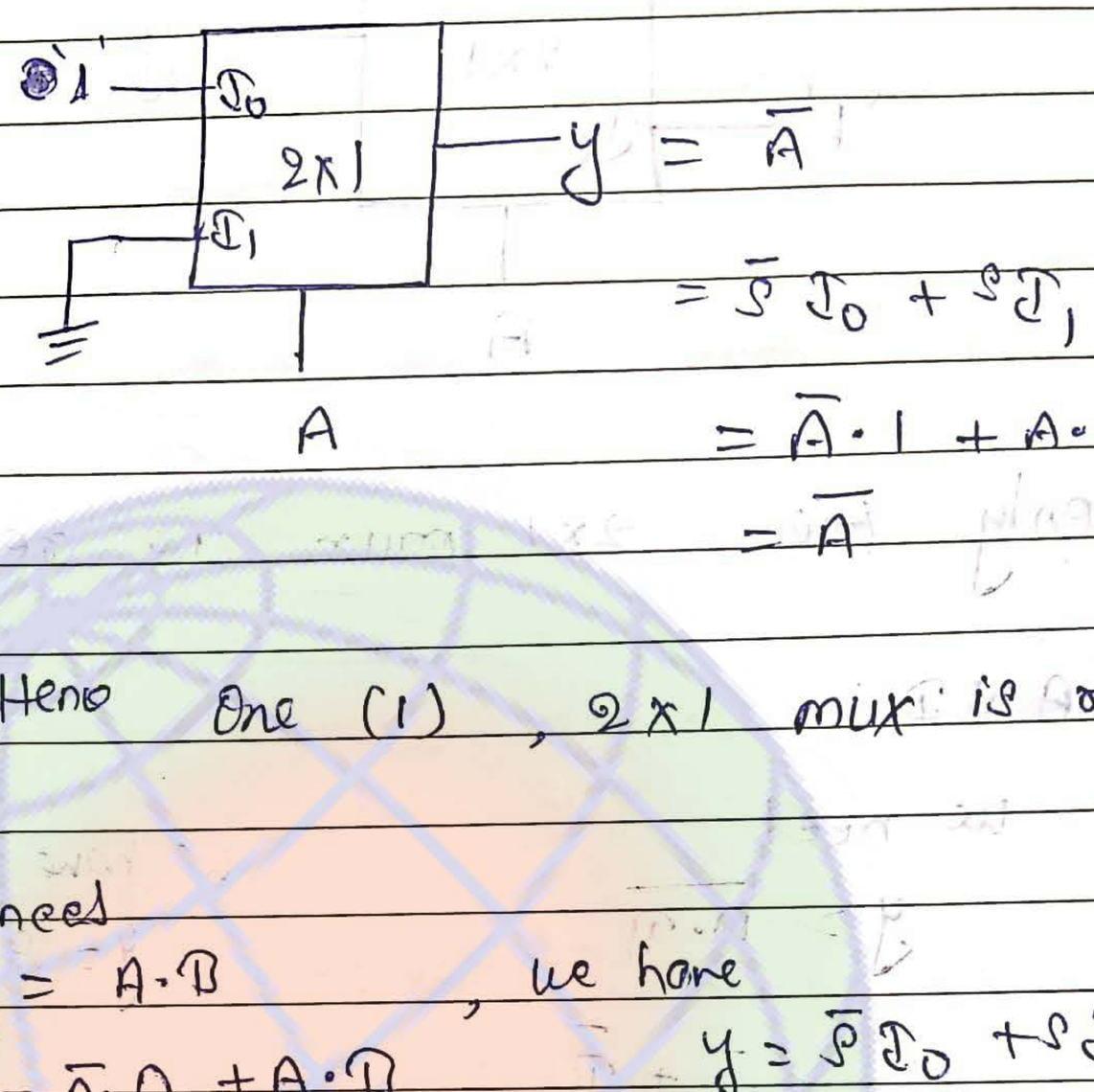
we need

$y = \bar{A}$

$y = \bar{A} \cdot 1 + A \cdot 0$

Now, compare,

$S = A$, $I_0 = 1$, $I_1 = 0$



$= \bar{S} I_0 + S I_1$

$= \bar{A} \cdot 1 + A \cdot 0$

$= \bar{A}$

Hence One (1), 2x1 mux is required.

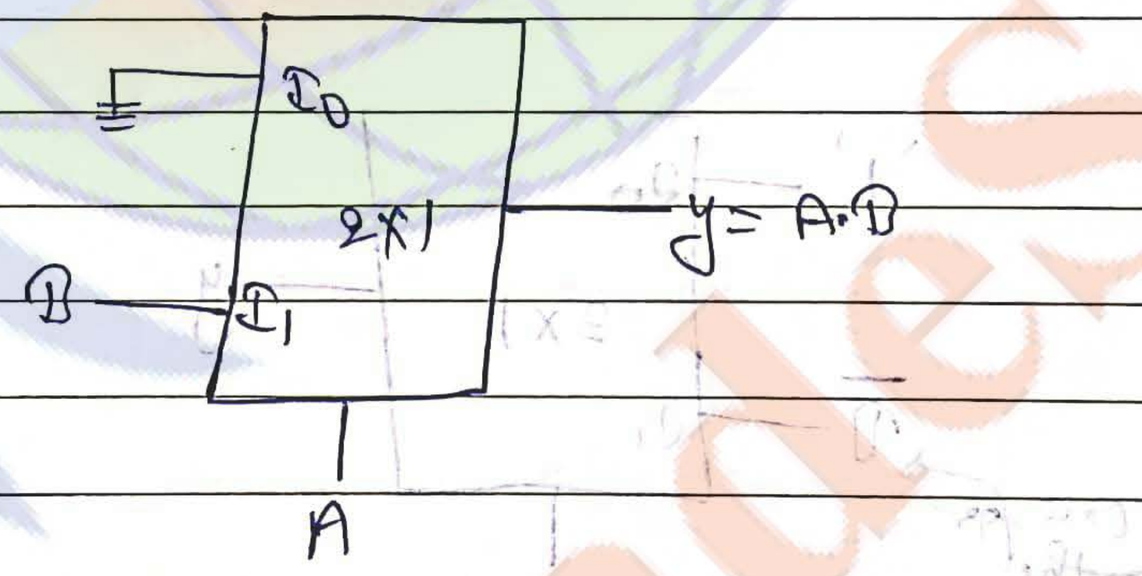
(b) AND:-

we need $y = A \cdot B$, we have $y = \bar{S} I_0 + S I_1$

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

So

$S = A$, $I_0 = 0$, $I_1 = B$



Hence One 2x1 mux is required.

(c) OR: we need $y = A + B$, we have $y = \bar{S} I_0 + S I_1$, ✓

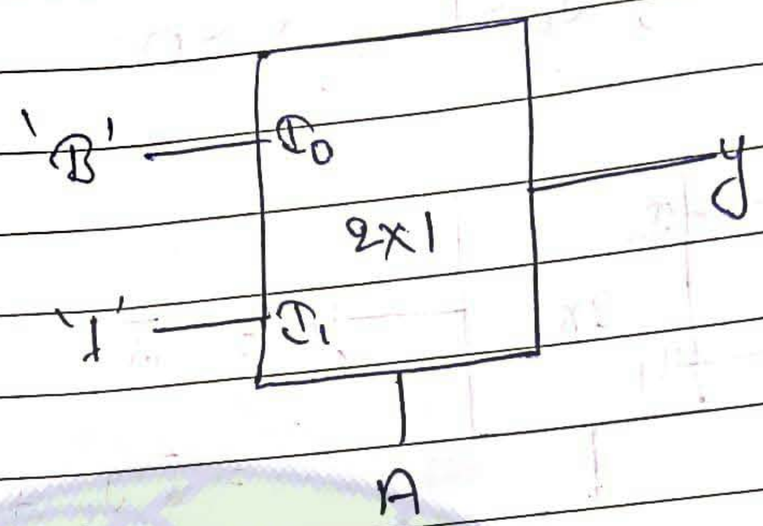
$= \cancel{A + \bar{A} B}$

$= \bar{A} B + A$

$= \bar{A} B + A \cdot 1$ ✓

Compare,

$S = A$, $I_0 = B$, $I_1 = 1$



Only one 2x1 mux is required.

(d) NAND :-

we need

$$y = \overline{A \cdot B}$$

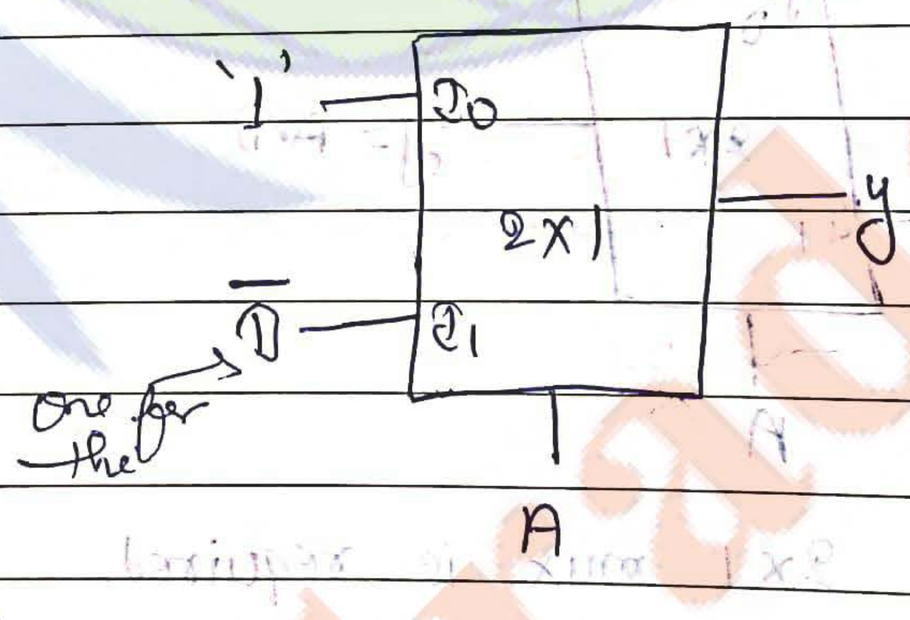
$$= \bar{A} + \bar{B}$$

$$= \bar{A} \cdot 1 + A\bar{B}$$

we have

$$y = S I_0 + S I_1$$

$$S = A, \; I_0 = 1, \; I_1 = \bar{B}$$



Hence two 2x1 mux are required.

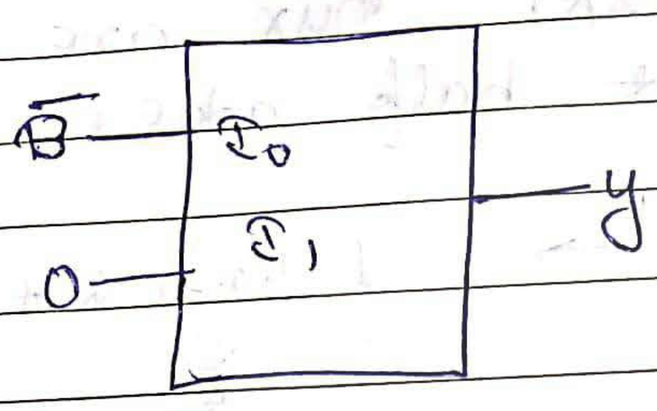
(c) NOR :-

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

$$y = \overline{A + B} = \bar{A} \cdot \bar{B}$$

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

$$S = A, \; I_0 = \bar{B}, \; I_1 = 0$$



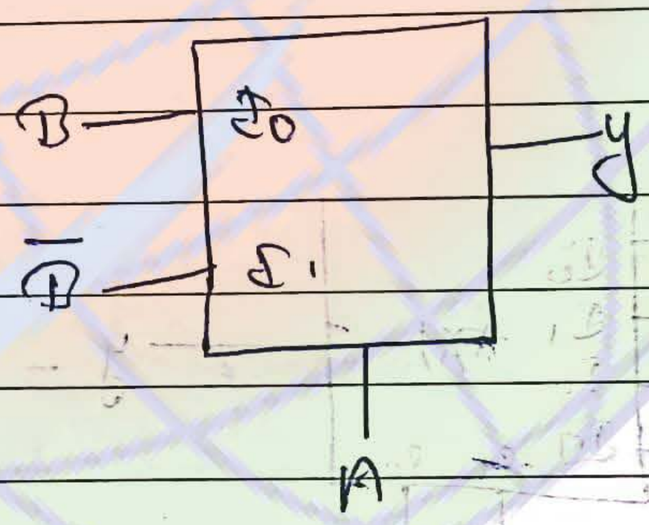
Hence, two 2×1 mux are required.

(5) Ex-OR

$$y = \bar{S} I_0$$

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

$$S = A, \; I_0 = B, \; I_{0_1} = \bar{B}_0$$



two 2×1 mux are requel.

(6) Ex-NOR

$$y = \bar{A}\bar{B} + AB$$

$$S = A, \; I_0 = \bar{B}, \; I_1 = B$$

two 2×1 mux

Q) How many 2×1 mux are required to construct half adder:

Sol^n H.A $\xrightarrow{2\times1}$ 1 Ex-OR + 1 AND
(2) (1) = three 2×1 mux

So, three 2×1 mux is required.

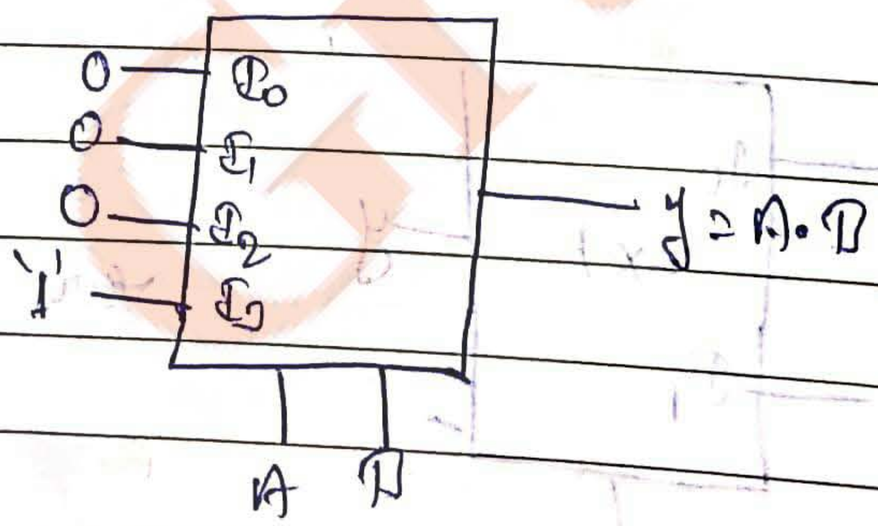
Note:
half subtractor = three 2×1 mux is requir

★ Using 4×1 mux :-

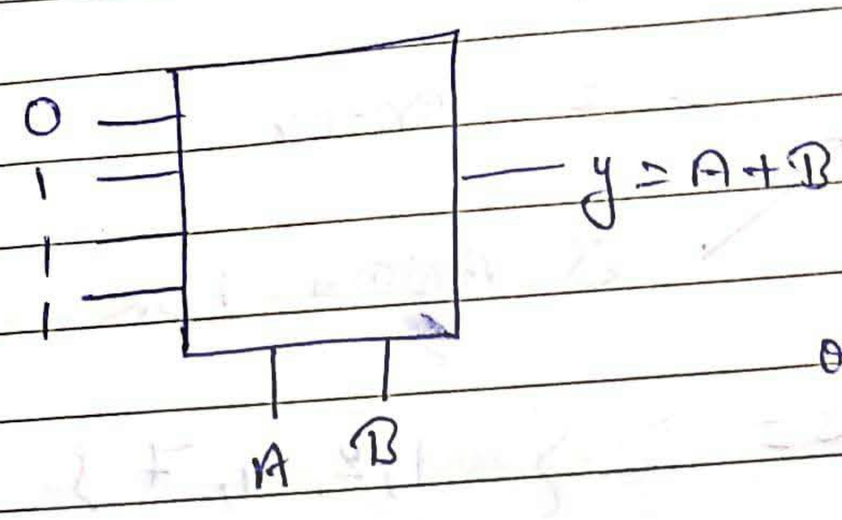
(1) NOT :

One 4×1 mux is required.

(2) AND :-

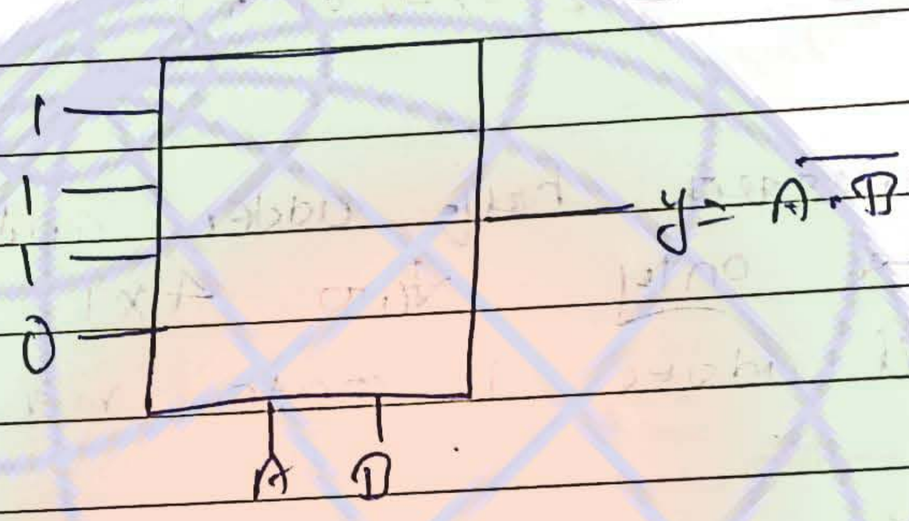


(III) OR

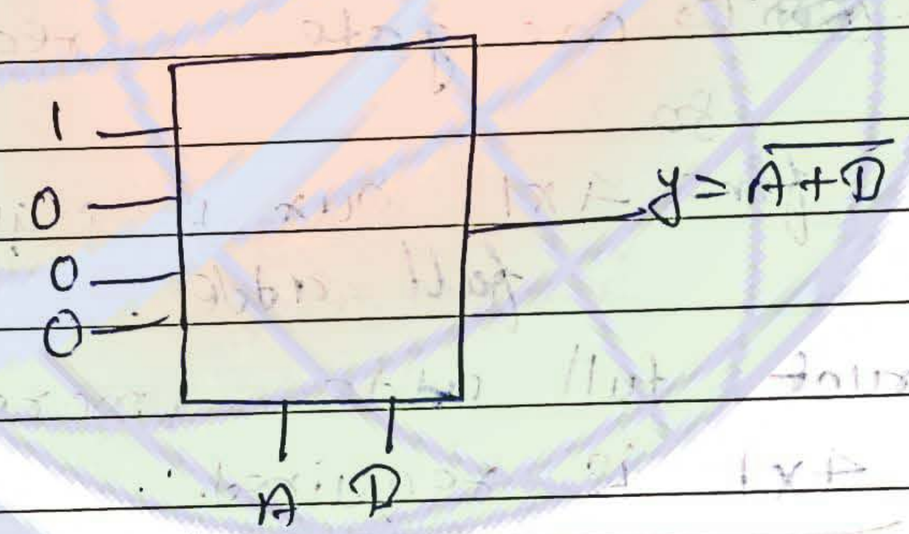


only one

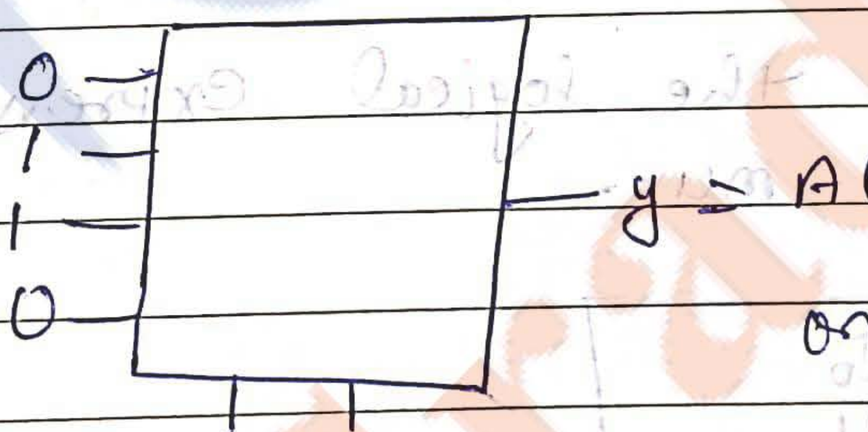
(IV) NAND



(V) NOR

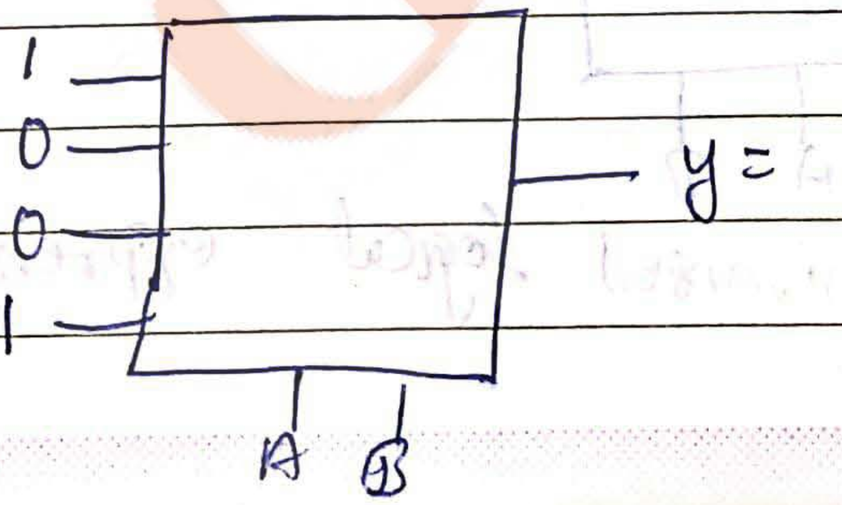


(VI) EX-OR



only one

(VII) EX-NOR



H.A > 2
H.S = 2

* Full adder using 4:1 :-

F.A < 2 ex-or
3 AND + 1 OR

$S = \Sigma\{1, 2, 4, 7\}$

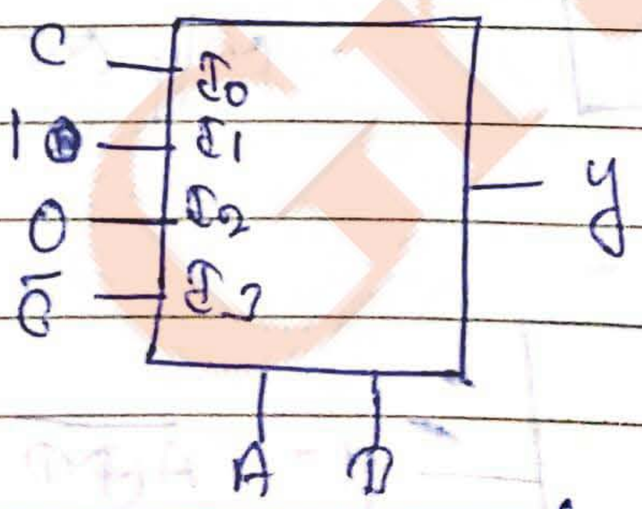
$C = \Sigma\{3, 5, 6, 7\}$

Note:
(i) To construct half adder and half Substracter only two 4x1 mux is requires
(ii) If full adder is made via half adder, then
four 4x1 mux and one 4x1 mux for ~~To~~ one gate i.e requires
so
five 4x1 mux is requires for full adder.

* To construct full adder, more than one chip of 4x1 is required.

Imp:
* model No. 3

Determine the logical expression at the output of mux.



what is minimised logical expression for y.

Date

$$y = \bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3$$

$$= \bar{A}\bar{B}C + \bar{A}B\cdot 1 + A\bar{B}\cdot 0 + AB\bar{C}$$

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

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

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

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

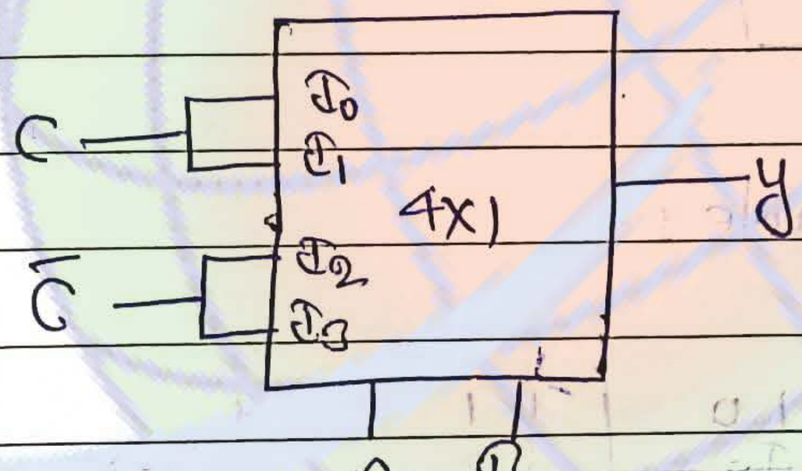
$$= \bar{A}C + B\bar{C}$$

Imp V.V.I

(using consus theorem)

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

*



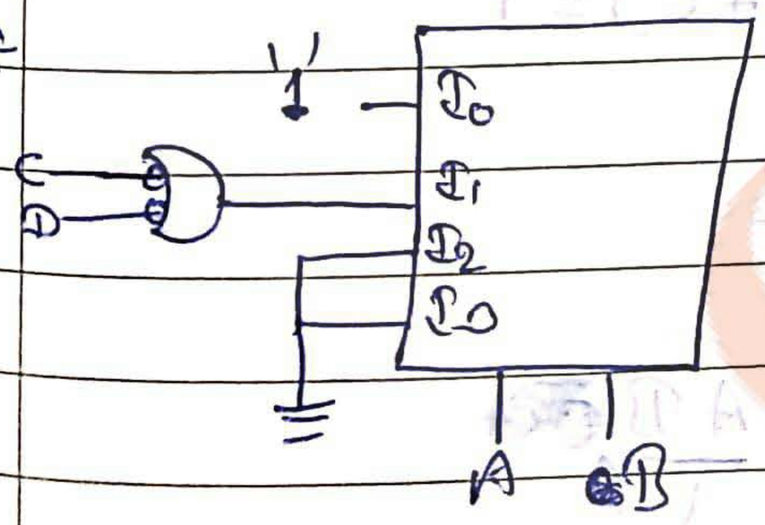
~~$y = S_1\bar{S_0}I_2 + S_1S_0I_3$~~

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

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

$$= A \oplus C$$

*



$$y = \bar{A}\bar{B}\cdot 1 + \bar{A}B\cdot C\bar{D} + 0$$

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

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

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

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

Page No.
Date

model 04: Implementation of given logical expression:

Implement function

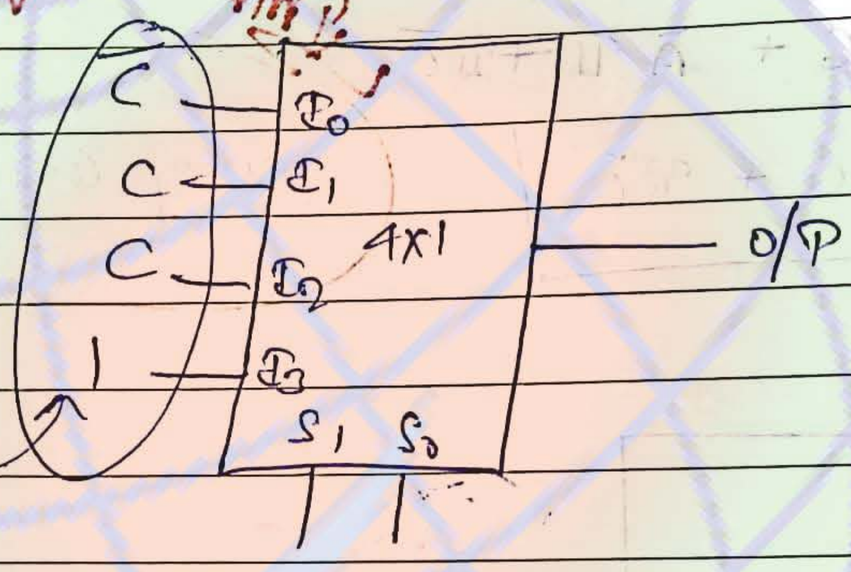
$f(A,B,C) = \sum m(0,3,5,6,7)$

using

4x1 mux

Assume one NOT gate is available.

Ans Let A, B are select lines



Implementation table :-

AB	00	01	10	11
	I_0	I_1	I_2	I_3
$\bar{C}$	0	(0)	2	4	(6)
	1	1	(3)	(5)	(7)
C				
	$\bar{C}$	C	C	1

$(\bar{C}+C) = 1$

Note:

0, 3, 5, 6, 7

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

$\bar{C}$ C C 1

Q:- Implement $f(A,B,C) = \sum m\{1,2,3,6,7\}$

Using
(i) A, B as select line
(ii) B, C as select line
(iii) B, C as select line

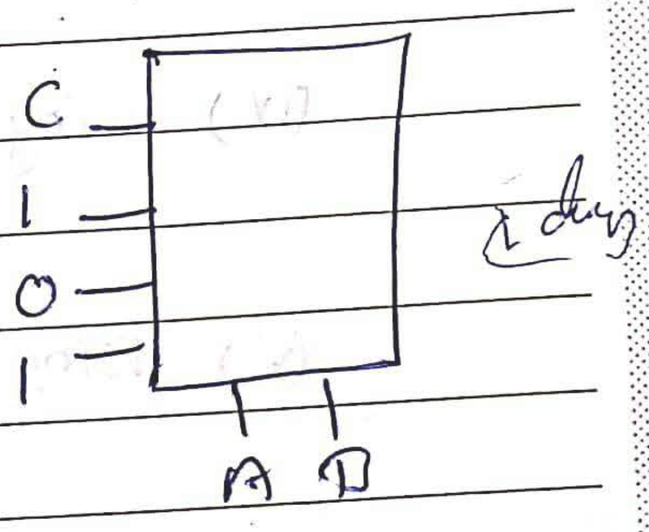
using 4x1

A. Implementation table:-

(a)

	AB	I_0 00	I_1 01	I_2 10	I_3 11
$\bar{C}$	0	0	(2)	4	(6)
C	1	(1)	(3)	5	(7)

C | 1 | 0 | 1

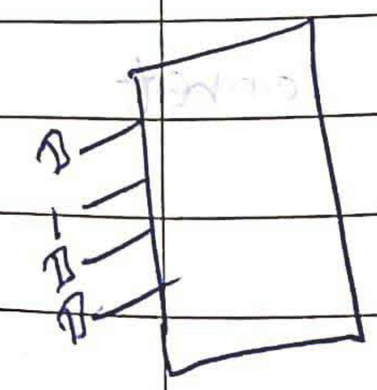


$f = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + AB$

(b) $f = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C} + AB\bar{C}$

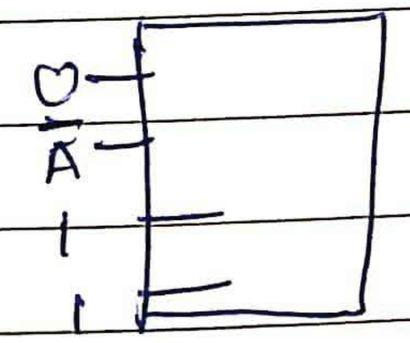
AC:

	I_0 00	I_1 01	I_2 10	I_3 11
$\bar{B}$	0 (000)	(1) (001)	4 (100)	5 (101) $\bar{B}$
B	(2) (010)	(3) (011)	(6) (110)	(7) (111) B



(c) $\bar{B}C$

	I_0 B=0	I_1 B=1	I_2 B	I_3 B
$\bar{A}$	0	(1)	(2)	3
A	4	5	(6)	(7)
	0	$\bar{A}$	1	1



Note.

(i) Using 2x1 mux — all 1 var / some of 2 variable

(ii) Using 4x1 mux — all 2 variable / some of 3 variable.

(iii) Using 4x1 mux + NOT — all 3 variable / some of 4 variable.

(iv) Using 8x1 mux — All 3 variable / some of 4 variable

(v) Using 8x1 mux + NOT — All 4 variable / some of 5 variable.

(vi) Since inside of a multiplexer, two level AND-OR exists.

Hence

NAND is most efficient to realise mux.

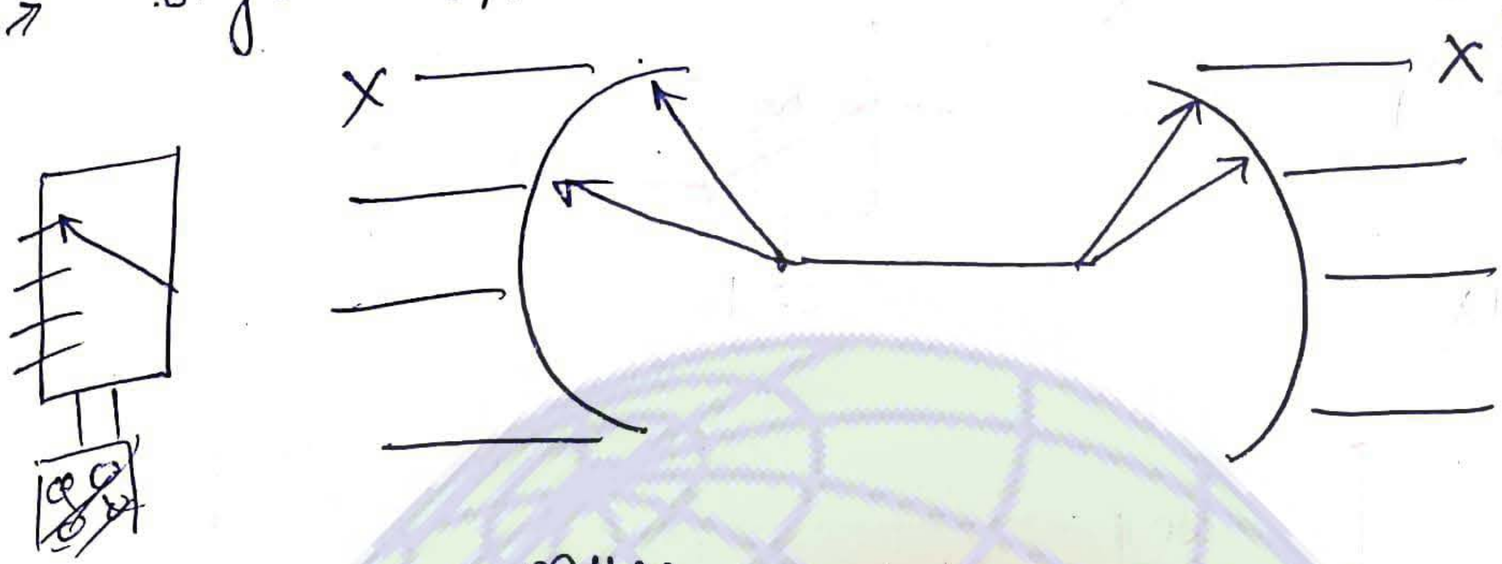
(vii) For mux, SOP expression only can be realised directly.

(viii) To realise any POS using mux, first convert POS into SOP and then realise.

Demultiplexer :

→ one to many circuit

→ single I/P and multiple O/P



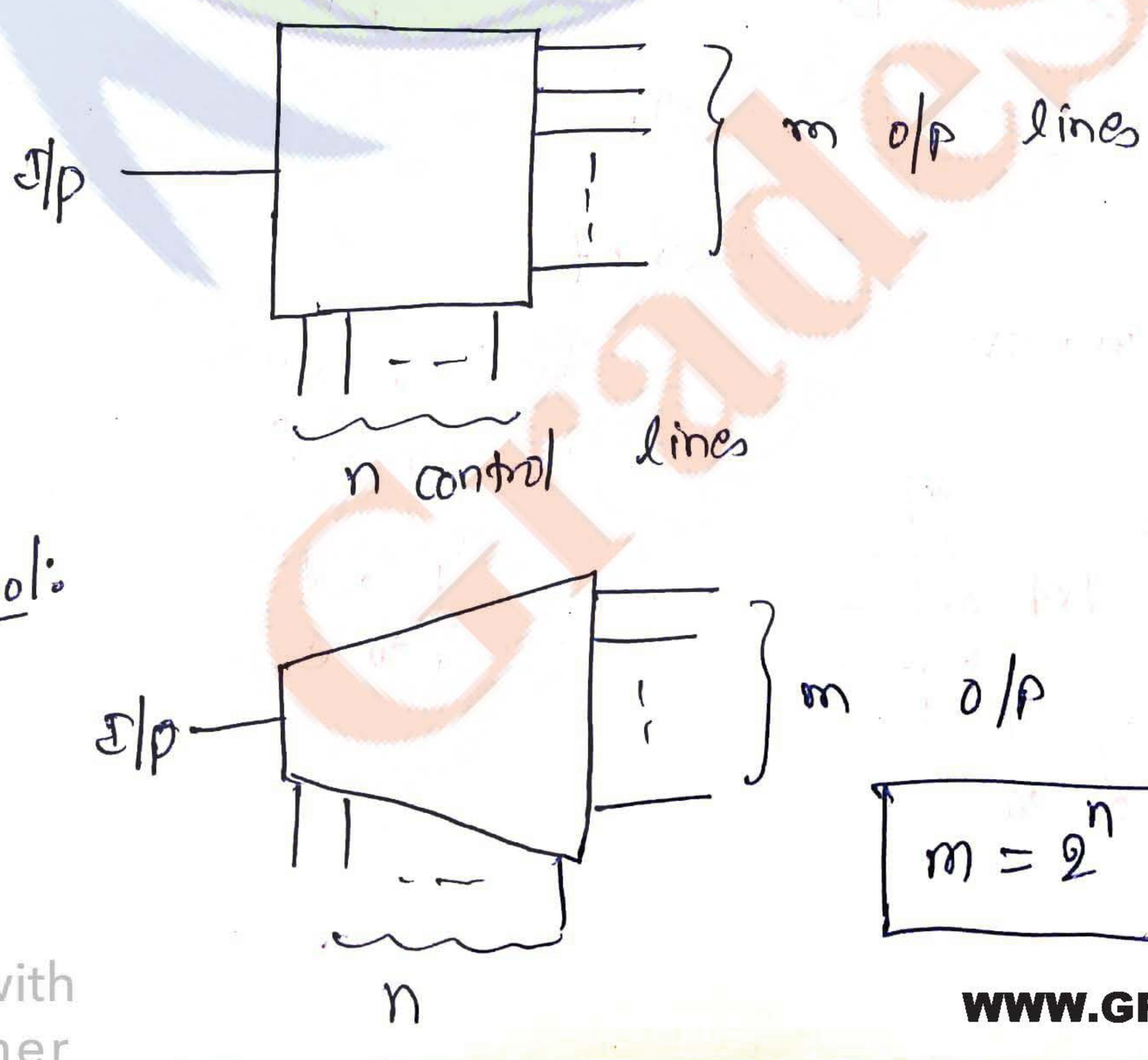
MUX
(many to one)

Demux
(one to many)

→ mux is a Universal circuit, but not demux

→ Demultiplexer is a combinational circuits, which has one input and many output depending on select or control line.
Input is transferred to one of the output line.

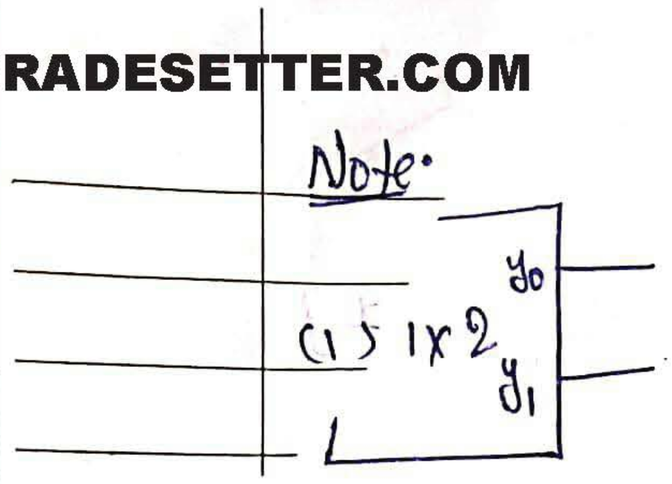
→



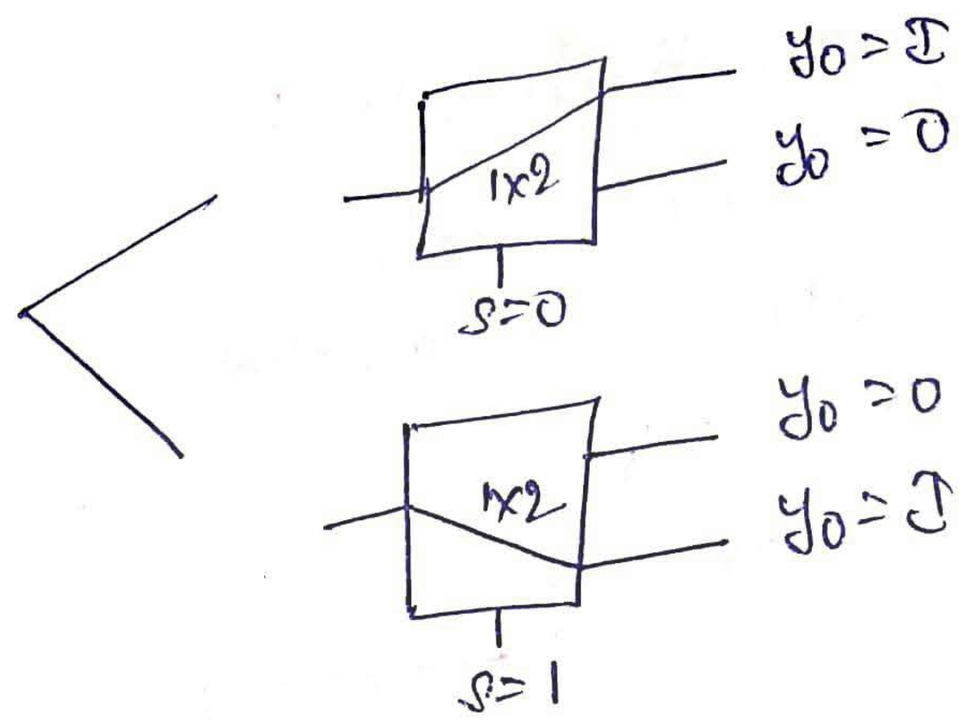
Symbol:

$m = 2^n$

Note:

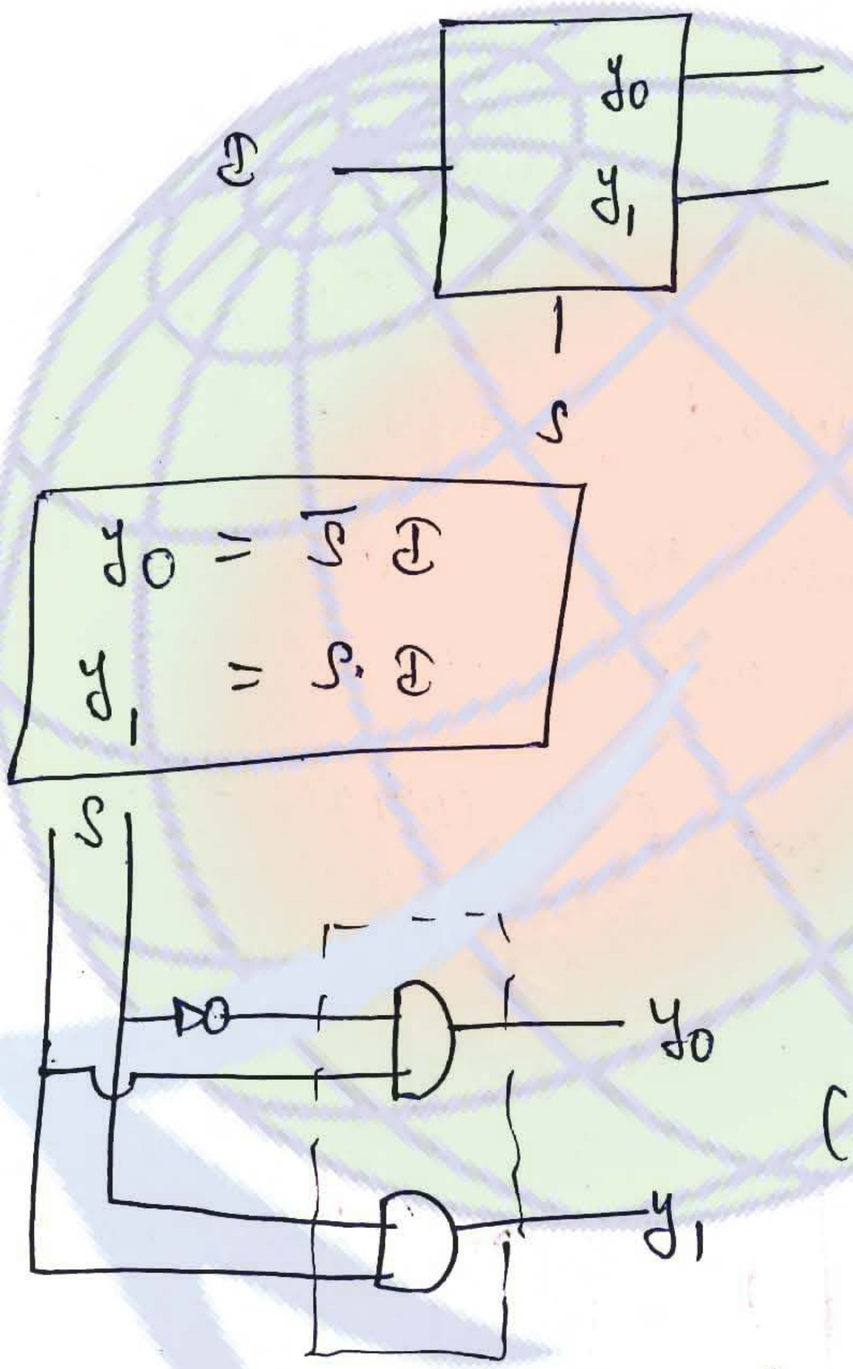


(minimum de two)



Circ

* 1x2 Demux



Demux circuit

S	y0	y0
0	I	0
1	0	I

expended truth table

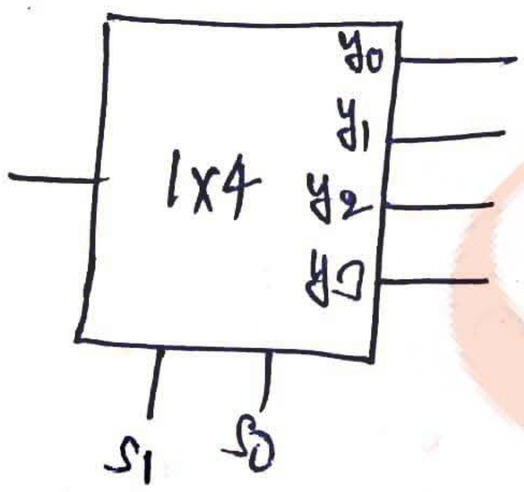
S	I	y0	y1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

*

(circuit)

*

* 1x4 Demux

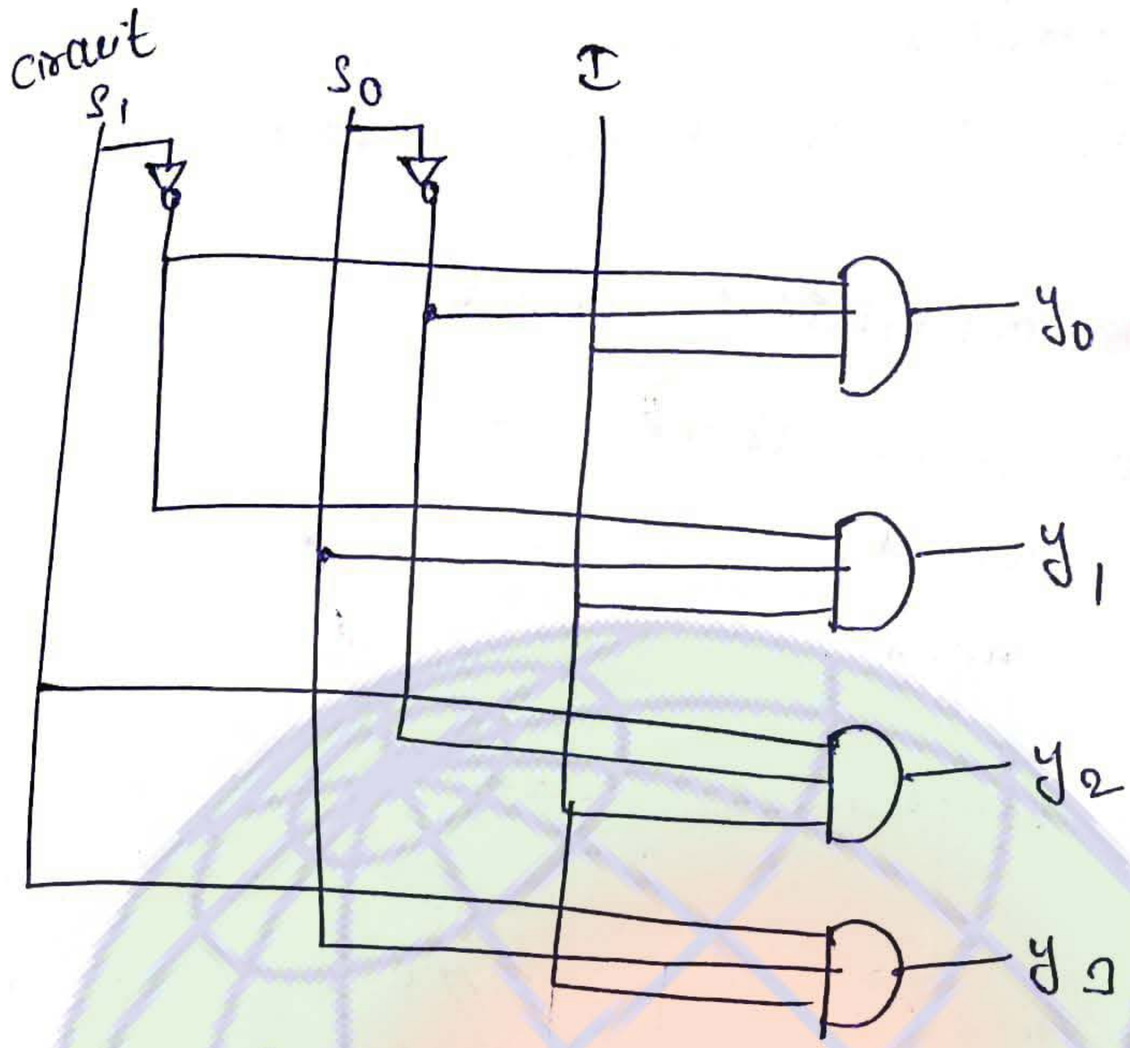


$y_0 = \bar{S}_1 \bar{S}_0 I$

$y_1 = \bar{S}_1 S_0 I$

$y_2 = S_1 \bar{S}_0 I$

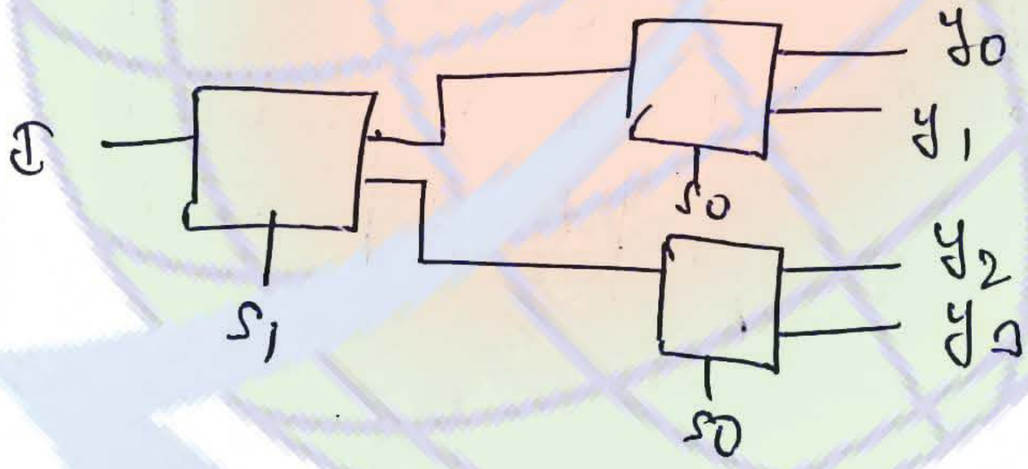
$y_3 = S_1 S_0 I$



* 1x4 using 1x2

$$\frac{4}{2} + \frac{2}{2} \Rightarrow 2+1 = 3$$

(from right most to left)



* 1x16 using 1x128

$$= \frac{128}{16} + \frac{8}{16}$$

$$= 8 + \frac{1}{2}$$

$$= 9 \text{ chips}$$

Decoder

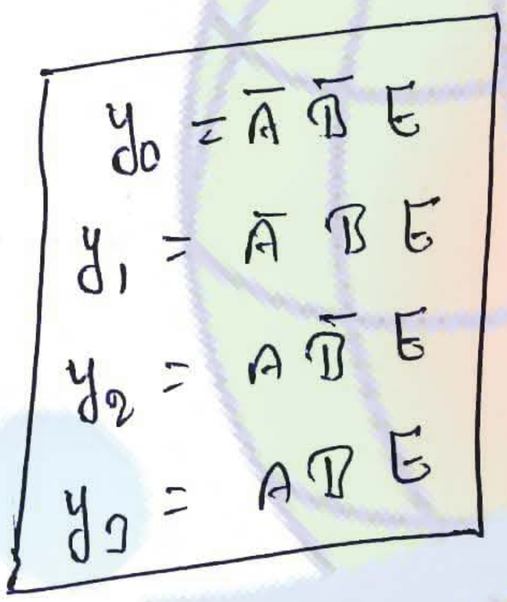
i) Decoder is a combinational circuit, that converts binary info. into other's.

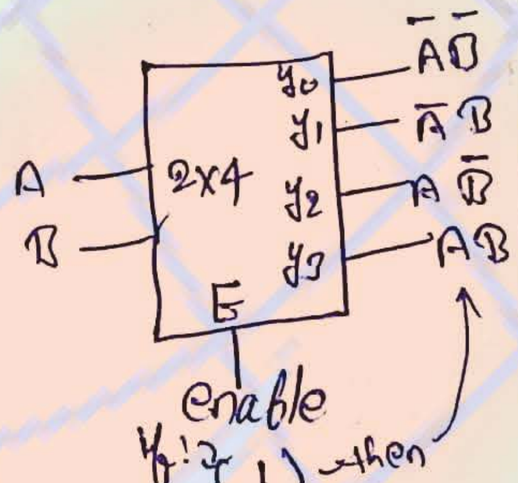
ii) It is a multiinput, multioutput circuit.

eg.> 3x8 (Binary to octal Convertor)
4x10 (BCD to Decimal)
4x16 (Binary to Hexa)

iii) - Smallest size of decoder is 2x4

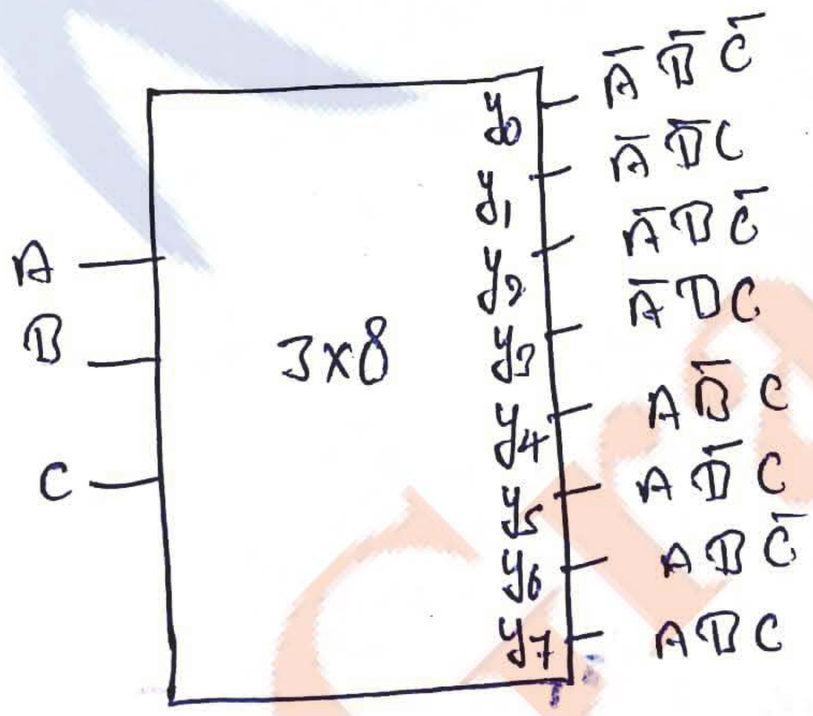
Convert 2 bit binary to decimal



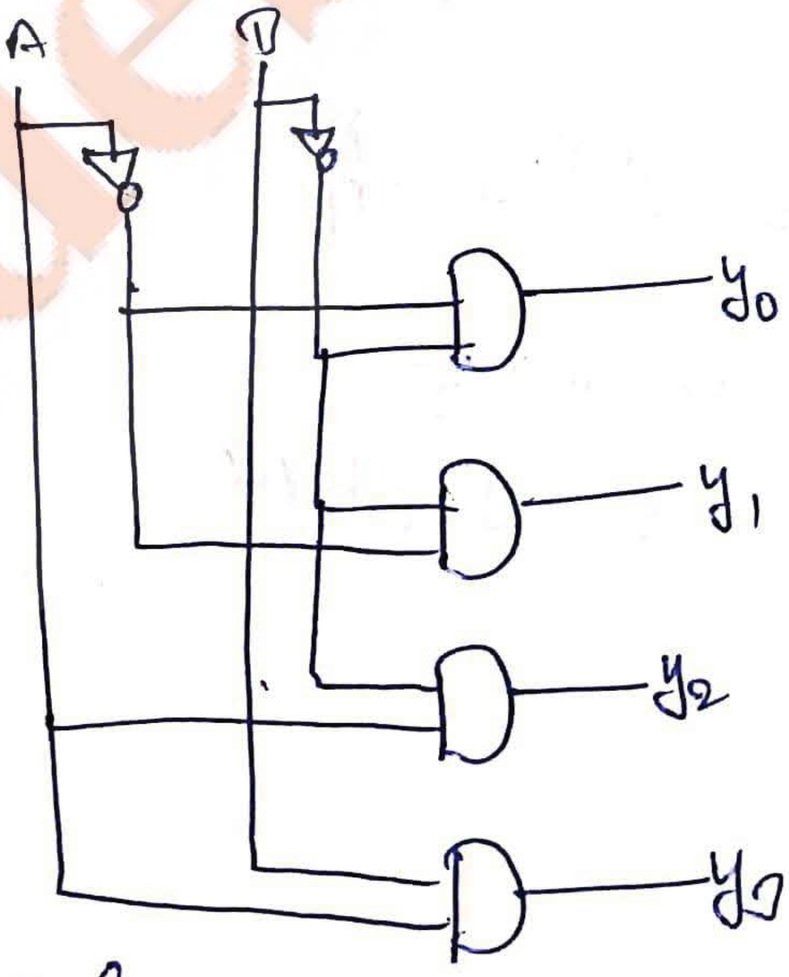


E	A	B	y_0	y_1	y_2	y_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

☆ 3x8 Decoder :-



Q.>

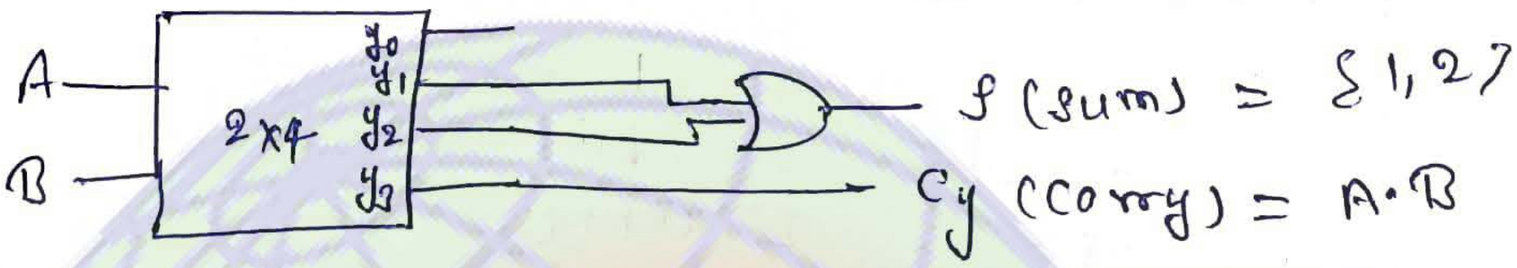


Note:
Decoder and demux have same internal circuitry

into Q: How many 2x4 decoders are requires, if we have OR gates?

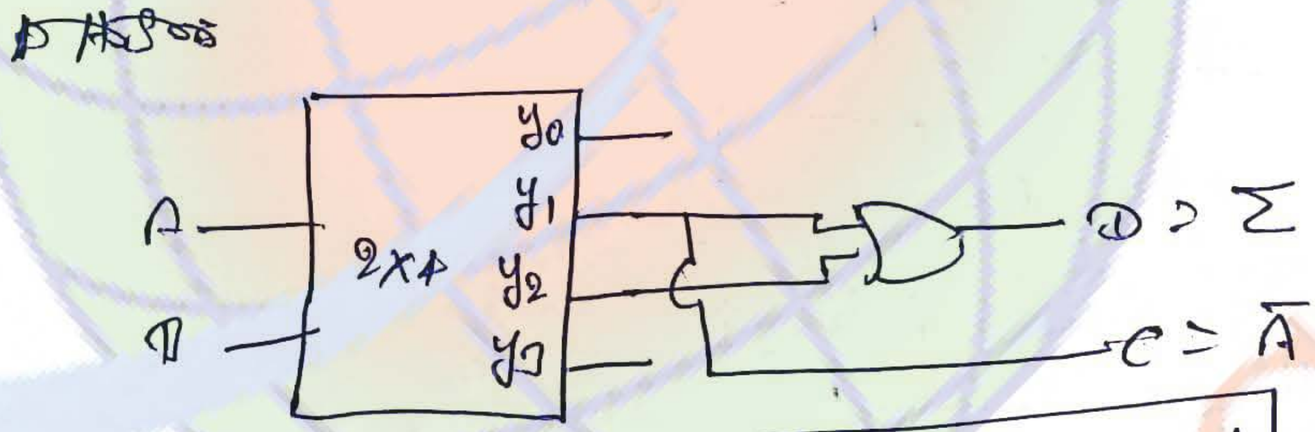
for half adder (HA): ① $S = \Sigma\{1,2\}$

$Cy = \Sigma\{3\}$



1 HA = One 2x4 decoder + 1 OR gate

For half Substractor:-



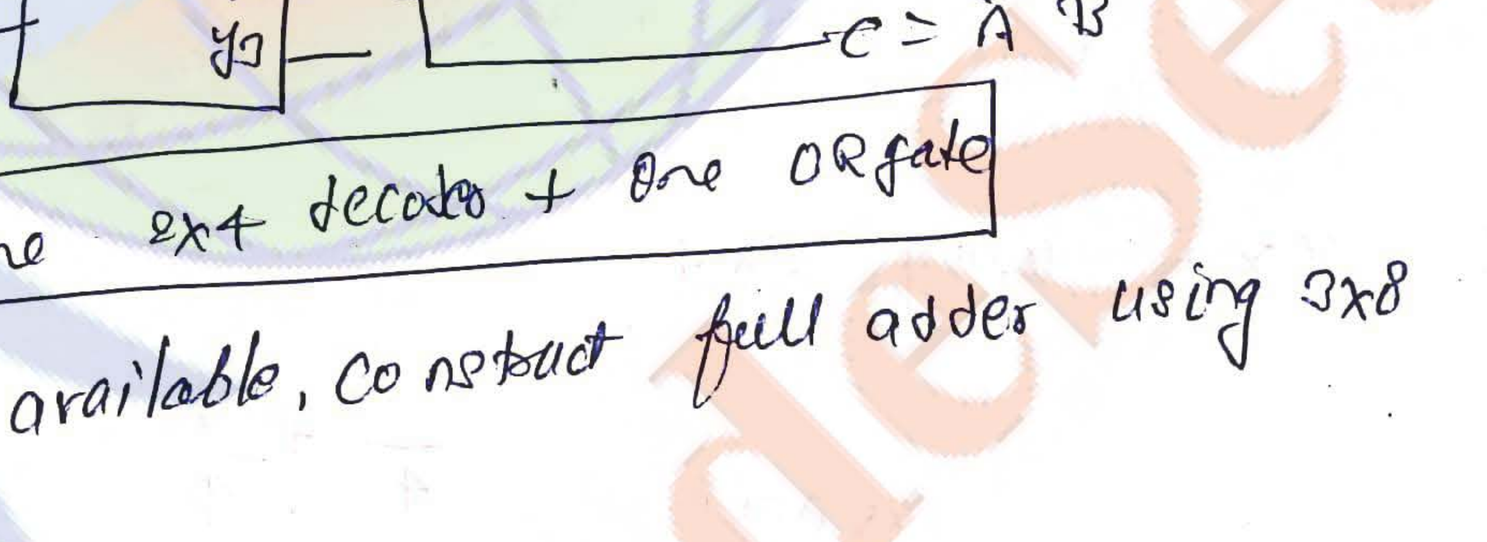
1 HS = One 2x4 decoder + One OR gate

Q) If OR gates are available, construct full adder using 3x8 decoder.

Ans FA using 3x8

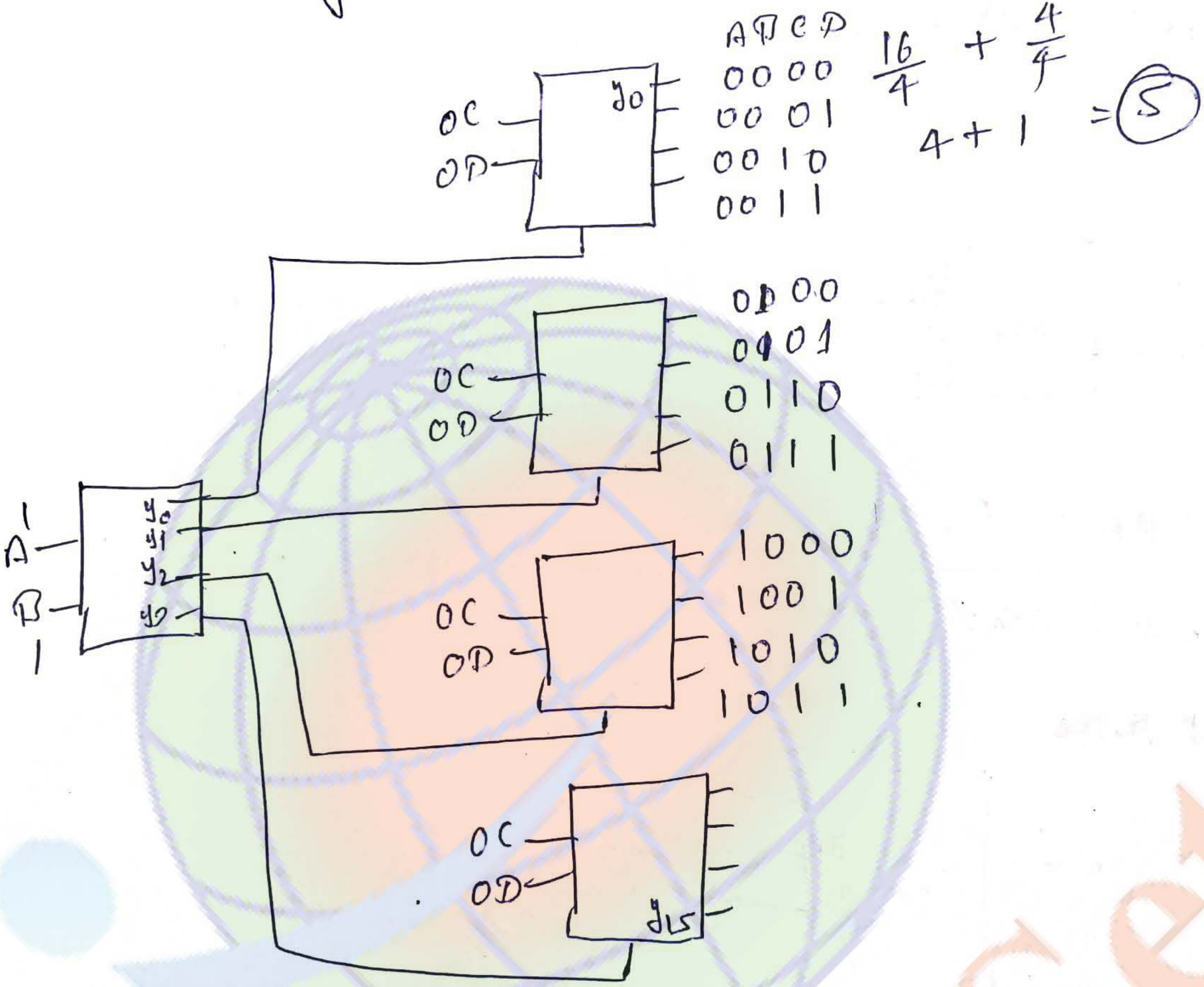
$S = \Sigma\{1,2,4,7\}$

$C = \Sigma\{3,5,6,7\}$



☆ Implementation of Higher order decoder using lower order decod

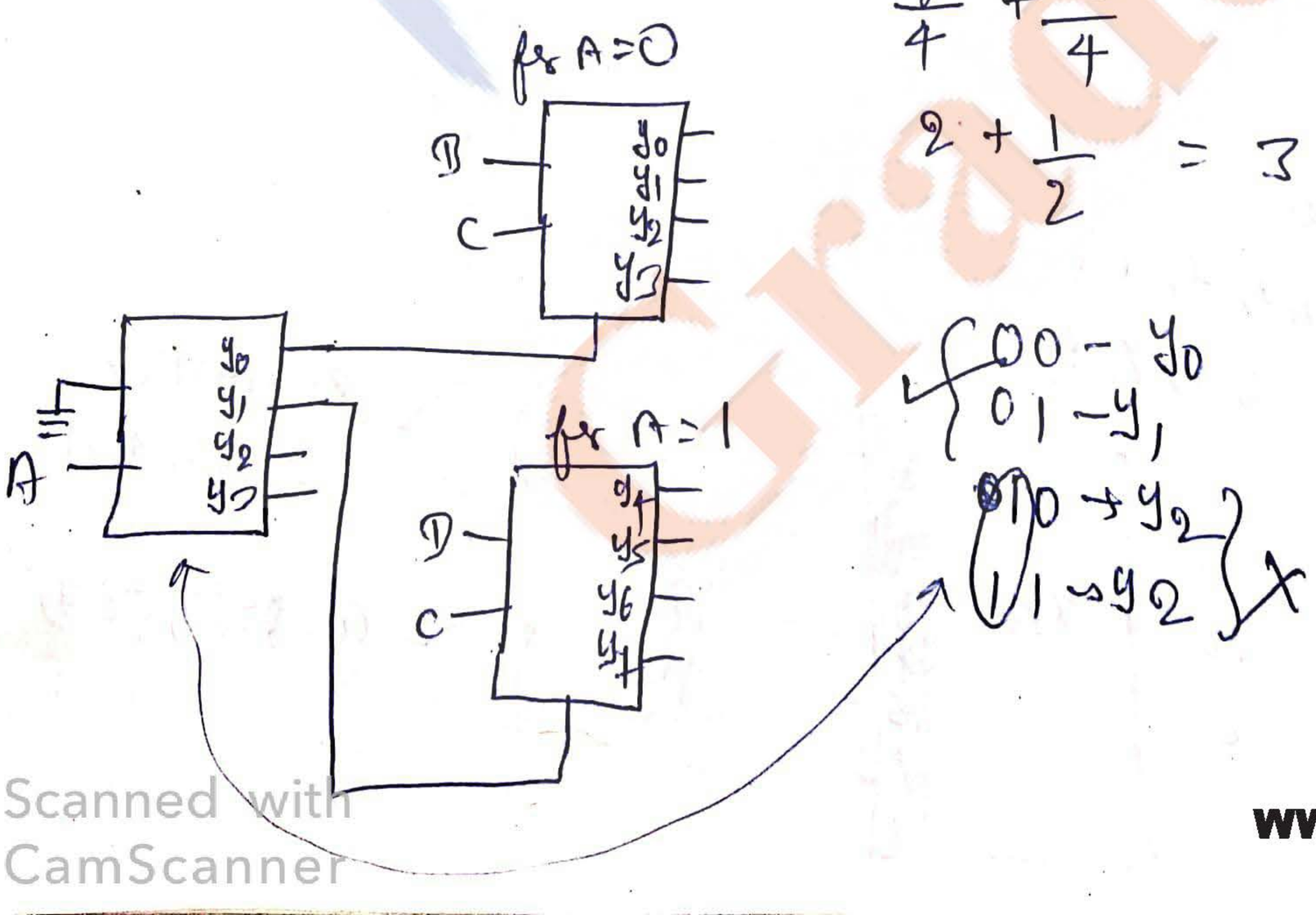
= 4x16 using 2x4



$\frac{16}{4} + \frac{4}{4}$

$4 + 1 = 5$

☆ 3x8 using 2x4



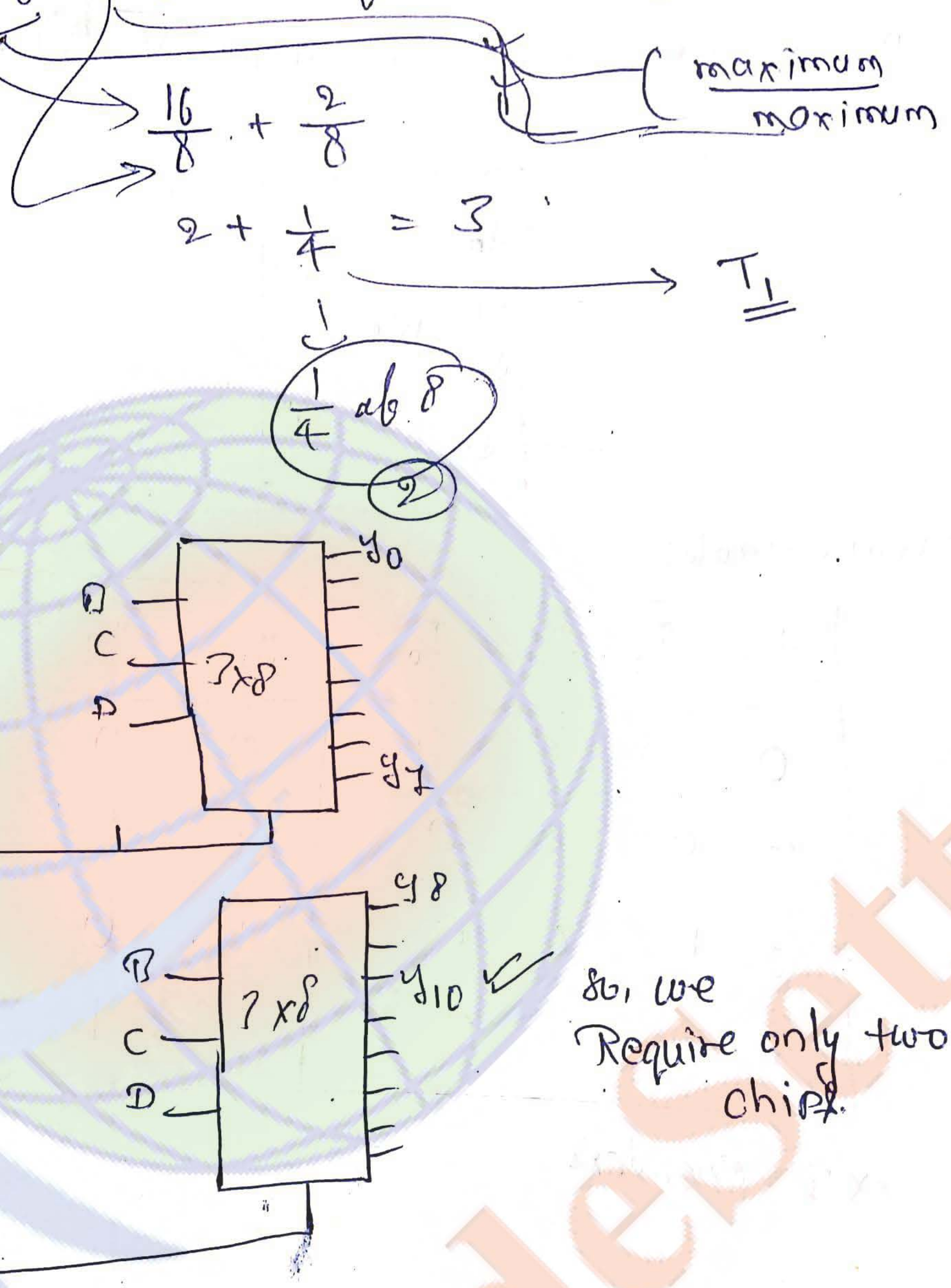
$\frac{8}{4} + \frac{2}{4}$

$2 + \frac{1}{2} = 3$ chip's.

00 - y_0
01 - y_1
10 → y_2
11 → y_2 } ✗

decoder

* How many 3x8 decoders are required to construct 4x16 decoder if one not gate is available.



So, we Require only two chips.

* Encoder :-

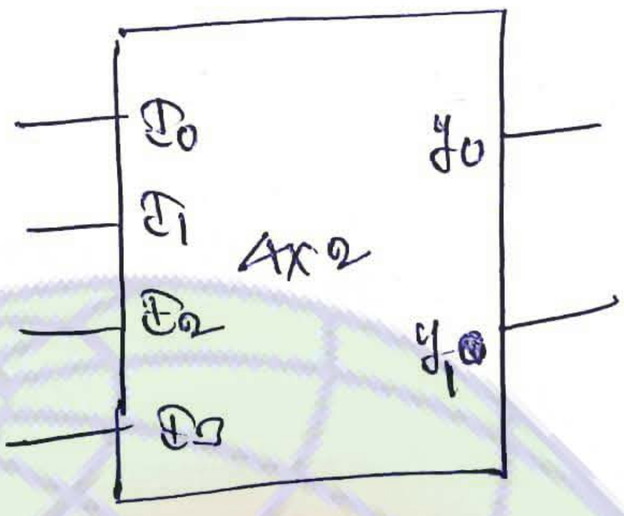
① Encoder is a combinational circuits which has many inputs and many outputs.

② It is used to convert other codes into binary such as - octal to binary
- decimal to BCD
- Hexa to binary etc.

③ multiple input and multiple output circuits.

(4) minimum size = 4x2 encoder

(5) In decoder one output is hight at a time, while in encoder only one input is high at a time.

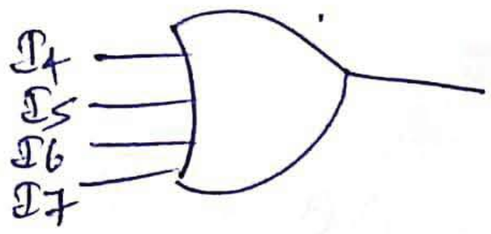


Truth table:-

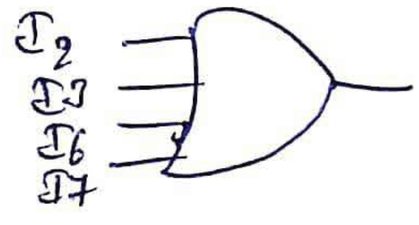
D_3	D_2	D_1	D_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

(6) 8x3 encoders

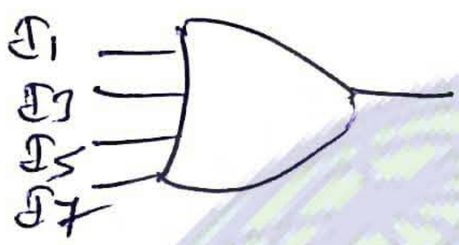
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1



$y_2 = I_4 + I_5 + I_6 + I_7$



$y_1 = I_2 + I_3 + I_6 + I_7$

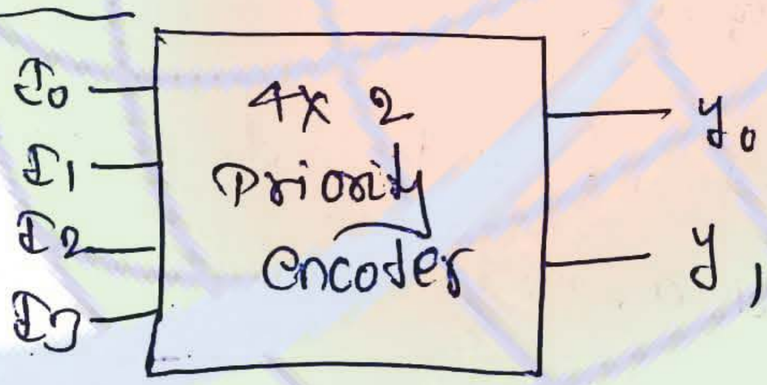


$y_0 = I_1 + I_3 + I_5 + I_7$

Problem arises:-
main problem with encoder is that, encoder fails in case of multiple high inputs

to overcome this problem, Priority encoder is used.

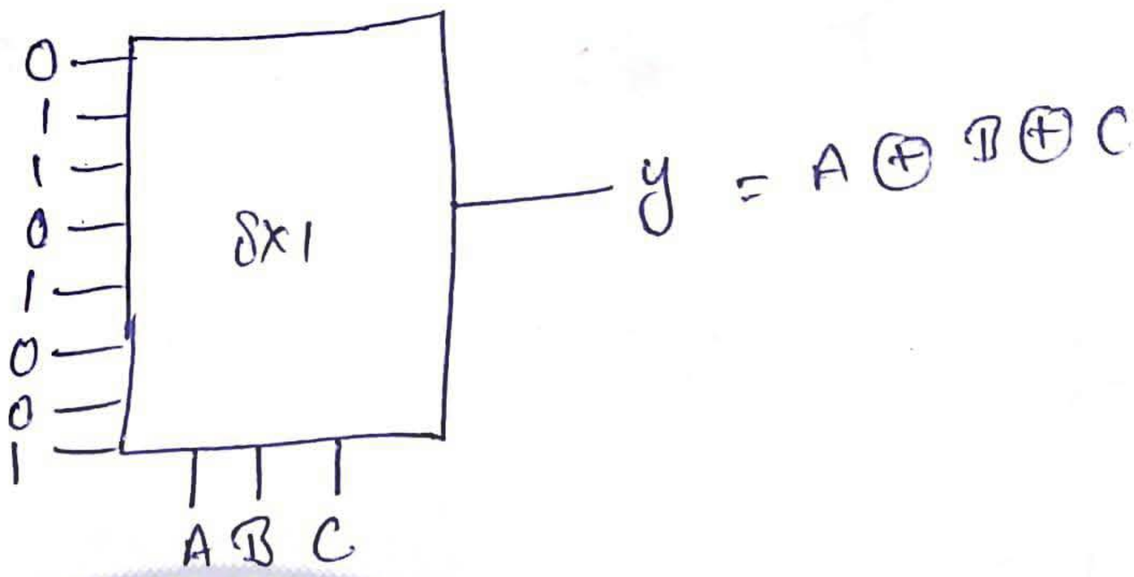
*Priority encoder:

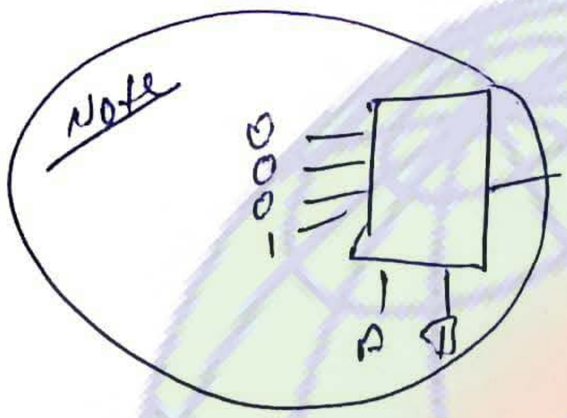


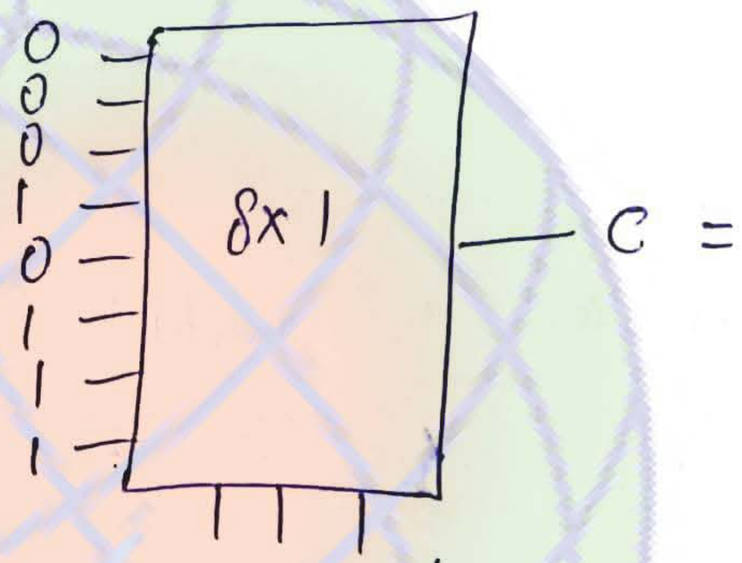
I_3	I_2	I_1	I_0	y_1	y_0
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

Priority encoder is used in flash type Analog to digital converter. (a/d)

Q) IFA = ? 8×1 mux, How many 8×1 mux are needed to construct one full adder







Q) How many 4x1 mux are required

* Full adder :-
No. of 4x1 mux requires if not gate is available
is equal to 2 for full adder.

4.4.) Hazards :

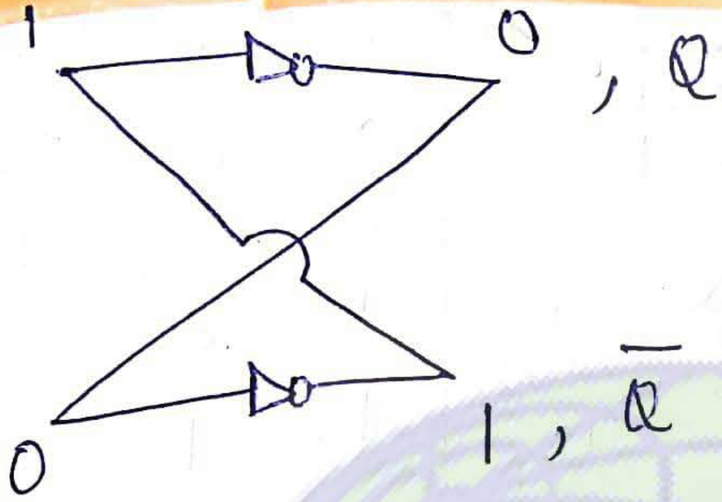
Hazards is a phenomenon that may cause the digital circuit to malfunction.

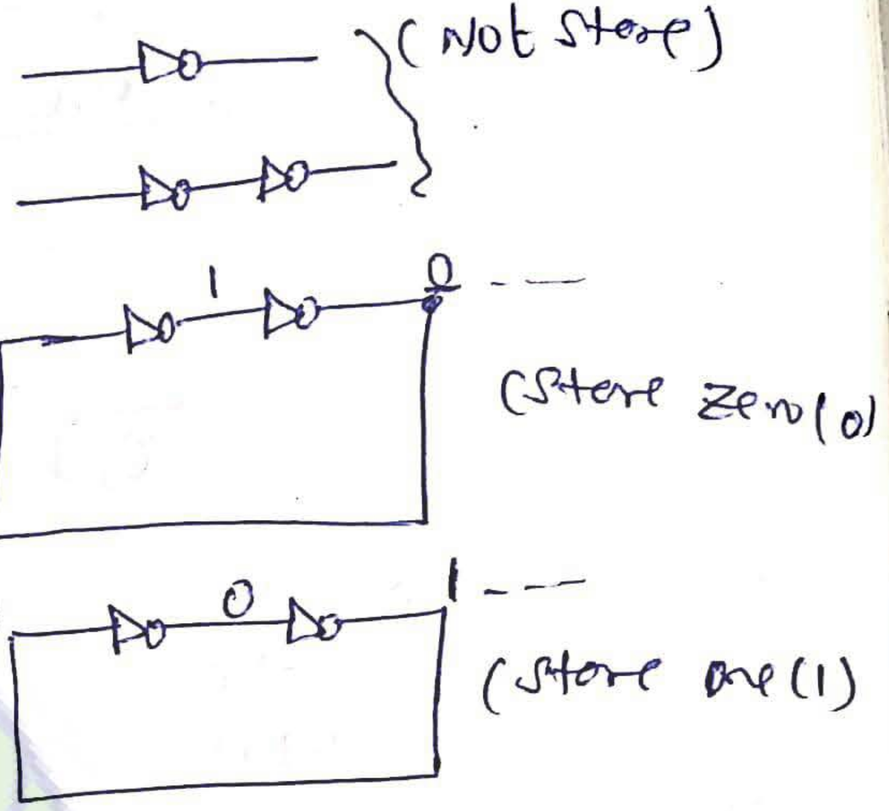
cause of hazards →
- Combination circuit ⇒ temporary false output value
- Sequential circuit ⇒ transition to a wrong stable state

Type of hazards :-
1) static hazards → static-1 hazards, static-0 hazards
2) dynamic hazards
3) essential hazard

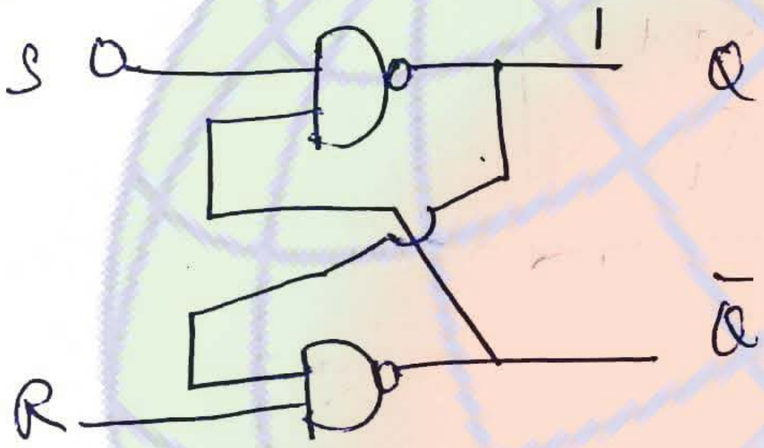
5.) Sequential Circuits

5.1) Latches and flip-flop

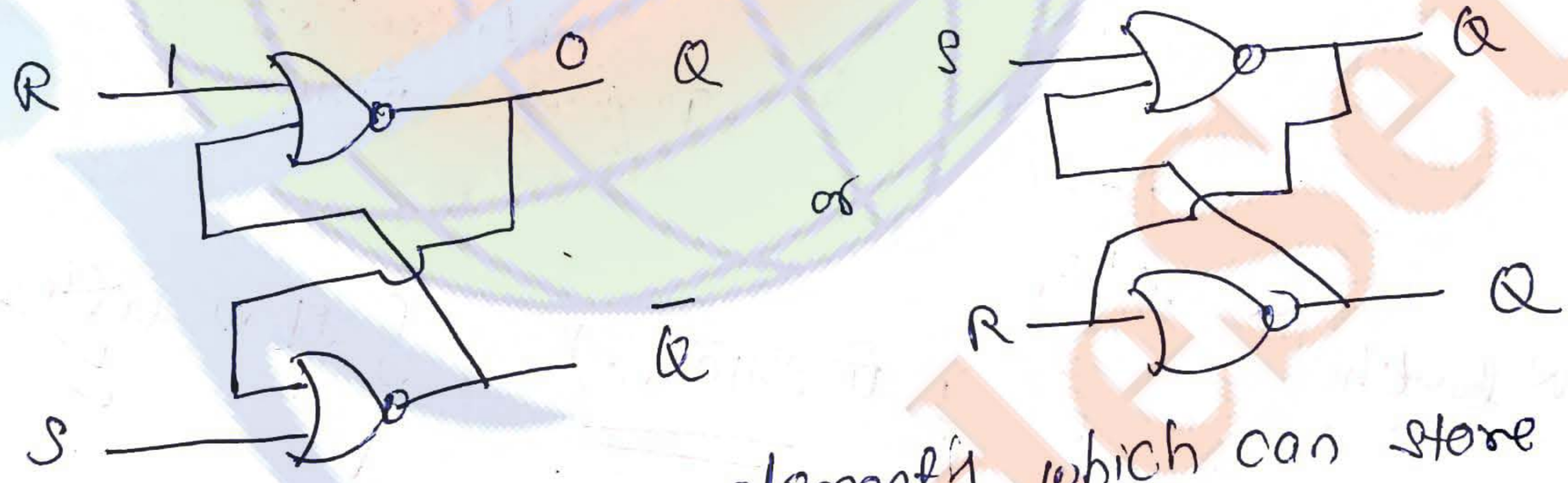




* NAND Latch:-



* NOR latch:-



① Latches are basic memory elements, which can store one bit at a time.

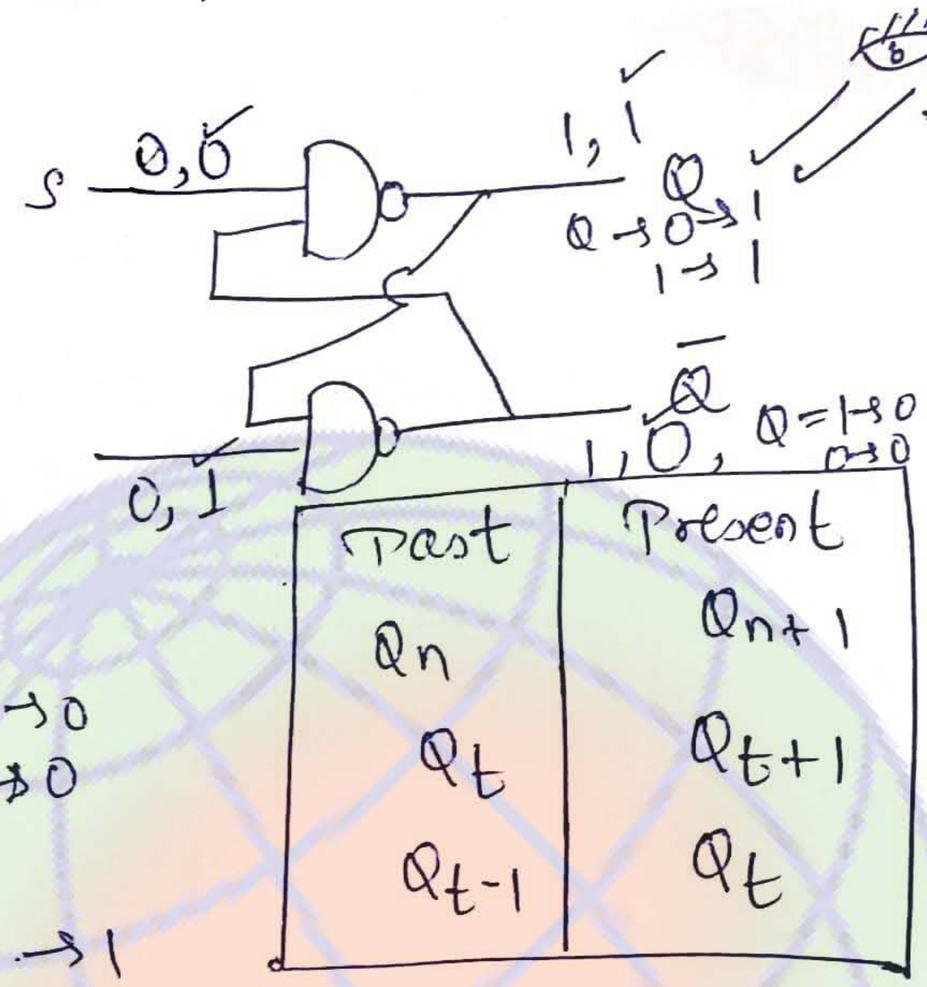
② Latches may be constructed using NAND or NOR.

③ Output is Q and Q̄ is complement of the output

④ In NAND latch and NOR latch portion of Q and Q̄ are different

✡ NAND Latch:

Any '0' at the input make output '1'



S	R	Q_{n+1} $\bar{Q}$
0	0	Indeterminate or Invalid
0	1	1
1	0	0
1	1	Q_n = (Previous value)

1 Q 0→0 1→1 1→1 0→0 } अगर Directly '0' को नहीं apply करते तो feedback से Zero को Prefer करता है।

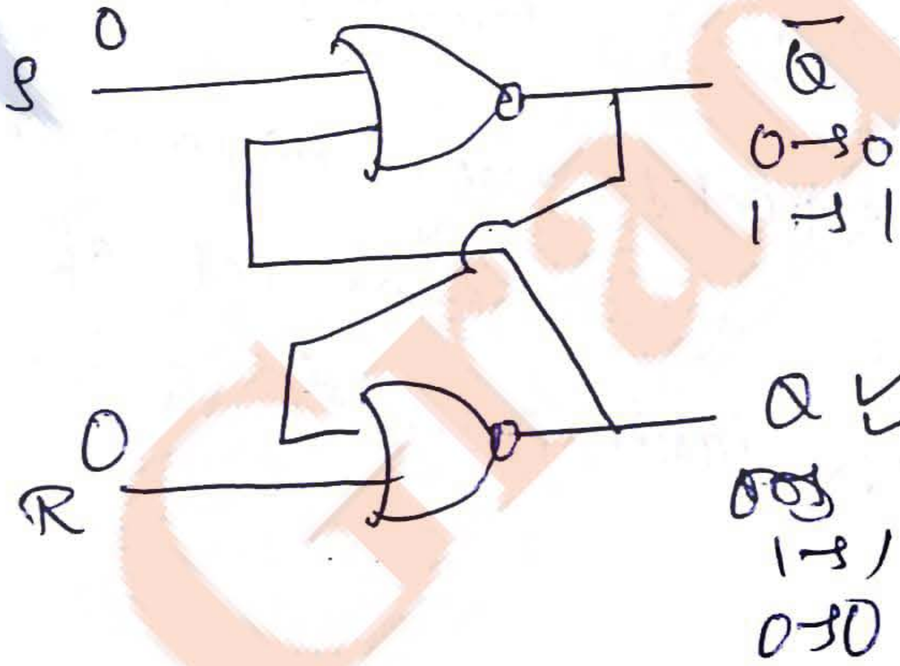
0

1

1

✡ NOR latch:

('1' पे हमारा है) { It is better }



S	R	Q
0	0	Q_n (Previous state)
0	1	0
1	0	1
1	1	invalid

Nor is better in term of truth table

(3) flip flops are converted into flip-flops by applying clock Pulses.

(4) clock is a square wave that is used to decide exact timing of operation of circuit.

(5) Synchronous → same time
Asynchronous → different time

(4) clock is used to synchronise operations.

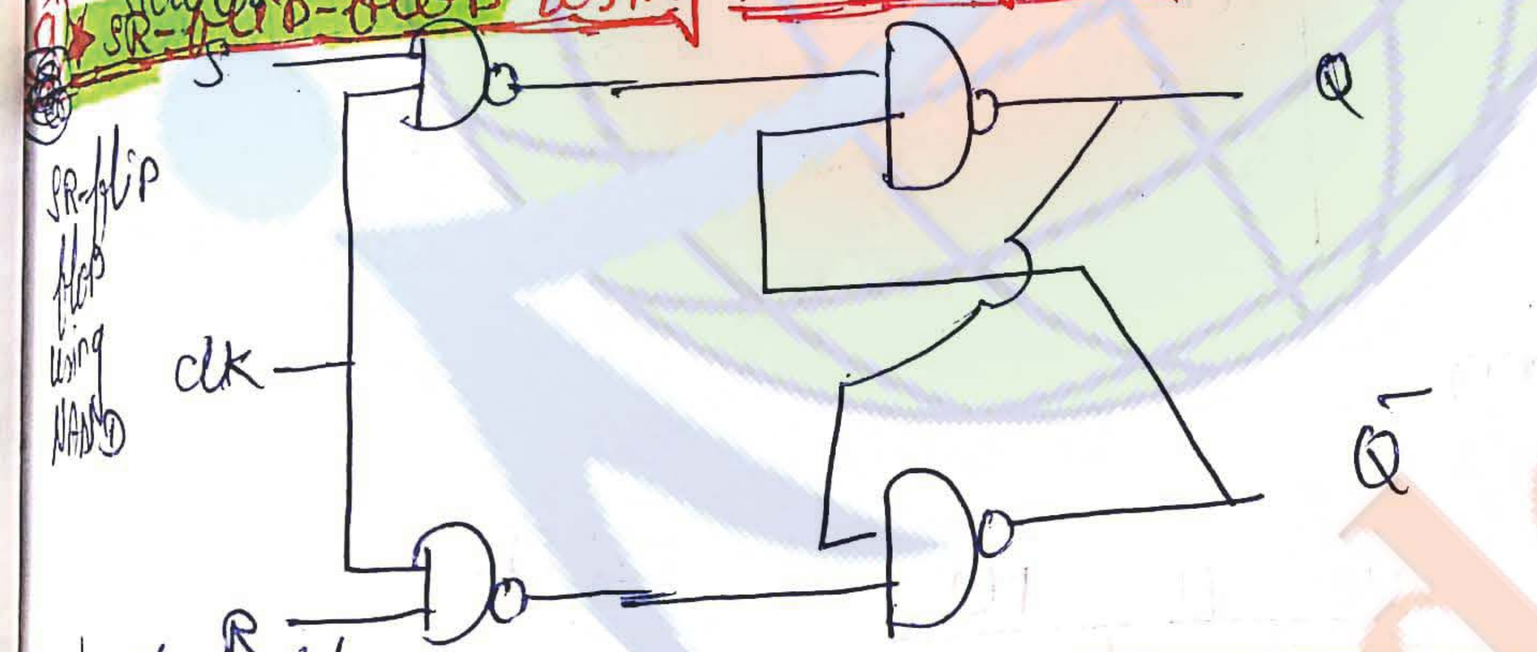
(5) circuits which are synchronous have all of their operations defines at Precised moments.

so, synchronous circuits are faster, error immune and flexible in designing.

(6) circuits which do not use clock i.e asynchronous.

Asynchronous circuits are less flexible in design, slower and error prone.

SR-flip-flop using NAND gate :-

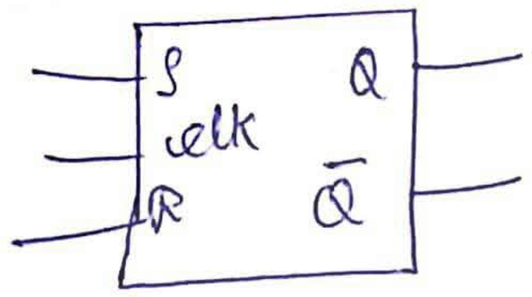


SR-flip flop using NAND

Truth table :-

clk	S	R	Q_{n+1}
0	x	x	Q_n → memory
1	0	0	Q_n → hold
1	0	1	0 → Reset
1	1	0	1 → Set
1	1	1	x → Invalid

Symbol

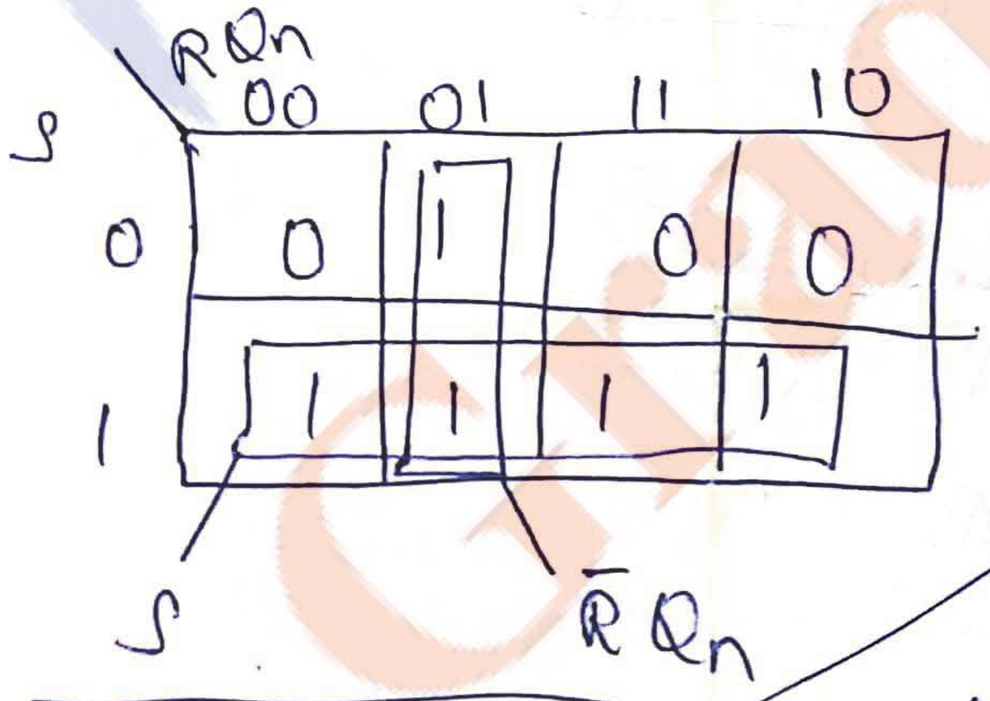


Characteristics/extented truth table

S	R	Q	Q_{n+1}	
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	invalid (X)	} means, it should not be used.
1	1	1	X	

Characteristic equation

K-map



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

This eqn is called characteristic eqn of SR-flip flop

with $S\bar{R}$ condition

$$S\bar{R} = 0$$

} Result

excitation → means input

excitation table

Q_n	Q_{n+1}	S	R
0	0	0	X → don't care
0	1	1	0
1	0	0	1
1	1	X	0
		0	0
		1	0

1 → No change
↘ set

excitation - input
Response - output