DIGITAL DESIGN NOTES

AKSHANSH CHAUDHARY

Digital Design Notes, First Edition

Presented by: Akshansh Chaudhary
Graduate of BITS Pilani, Dubai Campus
Batch of 2011

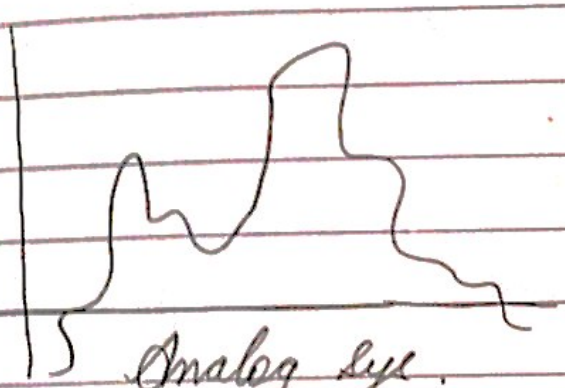
Course content by: Dr. R. Mary Lourde
Then Faculty, BITS Pilani, Dubai Campus

Layout design by: AC Creations © 2013

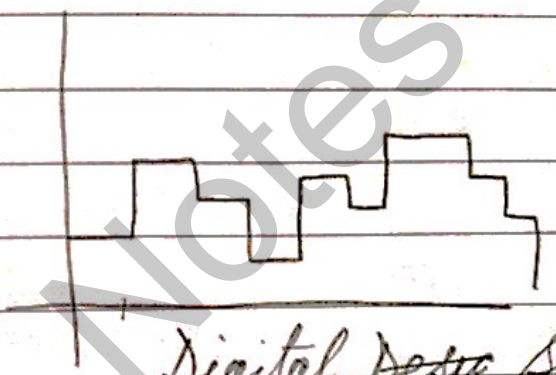
The course content was prepared during Fall, 2012.
More content available at: www.Akshansh.weebly.com

Date 10/9/2012
Page

DIGITAL SYSTEM



Analog sys.



Digital ~~Data~~ sys.

Discrete sys.

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

* BCD (Binary Coded Decimal)
Binary represent^{n} of each of Decimal digits (from 0 - 9)
eg: BCD for $(368)_{10}$ = (0011 0110 1000)

Puffin
Date
Page

Q. $(368)_{10} = (?)_2$

M1

2	368	
2	184	0
2	92	0
2	46	0
2	23	0
2	11	1
2	5	1
2	2	1
	1	0

$(101110000)_2$

M2 $368 = 256 + 112$

$112 = 64 + 48$

$48 = 32 + 16$

* +ve logic :
On, High, yes, true are represented as 1 & off, of that by 0, its called +ve logic in Digital sys

* -ve logic :-

High	Low	Notation by
On	off	
Yes	No	
True	False	
1	0	: +ve logic
0	1	: -ve logic

* AND gate defin^{n} (fn) :-
Its o/p will be high only when all the i/p's are high.

* Truth table :
A table which gives all o/p's state by state for each combin^{n} of i/p's state.

* Truth table for AND gate

A	B	o/p (A·B) = F
High (H)	low (L)	L
H	H	H
L	L	L
L	H	L

4 possibilities
no. of var.
2

go from low to high in general

Assuming +ve logic		
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

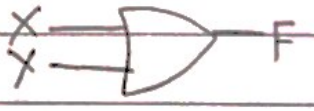
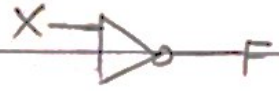
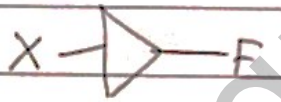
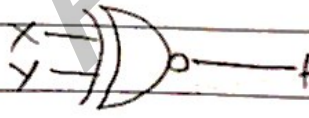
Assuming -ve logic		
A	B	F
1	1	1
1	0	1
0	1	1
0	0	0

+ve logic AND = -ve logic OR.

* Buffer is used to cause delay, store same data for a while.

Puffin
Date
Page

* Gates

Name	Graphic symbol	Algebraic fn (Boolean fns)
• AND		$F = X \cdot Y$
• OR		$F = X + Y$
• Inverter (NOT)		$F = X'$
• Buffer		$F = X$
• NAND		$F = (X \cdot Y)'$
• NOR		$F = (X + Y)'$
• Exclusive-OR (XOR)		$F = XY' + X'Y$ $= X \oplus Y$
• Exclusive-NOR or Equivalence (XNOR)		$F = XY + X'Y'$ $= (X \oplus Y)'$

→ odd fn logic gate (o/p is high ONLY when odd no. of i/p are h

* $X' = \overline{X}$
* For n no. of bits, ∃ 2^n combin^{ns} (0 to $2^n - 1$)

• How to write a Boolean expr^n.
Possible expression for i/p giving a high o/p.

- Way to remember binary equivalent of some digits.

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

A ⊕ B

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

✡ MINTERMS

x	y	z	Term	Designation
0	0	0	$x'y'z'$	m_0
0	0	1	$x'y'z$	m_1
0	1	0	$x'yz'$	m_2
0	1	1	$x'yz$	m_3
1	0	0	$xy'z'$	m_4
1	0	1	$xy'z$	m_5
1	1	0	xyz'	m_6
1	1	1	xyz	m_7

(high)

✡ MAXTERMS

— DO —

* High o/p (1) gives minterms.
* Low o/p (0) gives maxterms.

	x	y	z	f_1	f_2
Q.	0	0	0	0	0
	0	0	1	1	0
	0	1	0	0	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

$f_1(x,y,z) = x'y'z + xy'z' + xyz$

$= m_1 + m_4 + m_7$

= Sum of Product (SOP)

= SO minterms

$= \sum(1,4,7)$ (canonical form)

$f_1(x,y,z) = (x+y+z)(x+y'+z)(x+y'+z')$
$(x'+y+z')(x'+y'+z)$

$= M_0 M_2 M_3 M_5 M_6$

= Product of Sum (POS)

$= \pi(0,2,3,5,6)$

Similarly, $f_2 = \sum(3,5,6,7)$, $f_2 = \pi(0,1,2,4)$

* Two level & 3 level circuit.

↳ only 2 times the delay of each logic gate.

↳ delay in circuit before it reaches o/p.

Puffin
Date
Page

✱ Rules of Boolean Algebra.

- $X + XZ = X$
- $X(X+Y) = X$
- $(X+Y)(X+Z) = X+YZ$
- $X + \bar{X}Y = X+Y$
- $XY + YZ + \bar{Y}Z = XY+\bar{X}Z$

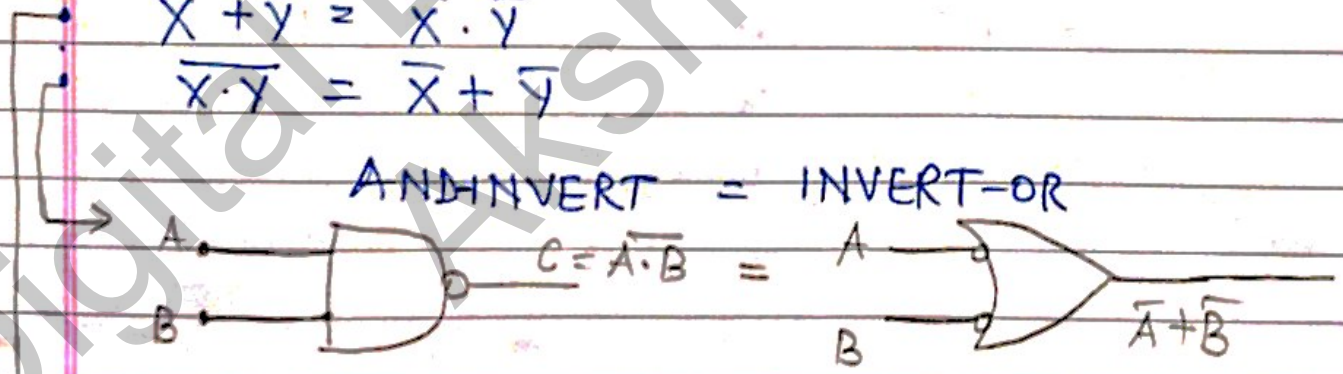
✱ Operator precedences: assigning priority

a. ( )
b. NOT
(c) AND
(d) OR

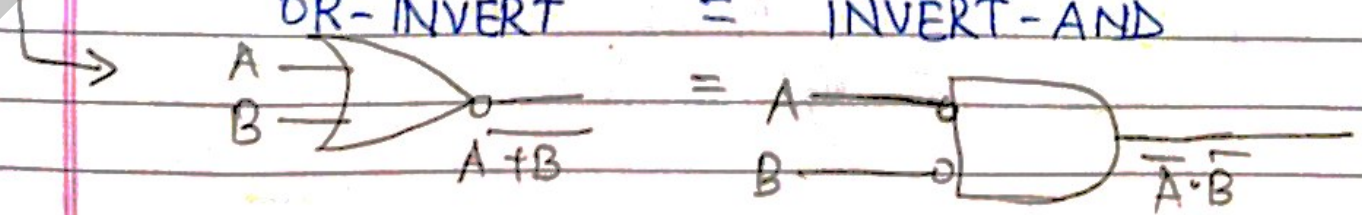
✱ De' morgan's theorem:-

- $\overline{X+Y} = \bar{X} \cdot \bar{Y}$
- $\overline{X \cdot Y} = \bar{X} + \bar{Y}$

AND-INVERT = INVERT-OR



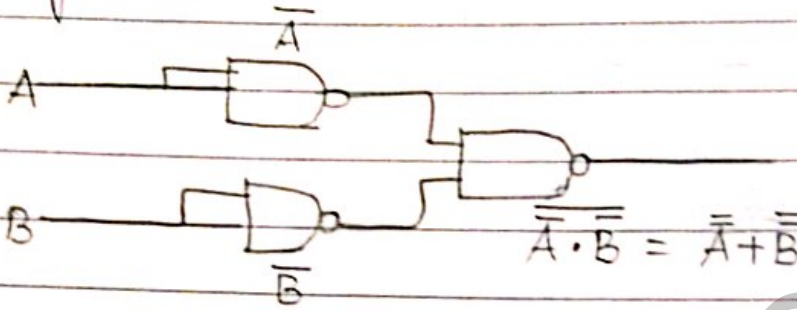
OR-INVERT = INVERT-AND



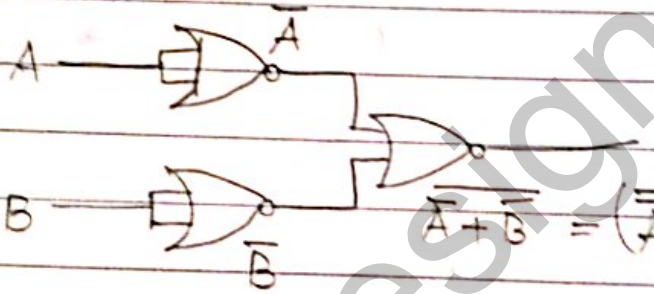
✱ Universal logic Gate:

NAND
NOR.

- OR gate using NAND



- AND gate using NOR



Q. $A \overset{XOR}{\oplus} 0 = A$

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

$A \oplus A = 0$

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

A	0	$A \oplus 0$
0	0	0
1	0	1

→ A

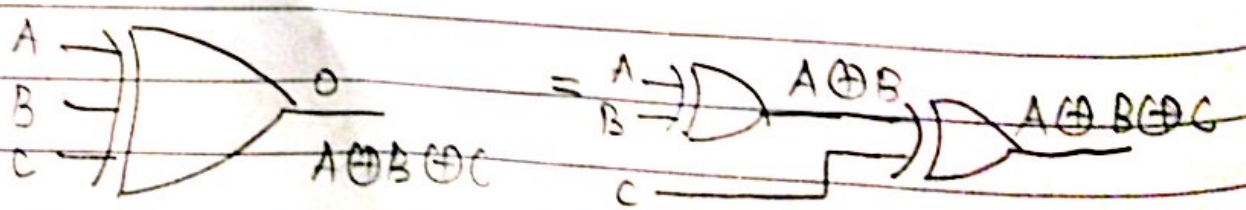
A	1	$A \oplus 1$
0	1	1
1	1	0

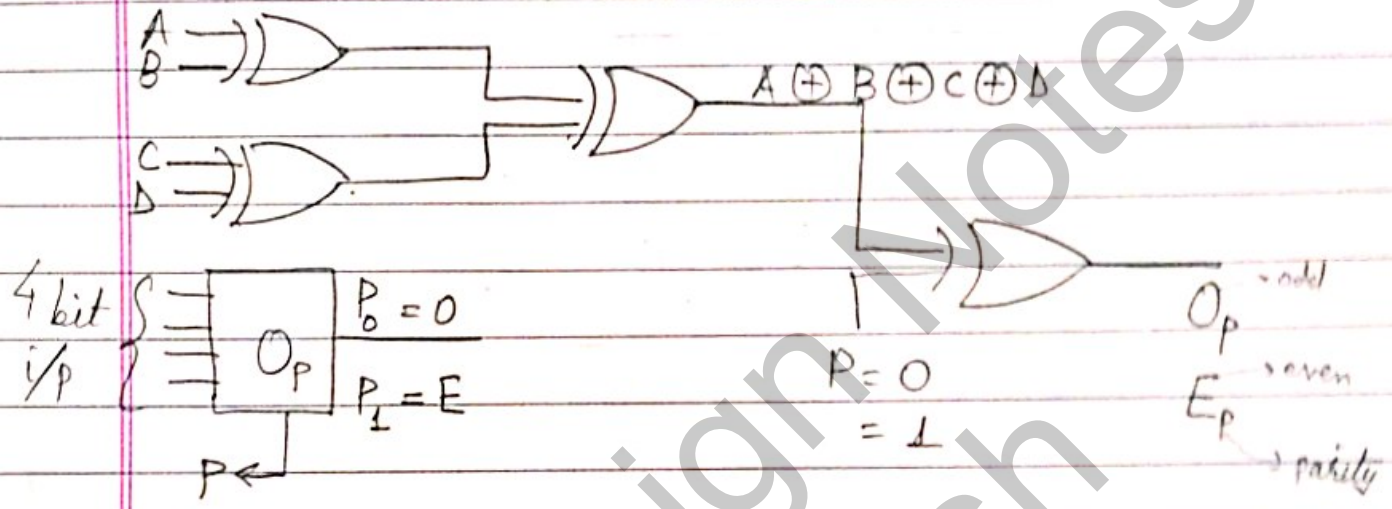
→ $\bar{A}$

★ ODD PARITY GENERATOR

If no. of high i/p is ODD, o/p = high.

3 bit odd parity generator circuit





Q. Implementation using NAND gate.

$F(A, B, C) = \sum(3, 5, 6, 7)$

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

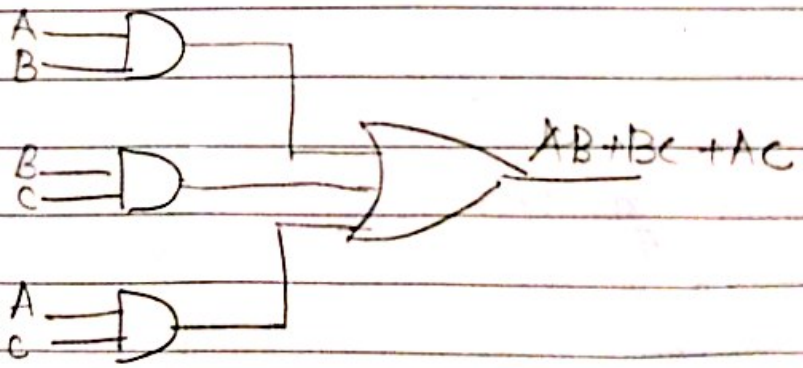
$+ AB(C + \bar{C})$ → 1

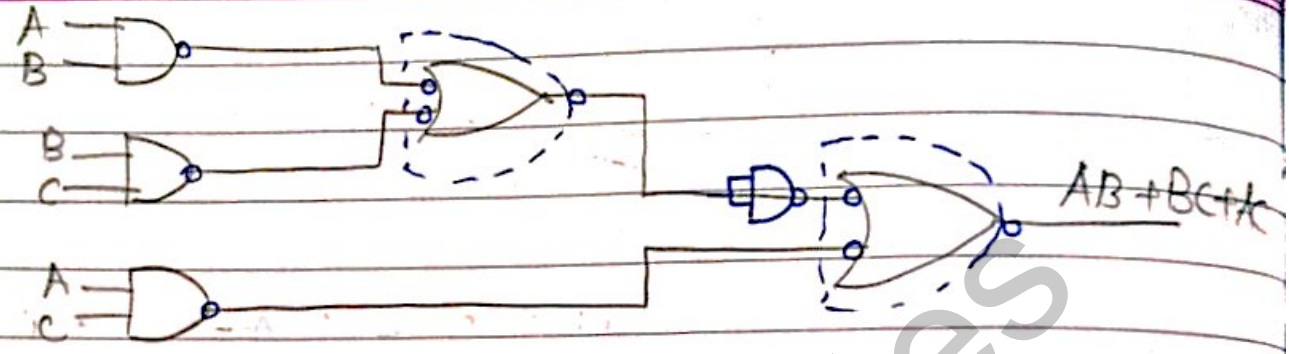
$+ AC(B + \bar{B})$ → 1

$+ BC(A + \bar{A})$ → 1

{ Taken ABC 3 times ∵ $x + x + x = x$ }

$= AB + BC + AC$





logically : $\overline{\overline{AB + BC + AC}}$

$= \overline{\overline{AB} \cdot \overline{BC} \cdot \overline{AC}}$

$= \overline{\overline{AB} \cdot \overline{BC}} + \overline{\overline{AC}}$

$= \overline{\overline{AB}} + \overline{\overline{BC}} + \overline{\overline{AC}}$

$= AB + BC + AC$

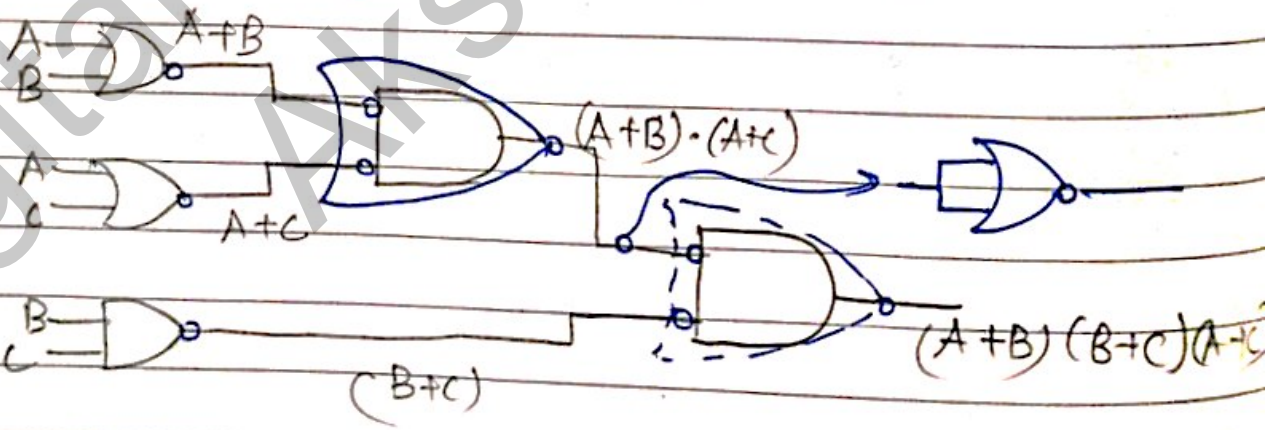
Implementⁿ using NOR gate

POS form = $\Pi(0, 1, 2, 4)$

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

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

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



logically : $\overline{\overline{(A+B)\cdot(A+C)(B+C)}}$

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

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

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

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

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

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

$= \sum(1,4,5,6,7)$

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

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

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

$\therefore 0$

* BCD code : Binary Coded Decimal : (0-9)

* 8421 code (Binary code) : 0 - 15
(2^3 2^2 2^1 2^0)

Puffin Date 17/9/2 Page

✱ NUMBER SYSTEMS & BINARY CODES

• BINARY ADDITION

* 1 + 1 = 0 & a CARRY

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{array}{r} 1\ 0\ 1\ 0 \\ +\ 1\ 1\ 0\ 0 \\ \hline (1)\ 0\ 1\ 1\ 0 \end{array} \equiv \begin{array}{r} 10 \\ +12 \\ \hline 22 \end{array}$$

$= 2 + 4 + 16$

• BINARY SUBTRACTION

0 - 0 = 0

1 - 0 = 1

10 - 1 = 1

1 - 1 = 0

$0 \times 2^0 + 1 \times 2^1 = (2)$

$k2 - 1 = 1$

borrow from second bit

A	B	Difference = A - B	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$\begin{array}{r} 1\ 1\ 1 \\ -\ 1\ 0\ 1 \\ \hline 0\ 1\ 0 = 2 \end{array} \equiv \begin{array}{r} 7 \\ -5 \\ \hline 2 \end{array}$$

✱ ~~GRAY CODE CIRCUIT~~

MSB

✱ (2)421

For 1st 5 representⁿ (0-4) keep MSB = 0.

From 5-9, keep MSB = 1 (weightage = 2)

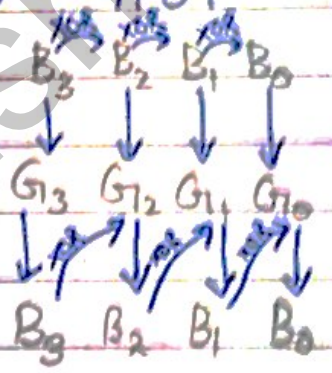
& distribute the rest accordingly.

Add 3 to every decimal & convert to binary

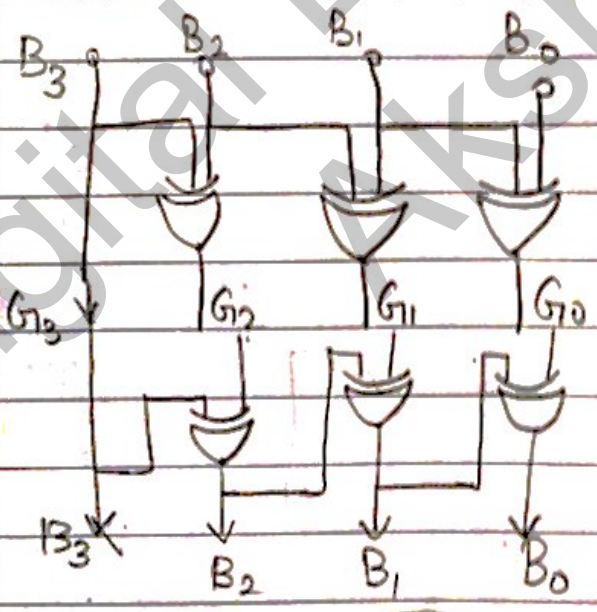
only 1 bit change b/w consecutive.

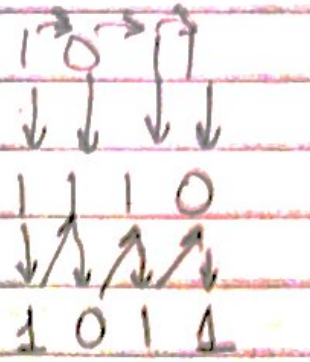
Puffin

Decimal no.	Binary BCD/8421	X_s-3	Gray code	2421
0	0000	0011	0000	0000
1	0001	0100	0001	0001
2	0010	0101	0011	0010
3	0011	0110	0010	0011
4	0100	0111	0110	0100
5	0101	1000	0111	1011
6	0110	1001	0101	1100
7	0111	1010	0100	1101
8	1000	1011	1100	1110
9	1001	1100	1101	1111



GRAY CODE CIRCUIT





☆ CHARACTERISTICS OF IC :

1. NOISE MARGIN
2. PROPAGATION DELAY
3. POWER DISSIPATION ($P_{avg} = V_{cc} \cdot I_{cc\,avg}$)
4. FAN IN / FAN OUT
   → no. of i/p & o/p that can be connected to an IC

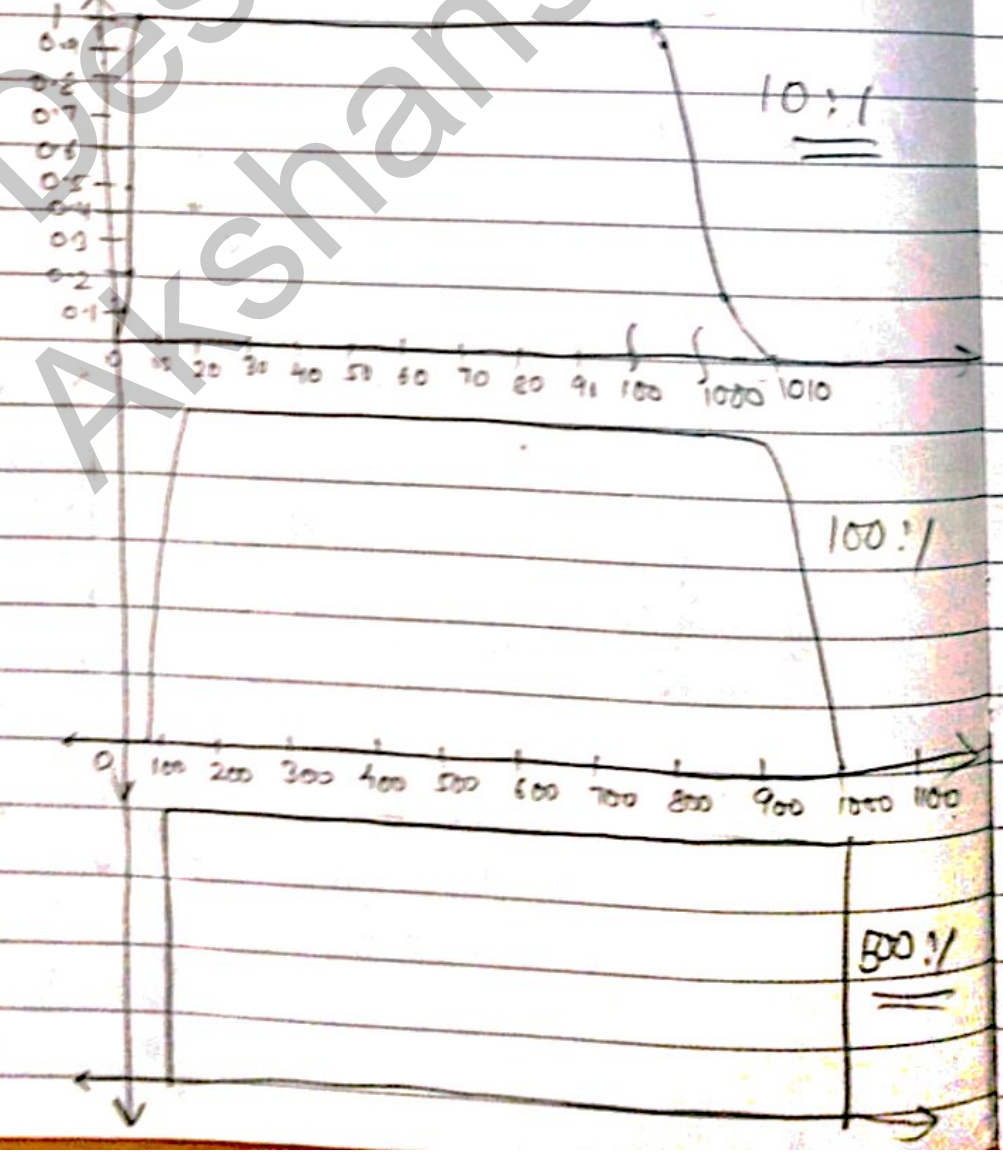
what's current driven when off is less

$\therefore I_{cc\,avg} = \frac{I_{cc\,High} + I_{cc\,Low}}{2}$

Q, Plot the following

x	y
0	0
1	0.1
2	0.2
5	0.5
10	1
100	1
1005	1
1001	0.9
1005	0.2
1010	0

① 10:1
② 100:1
③ 500:1



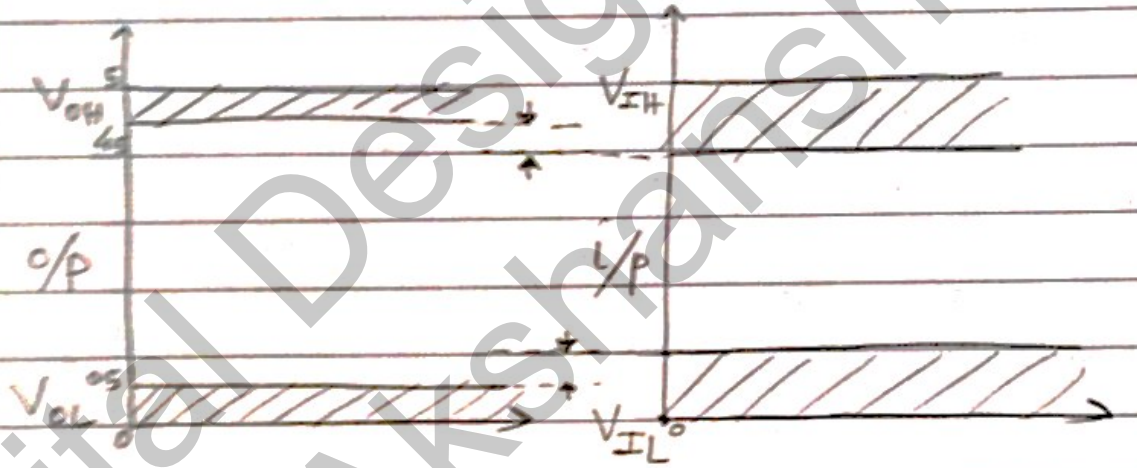
* t_p : time$_{propagation}$ = time delay = $\frac{t_{P_{LH}} + t_{P_{HL}}}{2}$.

* $t_{P_{LH}}$: time delay from low to high (rising edge)
* $t_{P_{HL}}$: time delay from high to low (falling edge)

-x-

* Noise Margin (NM)

Noise signal each of the logic gate can accomodate in the i/p without having any logic change in the o/p.



NM_L (Low Noise Margin) = $V_{IL} - V_{OL}$

NM_H = $V_{OH} - V_{IH}$

* FAN IN / FAN OUT : No. of i/p or o/p's that we can connect to a particular logic gate. handling

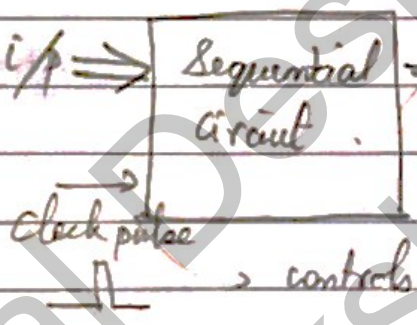
• FAN OUT : will indicate the current carrying/ capability of an IC.

☆ COUNTER CIRCUITS

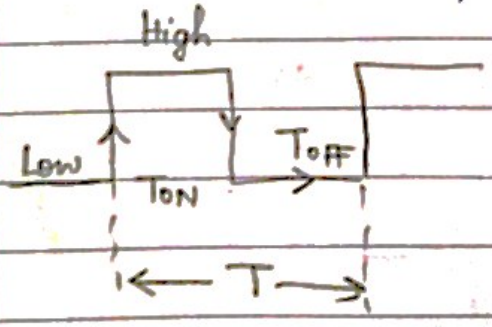
4-BIT BINARY COUNTER

- Clock pulse : A square pulse with 50 % duty cycle.
- Duty cycle : Dur^{n} of transfer of power from i/p to o/p.

  eg : DC = 50 % ⇒ 50 % of power is transfered from i/p to o/p.



* LATCH : Stores 1 bit of info. (So, very basic device for holding data)

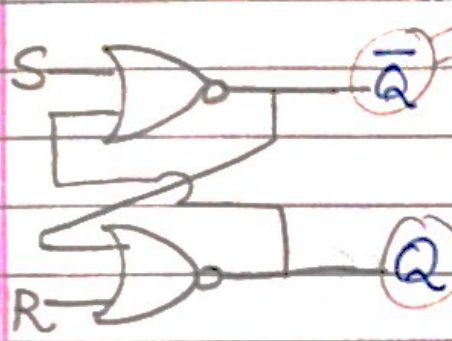


- Level triggered : when triggering done at high or low level.
- Edge triggered : when triggering done on Rising edge (+ve Edge) or Falling edge (-ve edge).

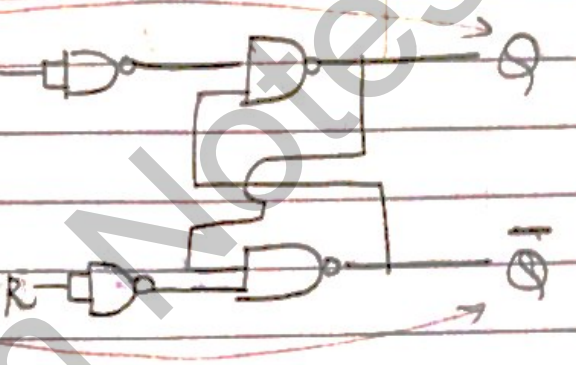
- Latch is a level triggered device.
- Flip-flop is an edge triggering device.

Latches

SR Latch (using NOR gate)

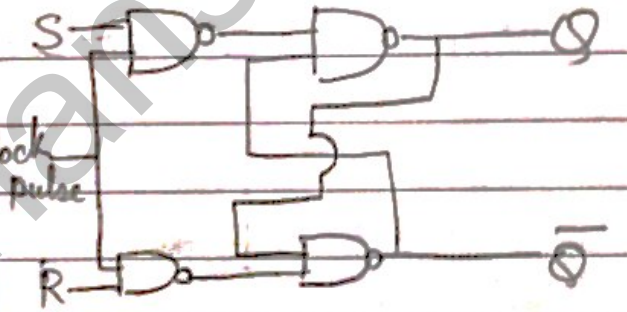


SR Latch (using NAND gate)



- Is designed to give complementary o/p.

✱ SR flip-flop.



- If both of S & R are 1, (Q & $\bar{Q}$) are 0 (∵ S is 1 & o/p is 0 for NOR. 11y for R)

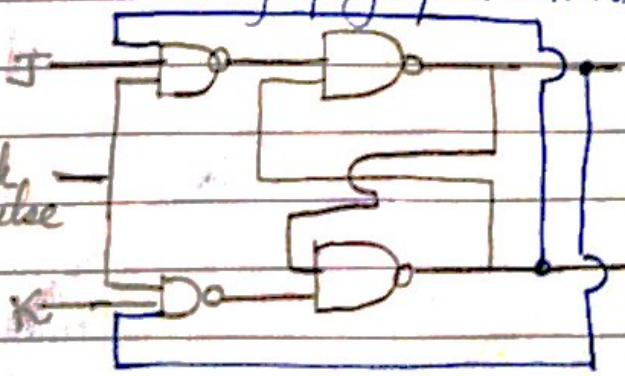
Way to remember: Set, Reset

S	R	$Q_{(t+1)}$	$\bar{Q}_{(t+1)}$
0	0	Q_t	$\bar{Q}_t$ (Previous state)
0	1	0	1
1	0	1	0
1	1	0	0

redundant state ↓ Invalid.

its setting i/p — its clearing the o/p

✱ JK flip flop (removing redundant state)

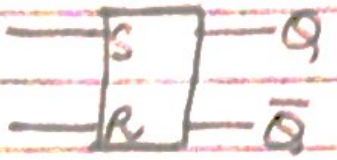


J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	$\bar{Q}_t$

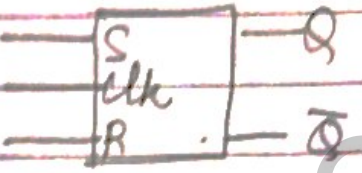
↙ Invalid state rectified.

* Representation

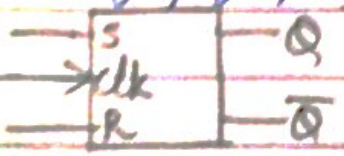
- SR Latch



- SR flip flop



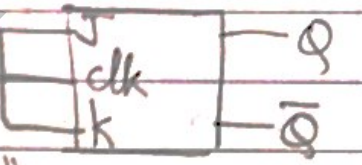
- +ve edge triggered SR flip flop



- -ve edge triggered SR flip flop.

- JK flip flop

- Characteristic JK flip flop.



⇓ same i/p for both terminals

i.e, J K

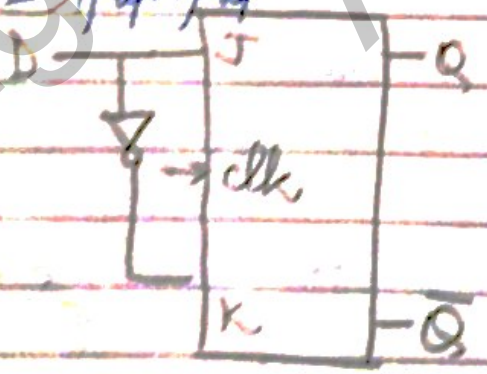
0	0	$Q_{(t)}$
1	1	$\bar{Q}_{(t)}$

called as T input

T	$Q_{(t+1)}$
0	$Q_{(t)}$
1	$\bar{Q}_{(t)}$

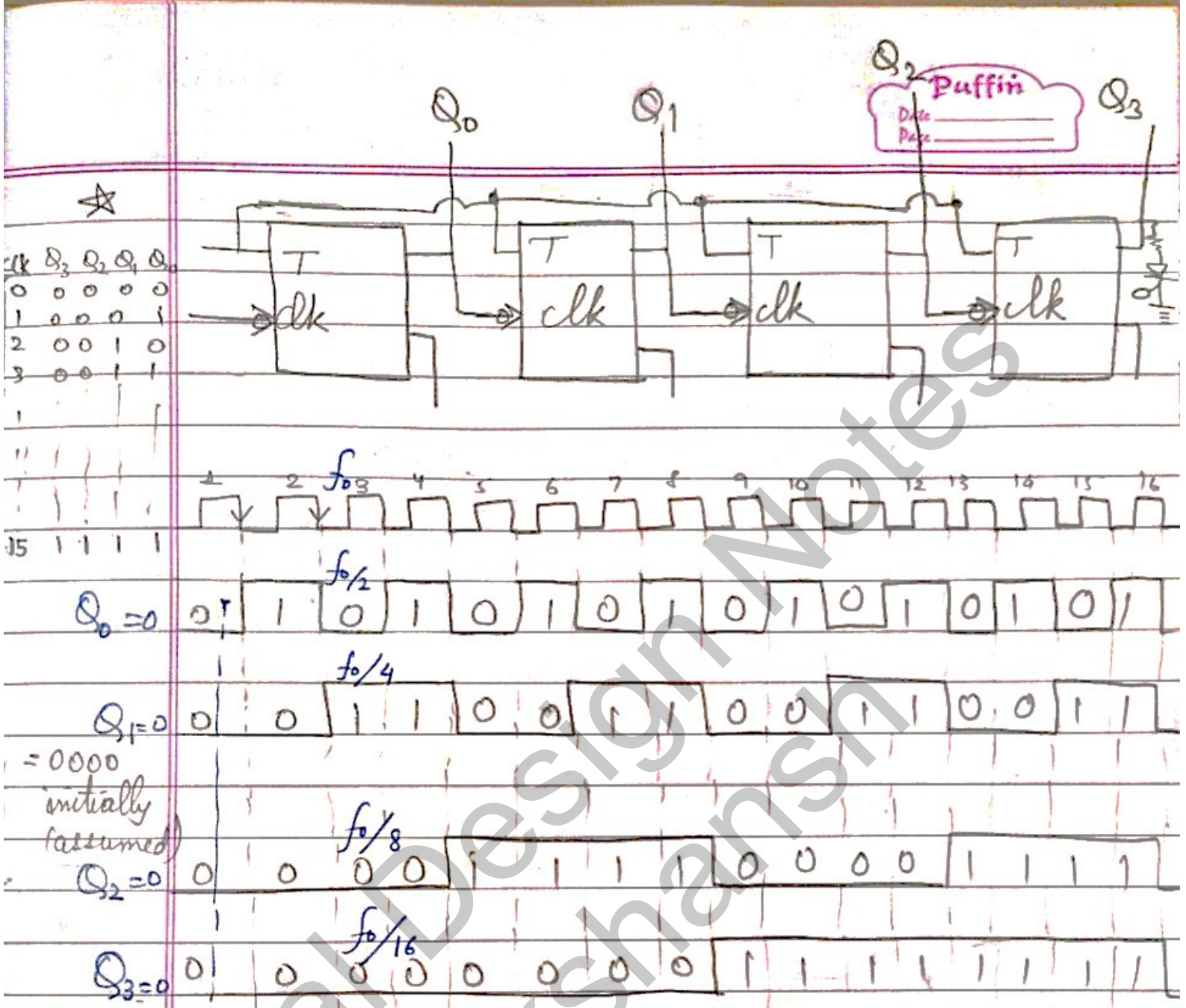
→ for delay can be used as storage device

- D flip flop



D	$Q_{(t+1)}$
0	0
1	1

* T flip flop with $T_{i/p} = 1$ is also called as frequency divider circuit.



clk	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
⋮				
15	1	1	1	1

= 0000 initially (assumed)

See column wise.

Q. Given 8 KHz clock pulse. Design a circuit to get 1 KHz as o/p. :- Instead of 4 bits, use 3 bits i.e 3 flip flops that divide →

$$8 \longrightarrow \frac{8}{2} \longrightarrow \frac{8}{2^2} \longrightarrow \frac{8}{2^3} = 1$$ ✓

NUMBER SYSTEMS

* Binary : Dec → Binary

Q $(23.575)_{10}$ = 610111 100100 1

$= (27.444)_8$ → group 3 bits together

$= (17.93)_{16}$ → group 4 bits together.

* Error

Q. $(AF.3B)_{16} = (?)_{10}$

$15 \times 16^0 + 10 \times 16^1$

$3 \times 16^{-1} + 11 \times 16^{-2}$

$= (175.2304)_{10}$

Q. $18\frac{25}{64} = 18.390625$

2 | 18
2 | 9 — 0
2 | 4 — 1
2 | 2 — 0
2 | 1 — 0

$(10010.01100)_2$

Q2. $10010 . \frac{16+8+1}{64}$

$\cdot \frac{16}{64} + \frac{8}{64} + \frac{1}{64}$

| 78 | 56 | 1
e2 | e2 | e2
156 | 112 |

$10010.\frac{1}{4}+\frac{1}{8}+\frac{1}{64}$

$.011001$

$\frac{1}{2}$ isn't there $\frac{1}{4}$ is there & so on till $\frac{1}{64}$

Q. $(517)_8 = (\ ?\ )_{16}$

→ BCD

→ (101 001 111) group into 4

$(14F)_{16}$ $(335)_{10}$

18
(00011000)

13
0111

0101 0001 0111

Q $(517)_{16} = (\ ?\ )_2 = (\ ?\ )_{10}$

$= 7\times16^0 + 1\times16 + 5\times16^2$

$= 7 + 16 + 1280$

$= 1303$

→ $(\ ?\ )_2$

$(0101\ 0001\ 0111)_2$

4 bit equivalent for each digit of hexadecimal

✡ SUBTRACTION USING COMPLEMENTS:

Complement = Max no. it can represent − our given no/bits.

eg: A = 0110

A' = 1111
−0110
1001

1's Complement 1001
+1
2's complement 1010

* In decimal sys, max. no. it can represent is 9.
So, ∃ 2 complements
9's complement
& 10's complement.
* For octal, 7's & 8th complement
* For hexadecimal, 15's & 16's complement.

eg :- $(365)_{16}$ → 15's C9A
16's C9B

eg 15's complement = 29F
16's. = 29F
+1
2A0

$(365)_8$ 7's → 777 − 365 = 412
8's → 412 + 1 = 413

* One's complement method of Subtraction.
Take 1's complement of subtrahand & add.
If ∃ a carry, add it too. This carry is called end-around carry.
If ∃ carry ⇒ ans. is +ve.
∄ ⇒ ans. is -ve.
↳ to get result, take 1's complement of resultant value.

• So, complement method is used to represent -ve nos.

eg: 10 - 13.

1010 +
-1101

= 10 + (-13) = 1010 → 1111 - 1101
+0010
1100 → no end around carry
1's complement of 13 has to be done.
1's
⇓
-ve no.
-0011
So, put a -ve sign & take 1's complement of result.

eg: 13 - 10. = 1101 +
-1010

1101
+0101 → 1111 - 1010
(1)0010 → end around carry ∃ ⇒ +ve
↓ add it to result.
+1
3 = 0011 → +ve

* Subtraction using 2's complement.
  ↳ Take 2's complement of subtrahand.
    → If ∃ carry, discard
      ⇓
      result: +ve
    → If ∄ no carry
      ⇓
      ans: -ve.
      ↳ take its 2's complement & put -ve sign

eg :- 13 − 10 = 1101 + 1010 → 1101 + 0101 → 1101 + 0110

1's

2's

discard ← (1) 0011

+ 3 Ans

10 − 13 = 1010 + 1101 → 1010 + 0010 → 1010 + 0011

1's

nothing as carry ⇓ −ve ← 1101

↳ Take its 2's complement.

1's ↳ 0010

2's ↳ 0011 with −ve sign

⇓

−0011 = −3

Ans

✱ BINARY MULTIPLICATION

✱ Convention :- SIGN BIT : 0 for +ve
1 for -ve

Decimal	Signed 2's compl.	Signed 1's compl.	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	—	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	—	—

1's compl. of only -ve no's.

→ It is -8 for unsigned no.

Puffin
Date
Page

this bit for ↑ 0

* Max. no. for unsigned represent^n :- n bits : $2^n - 1$

for signed bit

* For signed represent^n, n bits : MAX no. : $2^{n-1} - 1$
  Min no. : $-(2^{n-1} - 1)$
* 1's complement represent^n, n bits :- Max. : $2^{n-1} - 1$
  Min. : $-(2^{n-1} - 1)$
* 2's complement represent^n, n bits :- Max. : $2^{n-1} - 1$
  Min : -2^{n-1}

-1 not there as 7 only 1 zero for +ve See table.

---

* **BCD** : BCD correction :- Add 6 to whatever result you get ONLY IF resultant value is invalid or gives end around carry.

eg 8 + 4 = 12

1000 + 0100 = (1100)$_2$ = 12_{10} (binary)

convert binary to BCD

12 → 0001 0010 → BCD

Invalid BCD code.

eg: 8 + 9 = 17

1000 + 1001 = 1 0001

= $(11)_{10}$ (by grouping in 4)

∃ end around carry

☆ Signed Binary Arithmetic

sign bit

eg :- +6 0000 0110
+13 0000 1101
+19 0001 0011

1's → 1111 1001
2's → 1111 1010

→ make its 2's complement 1000 0110.

eg :- −6 1111 1010
+13 0000 1101
+7 (1) 0000 0111

→ carry ⇒ +ve no. = +7

eg :- +6 0000 0110
−13 1111 0011
−7 (1)111 1001

→ msb = 1 ⇒ −ve no.

So take its 2's complement
−ve sign put

1000 0111

eg :- −6 1111 1010
−13 (1)1111 0011
−19 1110 1101

carry ⇒ −ve no.

↓ Take 2's complement for Ans

1's → 1001 0010
2's → 1001 0011 → Ans

BCD ARITHMETIC

eg
$$\begin{array}{rl} 4 & 0100 \\ +3 & 0011 \\ \hline 7 & 0111 \end{array}$$

eg
$$\begin{array}{rl} 8 & 1000 \\ +5 & 0101 \\ \hline 13 & 1101 \end{array}$$

→ exceeds 9, so, not a valid code

$$+0110$$

$$\begin{array}{rl} & 10011 \\ & 3 \\ 1 & \end{array}$$

Rule: Add 6 to result

= 13 in BCD.

eg
$$\begin{array}{rl} 9 & 1001 \\ +7 & +0111 \\ \hline 16 & 10000 \end{array}$$

→ end around carry. So apply rule.

$$+0110$$

$$\begin{array}{l} 10110 \\ 16 \end{array}$$

16 = 16 Ans

eg
$$\begin{array}{r} 1184 \\ +1576 \\ \hline 2760 \end{array}$$

Do this first

$$\begin{array}{rrr} 0001 & 1000 & 0100 \\ +0101 & 0111 & 0110 \\ \hline 0111 & 10000 & 1010 \end{array}$$

1010 > 9, so rule

$$\begin{array}{rrr} & +0110 & +0110 \\ \text{(∵ Rule)} & & 10000 \\ \hline & 0110 & \end{array}$$

end around carry. So, rule.

$$\begin{array}{rrr} 0111 & 0110 & 0000 \\ 7 & 6 & 0 \end{array}$$

= 760

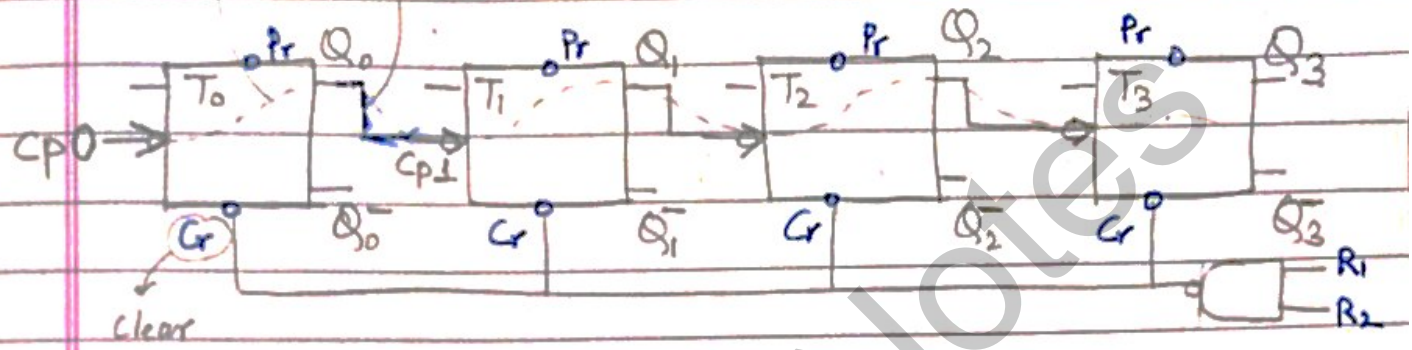
Ans

connection made externally → feedback connection

asynchronous

★ 4-bit binary counter : IC 7493

a sort of a RIPPLE. So, it is called as RIPPLE COUNTER



- Terminologies used : Clear $\xrightarrow{(Cr)} 0$; Preset $\xrightarrow{(Pr)} 1$
- Asynchronous circuits :- Clock i/p is not synchronised with o/p changes simultaneously.
- To get o/p as of flip-flop cleared put R_1 & R_2 as 1. That would give $C_r = 0$. This will clear. Also, keep $P_r = 1$
- Clear signals are asynchronous inputs.
- To get o/p of flip flops presetted (set to 1) Put R_1 & R_2 as 0
- We made R_1 & R_2 as grounded, i.e $R_1 = R_2 = 0$. So, $C_r = 1$. So, we want it to take values.

★ Note :-

If any i/p is not grounded i.e, we are applying Vcc.

★ If we want to start o/p from 5, say, we have to give Cr & Pr separately for each.

5 : 0 1 0 1

* They have to be made 0 & 1 manually

whichever out of them has 0, it comes into picture.

$C_r = 0$, $P_r = 1$; $C_r = 1$, $P_r = 0$; $C_r = 0$, $P_r = 1$; $C_r = 1$, $P_r = 0$

TUTORIAL

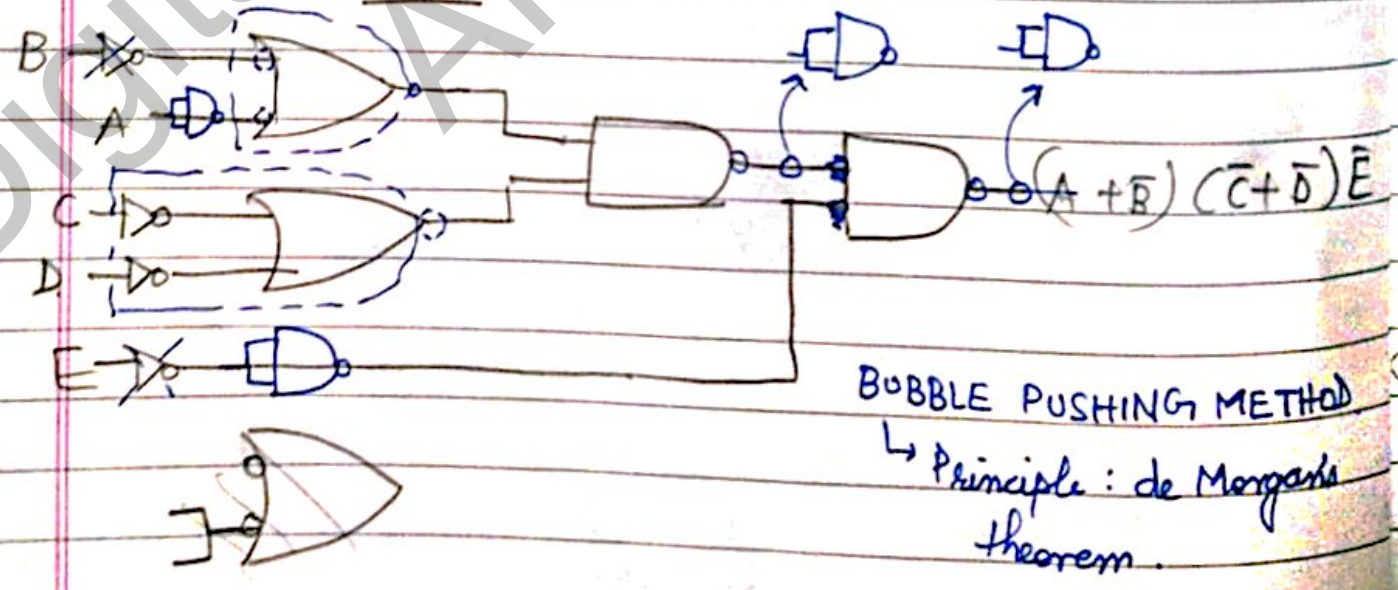
Q Simplify the expression to min. literal.

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

Q Find complement of
$$(A'B + CD)\cdot E' + E$$

$$\overline{(A'B + CD)\cdot E' + E}$$
$$= \overline{(A'B + CD)\cdot E'}\cdot \overline{E}$$
$$= \overline{A'B + CD + E'}\cdot \overline{E}$$
$$= \overline{[(A'B)\cdot \overline{CD}] + E'}\,\overline{E}$$
$$= A'B\,CD\,E$$

$$\overline{(A'B + CD)\cdot E' + E}$$
$$= \overline{(A'B + CD)\cdot E' + E}\;\overline{E}$$
$$= \overline{A'B + CD}\cdot E$$
$$= \cancel{AB}(A+\overline{B})(\overline{C}+\overline{D})\,\overline{E}$$



BUBBLE PUSHING METHOD
↳ Principle : de Morgan's theorem.

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

Obtain Truth table & express in sum of minterms & product of maxterms.

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

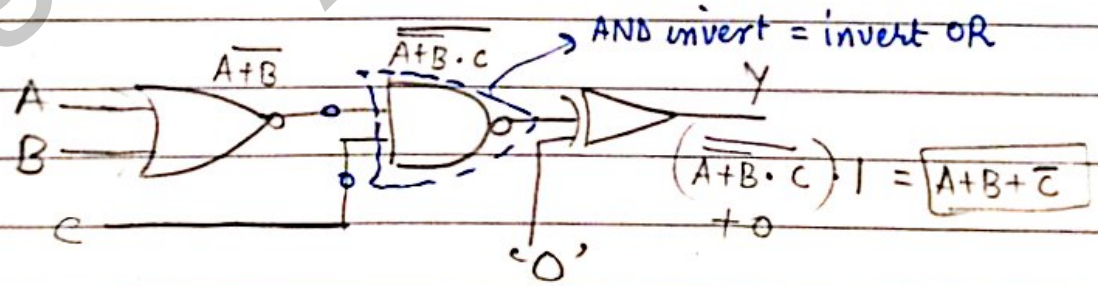
x	y	z	F = xy' + xz + y'z + y'	Min	Max
0	0	0	1	$x'y'z'$	~~x+y+z~~
0	0	1	1	$x'y'z$	
0	1	0	0		$x+y'+z$
0	1	1	0		$x+y'+z'$
1	0	0	1	$xy'z'$	
1	0	1	1	$xy'z$	
1	1	0	0		$x'+y'+z$
1	1	1	1	xyz	

SOP = $x'y'z + xy'z' + xy'z + xyz + x'y'z$

$= \sum(0,1,4,5,7)$

POS = $\pi(2,3,6) = (x+y'+z)\cdot(x+y'+z')\cdot(x'+y'+z)$

Q.



A'B + A'B

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

Q. Write o/p Y for given logic diagram & find what is y in 'y'.

$\overline{A+B\cdot C}$	'0'	Y
0	0	0
1	0	1

Q Given :- $F = \bar{A}\bar{B} + AB$

TPT :- F is complement of XOR

let $Y = A'B + AB'$ (XOR fn)

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

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

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

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

$$= AB + \bar{A}\bar{B} \quad \checkmark \quad = F$$

So, it is.

★ Note :- $\overline{AB} \neq \bar{A}\bar{B}$

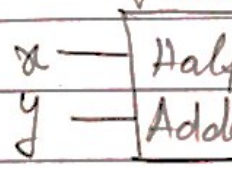
$= \bar{A} + \bar{B}$ (de Morgans)

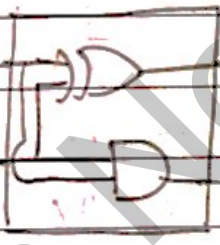
COMBINATIONAL CIRCUIT

1) ARITHMETIC CIRCUITS.

(a) ADDITION

Symbolic





x	y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Half adder circuit :-
  Combinational circuit in which addⁿ takes b/w 2 single bit i/p's.
- Full adder circuit :-
  ——— : 3 single bit binary bits.

O/p is coming as

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

$= A \oplus B$

$C = A \cdot B$

x	y	z	Sum	Carry
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

- Carry f^n (See 1's)

$C(A,B,C) =$

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

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

$= AB + BC + AC$

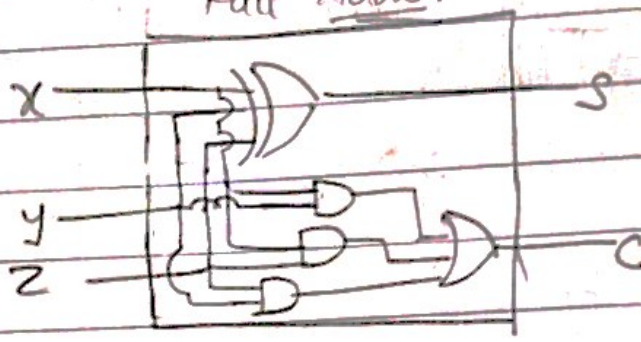
$= \sum(3,5,6,7)$ → MAJORITY FUNCTION

- Sum f^n $(S(A,B,C)) = \sum(1,2,4,7)$

$\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$ ~~→ MAJORITY FUNCTION~~

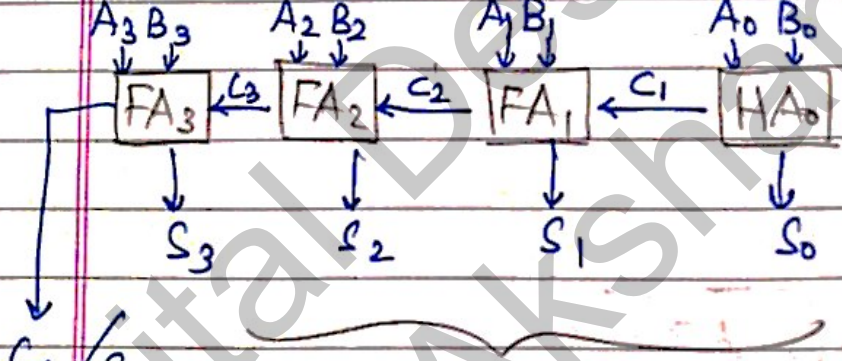
$= A \oplus B \oplus C$ → XOR FUNCTION

Full Adder

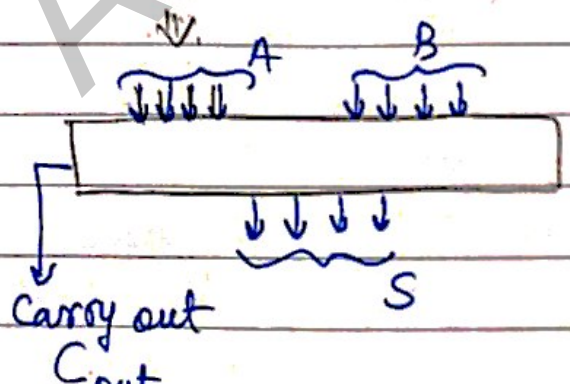


eg: Adding decimal digits 6 & 7 → make circuit

	FA_3	FA_2	FA_1	HA_0	
					$C_3\ C_2\ C_1$
6	0	1	1	0	$A_{BCD} \rightarrow A_3 A_2 A_1 A_0$
7	0	1	1	1	$B \rightarrow B_3 B_2 B_1 B_0$
	1	1	0	1	$C_4/S_4 \quad S_3\ S_2\ S_1\ S_0$

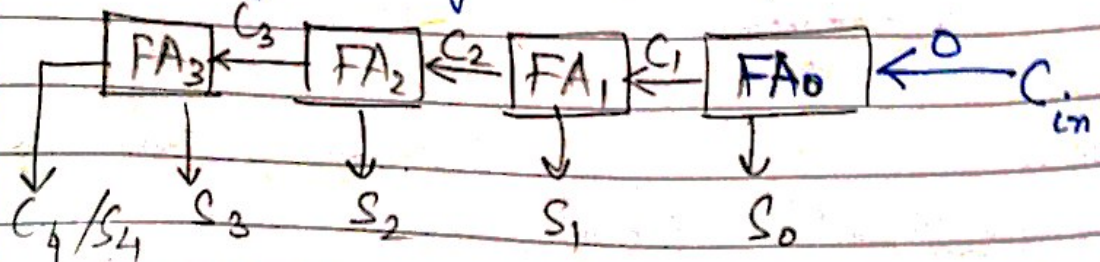


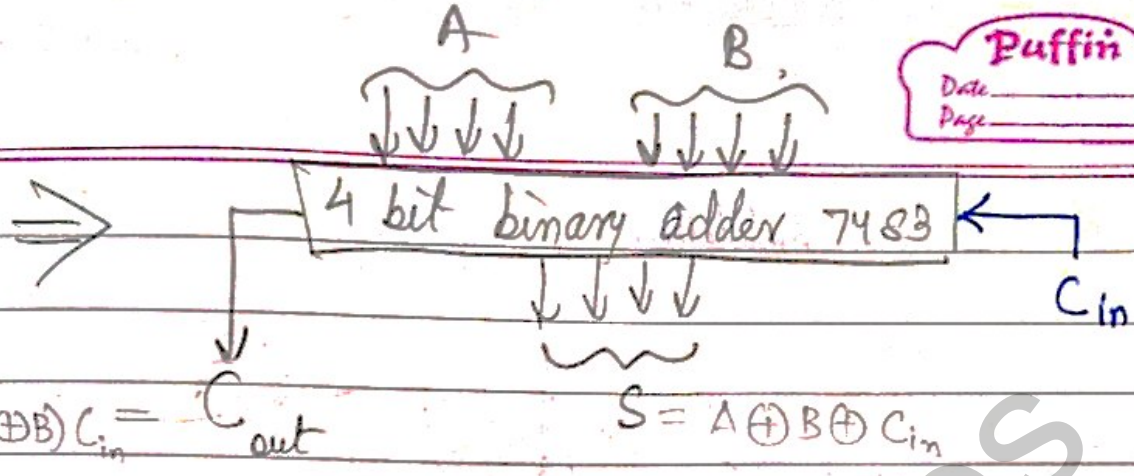
∴ 4 bit binary word adder circuit.

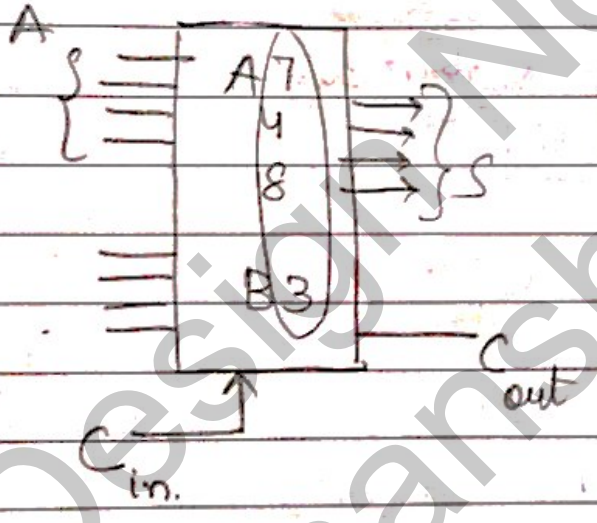


Universal.

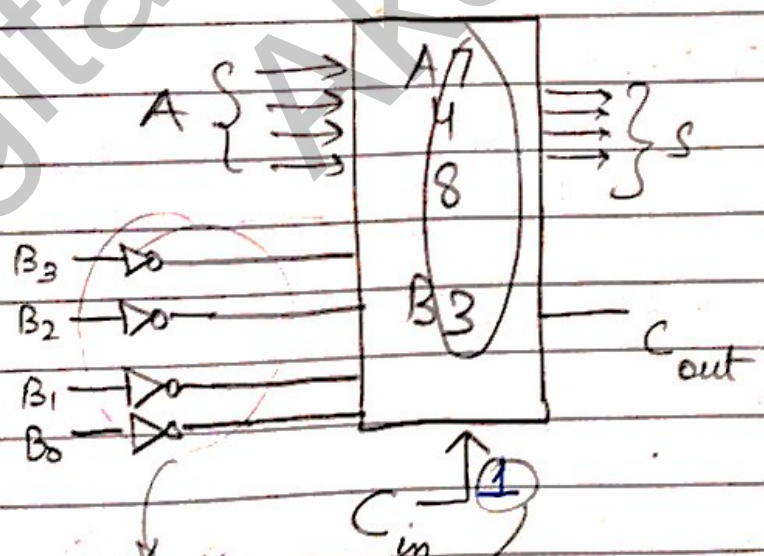
*) For adding 3 digit nos. (eg: 369 & 205)



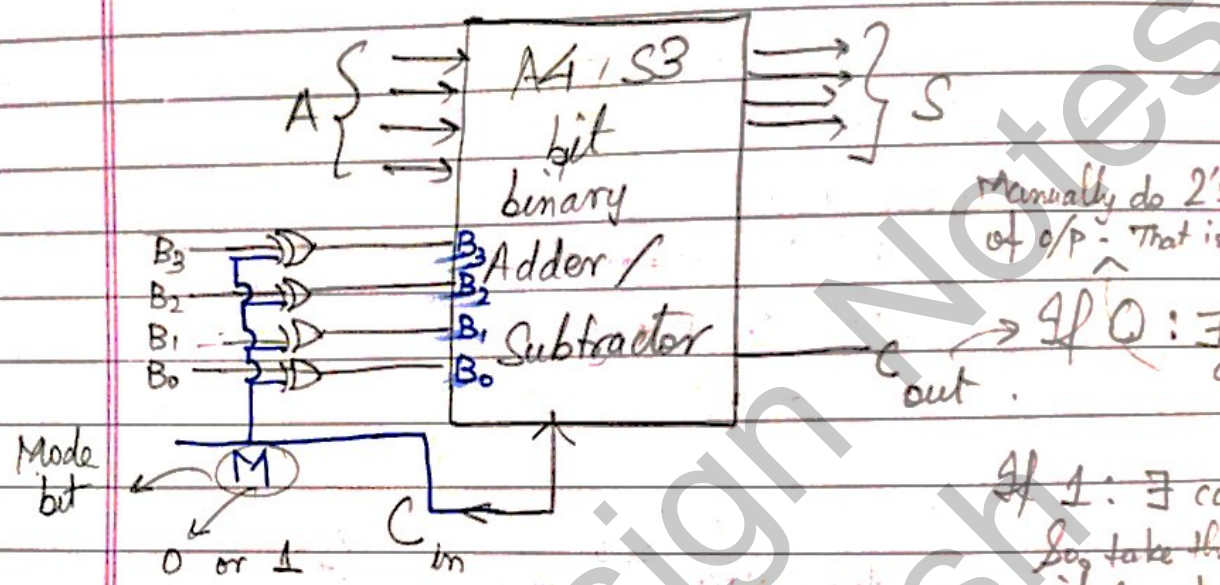




(b) SUBTRACTION



• Either Addition or Subtraction



Manually do 2's complement of o/p - That is ans (-ve sign)

If 0 : ∄ no carry

If 1 : ∃ carry. So, take the o/p as it is, discard carry.

• Mode control: use it to give the logic of sum (M) or difference to IC.

They are equivalent (only 1 bit change)

K-MAPS

↳ it gives optimum, minimised solution.

- 3 Variable K-maps

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

MSB ← A ; LSB ← C

: Only 1 bit diff. b/w adjacent rows and/or adjacent columns. (like in Gray code)

They are equivalent (only 1 bit change for writing A, B & C in both)

We will take as standard.

We can also be used.

ex :- $F(A, B, C) = \sum(3, 5, 6, 7)$

Mapping these values of minterms into cell,

A \ BC	00	01	11	10
0	0 (m_0)	0 (m_1)	1 (m_3)	0 (m_2)
1	0 (m_4)	1 (m_5)	1 (m_7)	1 (m_6)

A \ BC	00	11	01	10
0	m_0	m_3	m_2	m_1
1	m_4	m_7	m_6	m_5

Now, finding simplified expression.

↳ Start grouping 1's. The No. of elements in each group should be in terms of 2^n. eg :- we can make group of 1, 2, 4, 8.

- reduce no. of groups & no. of elements in each group.

✱ Way to find

m_5 m_7

$A\boxed{B}C + A\boxed{B}C$

$= 1$

$= AC$

for both m_5 & m_7,
A is there (A = 1) in both terms.
B changes.
C is there (C = 1) in both terms

So, net :-

- Value remains same for both terms.
  Value = 0 : $\bar{A}$
  Value = 1 : A.
- Value changes. remove that

Ily,
$A\bar{B} + BC$

ex $F(x, y, z) = \sum(0, 2, 3, 4, 6)$

x \ yz	00	01	11	10
0	1	0	1	1
1	1	0	0	1

Fold these. We see that still, ∃ 1 literal change
So, its fine

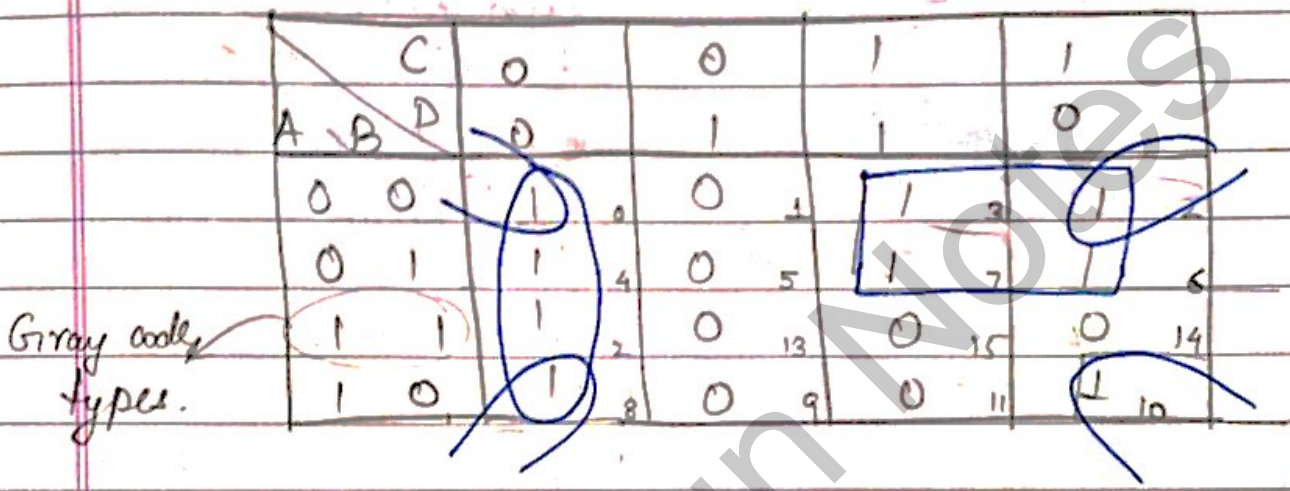
$= \bar{z} + \bar{x}y$

✱ For each group, try including max. no. of literals

* Idea :- Try reducing the no. of groups
& Try increasing no. of elements in each group.

Puffin Date Page

- 4 variable k-maps



ex :- $F(A,B,C,D) = \sum(0,2,3,4,6,8,10,12)$

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

- 5 variable k-maps (A, B, C, D, E)

Method :- Take two 4-variable k-maps

1st → A = 0

2nd → A = 1

A = 0 BCDE → 0 – 15 → $\bar{A}$. (expression for BCDE for 1st k-map)

A = 1 16 – 31 → A . (expression for 2nd k-map)

Simplify further if possible.

Q. Using 5 var. k-map, simplify →

$$F(A,B,C,D,E) = \Sigma(0,1,4,5,16,17,21,25,29)$$

A = 0

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

$\bar{A}(\bar{B}\bar{D})$

A = 1

BC \ DE	00	01	11	10
00	1 (16)	1 (17)	0 (19)	0 (18)
01	0 (20)	1 (21)	0 (23)	0 (22)
11	0 (28)	1 (29)	0 (31)	0 (30)
10	0 (24)	1 (25)	0 (27)	0 (26)

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

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

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

$(A+\bar{A})$ ← $(\bar{A}+\bar{C})$

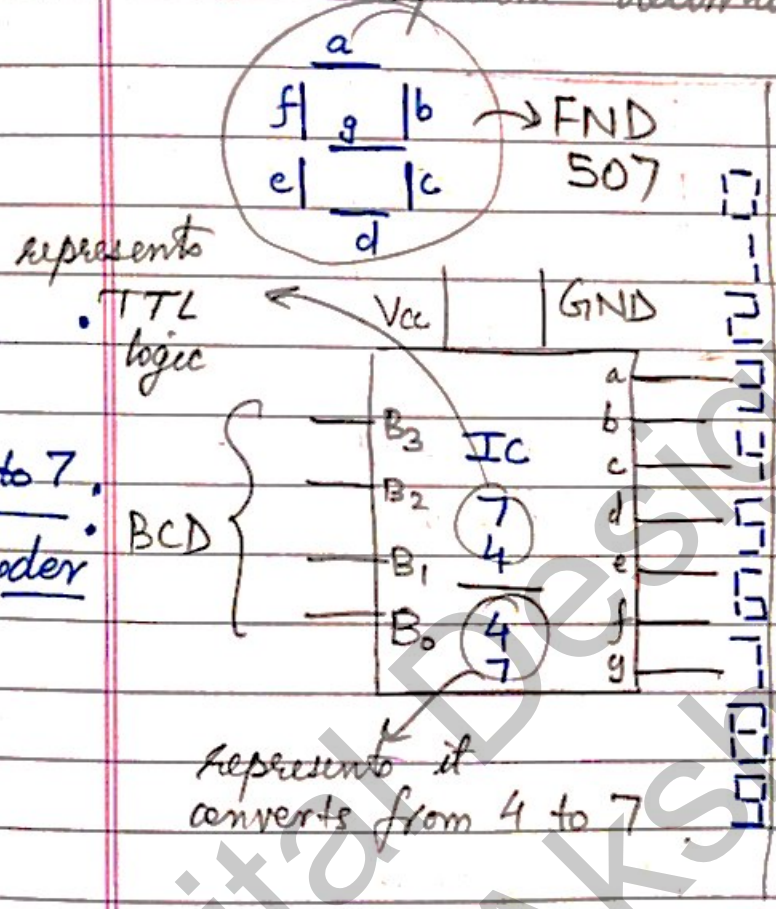
$$= \bar{B}\bar{D}(\bar{A}+\bar{C}) + A\bar{D}E$$

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

Puffin
Date 3/10/12
Page

☆ CODE CONVERTER CIRCUITS.

- 7 segment digital device
  - ↳ represent decimal nos. (0-9)



B_3	B_2	B_1	B_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

☆ Method: Put 7 K-maps for each fn.
a to g ⟶ all are fns.

☆ DON'T CARE TERMS. (d or X or ϕ) symbols.
Terms from 10 - 16, which don't take any value in BCD.
☆ They can be taken either as 1 or as 0 as per convenience (can be used in K-maps for grouping).

* for fn a :-

$B_3 B_2$ \ $B_1 B_0$	0 0	0 1	1 1	1 0
0 0	1	0	1	1
0 1	0	1	1	1
1 1	x	x	x	x
1 0	1	1	x	x

Taking Don't care terms as 1

* Try maximising the no. of elements in one group.

B_1

B_3

$B_2 B_0$

$\bar{B_2}\bar{B_0}$

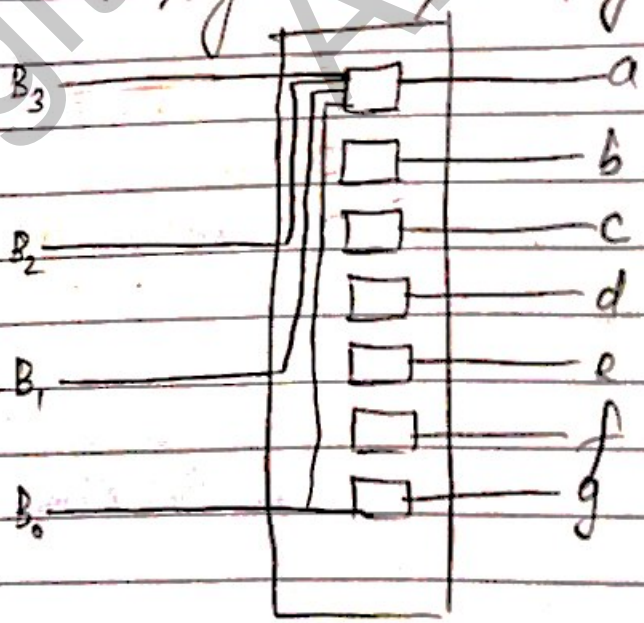
fn: $a(B_3 B_2 B_1 B_0) = \bar{B_2}\bar{B_0} + B_1 + B_2 B_0 + B_3$

⇓ Implies.

Either one of B_3 or B_1 or $B_2 B_0$ or $\bar{B_2}\bar{B_0}$ is high, a will lit.

* IIly, do for all other fns

* All 7 segments put together :



B_3, B_2, B_1 & B_0 are connected to each box

* Circuits which have no. of o/p > no. of i/p : DECODERS

So, decodes are :- Take n i/ps → give out → more than n or max of 2^n o/ps.

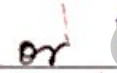
* ENCODER : no. of i/ps > no. of o/ps.

2^n max i/ps taken ⇒ give n o/ps

* Note :- It is possible to have multiple sol^{ns} of a problem.

eg :- 9 is written as

 → 1 1 1 0 0 1 1

or  → 1 1 1 1 0 1 1

So, correspondingly, we will get different results.

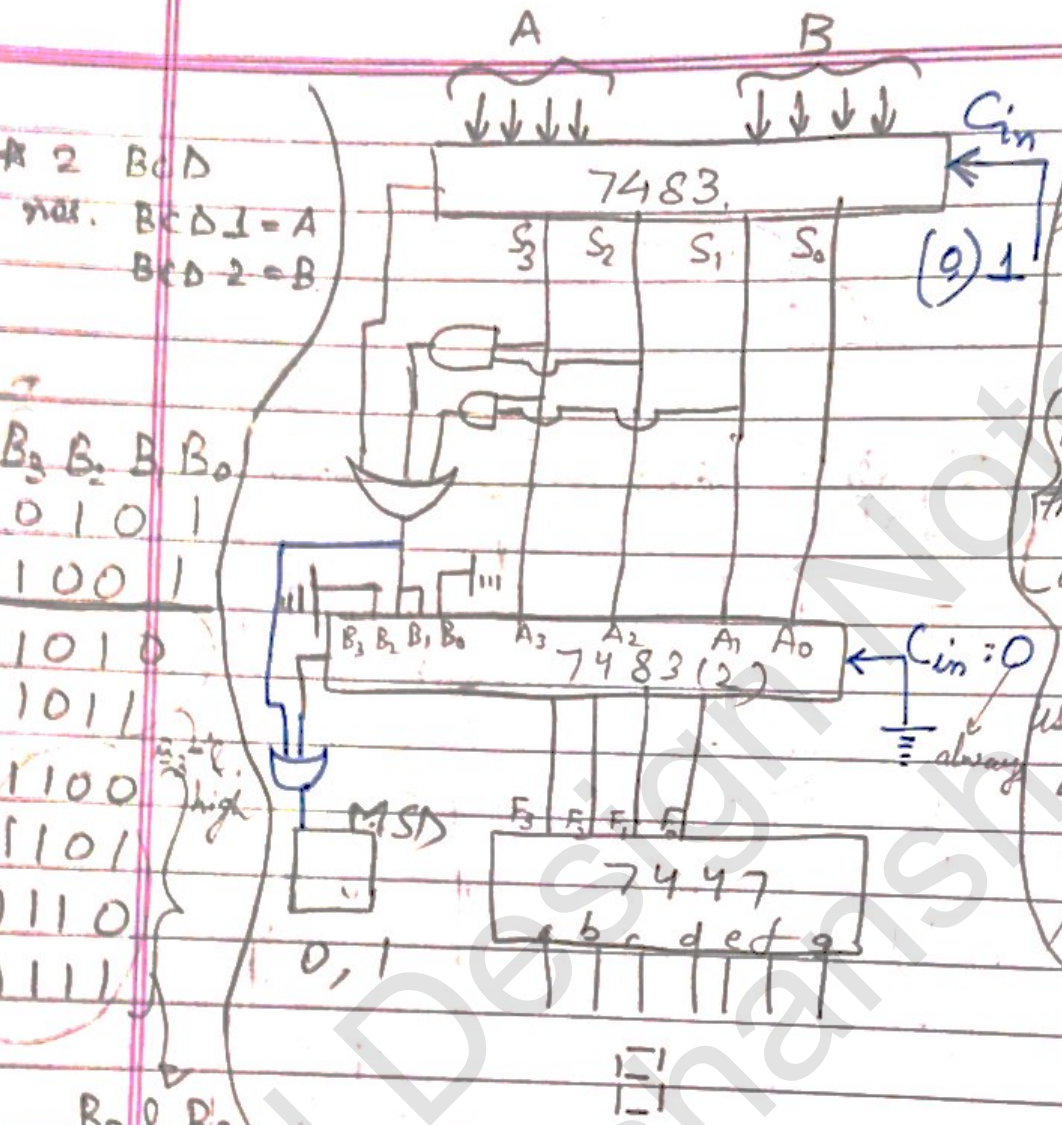
eg: We do some different grouping in K-maps.

* If we have binary i/p, we have to convert it to equivalent BCD.

eg : Req^d for 10 - 15.

Note :- ∃ carry out of conversion, for 10-15.

✱ EXPERIMENT : BCD ADDER



✱ 2 BCD var. BCD 1 = A
BCD 2 = B

B_3	B_2	B_1	B_0
0	1	0	1
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

4 bit binary adder / subtracter circuit.

↓

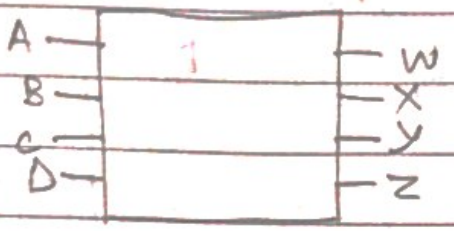
Convert binary op that we get from above to BCD

↓

use BCD to light LED's → digital values ? Use previous pg. circuit

✱ CONVERT :- BCD to Excess-3 CODE

Input BCD				Output Excess-3 Code			
A	B	C	D	W	X	Y	Z
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				



* We need to find expression. So, make k-map for each f^n.

for Z :-

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

for y :- $CD + C'D'$ or $(C \oplus D)'$

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

* Comparing both truth tables, we can directly say that $Z = \overline{D}$

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

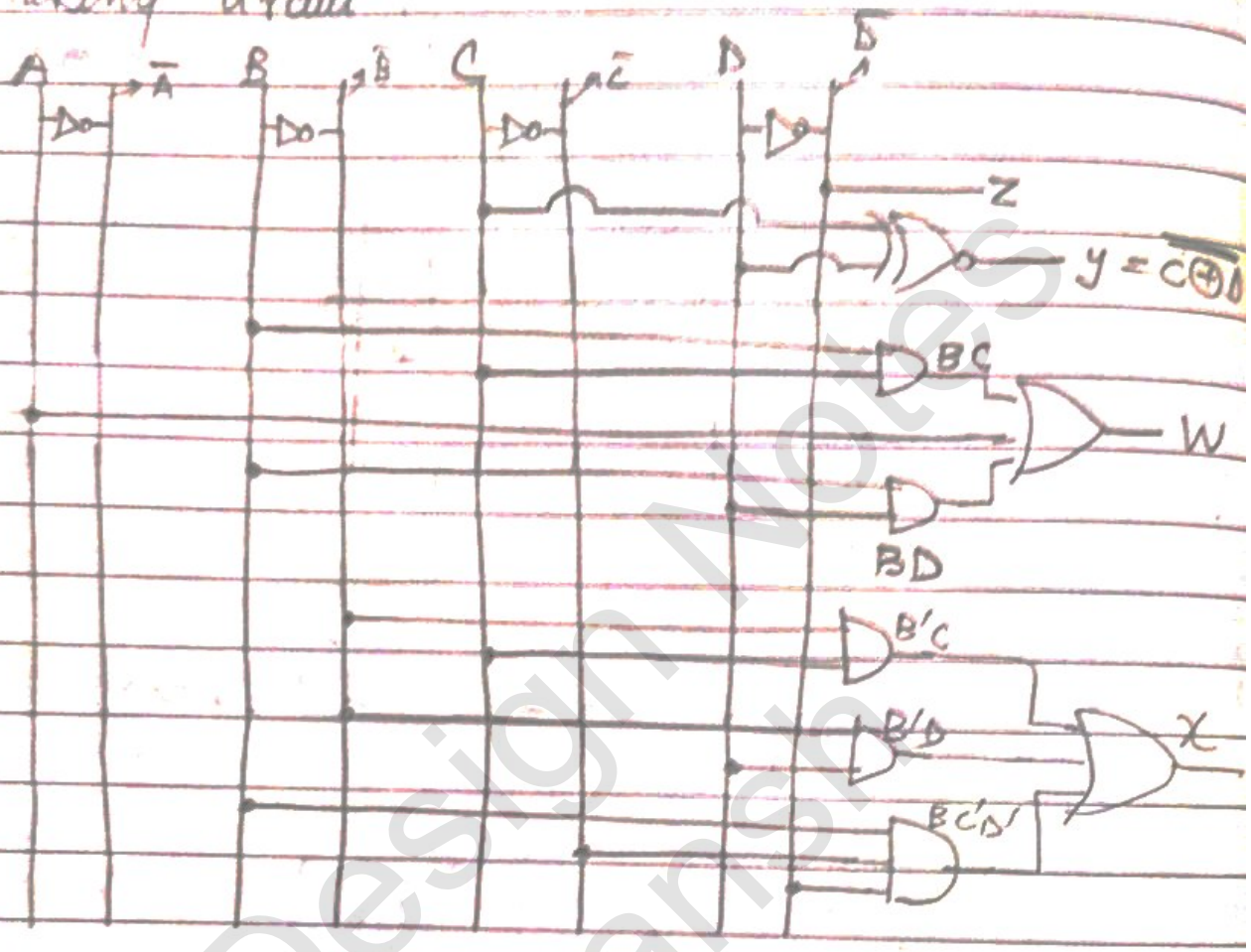
CD / AB	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

$W = A + BC + BD$

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

Puffin
Date
Page

Making circuit



Another way :- Use adder/subtractor circuit.

Like :-

* $A \oplus B \oplus C = \sum(1,2,4,7)$ → Sum fn" → full adder

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

→ no grouping possible

* $\overline{A \oplus B \oplus C} = \sum(0,3,5,6)$

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

Puffin
Date
Page

★ $F = A \oplus B \oplus C \oplus D$: Odd parity

$= \sum(1, 2, 4, 7, 8, 11, 13, 14)$

AB \ CD	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$\overline{F} = (0, 3, 5, 6, 9, 10, 12, 15)$

★ $F = A \oplus B \oplus C \oplus D$: even parity

AB \ CD	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

★ Note :-

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

$= A' \oplus B' \oplus C'$.

So, given a f^n, like $A' \oplus B \oplus C$

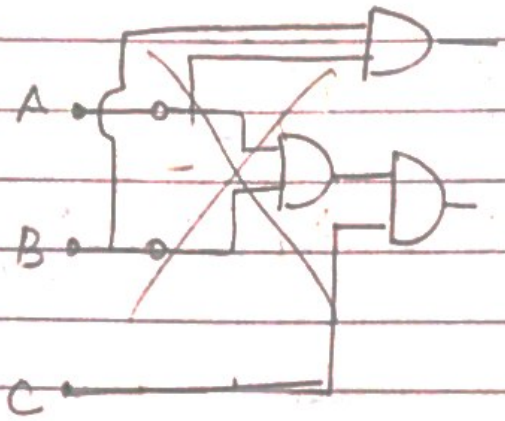
⇓

X-NOR gate

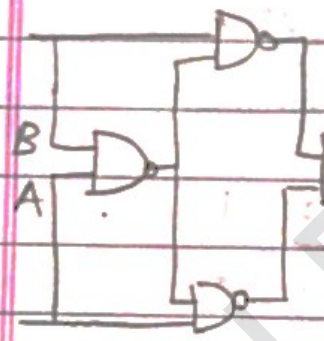
Q Implement 2 i/p XOR gate using NAND gates

$A \oplus B \oplus C = \sum(1, 2, 4, 7)$

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



Optimum solution



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

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

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

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

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

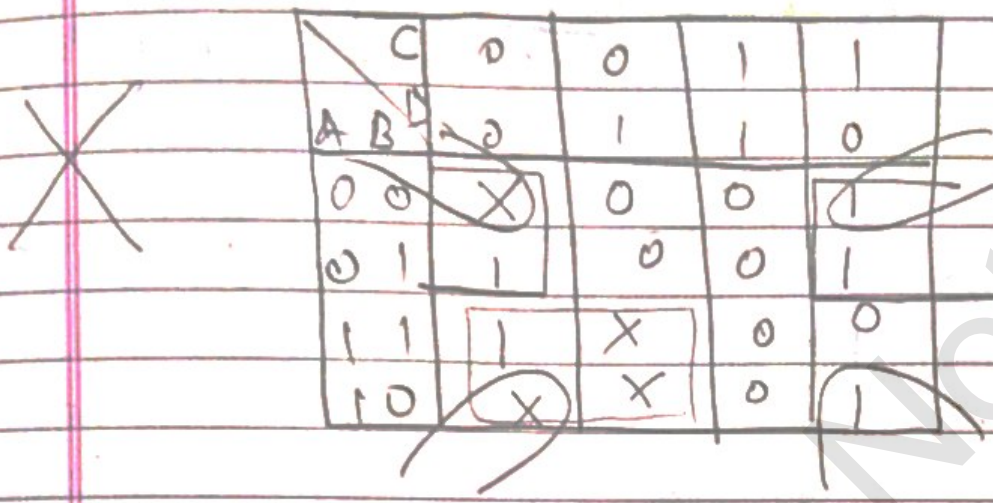
Q. Implement the fn F using not more than 2 NOR gates → i.e in POS form.

$F(A,B,C,D) = \sum(2, 4, 6, 10, 12)$

$(d)(A,B,C,D) = \sum(0, 8, 9, 13)$

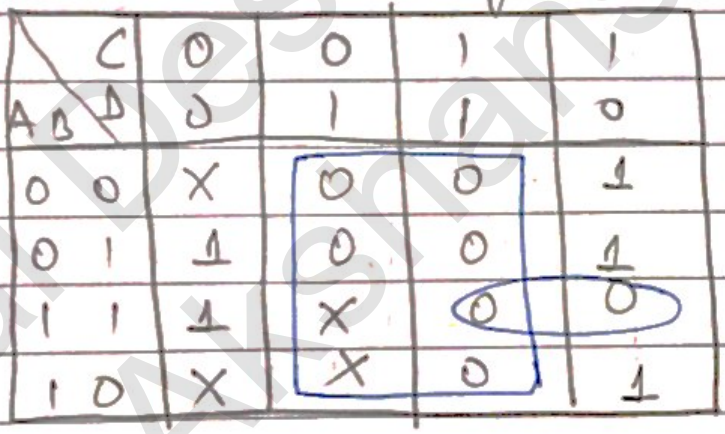
→ don't care terms.

* To implement using:
- NAND gates : We need SOP form
- NOR gates : We need POS form.



Here, I grouped 1's
⇓
SOP expression

I need POS form to group 0's.



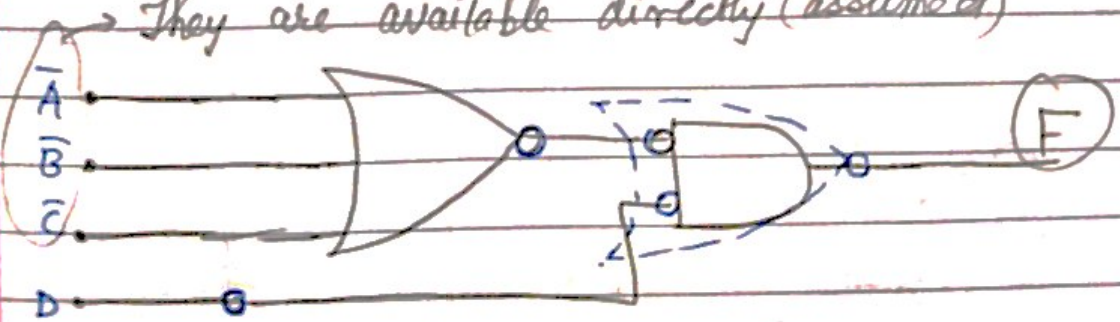
∵ grouped 0's

$\overline{F} = D + ABC$

$\Rightarrow F = \overline{D + ABC}$

$\Rightarrow F = \overline{D}.(\overline{A} + \overline{B} + \overline{C})$

→ They are available directly (assumed)



Q. Implement by two level (a) NAND - AND (c) OR - NAND
(b) AND - NOR (d) NOR - OR

$F(A,B,C,D) = \Sigma(0,4,8,9,10,11,12,14)$

(Group 1's) SOP form =

AB \ CD	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	0	0	1
10	1	1	1	1

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

Group (0's) POS form

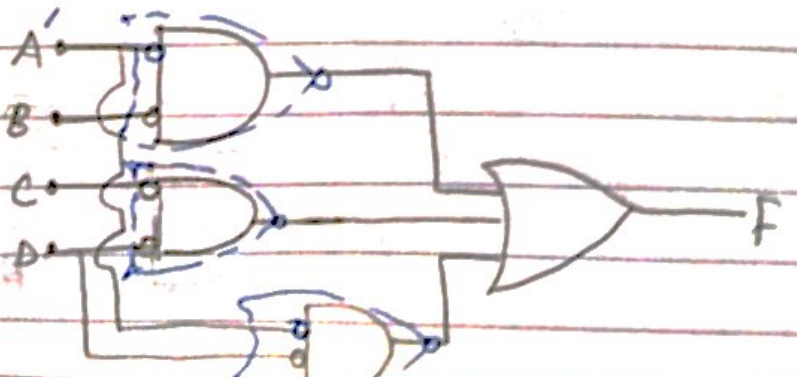
AB \ CD	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	0	0	1
10	1	1	1	1

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

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

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

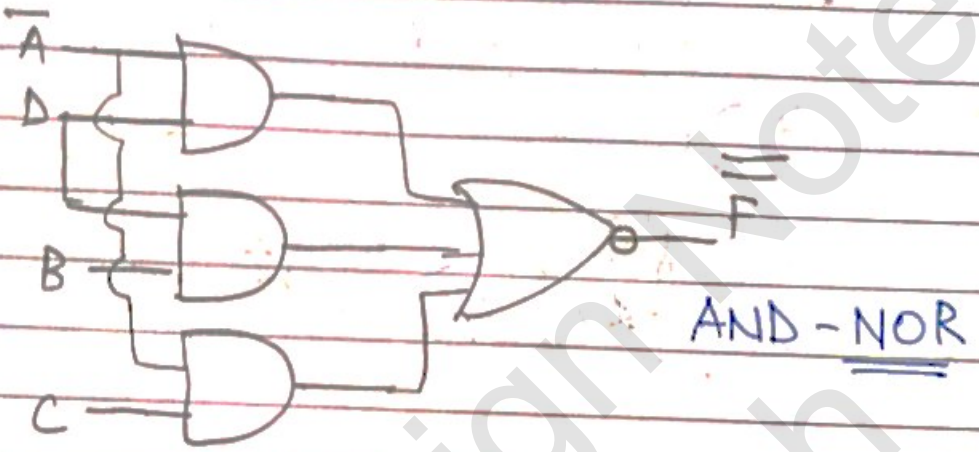
(d) NOR-OR, Using $F = \bar{C}\bar{D} + A\bar{B} + A\bar{D}$



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

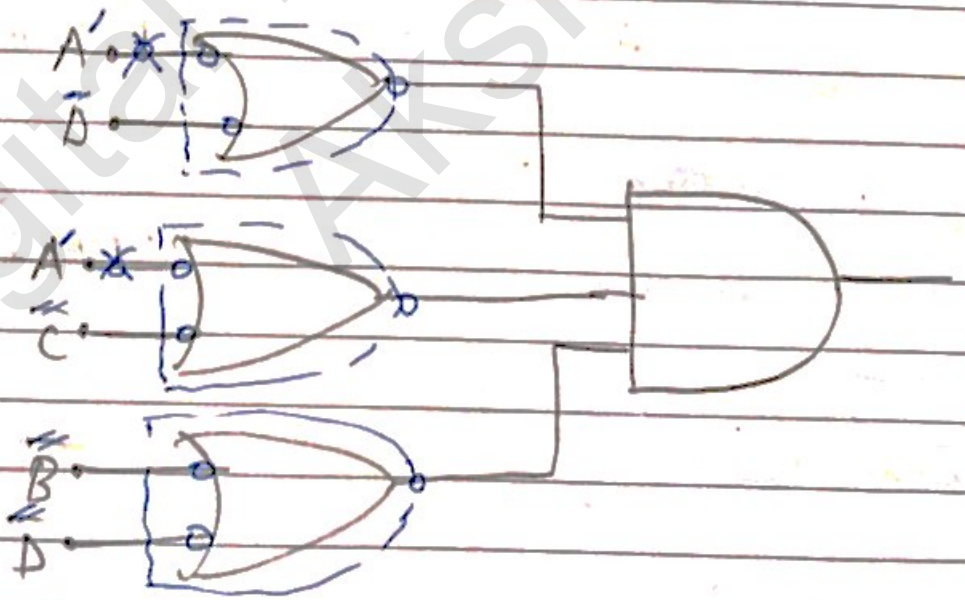
↳ Implementing this

(b)



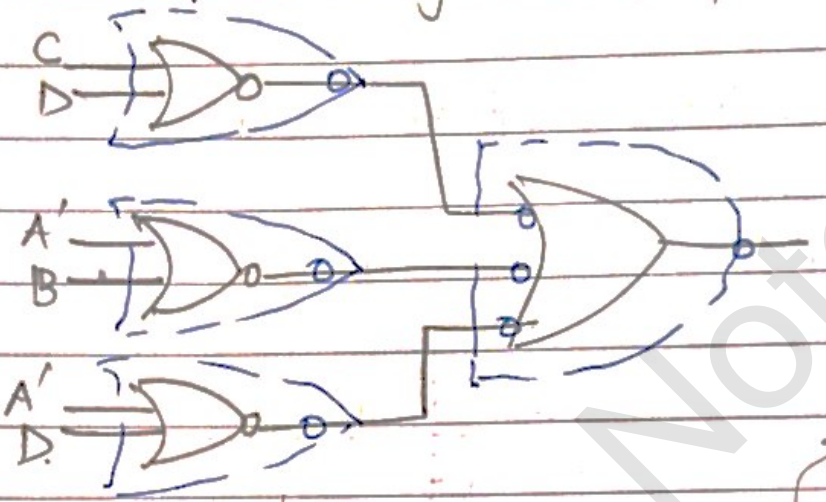
AND - NOR

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



* Note: part (a) can also be done by using the circuit of part (b), like its done in part (c) (next page)

(c) OR-NAND (Using the (d) part)



Sum & Carry
XOR AND

2 i/p 2 o/p.

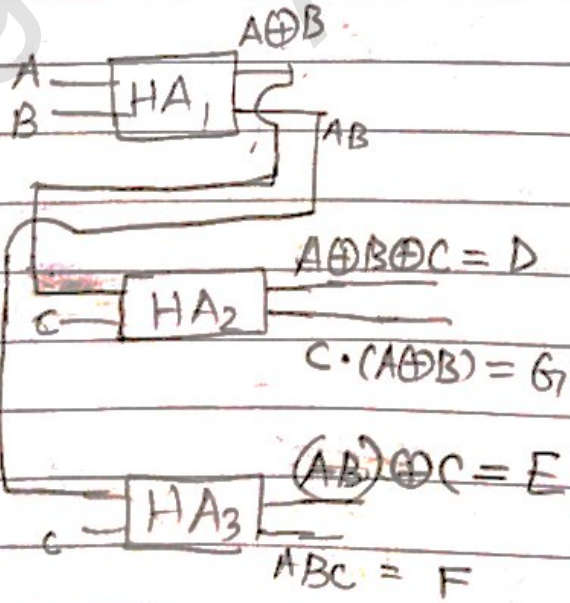
Q. Given 3 half adders. Implement these 4 fns in a SINGLE circuit

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

(2) $E = ABC' + (A'+B')C = (AB) \oplus C$; $ABC' + (\overline{AB})C$; $(A+B)C$

(3) $F = ABC$

(4) $G_1 = A'BC + AB'C = C(A'B + AB') = C(A \oplus B)$



Q $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

Find reduced SOP expression.

wx \ yz	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	0	1	1	0
10	1	0	0	1

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

$\bar{w}x$ or $\bar{w}\bar{z}$

$xz + \bar{z}\bar{x}$ → essential prime implicants (terms which will be there definitely)

2 solns :- $F = \bar{w}x + xz + \bar{z}\bar{x}$
$= \bar{w}\bar{z} + xz + \bar{z}\bar{x}$

Same no. of literals. Both are optimum soln's.

So, ∃ many soln's (no problem : It's possible.)

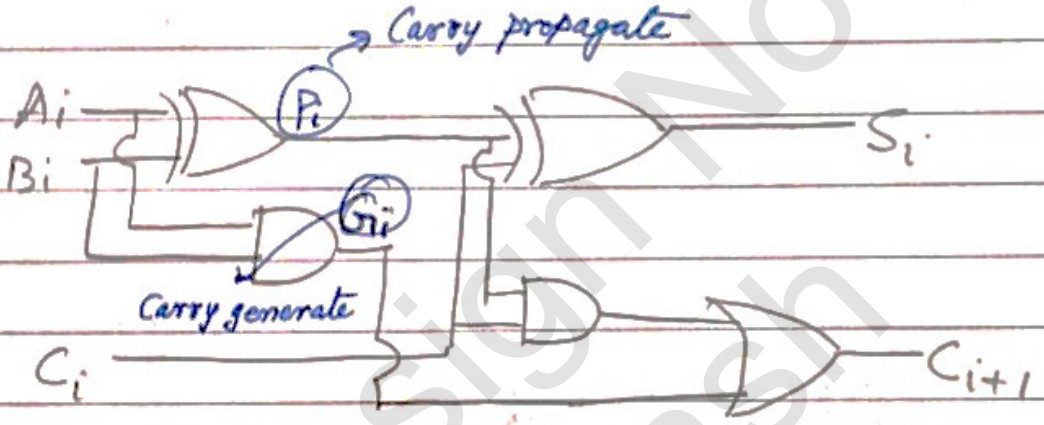
★ Prime Implicants : Each of product terms arising from K maps.

★ sth like local maxima/minima for a f^n $f(x)$. There can be many. Absolute maxima doesn't exist in this optimum soln of KMAP.

PARALLEL

✱ 4 BIT BINARY ADDER: Increasing circuit speed :-

- See the circuit made previously.
- Delay is being caused by each adder circuit to produce carry
- If we produce carry separately, then, that delay time is saved.



$P_i = A_i \oplus B_i$; $G_i = (A_i)\cdot(B_i)$

$S_i = P_i \oplus C_i$; $C_{i+1} = G_i + P_i \cdot C_i$.

These are the 4 carry bit generated separately

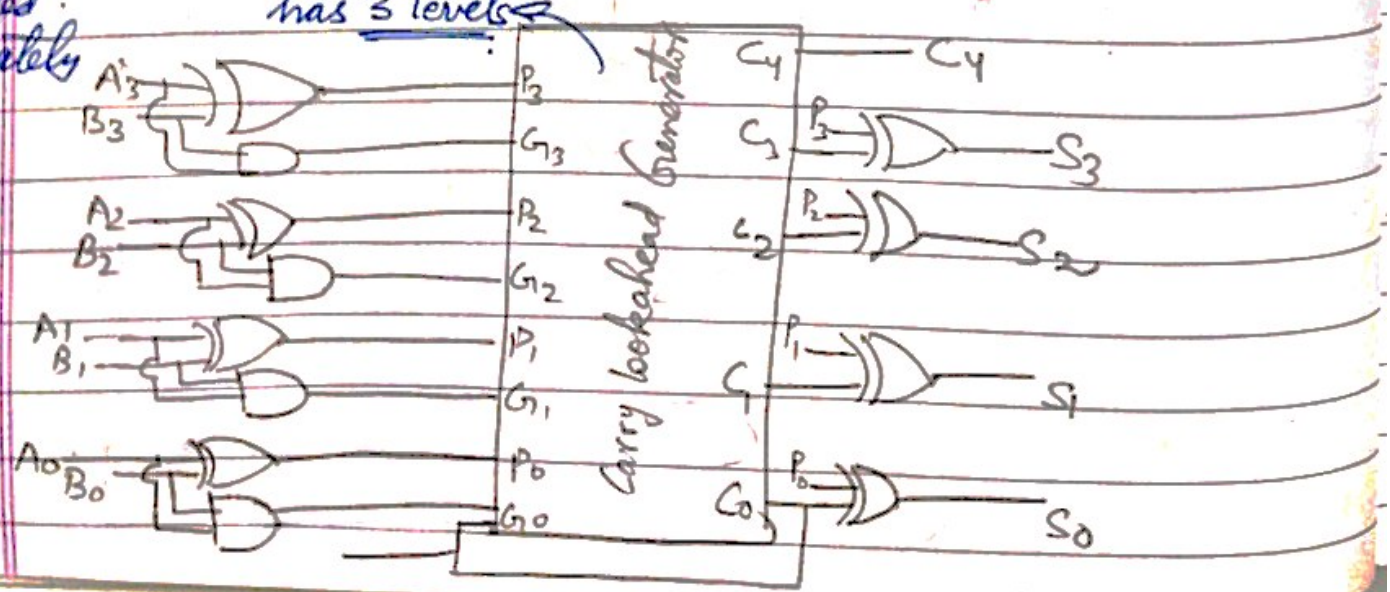
C_0 = input carry :- Put i = 0

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

has 3 levels



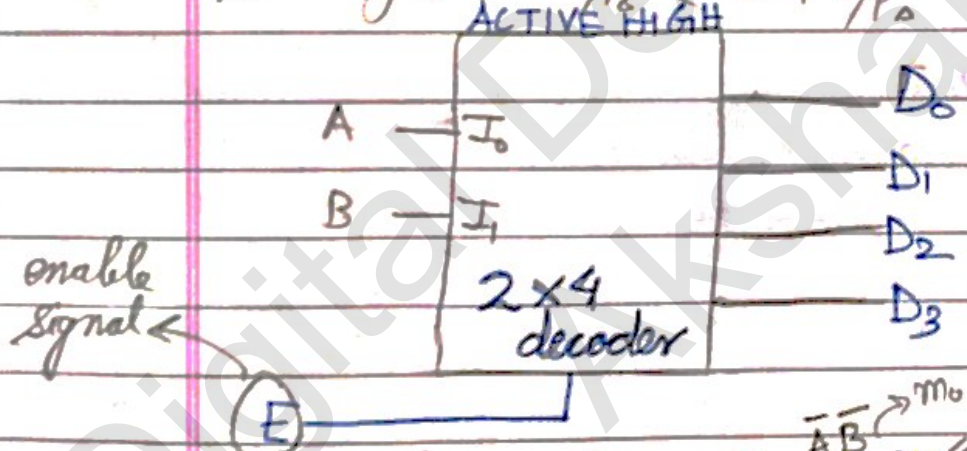
✓ circuit has 5 levels. The original circuit we used had 9 levels. (3 for each of the first 3 full adders).

So, speed ↑.

Decoders

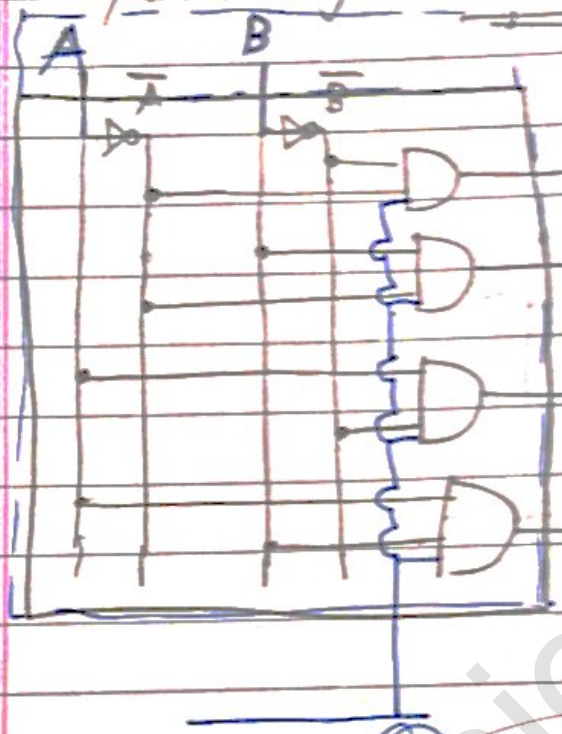
* Takes n i/p's & gives out 2^n o/p's.

* for 2 i/p's → 4 o/p's



A	B	D_0 $\bar{A}\bar{B}$ → m_0	D_1 $=\bar{A}B$	D_2 $=A\bar{B}$	$D_3 = AB$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

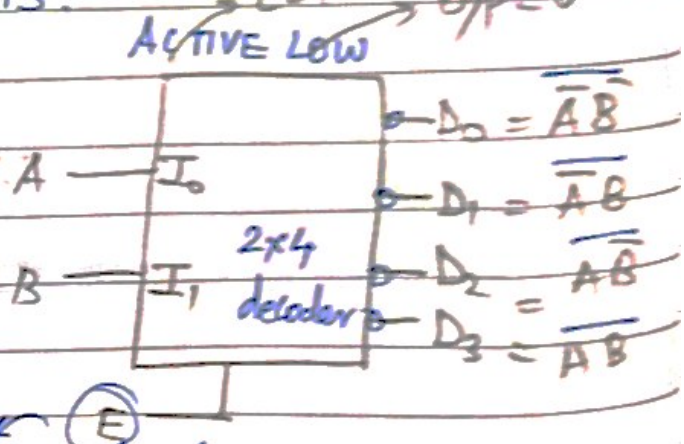
Implementing 2×4 decoder



gives MINTERMS.

→ when it high, it enabling circuit.

* Note :- If the "implement" is done using NAND gate (instead of AND), we get MAX TERMS.



Corresponding circuit

We will get low o/p when it is activated

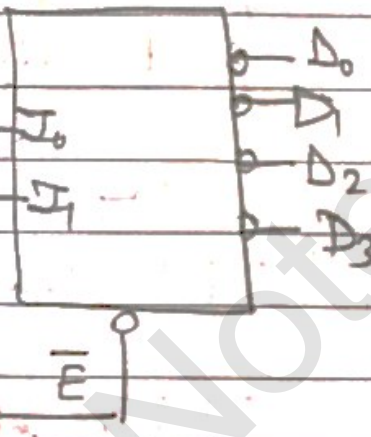
It is activating

A	B	D_0	D_1	D_2	D_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

* Enable pin :- If it's low ($E = 0$), we use $\bar{E}$ as i/p.

Invalid o/p → any value.

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



* Ily for 3 x 8 decoder.

A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

* make circuit accordingly.

✱ AIM: Add 2 7-bit nos.

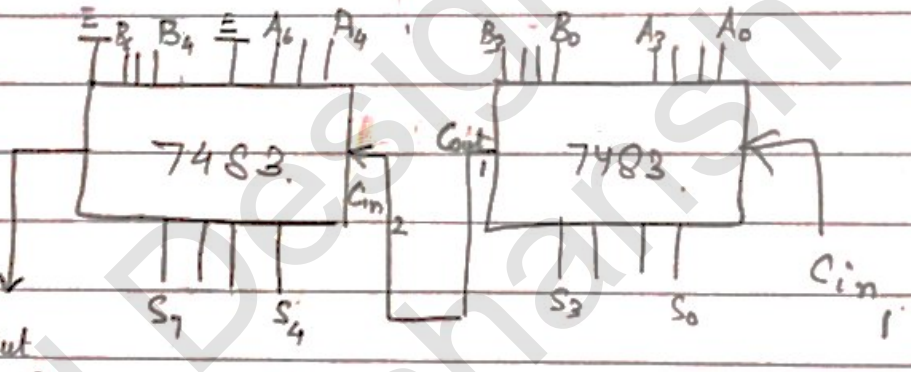
Do using std. circuits studied

A $A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

B $B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$

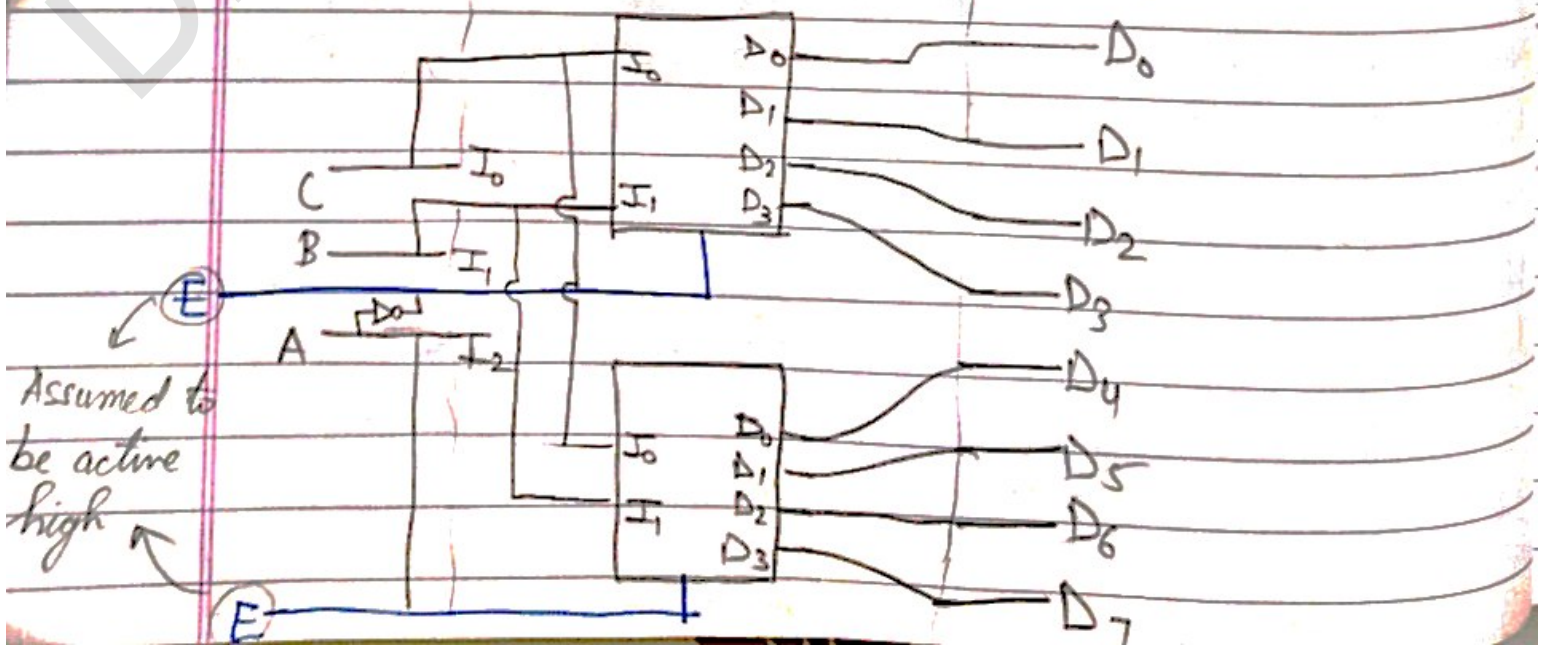
eg: 1101110 +
1011110

A	$A_6\ A_5\ A_4$	$A_3\ A_2\ A_1\ A_0$
B	$B_6\ B_5\ B_4$	$B_3\ B_2\ B_1\ B_0$



3 bits → MSB A B C LSB

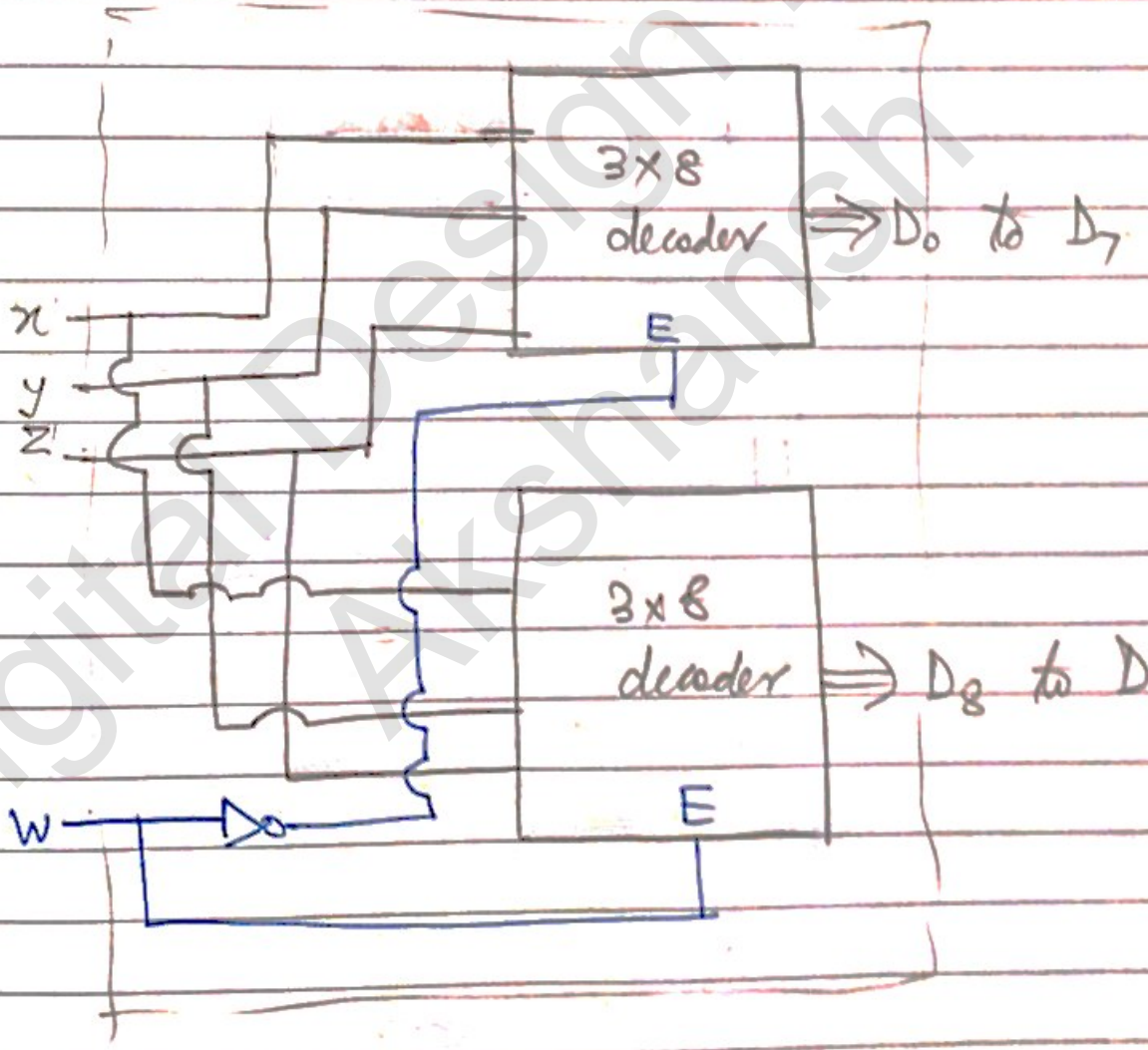
✱ AIM:- Make 3×8 decoder using 2×4 decoder



⁂ A is taken as enable pin

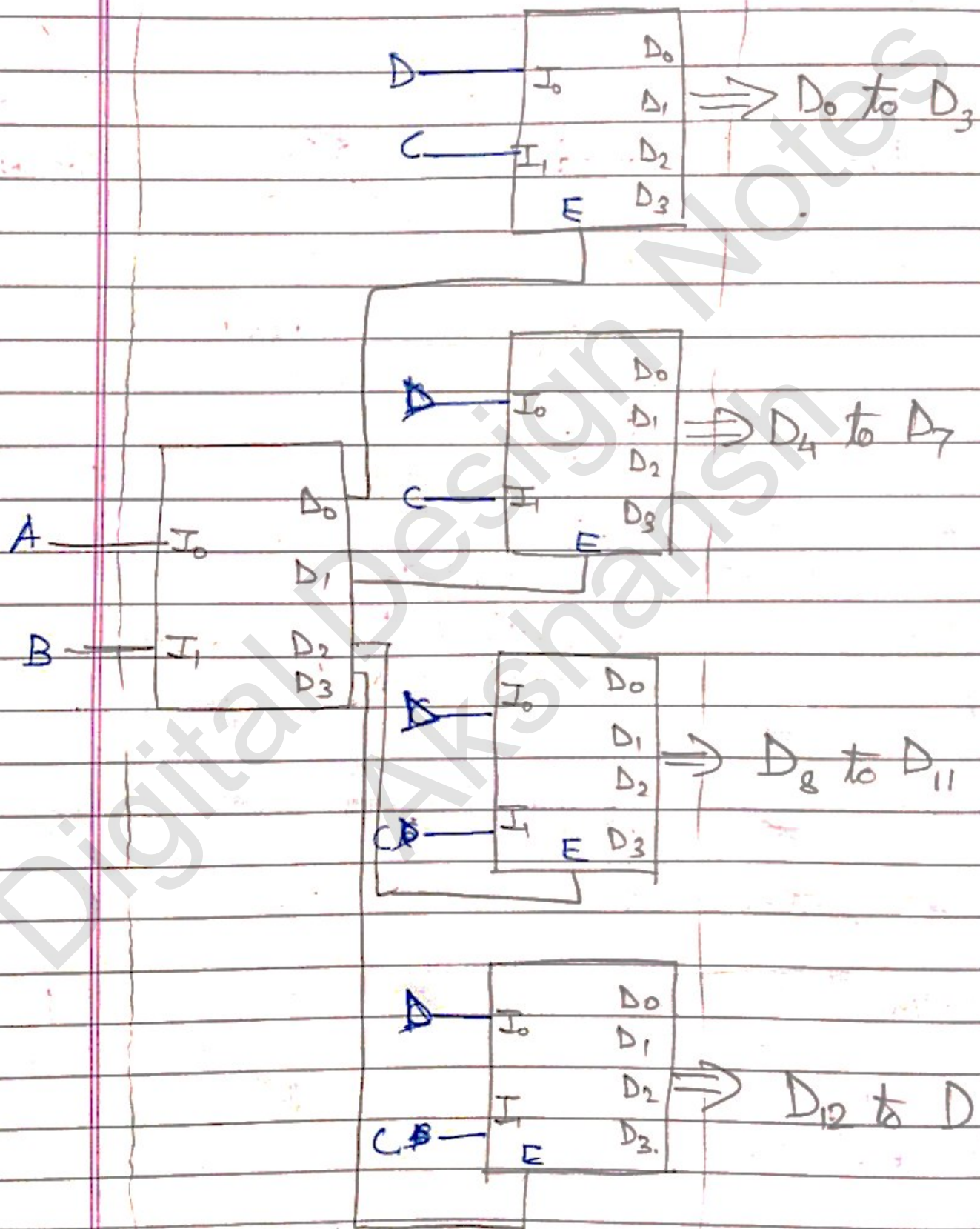
- When A = 0 ⇒ enable pin is activated for upper IC.
  When A = 1 ⇒ enable pin is activated for lower IC

⁂ AIM: Make 4×16 decoder using 3×8 decoder



Date
Page

✡ AIM: Make 4x16 decoder, using 2x4 decoders.



✱ Function Implementation using decoders.

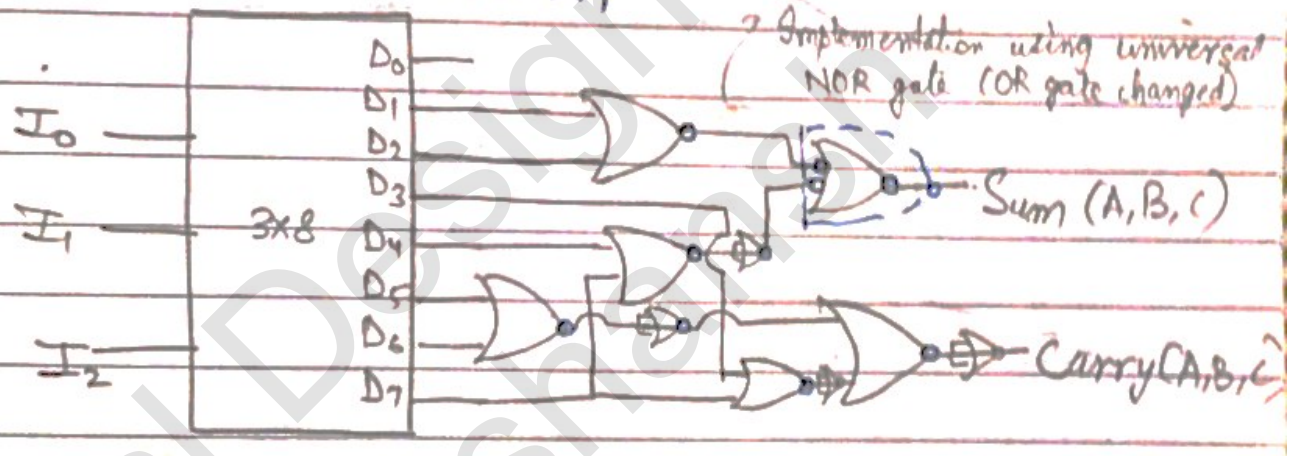
- Using adder circuit

$S(A,B,C) = A \oplus B \oplus C = \sum(1,2,4,7) = m_1 + m_2 + m_4 + m_7$

$Carry(A,B,C) = AB + BC + AC = \sum(3,5,6,7)$

$= m_3 + m_5 + m_6 + m_7$

NOT OPTIMUM



Idea Implementation :- I have to make sum f^n, say ($\sum(1,2,4,7)$) as $D_1 + D_2 + D_4 + D_7$. So, use OR gates. Why, for Carry f^n.

✱ Note :- D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7

• Active high.	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
• Active low	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7

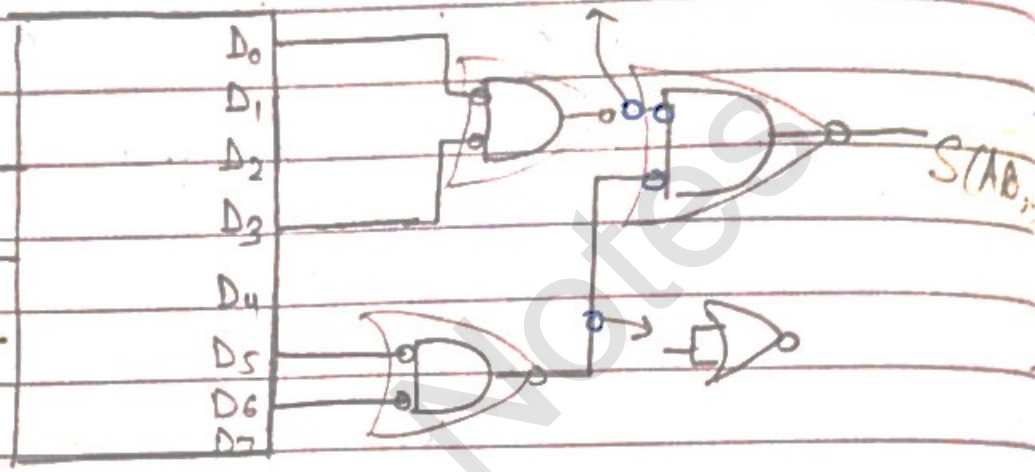
✱ For using only NAND / NOR gate for implement^n, use maxterms. i.e, complement each o/p & use.

Optimum solution

$S = \Pi(0, 3, 5, 6)$

$C = \Pi(0, 1, 2, 4)$

2 i/p NOR gate alone



↓ Implementⁿ of Sum fⁿ using

✓ Active high

✓ NOR gate

Ily. do for NAND gate.

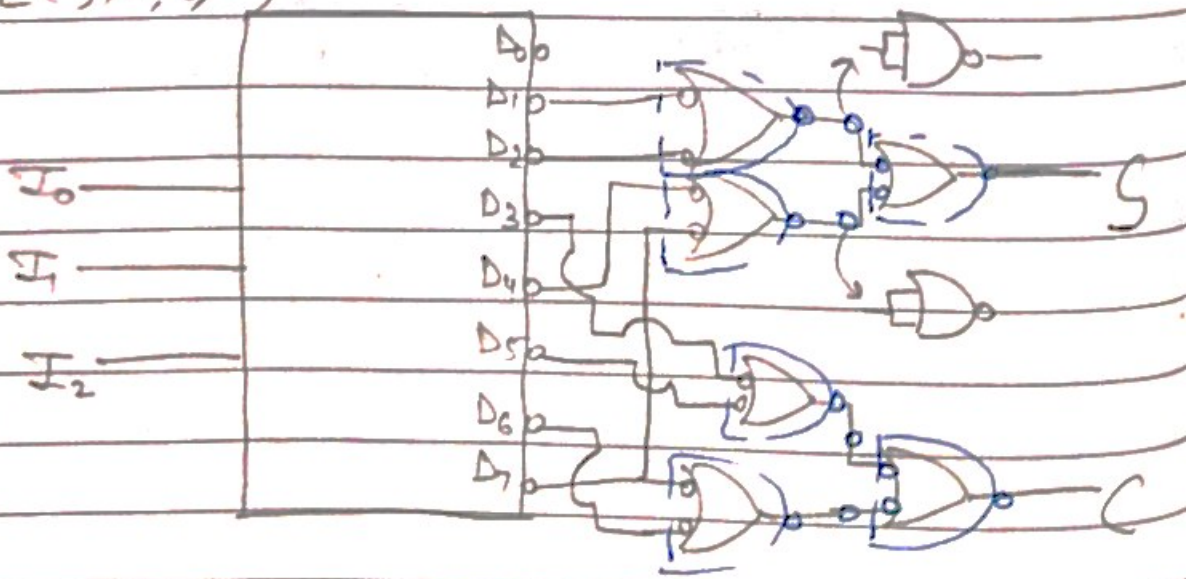
Now, Sum fⁿ implementation

Given ACTIVE LOW

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

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

2 i/p NAND gate alone



Puffin
Date
Page

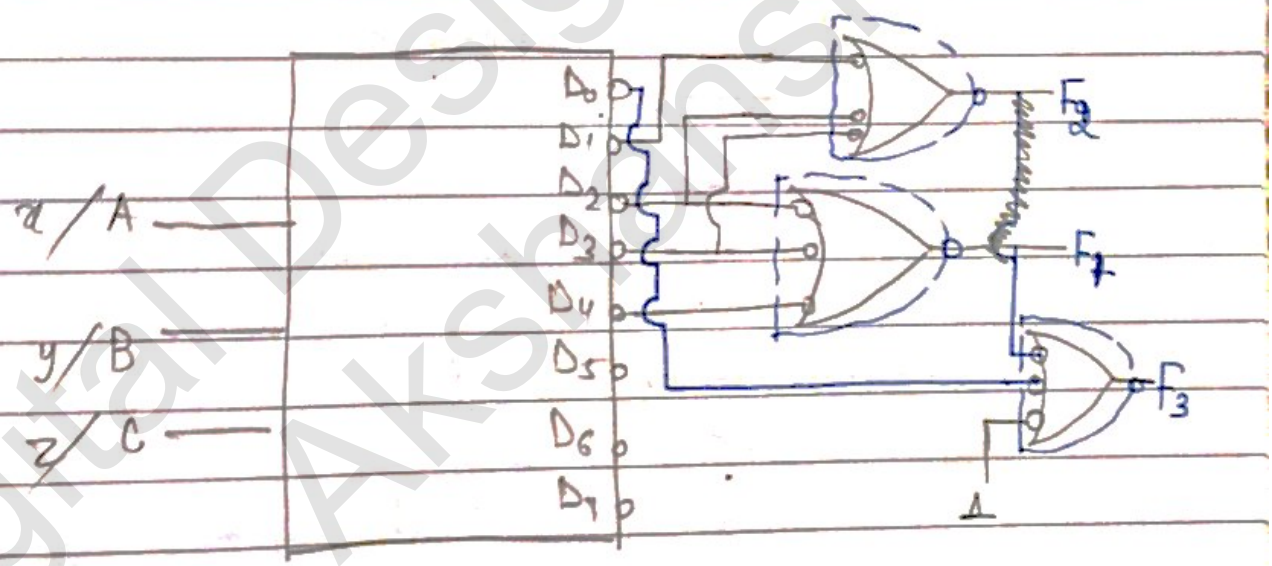
Q. $F_1(A,B,C) = \Pi(0,1,5,6,7) = \Sigma(2,3,4)$

$F_2(A,B,C) = \Sigma(1,2,3)$

$F_3(x,y,z) = \bar{x}y + \bar{y}\bar{z} = \Sigma(0,2,3,4)$
$= \Pi(1,5,6,7)$

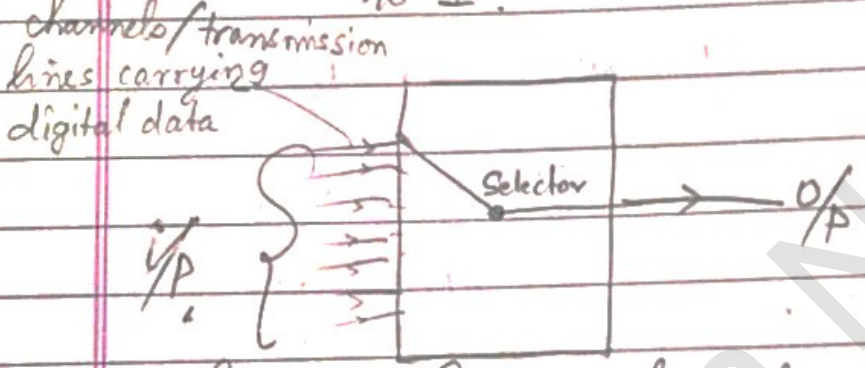
Implement given fn using

- ✓ Active low
- ✓ 3 x 8 decoder
- ✓ 3 i/p
- ✓ AND, OR, NAND gates to be used only.
- ✓ Use ONLY 3 logic gates



Multiplexer / Demultiplexer

→ Many multiplexed to 1.



how to select 1 channel out of (2^n) — under std. case

* 1 K = 2^{10} = 1024

1 K X 8 ⇒ each space is 8 bit in size ⇒ called as words.

∴ each of the 1024 words would be 8 bit in size

Memory location of 1 K
↳ 1024 locations having data (in form of 0's & 1's)

A comp. sys. has:
- Address (wires) (bus → group of electrical line) : bus carrying address values
- Data : " " data "
- Control : " " control of info

To represent the 1024 addresses (from 0 - 1023), we require 10 bits so, we need 10 address lines

Total data available to me = 1 K x (8 bit) of memory
= 1 KB of memory

2^{10} : kilo
2^{20} : mega
2^{30} : giga

10 : address lines
20 : "
30 : "

* Here, for memory location for 1K x (8)

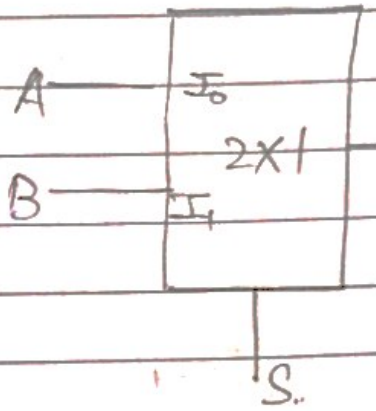
we have taken 8 bits space for each address.

For 2^n data lines ⇒ we need n address lines

In multiplexer :- address lines are called as select lines

So, a multiplexer will have map of 2^n data lines & n address lines. Out of that 1 data info is taken to o/p.

✡ 2×1 multiplexer



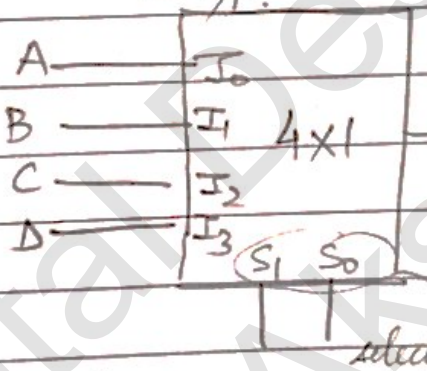
$Y = A\bar{S} + BS$

For 2×1 → If $S=0$ ⇝ I_0 is o/p
$S=1$ ⇝ I_1 is o/p

✡ (4×1) multiplexer

4 i/p's

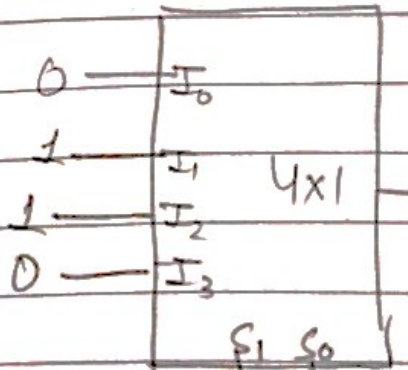
1 o/p.



$Y = \bar{S}_0\bar{S}_1A + S_0\bar{S}_1B + \bar{S}_0S_1C + S_0S_1D$

I want to represent each i/p using 2 select lines. So, basically, generate 4 comb^n of select lines (00, 01, 10 or 11) & link to each i/p. (≅ 2×4 decoder)

✡ Implement Sum f^n of ½ adder using multiplexer.



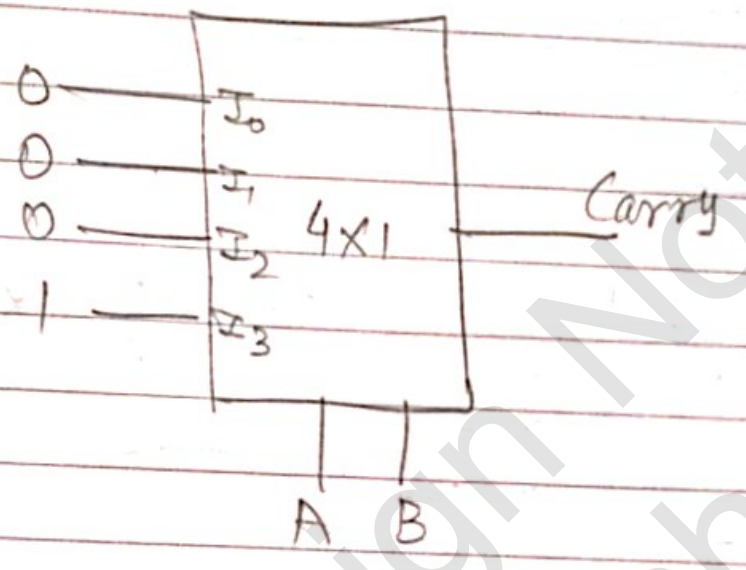
$Sum = \bar{S}_0\bar{S}_1(0) + S_0\bar{S}_1(1) + \bar{S}_0S_1(1) + S_0S_1(0)$ → 0

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

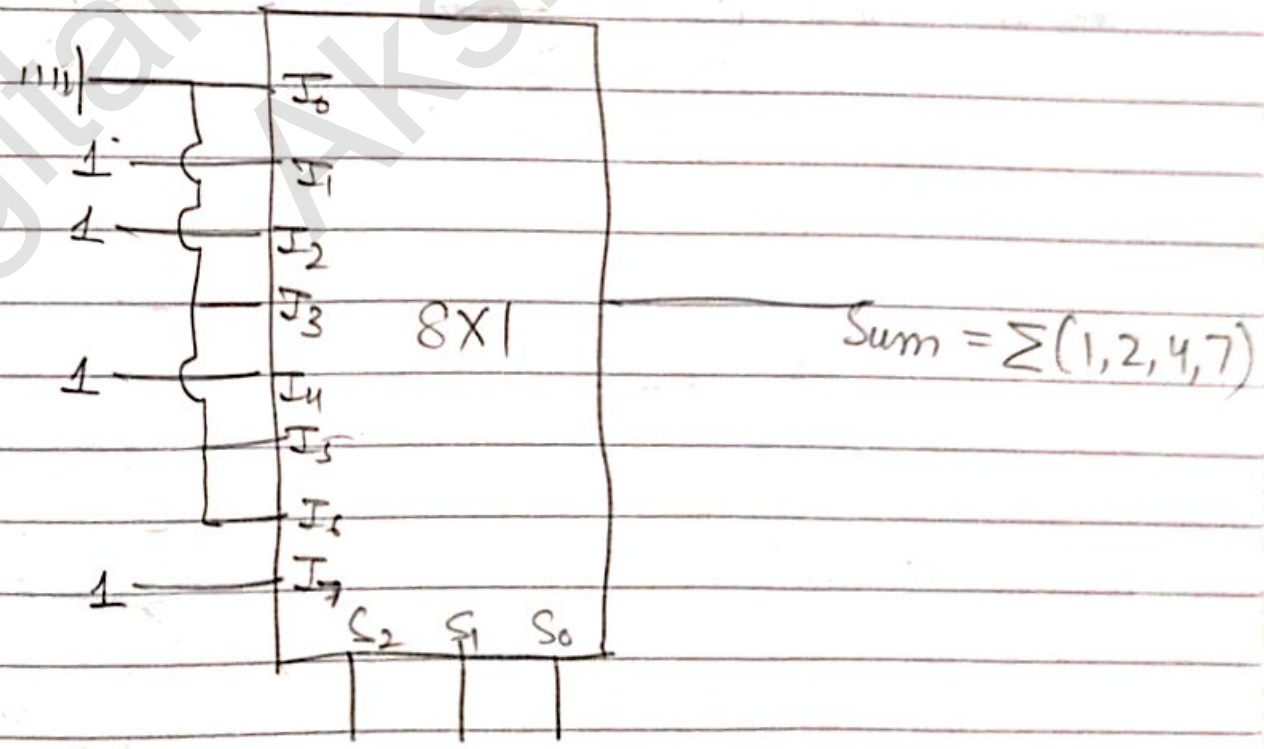
$= \bar{A}B + A\bar{B} = A \oplus B$

= Sum f^n

★ Implement carry fⁿ of ½ adder using multiplexer.

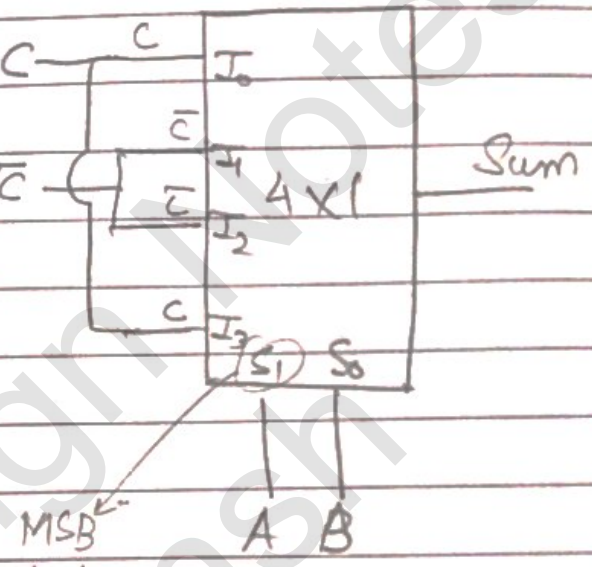


★ For implementing full adder sum using multiplexer ⟶ 8×1



✡ Implement a full adder using two 4×1 multiplexers.

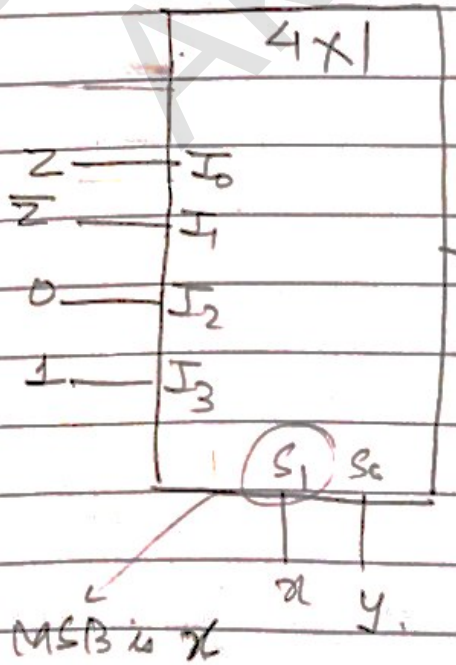
A	B	C	F	
0	0	0	0	
0	0	1	1	C
0	1	0	1	
0	1	1	0	$\bar{C}$
1	0	0	1	
1	0	1	0	$\bar{C}$
1	1	0	0	
1	1	1	1	C



(Sequence matters here)

✡ Implement using ~~8×1~~ 4×1 multiplexer.

$F(x, y, z) = \sum(1, 2, 6, 7)$



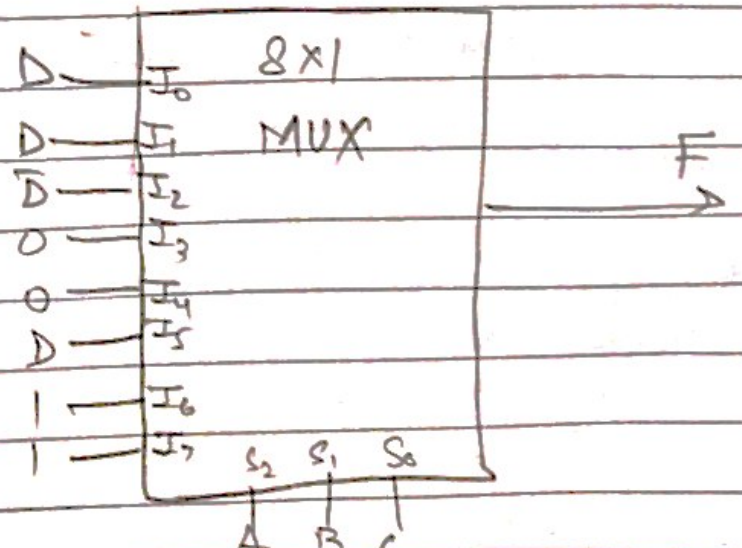
x	y	z	F	
0	0	0	0	z
0	0	1	1	
0	1	0	1	$\bar{z}$
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	

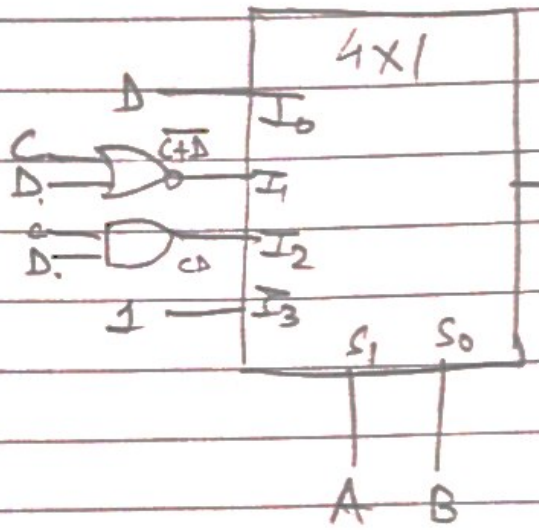
Puffin
Date
Page

✡ Implement using 8×1 multiplexer.

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

A	B	C	D	F	8×1	4×1
0	0	0	0	0	D	D
0	0	0	1	1		
0	0	1	0	0	D	
0	0	1	1	1		
0	1	0	0	1	$\bar{D}$	$\bar{C}\bar{D}$
0	1	0	1	0		
0	1	1	0	0	0	
0	1	1	1	0		
1	0	0	0	0	0	CD.
1	0	0	1	0		
1	0	1	0	0	D	
1	0	1	1	1		
1	1	0	0	1	1	1
1	1	0	1	1		
1	1	1	0	1	1	
1	1	1	1	1		





§ Encoders

↳ Take 2^n i/p, o/p → n.

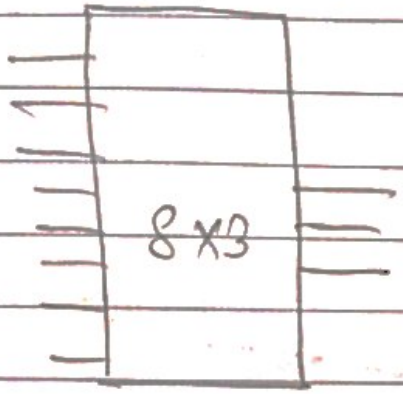
(opposite of decoder)

for n = 3

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

=> If D_4 goes high ⇒ corresponding o/p.

$m_4 \curvearrowright 100$



✱ x, y, z are my o/p fns in terms of D_0 to D_7

$x = D_4 + D_5 + D_6 + D_7$

fn. (∵ x is high when any one of them is high)

$y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$.

4 bit Priority encoder

↳ indicates the bit value corresponding the higher significant high bit in given word (eg $D_3 D_2 D_1 D_0$: D_3 is highest)

eg 0 1 0 1 : 0 is MSB.

But Most high significant bit = D_2 (∵ after D_3 (MSB), next high=1)

If I have 4 bits, I need 2 bits to represent them

eg: suppose → 0 0 D_0

0 1 D_1

1 0 D_2

1 1 D_3

Priority encoder

The code values

D_0	D_1	D_2	D_3	x	y	V (Valid bit)
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

fn table

Valid bit says whether i/p code is valid or not. It's valid when atleast 1 bit is high

→ when $D_3 = 0$, don't look at D_2, D_1 or D_0 & indicate $D_3 = 1$ by $x = 1$, $y = 1$

→ when all bits are low, we get $x = X, y = X$ (don't care)

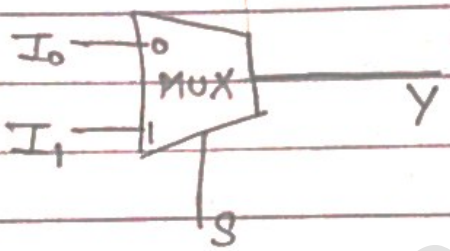
	D_3	D_2	D_1	D_0	X	Y	
d(0)	0	0	0	0	X	X	none high (Invalid o/p)
Σ(1)	0	0	0	1	0	0	D_0 = high
Σ(2,3)	0	0	1	X	0	1	D_1 = high
	0	0	1	X	0	1	
Σ(4,5,6,7)	0	1	X	X	1	0	D_2 = 1 Others dont matter.
	0	1	X	X	1	0	
	0	1	X	X	1	0	
	0	1	X	X	1	0	
Σ(8,9,10,11,12,13,14,15)	1	X	X	X	1	1	D_3 = 1. So, other terms don't matter. It'll give o/p as 1 D_3: higher significant high bit
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	

Making f^n $X = d(0) + \Sigma(4,5,6,\ldots,15)$

$Y = d(0) + \Sigma(2,3,8,9,\ldots 15)$

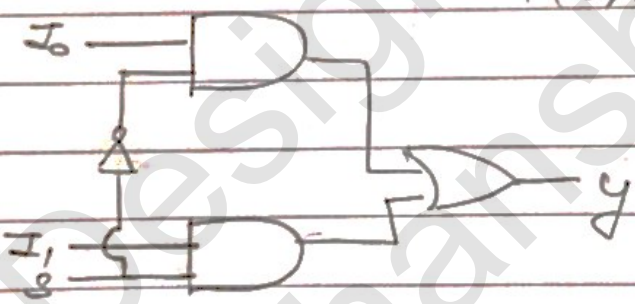
✱ Multiplexer Implementation

Symbolic Representation of Multiplexer

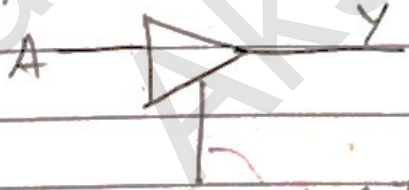


2×1 MUX

≡



✱ Buffer → stores data



(Tristated buffer)

→ activated with high i/p

→ control i/p

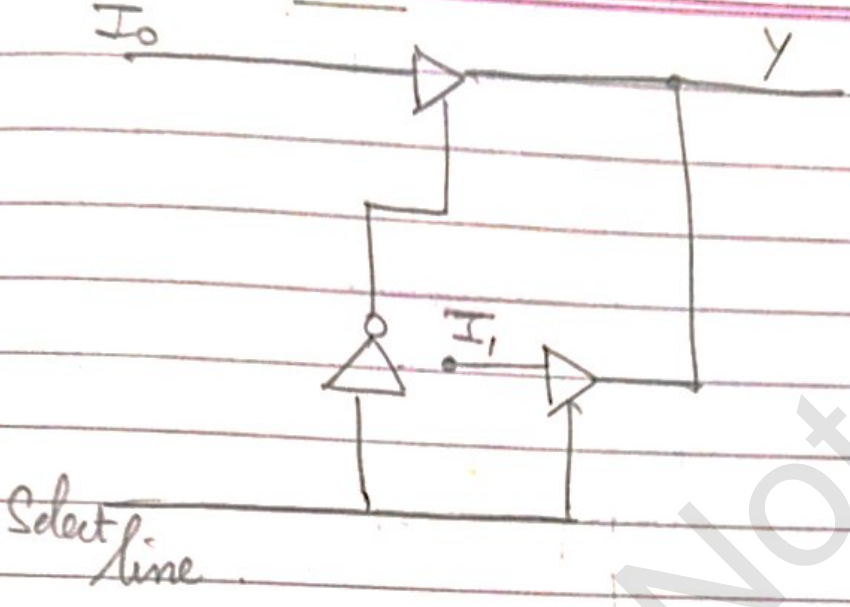
→ $Y = A$ when $C = 1$

$Y = Z$ when $C = 0$

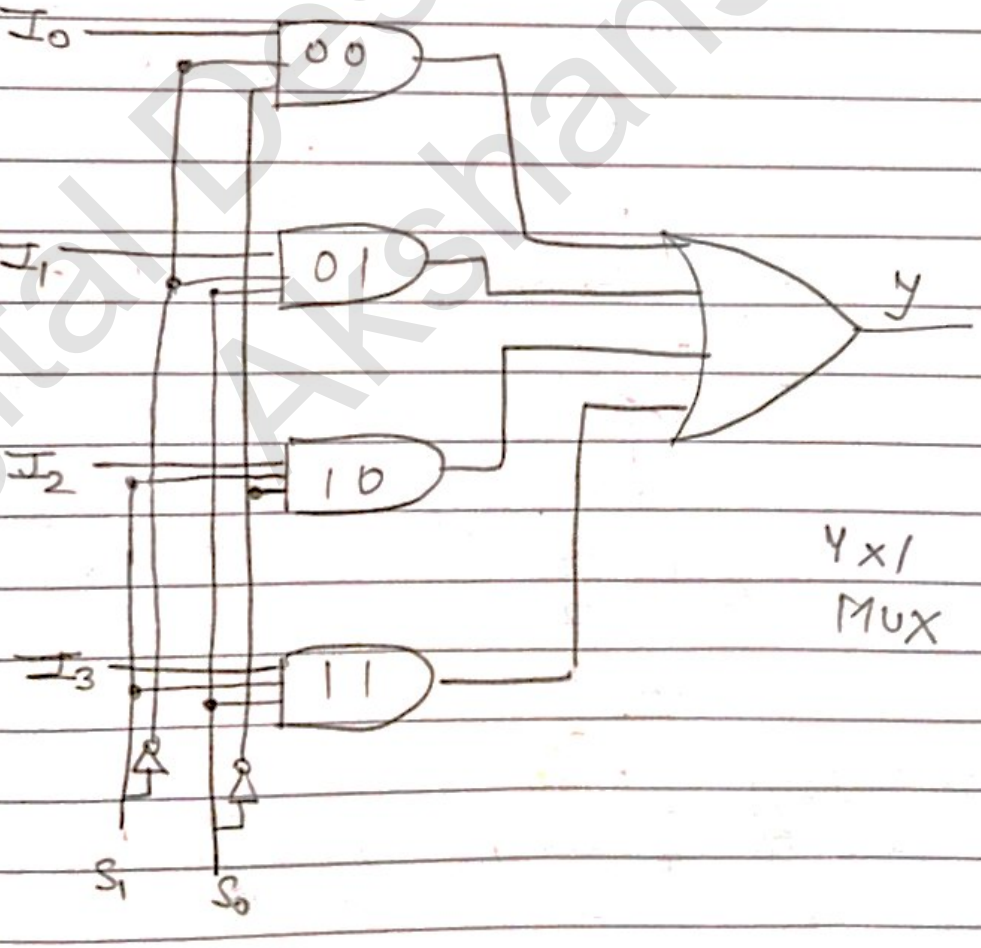
→ basically, the circuit didn't work.

or, a state reached ⇒ high impedance state

TRISTATED LOGIC

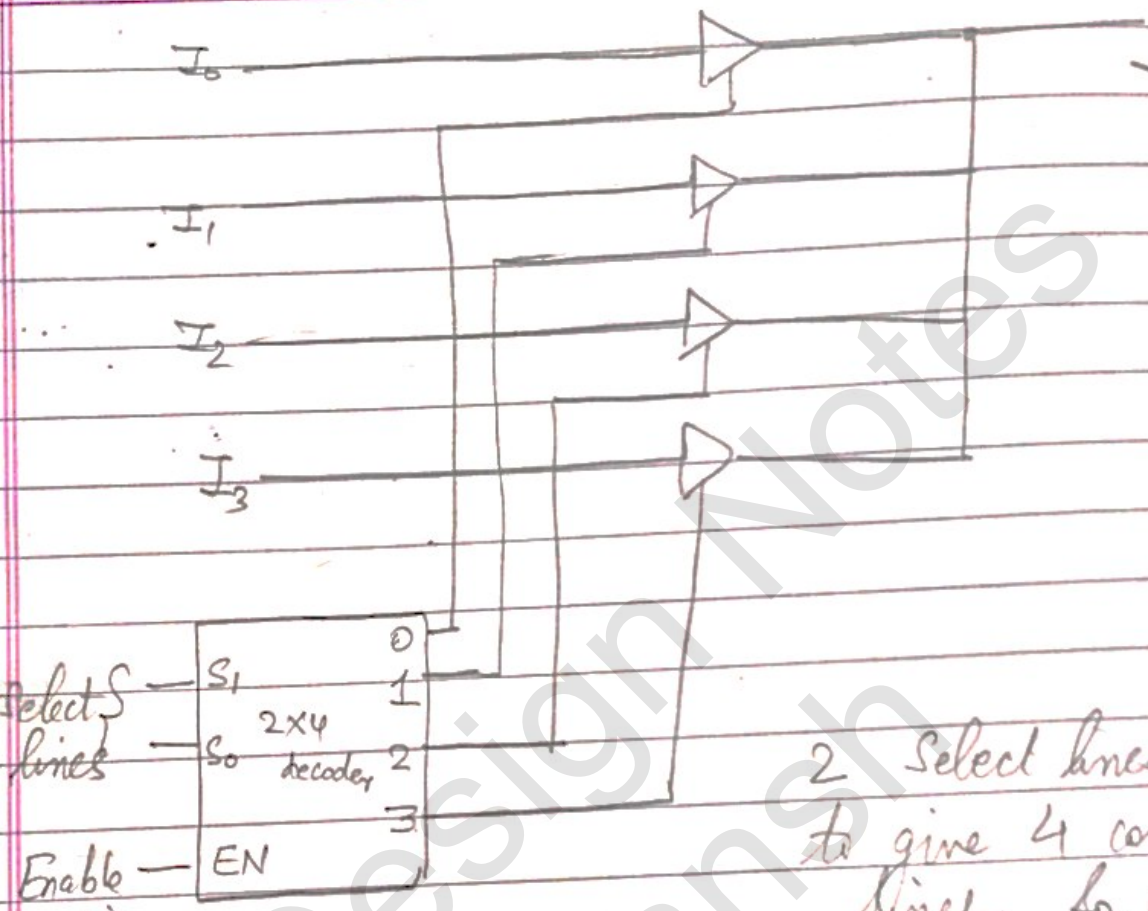


✡ 4×1 multiplexer



4×1 MUX

☆ Tristated logic using buffer & decoder.

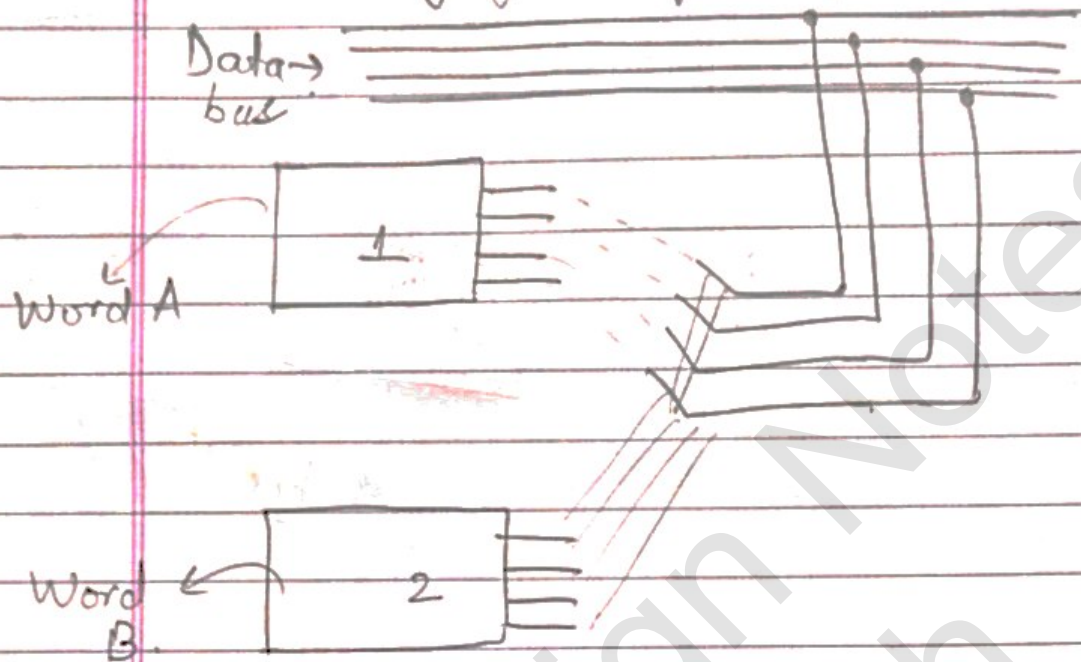


2 Select lines have to give 4 control lines. So, we used decoder.

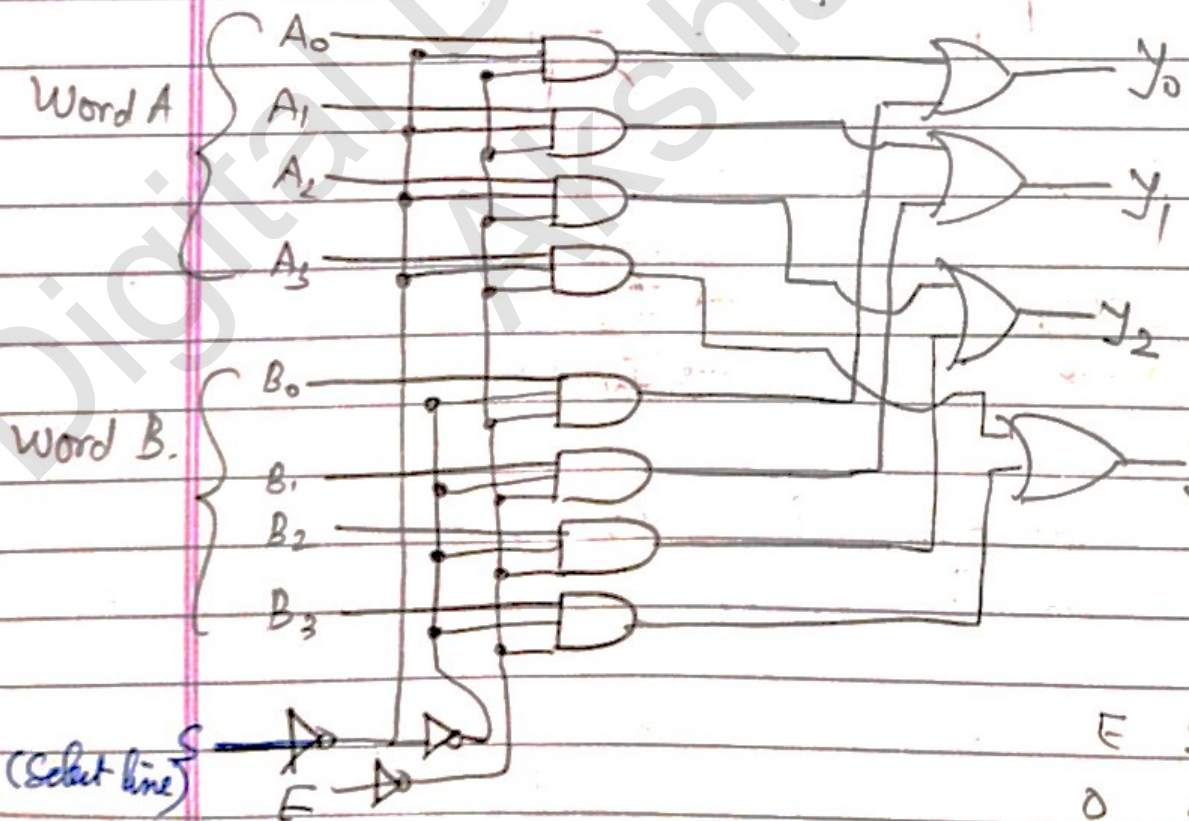
☆ 8/16/32/64 bit sys. (computer)
Size of data (no. of bit) the processor can handle at a time.

☆ For 1 K → 10 address lines
Due to limit^n of address lines, I have limit^n on memory.
☆ For 1 Mega → I need 20 address lines

✱ Selecting from group of conductors

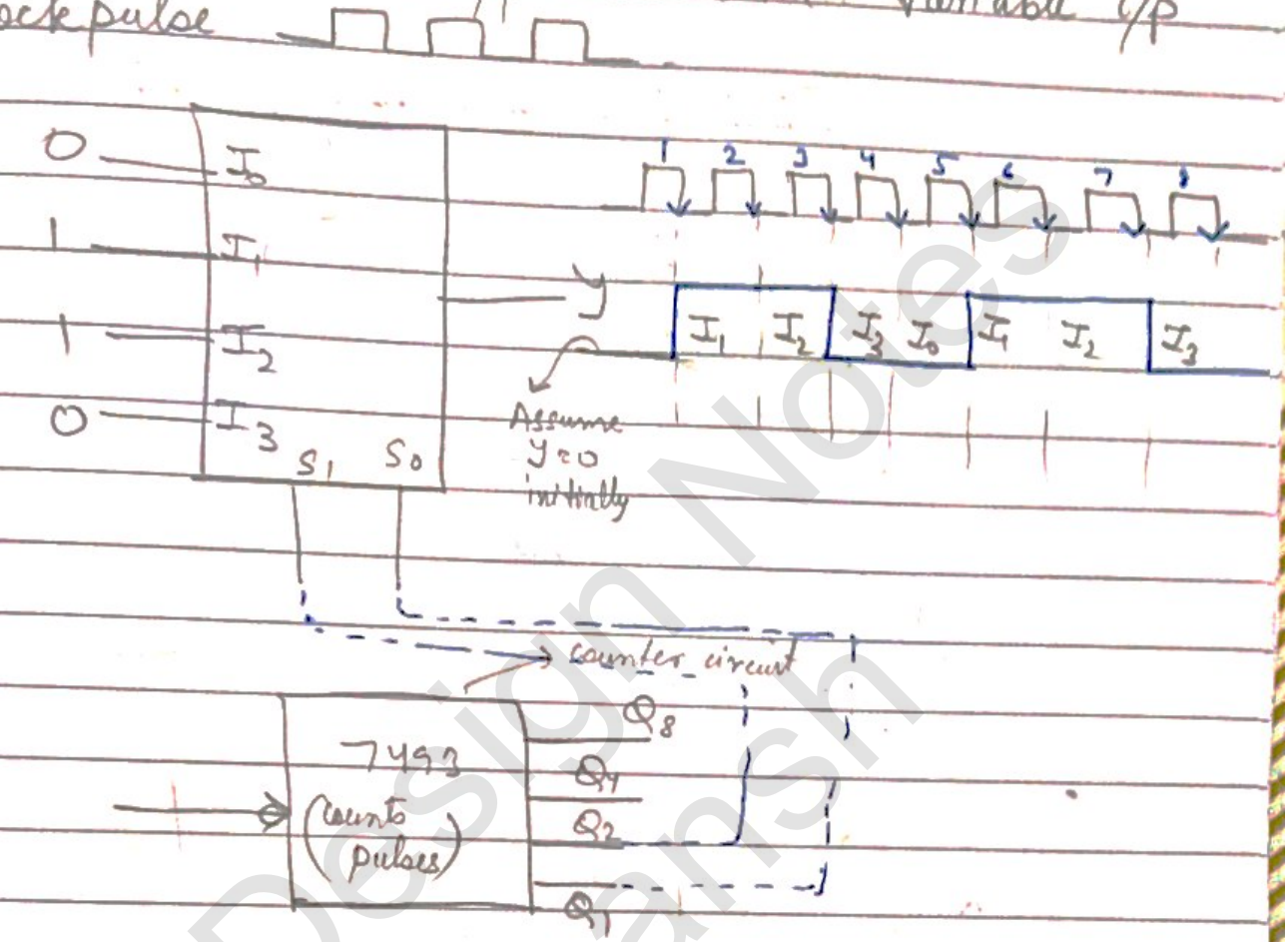


Choose any one of 2 words using multiplexer



E	S	O/P
0	0	A
0	1	B
1	X	0's

Q What will be o/p 'y' w.r.t variable i/p clock pulse



when 1st clock pulse comes, it will give o/p 0 0 0 1 ($Q_4 Q_3 Q_2 Q_1$)

				S_1 S_0	
1st clock pulse	→ I_1	1	0 0	0 1	I_1
2nd	→ I_2	1		1 0	I_2
3rd	→ I_3	0		1 1	I_3
4th	→ I_4	0	1 0 0	0 0	I_0
			1	0 1	I_1
			1	1 0	I_2
			1	1 1	I_3
			1 0	0 0	I_0

eg: Let select line i/ps be

$5 = I_1$ (1 0 1)

$1 = I_1$

$2 = I_2$

$1 = I_1$

$0 = I_0$

Q_2 Q_1
|| ||
S_1 S_0 ⇒ I_1
0 1

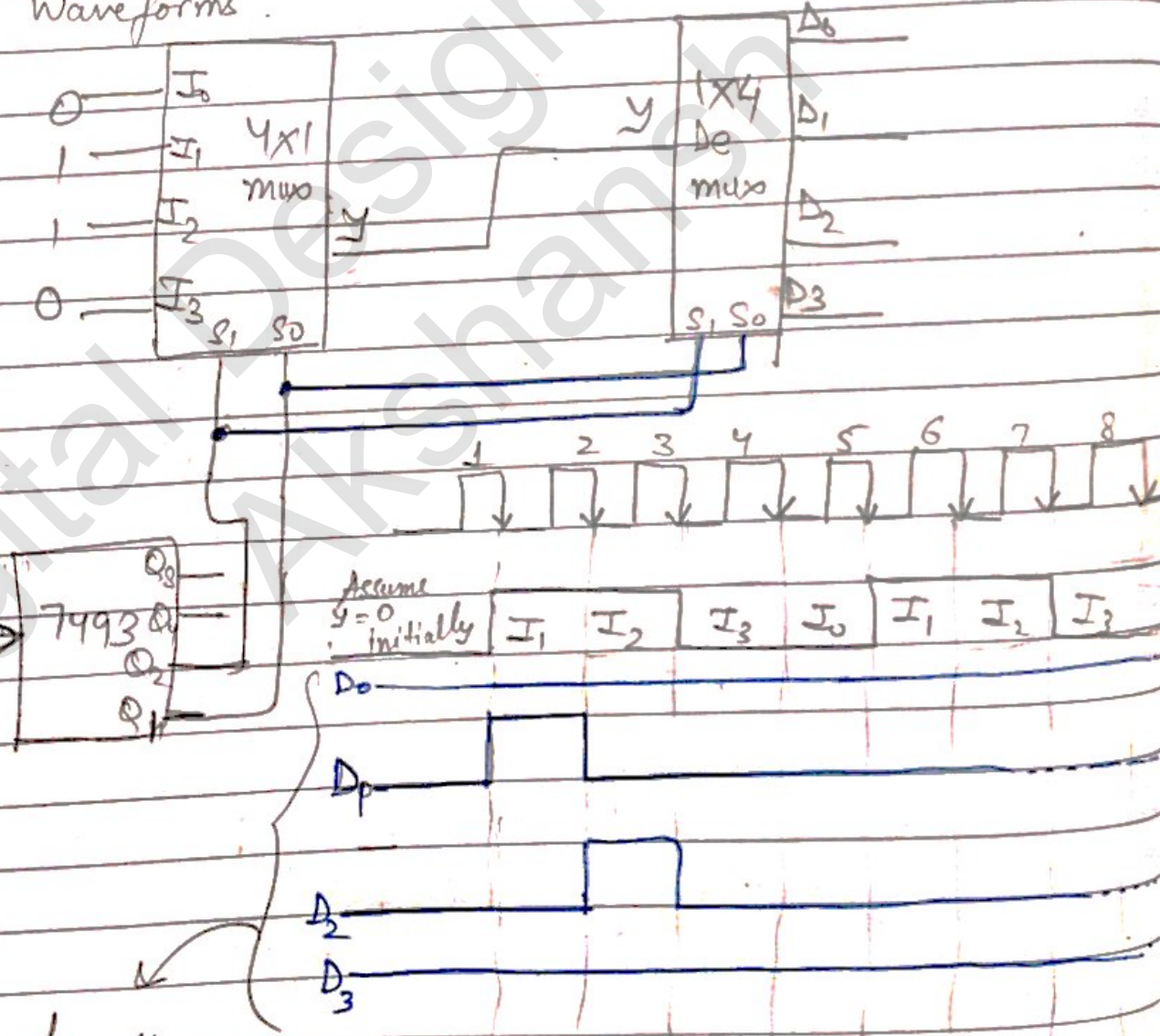
Q Why connect to Q_2 & Q_4. Select line i/ps change.

☆ In previous circuit, i/p of multiplexer is constt.

Select lines are varying ctsly with time.

So, time varying multiplexing is being done

☆ Now, consider a demux circuit attached. Find o/p waveforms.



☆ Using decoder as a demultiplexer

In decoder (3×8)

composed of two 2×4 decoders

A (MSB) B C (LSB) → i/p

i/p connected to enable pin

o/p gives out 8 fns s.t each fn is high only at specific i/p

* It has
  - 3 i/p bits :
    - ↳ MSB connected to EN
  - & 1 enable bit (EN)

Change →

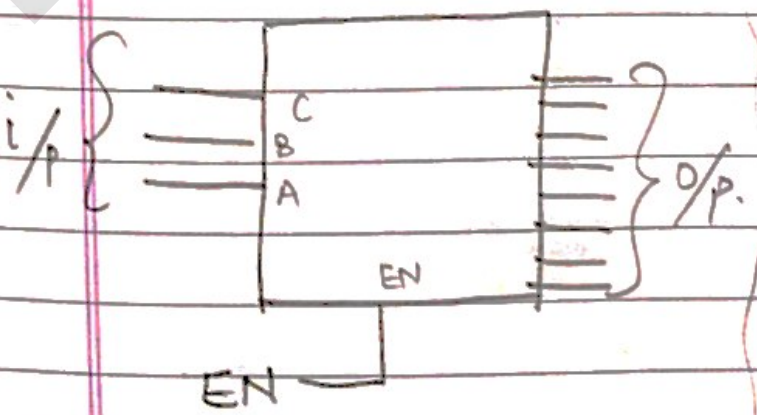
In demux (1×8)

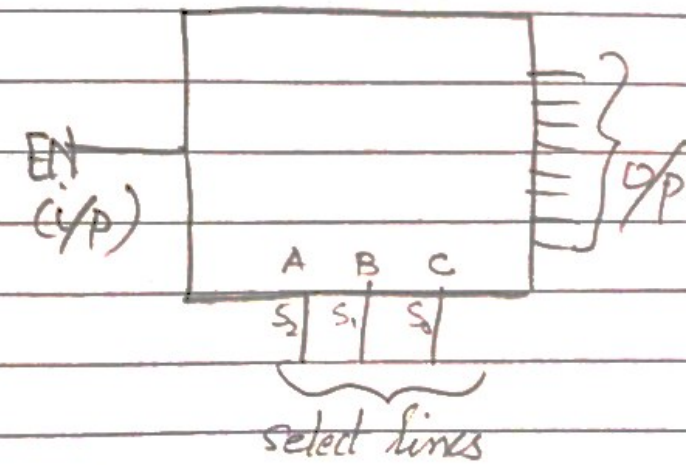
A (MSB) B C (LSB)

↳ select lines

Enable bit : It can be said as a fn which can be high or low. That means an IC giving output as fns (high only at one specific value) can be used as enable bit

* It has
  - 3 select lines
    - ↳ MSB connected to EN
  - &
  - 1 EN is i/p





Puffin
Date 31/10/12
Page

MAGNITUDE COMPARATOR

* MSI → Arithmetic circuits
  - → Adder
  - → Substractor
  - → Multiplier (left)
* → Converter circuits
  - → decoder
  - → encoder
  - → mux
  - → demux

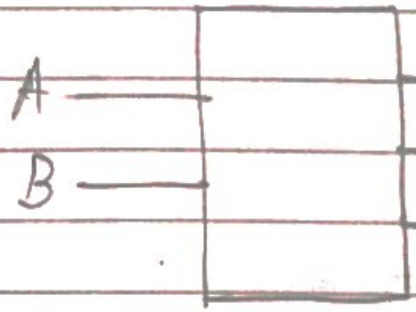
  (done so far)
* → Comparaters
  - → Single bit
  - → Multiple bit

☆ Comparing

3 expected results by comparing a & b

① $a = b$, $a \neq b$

$a > b$ ② $a < b$ ③



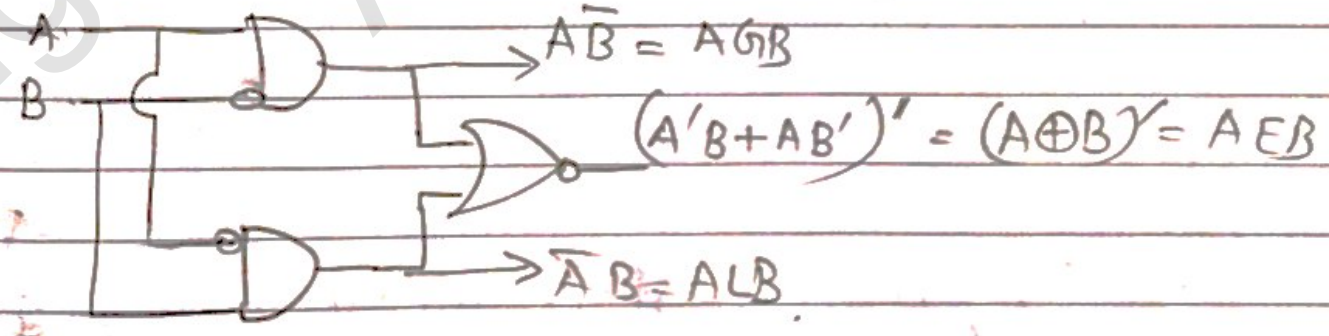
AGB : A > B
AEB : A = B
ALB : A < B

☆ Considering 1 BIT WORDS of A & B

AGB ⇒ A = 1, B = 0
⇒ AB̄

ALB = ĀB

AEB = AB + ĀB̄ = (A⊕B)′
= A⊙B
= A′⊙B
= A⊕B′



Puffin
Date
Page

Considering 2 BIT WORDS

$\Rightarrow A \rightarrow A_1 A_0 \quad B \rightarrow B_1 B_0$

$A > B = A_1 \overline{B_1} + x_1 A_0 \overline{B_0}$

$A < B = \overline{A_1} B_1 + x_1 \overline{A_0} B_0$

$A = B = x_1 \cdot x_0$

Assume, for any bit, $A_i = B_i = x_i$

i.e $A_1 = B_1 = x_1$ - - -

x is equality = 1 : if equal

0 : not equal

4 BIT WORD

$A \rightarrow A_3 A_2 A_1 A_0 \quad B \rightarrow B_3 B_2 B_1 B_0$

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

$A = B = x_3 x_2 x_1 x_0$

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

i.e, comparing MSBs of both words.

If MSB is same then comparing 2nd bits. - - - Ily others

Eg: A: 4 3 2 1

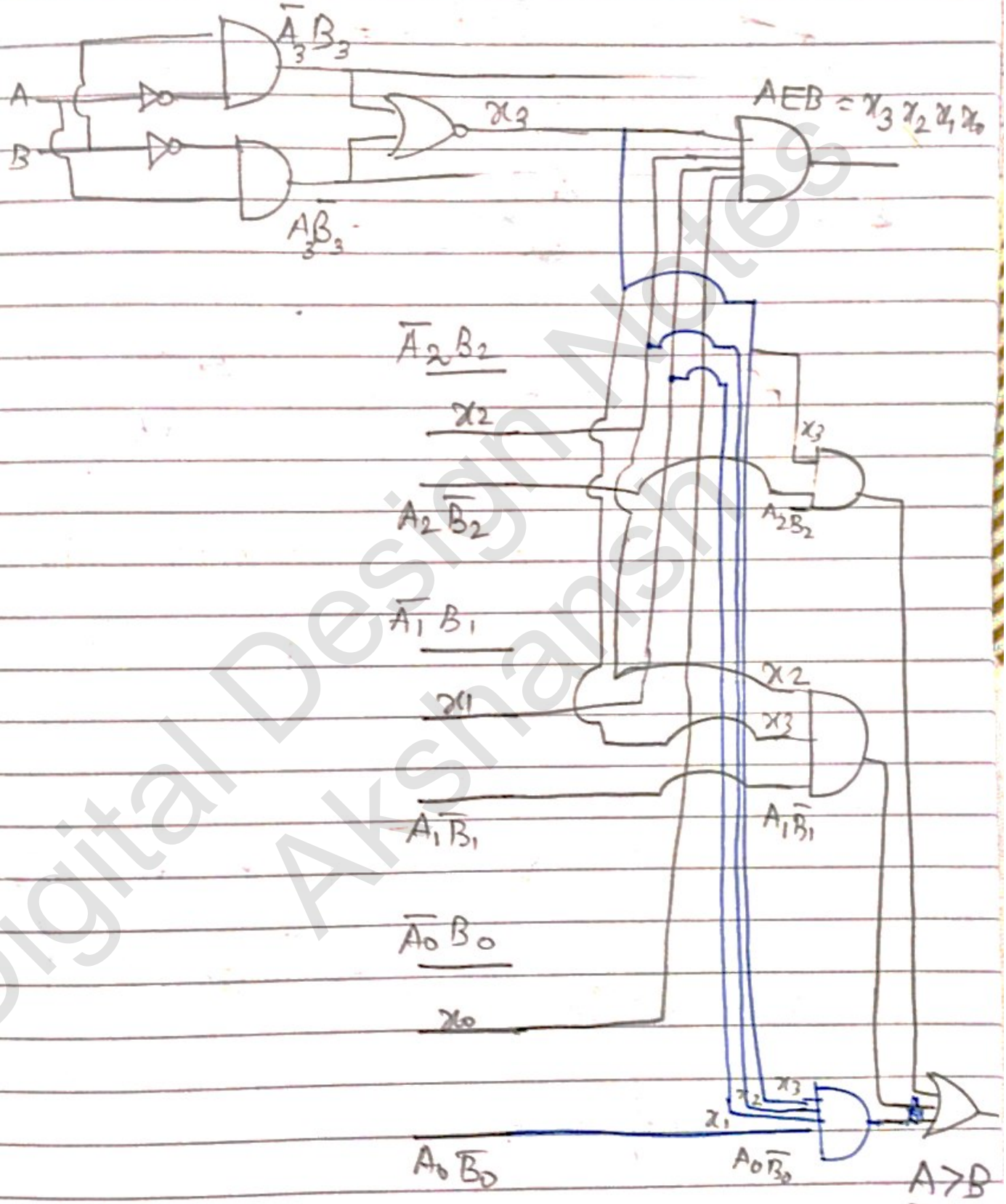
B: 5 6 7 8

$A < B$

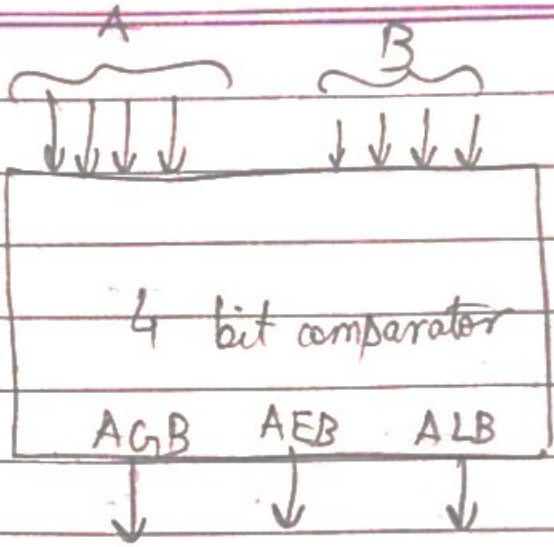
$\Rightarrow A \; L \; B$

$\Rightarrow 4 \; L \; 5$

Implementing 4 bit



Hy for A L B

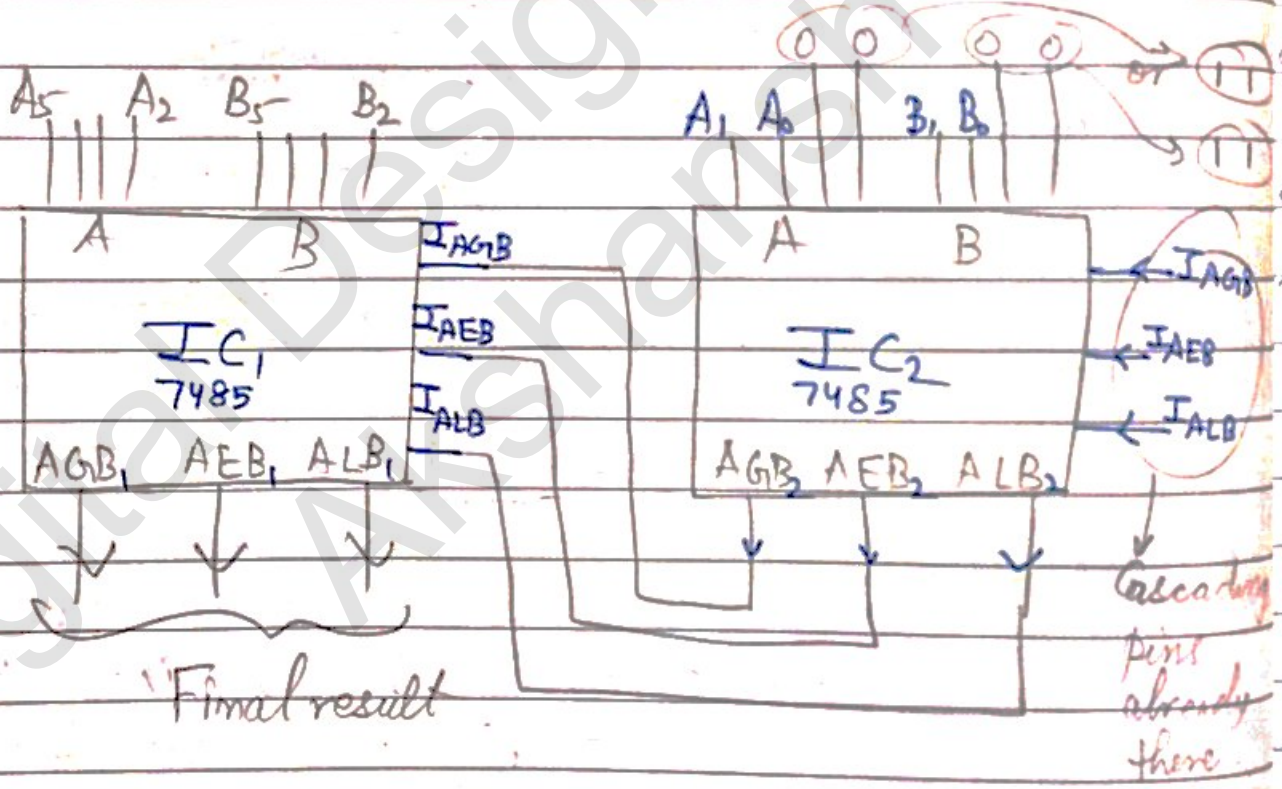


Std. circuit

eg: Comparing 6 bits words

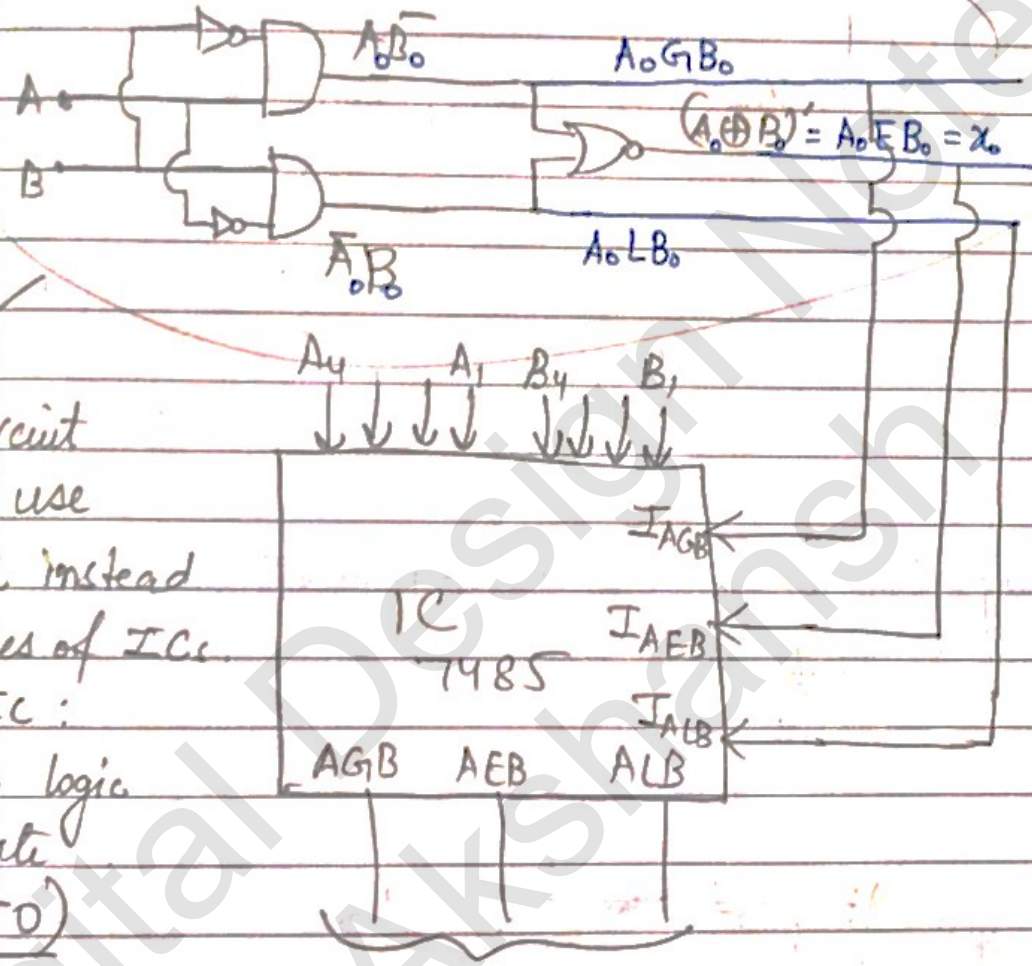
$A \rightarrow A_5 A_4 A_3 A_2 A_1 A_0$

$B \rightarrow B_5 B_4 B_3 B_2 B_1 B_0$



So, firstly IC_1 is being compared.
If result is equal, then IC_2 is used or compared to give its o/p to i/p pins of cascade & that is ANDed with AEB_1.
Finally o/ps are got.
Note: ANDing is internal.

eg Comparing 5 bit words : Only 1 7485 is given.



Make circuit efficient: use single IC instead of 3 types of ICs.
Single IC: XOR logic gate (PTO)

Idea : Compare first 4 MSB bits. Use the circuit of 2 bit comparator for comparing A_0 & B_0. If the o/p $A_4A_3A_2A_1 = B_4B_3B_2B_1$, then, that AEB is ANDed (=D) with the cascading i/ps coming from the 2 bit comparator circuit.

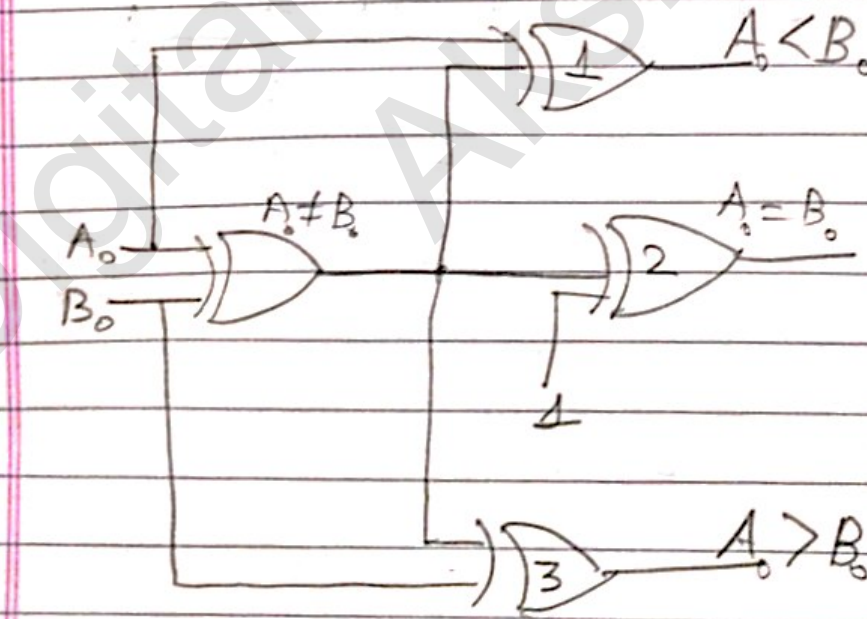
Note :- ANDing is INTERNAL

* lower significant bit has to be cascaded to Higher significant bit IC ∴ o/p comes from Higher IC.

F^n table *

Comparing I/P						Cascading I/Ps			O/Ps		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0			$I_{A_0>B_0}$	$I_{A_0<B_0}$	$I_{A_0=B_0}$	A>B	A<B	A=
① → $A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$			X	X	H	L	L	H
② → $A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$			H	H	L	L	L	L
③ → $A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$			L	L	L	H	H	L

* Single bit comparator EFFICIENT implement^n.
XOR gate



Suppose $A_0 \neq B_0$ is 1 (as o/p), that means A_0 & B_0 are of opp sign. (0 & 1). If IC ① gives 1 as o/p

That means A_0 is 0 & $B_0 = 1$. So, $A_0 < B_0$

Why for $A_0 > B_0$.

Suppose $A_0 \neq B_0$ is 0 (as o/p) & I(1) gives 1 as o/p ⇒ $A_0 = 1$ & $B_0 = 1$ → both A_0 & B_0 have 0 or 1

So, $I_{A_0 = B_0}$ is high

$I_{(A_0 > B_0)}$ is high

$I_{(A_0 < B_0)}$ is high

See ⇒ this should not happen

Now,

Suppose $A_0 \neq B_0$ is 0 (as o/p) [⇒ A_0 & B_0 are both 0 or 1). & I(1) gives 0 as o/p. ⇒

$A_0 = 0$ & $B_0 = 0$

⇒ $I_{(A_0 = B_0)}$ is high

$I_{(A_0 > B_0)}$ is low

$I_{(A_0 < B_0)}$ is low

$I_{(A_0 = B_0)}$ is high in both cases whatever be the value of $I_{A_0 > B_0}$ or $I_{A_0 < B_0}$

So, see F^n table (↖) point (1).

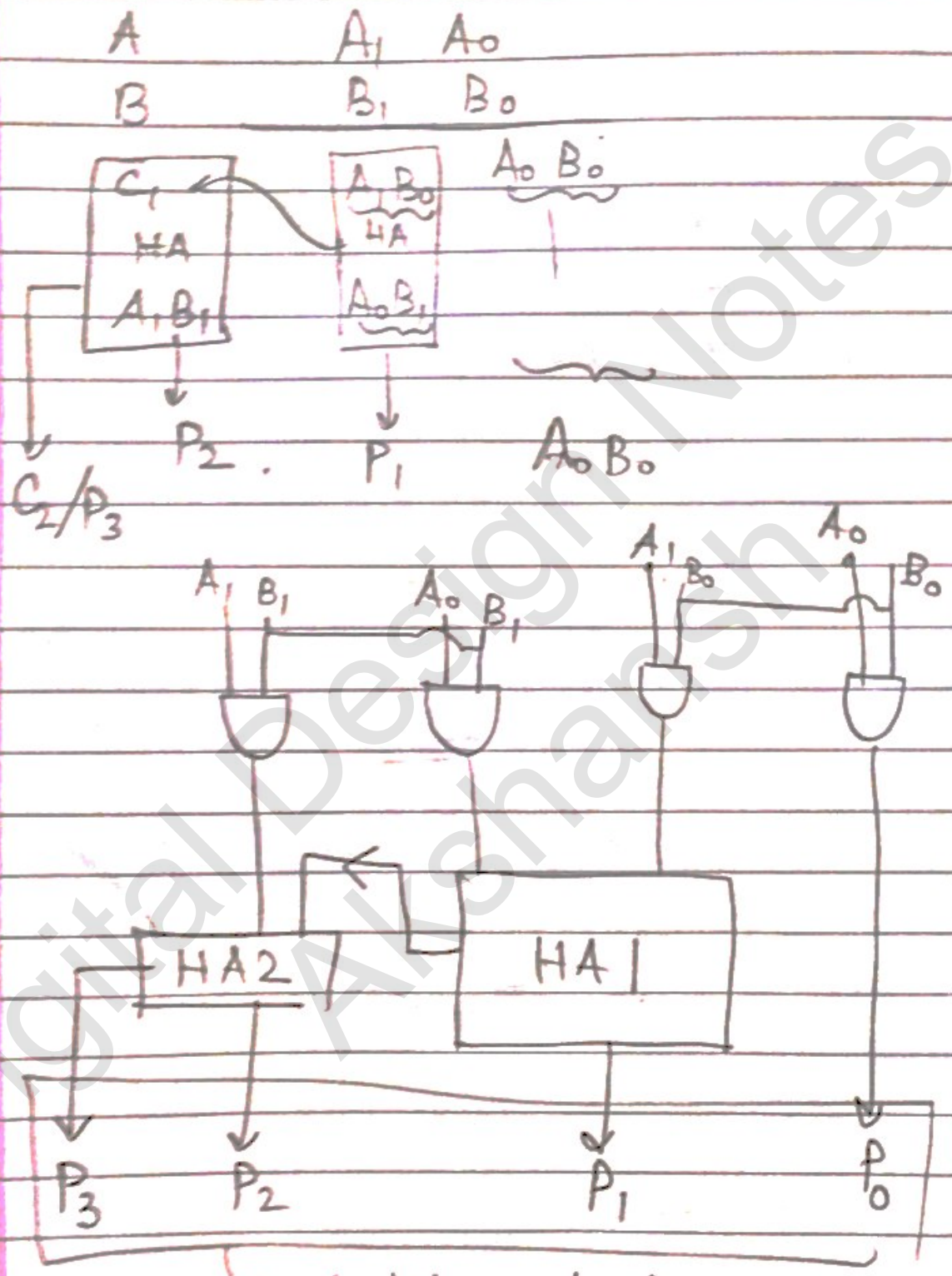
In this case, o/p received is

$\underline{\underline{A = B}}$ = high

$A > B$ } low

$A < B$ }

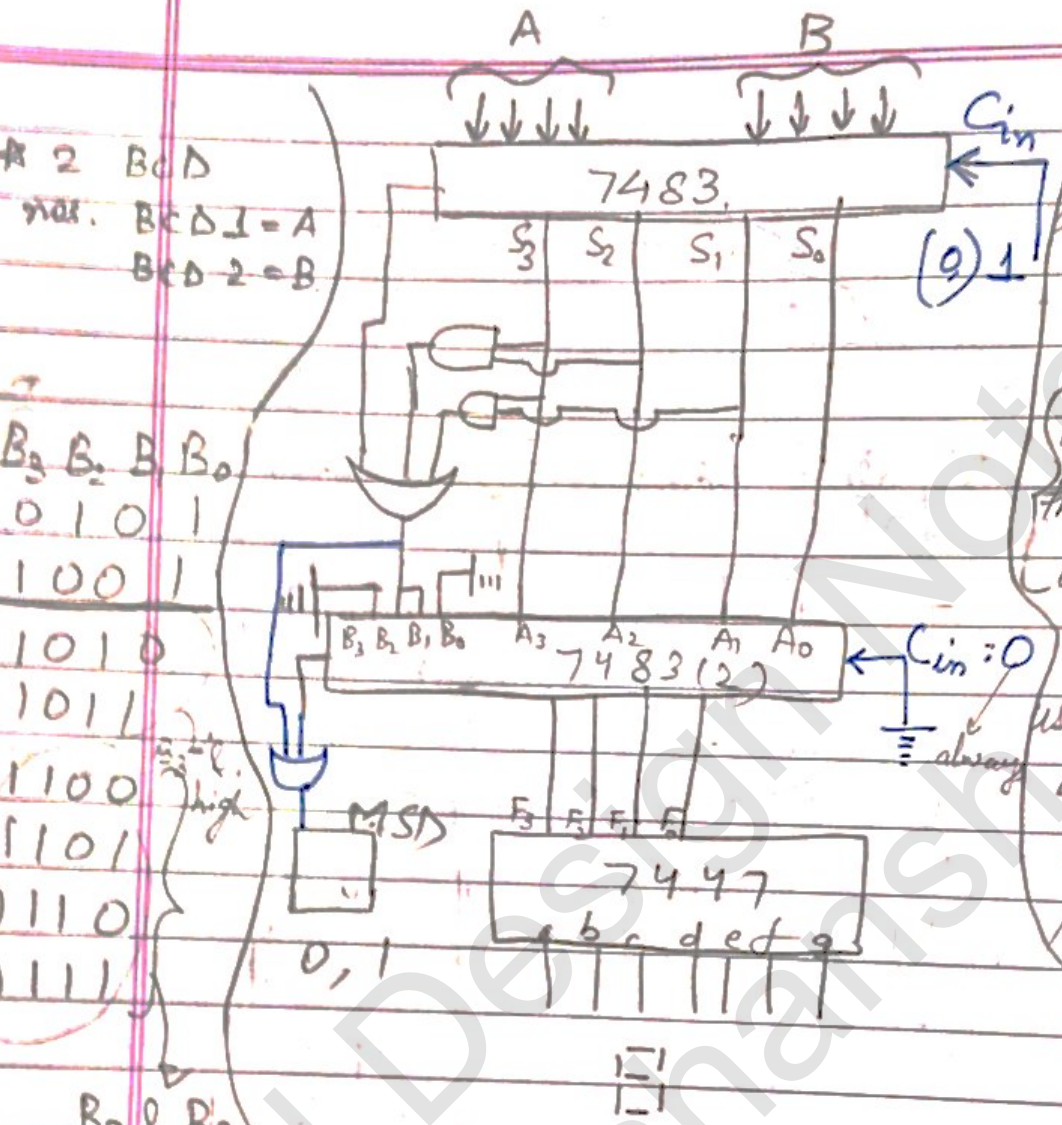
★ 2 BIT MULTIPLICATION



→ 4 bit word for 2 bit multiplic".

★ For m bit x n bit word, product term will have m+n bits.

✱ EXPERIMENT : BCD ADDER



✱ 2 BCD var. BCD 1 = A
BCD 2 = B

B_3	B_2	B_1	B_0
0	1	0	1
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

4 bit binary adder / subtracter circuit.

↓

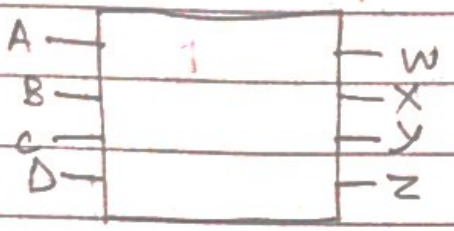
Convert binary op that we get from above to BCD

↓

use BCD to light LED's → digital values ? Use previous pg. circuit

✱ CONVERT :- BCD to Excess-3 CODE

Input BCD				Output Excess-3 Code			
A	B	C	D	W	X	Y	Z
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				



* We need to find expression. So, make k-map for each f^n.

for Z :-

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

for y :- $CD + C'D'$ or $(C \oplus D)'$

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

* Comparing both truth tables, we can directly say that $Z = \overline{D}$

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

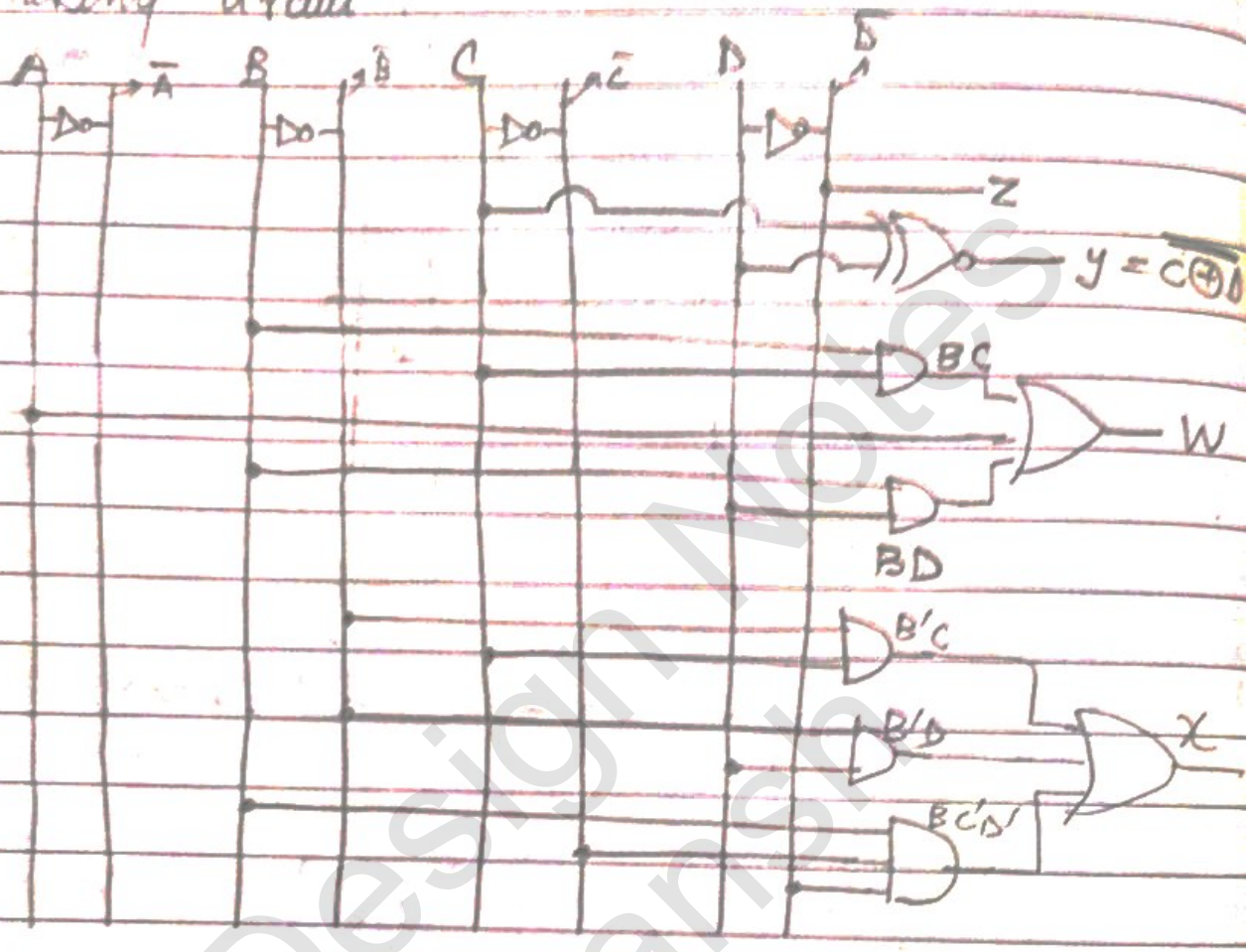
CD / AB	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

$W = A + BC + BD$

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

Puffin
Date
Page

Making circuit



Another way :- Use adder/subtractor circuit.

Like :-

* $A \oplus B \oplus C = \sum(1,2,4,7)$ → Sum fn" → full adder

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

→ no grouping possible

* $\overline{A \oplus B \oplus C} = \sum(0,3,5,6)$

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

Puffin
Date
Page

★ $F = A \oplus B \oplus C \oplus D$: Odd parity

$= \sum(1, 2, 4, 7, 8, 11, 13, 14)$

AB \ CD	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$\overline{F} = (0, 3, 5, 6, 9, 10, 12, 15)$

★ $F = A \oplus B \oplus C \oplus D$: even parity

AB \ CD	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

★ Note :-

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

$= A' \oplus B' \oplus C'$.

So, given a f^n, like $A' \oplus B \oplus C$

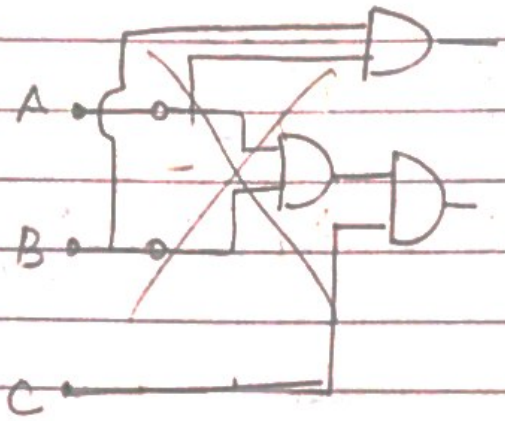
⇓

X-NOR gate

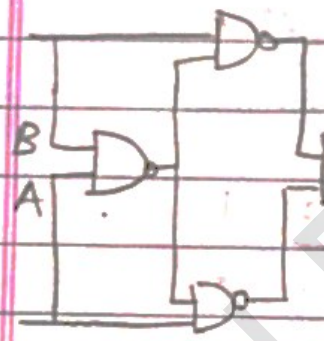
Q Implement 2 i/p XOR gate using NAND gates

$A \oplus B \oplus C = \sum(1, 2, 4, 7)$

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



Optimum solution



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

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

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

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

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

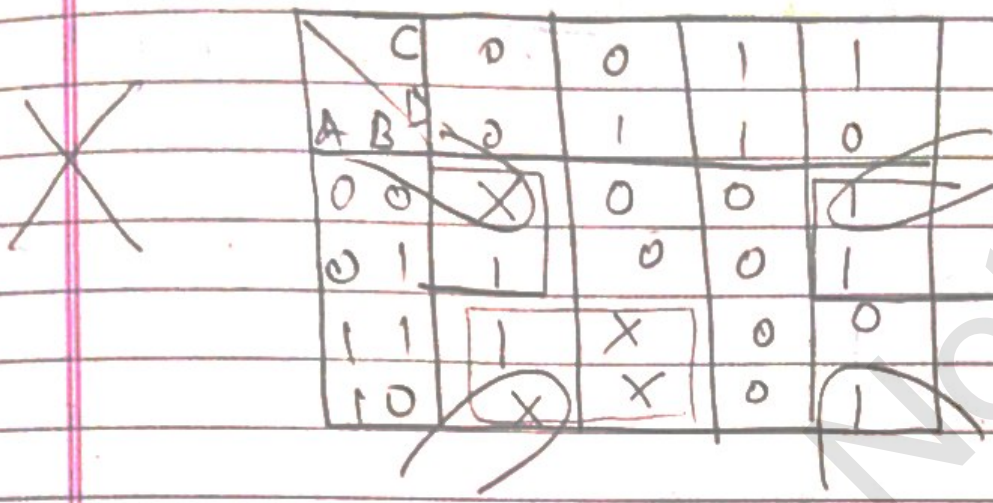
Q. Implement the fn F using not more than 2 NOR gates → i.e in POS form.

$F(A,B,C,D) = \sum(2, 4, 6, 10, 12)$

$(d)(A,B,C,D) = \sum(0, 8, 9, 13)$

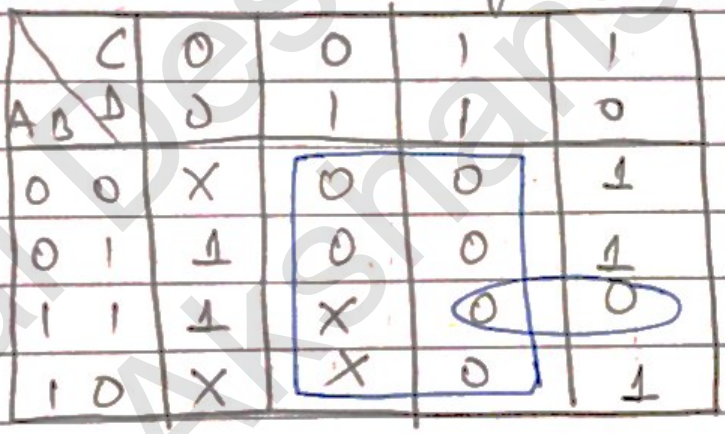
→ don't care terms.

* To implement using:
- NAND gates : We need SOP form
- NOR gates : We need POS form.



Here, I grouped 1's
⇓
SOP expression

I need POS form to group 0's.



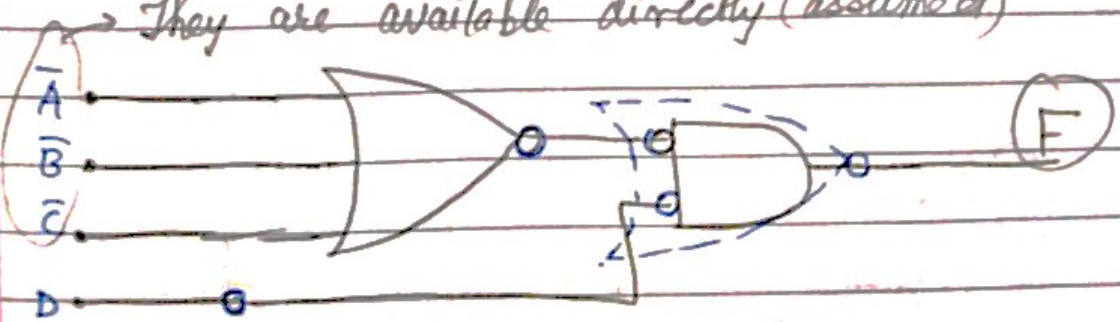
∵ grouped 0's

$\overline{F} = D + ABC$

$\Rightarrow F = \overline{D + ABC}$

$\Rightarrow F = \overline{D}.(\overline{A} + \overline{B} + \overline{C})$

→ They are available directly (assumed)



Q. Implement by two level (a) NAND - AND (c) OR - NAND
(b) AND - NOR (d) NOR - OR

$F(A,B,C,D) = \Sigma(0,4,8,9,10,11,12,14)$

(Group 1's) SOP form =

AB \ CD	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	0	0	1
10	1	1	1	1

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

Group (0's) POS form

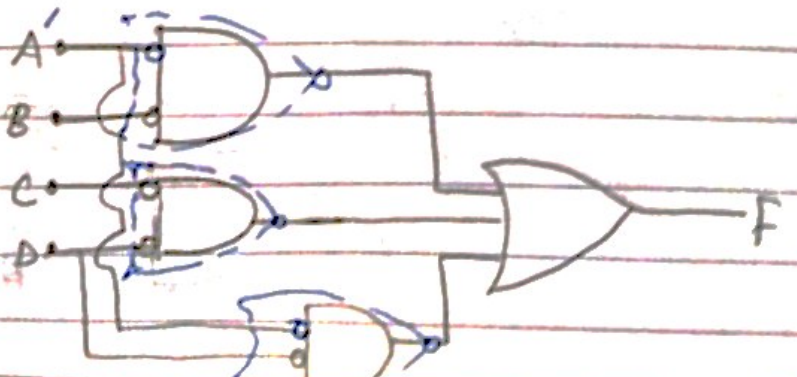
AB \ CD	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	1	0	0	1
10	1	1	1	1

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

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

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

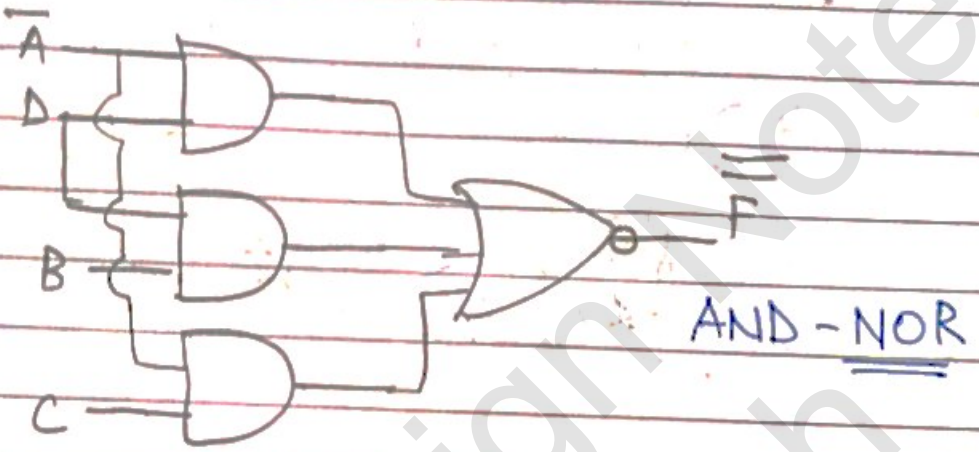
(d) NOR-OR, Using $F = \bar{C}\bar{D} + A\bar{B} + A\bar{D}$



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

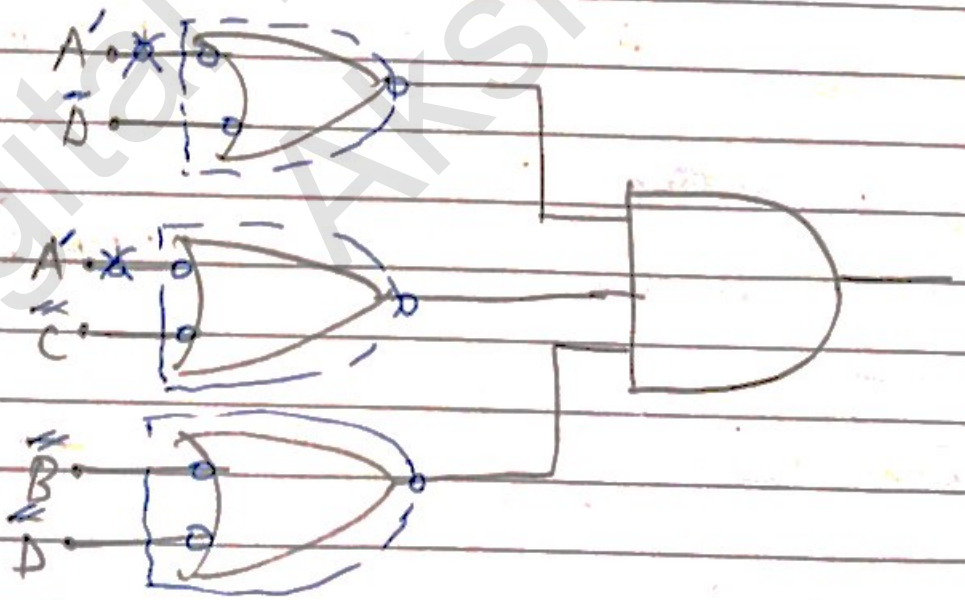
↳ Implementing this

(b)



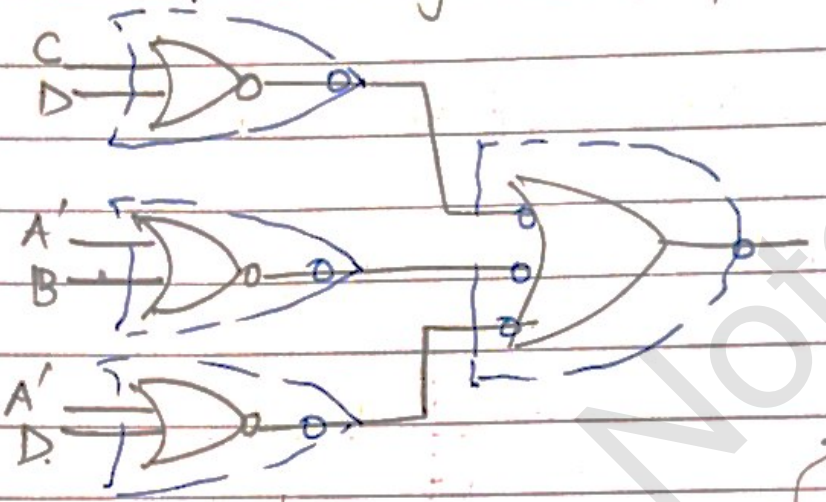
AND - NOR

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



* Note: part (a) can also be done by using the circuit of part (b), like its done in part (c) (next page)

(c) OR-NAND (Using the (d) part)



Sum & Carry
XOR AND

2 i/p 2 o/p.

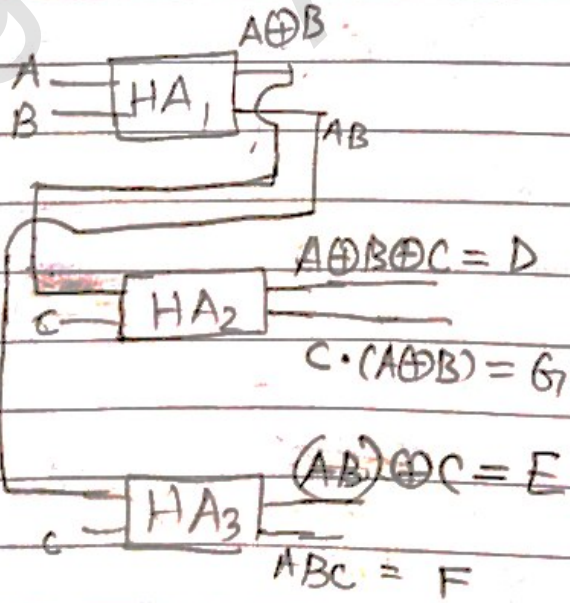
Q. Given 3 half adders. Implement these 4 fns in a SINGLE circuit

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

(2) $E = ABC' + (A'+B')C = (AB) \oplus C$; $ABC' + (\overline{AB})C$; $(A+B)C$

(3) $F = ABC$

(4) $G_1 = A'BC + AB'C = C(A'B + AB') = C(A \oplus B)$



Q $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

Find reduced SOP expression.

wx \ yz	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	0	1	1	0
10	1	0	0	1

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

$\bar{w}x$ or $\bar{w}\bar{z}$

$xz + \bar{z}\bar{x}$ → essential prime implicants (terms which will be there definitely)

2 solns :- $F = \bar{w}x + xz + \bar{z}\bar{x}$
$= \bar{w}\bar{z} + xz + \bar{z}\bar{x}$

Same no. of literals. Both are optimum soln's.

So, ∃ many soln's (no problem : It's possible.)

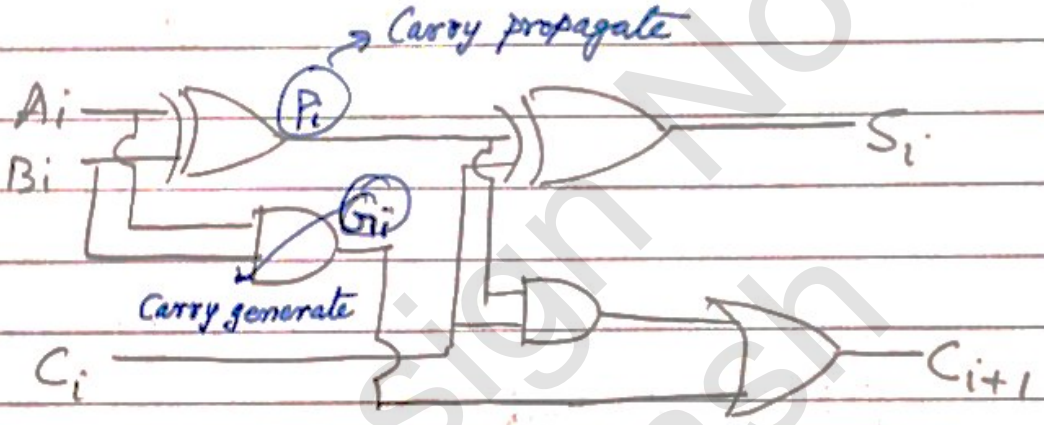
★ Prime Implicants : Each of product terms arising from K maps.

★ sth like local maxima/minima for a f^n $f(x)$. There can be many. Absolute maxima doesn't exist in this optimum soln of KMAP.

PARALLEL

✱ 4 BIT BINARY ADDER: Increasing circuit speed :-

- See the circuit made previously.
- Delay is being caused by each adder circuit to produce carry
- If we produce carry separately, then, that delay time is saved.



$P_i = A_i \oplus B_i$; $G_i = (A_i)\cdot(B_i)$

$S_i = P_i \oplus C_i$; $C_{i+1} = G_i + P_i \cdot C_i$.

These are the 4 carry bit generated separately

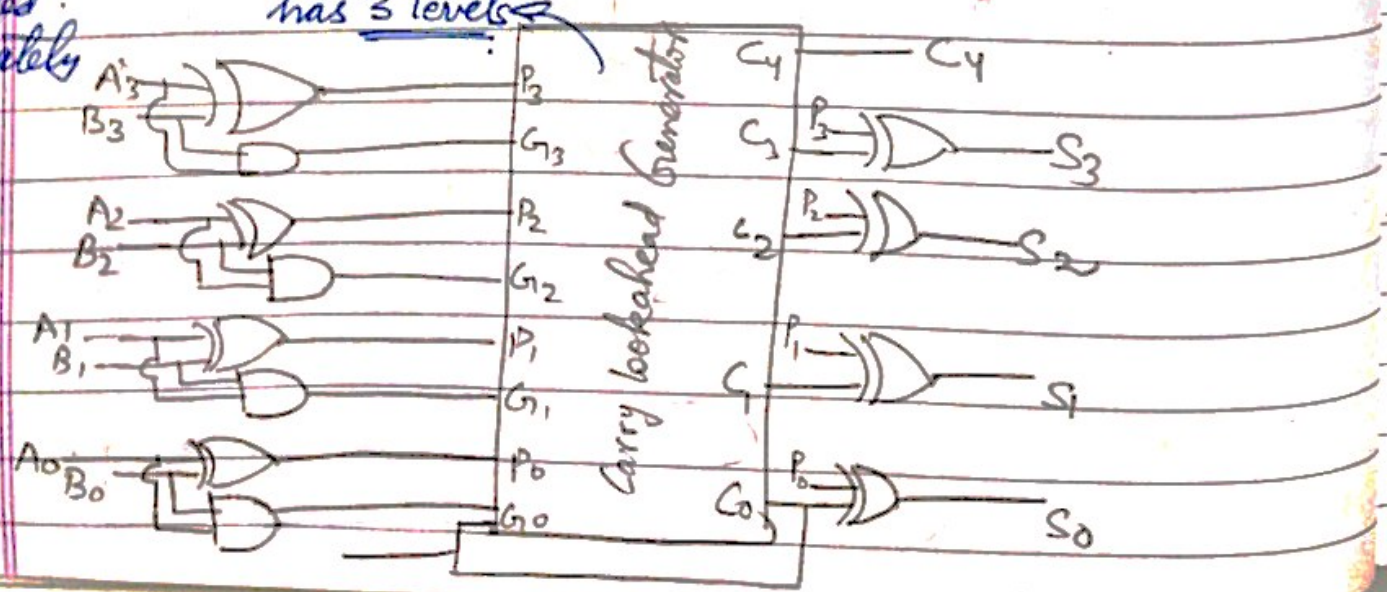
C_0 = input carry :- Put i = 0

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

has 3 levels



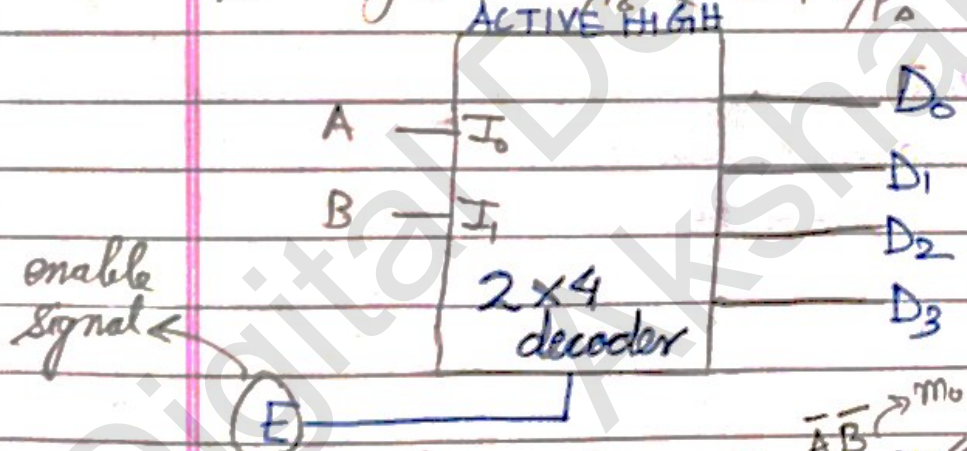
✓ circuit has 5 levels. The original circuit we used had 9 levels. (3 for each of the first 3 full adders).

So, speed ↑.

Decoders

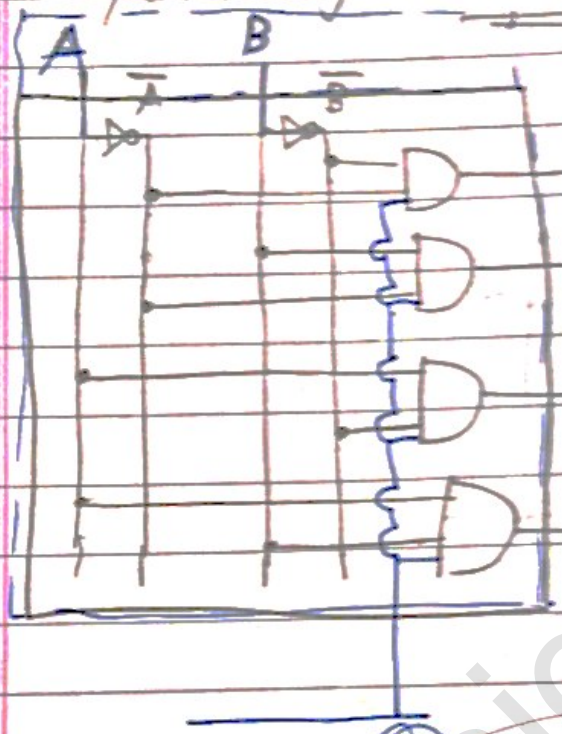
* Takes n i/p's & gives out 2^n o/p's.

* for 2 i/p's → 4 o/p's



A	B	D_0 $\bar{A}\bar{B}$ → m_0	D_1 $=\bar{A}B$	D_2 $=A\bar{B}$	$D_3 = AB$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

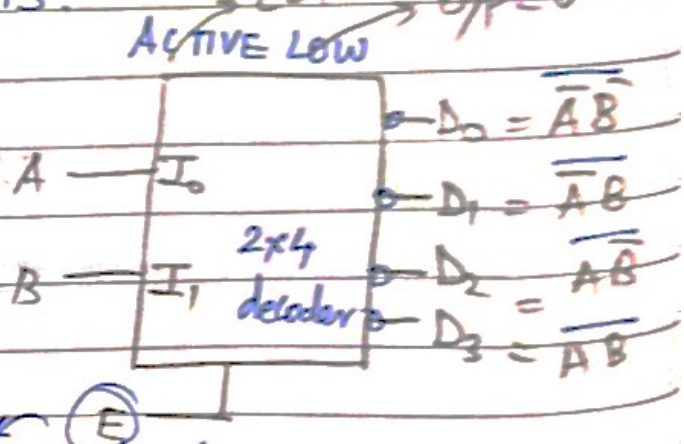
Implementing 2×4 decoder



gives MINTERMS.

→ when it high, it enabling circuit.

* Note :- If the "implement" is done using NAND gate (instead of AND), we get MAX TERMS.



Corresponding circuit

We will get low o/p when it is activated

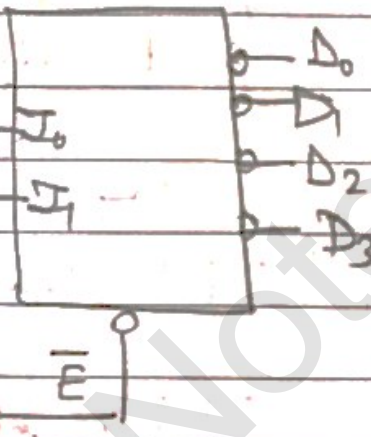
It is activating

A	B	D_0	D_1	D_2	D_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

* Enable pin :- If it's low ($E = 0$), we use $\bar{E}$ as i/p.

Invalid o/p → any value.

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



* Ily for 3 x 8 decoder.

A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

* make circuit accordingly.

✱ AIM: Add 2 7-bit nos.

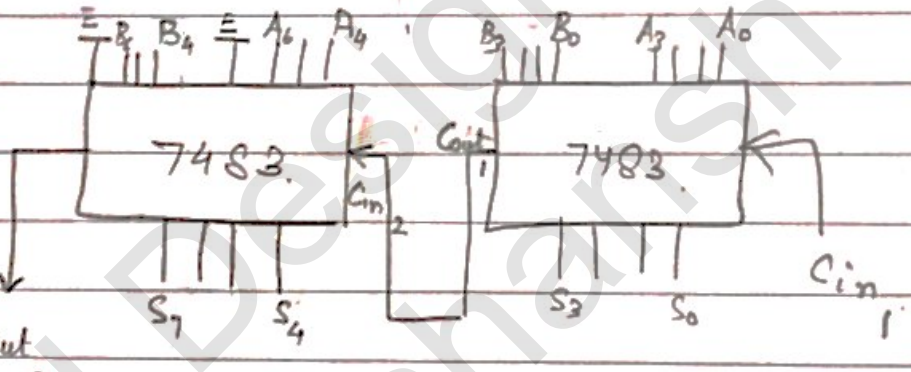
Do using std. circuits studied

A $A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$

B $B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$

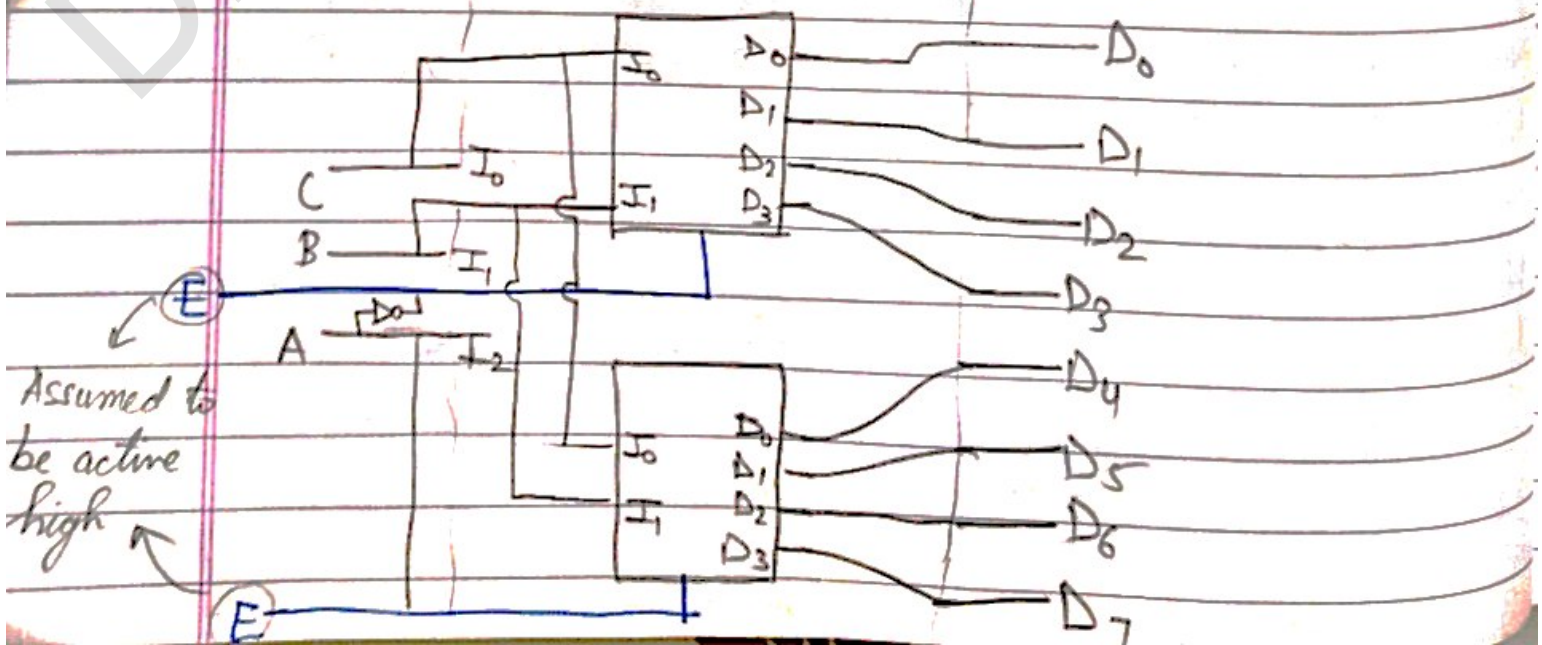
eg: 1101110 +
1011110

A	$A_6\ A_5\ A_4$	$A_3\ A_2\ A_1\ A_0$
B	$B_6\ B_5\ B_4$	$B_3\ B_2\ B_1\ B_0$



3 bits → MSB A B C LSB

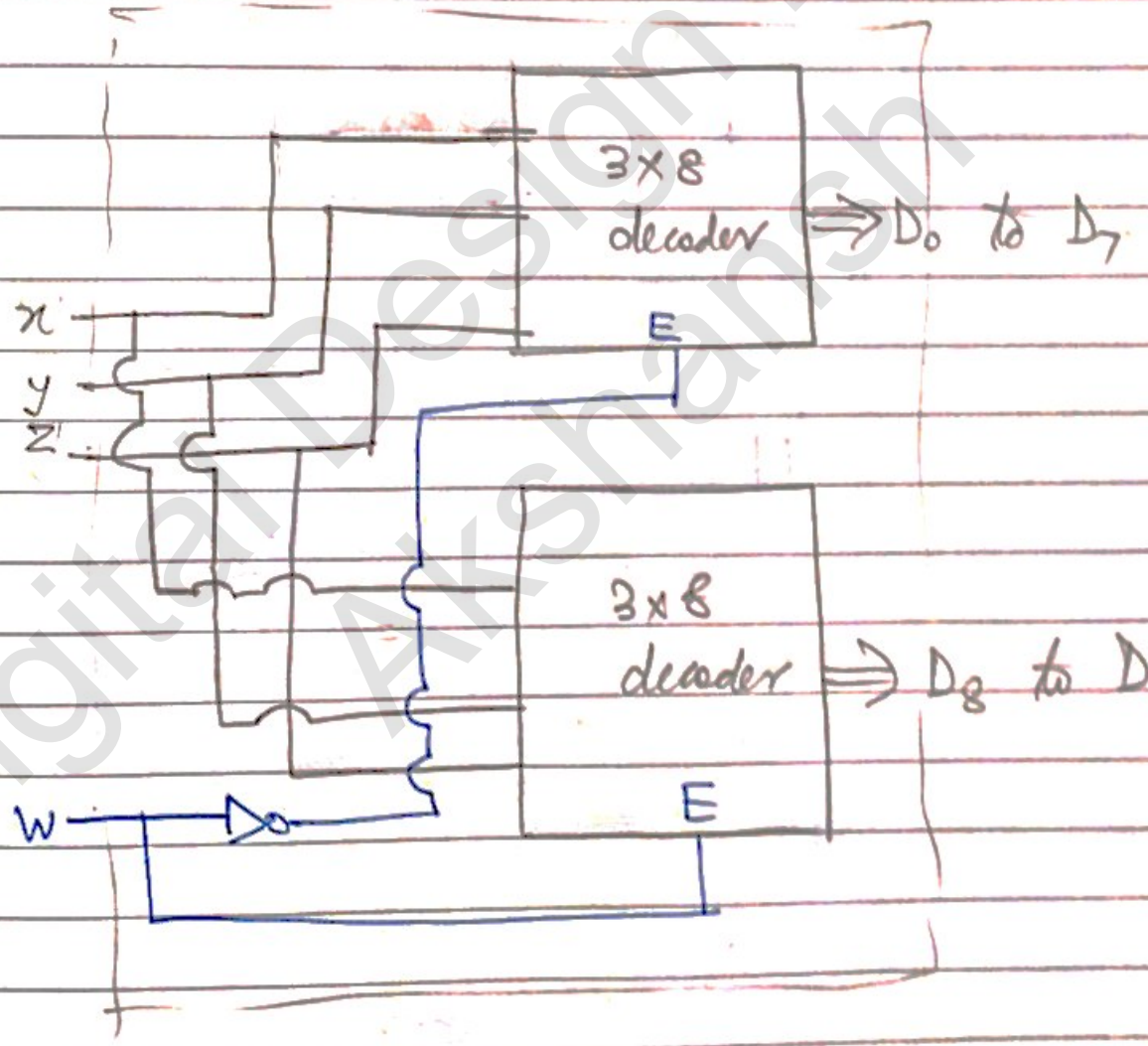
✱ AIM:- Make 3×8 decoder using 2×4 decoder



☆ A is taken as enable pin

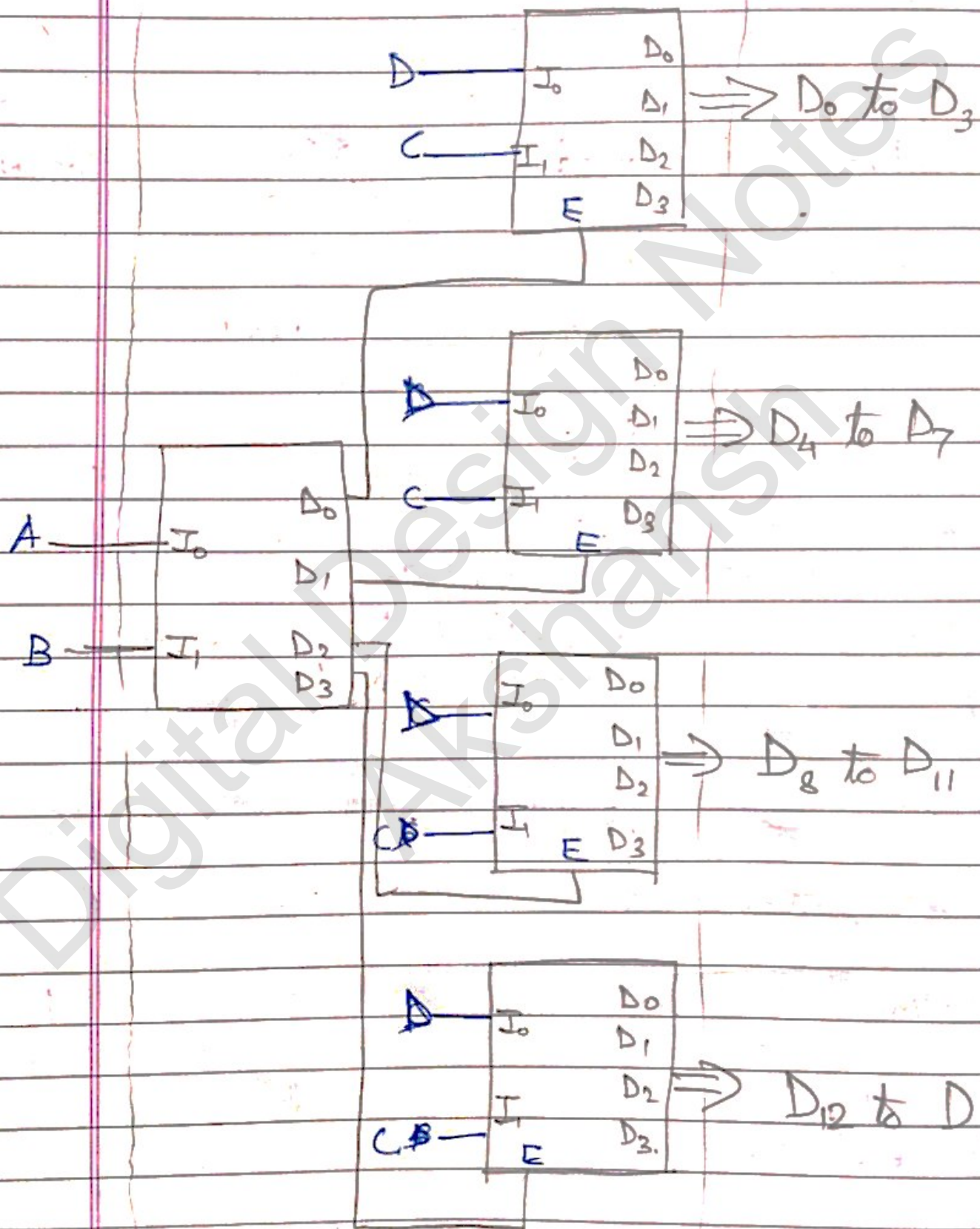
- When A=0 ⇒ enable pin is activated for upper IC.
  When A=1 ⇒ enable pin is activated for lower IC

☆ AIM: Make 4×16 decoder using 3×8 decoder



Date
Page

✡ AIM: Make 4x16 decoder, using 2x4 decoders.



✱ Function Implementation using decoders.

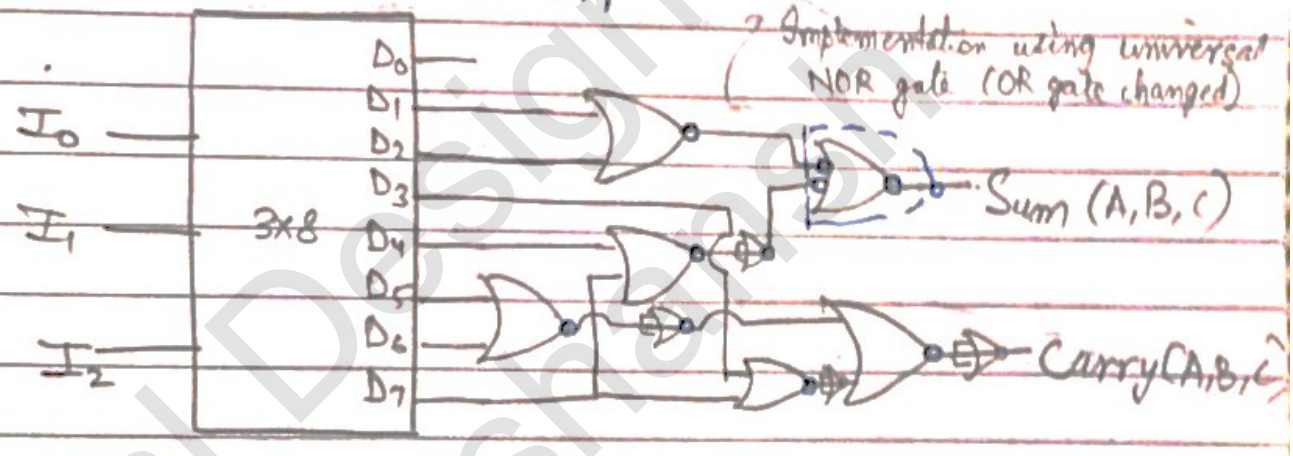
- Using adder circuit

$S(A,B,C) = A \oplus B \oplus C = \sum(1,2,4,7) = m_1 + m_2 + m_4 + m_7$

$Carry(A,B,C) = AB + BC + AC = \sum(3,5,6,7)$

$= m_3 + m_5 + m_6 + m_7$

NOT OPTIMUM



Implementation using universal NOR gate (OR gate changed)

Idea Implementation :- I have to make sum f^n, say $(\sum(1,2,4,7))$ as $D_1 + D_2 + D_4 + D_7$. So, use OR gates. Why, for Carry f^n.

✱ Note :- D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7

• Active high.	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
• Active low	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7

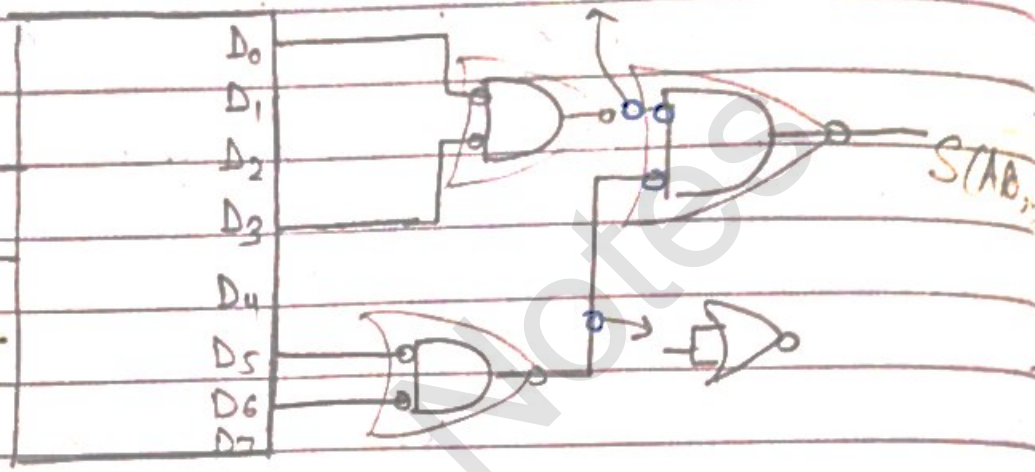
✱ For using only NAND / NOR gate for implement^n, use maxterms. i.e, complement each o/p & use.

Optimum solution.

$S = \Pi(0, 3, 5, 6)$

$C = \Pi(0, 1, 2, 4)$

2 i/p NOR gate alone



↓ Implementⁿ of Sum fⁿ using
- ✓ Active high
- ✓ NOR gate

Illy. do for NAND gate.

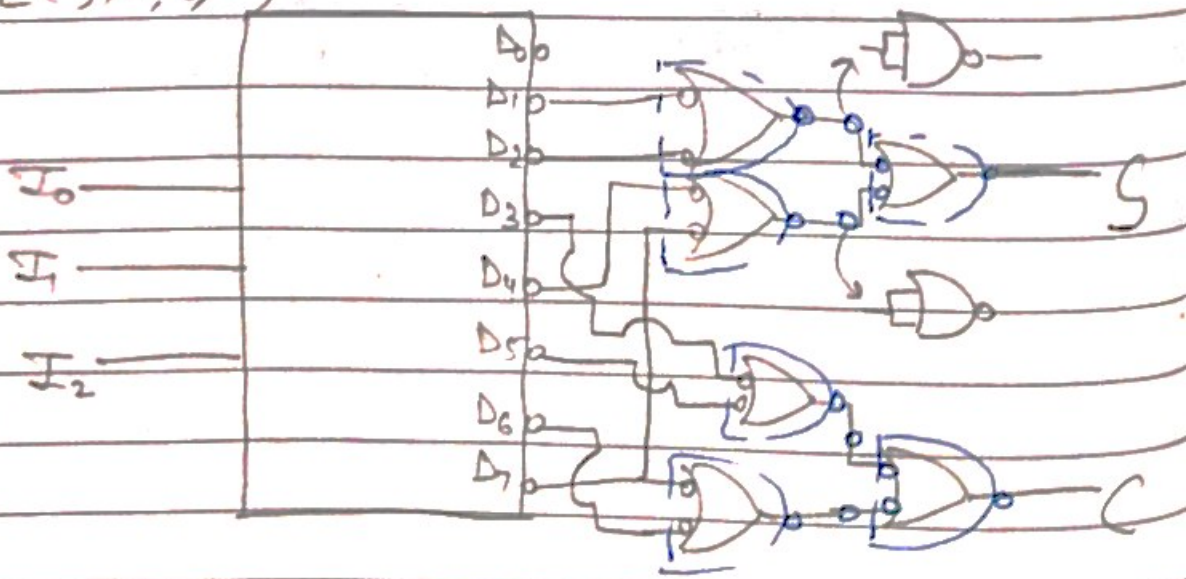
Now, Sum fⁿ implementation

Given ACTIVE LOW

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

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

2 i/p NAND gate alone



Puffin
Date
Page

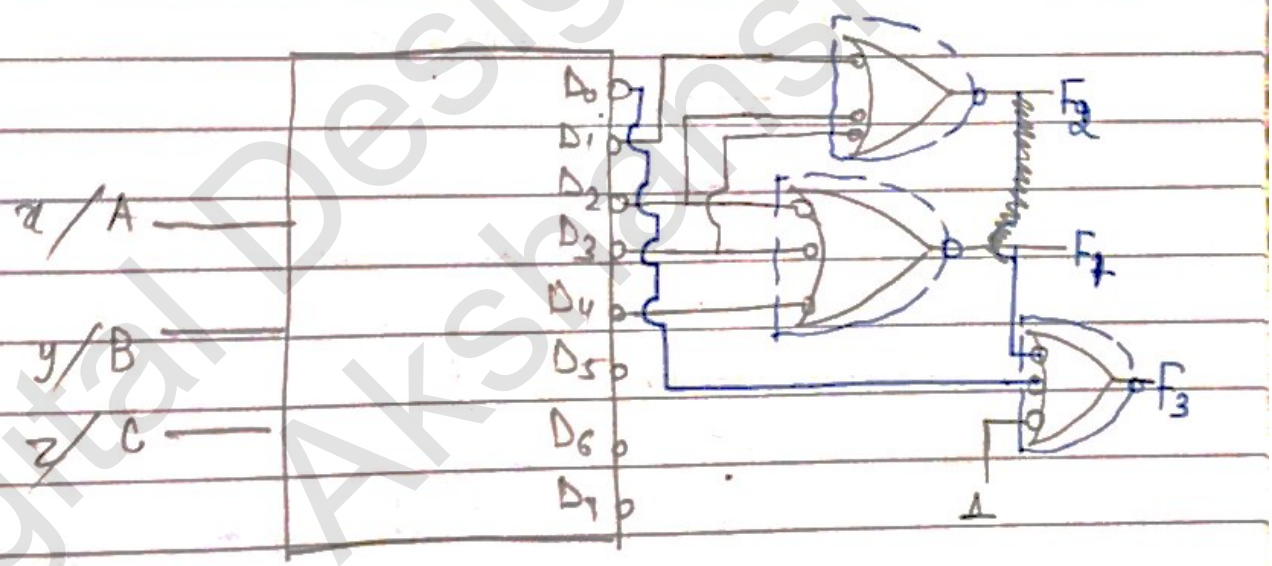
Q. $F_1(A,B,C) = \Pi(0,1,5,6,7) = \Sigma(2,3,4)$

$F_2(A,B,C) = \Sigma(1,2,3)$

$F_3(x,y,z) = \bar{x}y + \bar{y}\bar{z} = \Sigma(0,2,3,4)$
$= \Pi(1,5,6,7)$

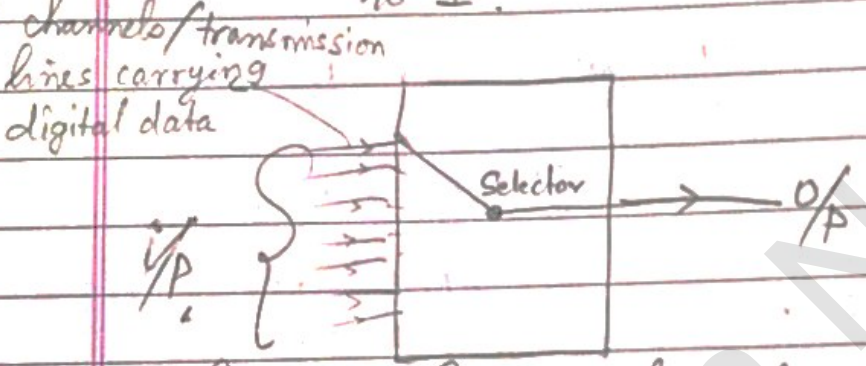
Implement given fn using

- ✓ Active low
- ✓ 3 x 8 decoder
- ✓ 3 i/p
- ✓ AND, OR, NAND gates to be used only.
- ✓ Use ONLY 3 logic gates



Multiplexer / Demultiplexer

→ Many multiplexed to 1.



how to select 1 channel out of (2^n) — under std. case

* 1 K = 2^{10} = 1024

1 K X 8 ⇒ each space is 8 bit in size ⇒ called as words.

∴ each of the 1024 words would be 8 bit in size

Memory location of 1 K
↳ 1024 locations having data (in form of 0's & 1's)

A comp. sys. has

- Address (wires) (bus → group of electrical line) : bus carrying address values
- Data : " " data "
- Control : " " control of info

To represent the 1024 addresses (from 0 - 1023), we require 10 bits (lines) so, we need 10 address

Total data available to me = 1 K x (8 bit) of memory
= 1 KB of memory

2^{10} : kilo
2^{20} : mega
2^{30} : giga

10 : address lines
20 : "
30 : "

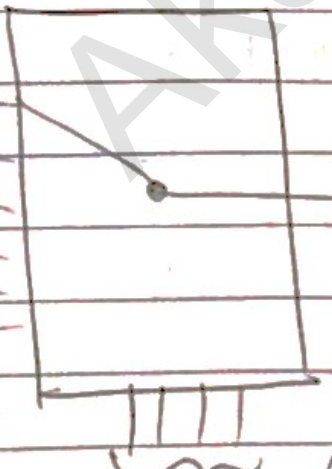
* Here, for memory location for 1K x (8)

we have taken 8 bits space for each address.

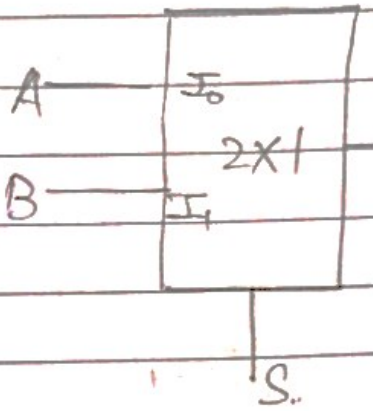
For 2^n data lines ⇒ we need n address lines

In multiplexer :- address lines are called as select lines

So, a multiplexer will have map of 2^n data lines & n address lines. Out of that 1 data info is taken to o/p.



✡ 2×1 multiplexer



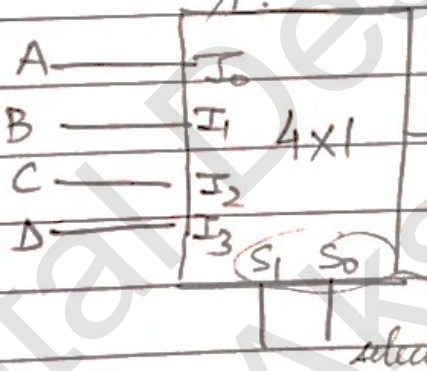
$Y = A\bar{S} + BS$

For 2×1 → If $S=0$ ⇝ I_0 is o/p
$S=1$ ⇝ I_1 is o/p

✡ (4×1) multiplexer

4 i/p's

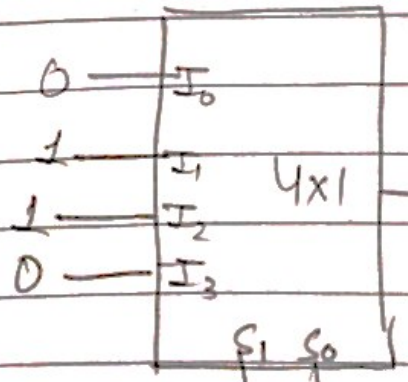
→ 1 o/p.



$Y = \bar{S}_0\bar{S}_1A + S_0\bar{S}_1B + \bar{S}_0S_1C + S_0S_1D$

I want to represent each i/p using 2 select lines. So, basically, generate 4 pairs of select lines (00, 01, 10 or 11) & link to each i/p. (≅ 2×4 decoder)

✡ Implement Sum f^n of ½ adder using multiplexer.



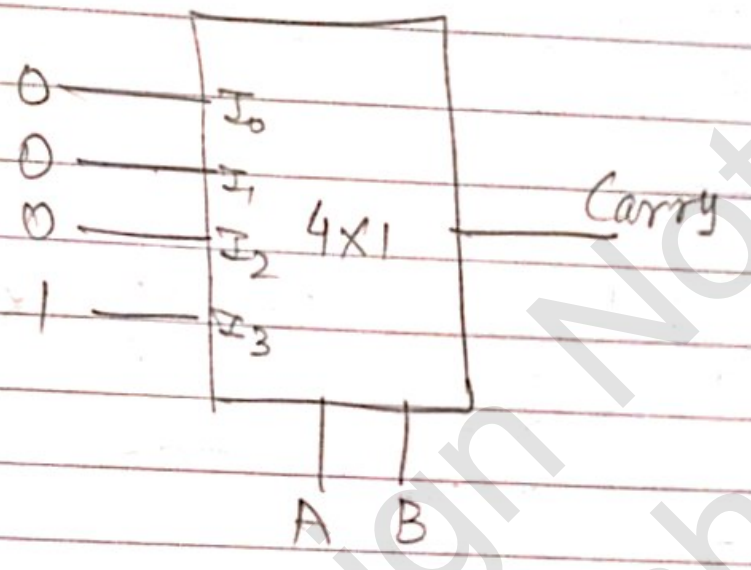
$Sum = \bar{S}_0\bar{S}_1(0) + S_0\bar{S}_1(1) + \bar{S}_0S_1(1) + S_0S_1(0)$ → 0

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

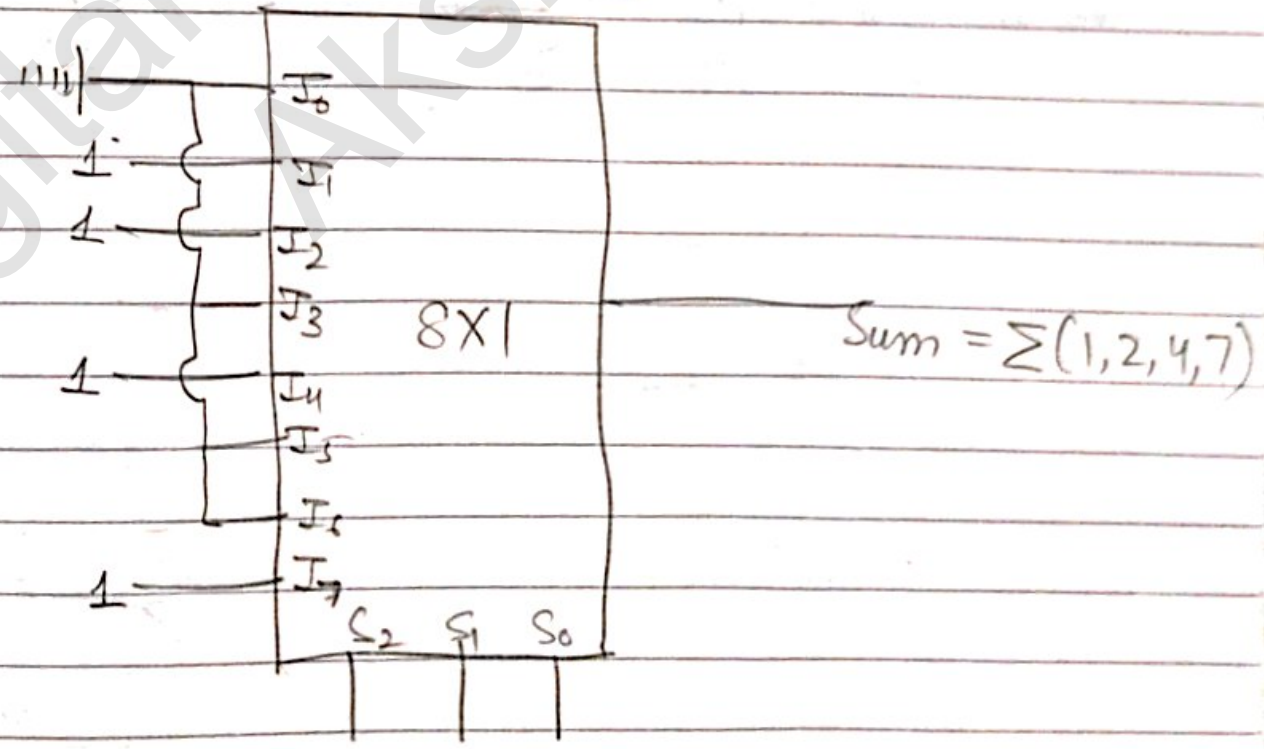
$= \bar{A}B + A\bar{B} = A \oplus B$

= Sum f^n

★ Implement carry fn of ½ adder using multiplexer.

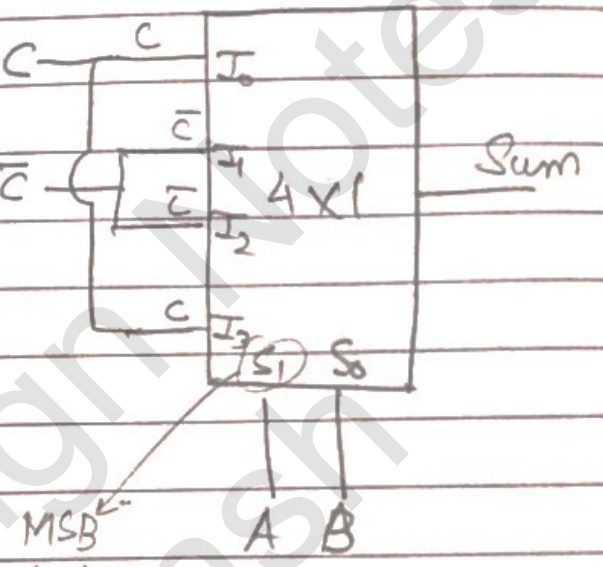


★ For implementing full adder sum using multiplexer ⟶ 8×1



✡ Implement a full adder using two 4×1 multiplexers.

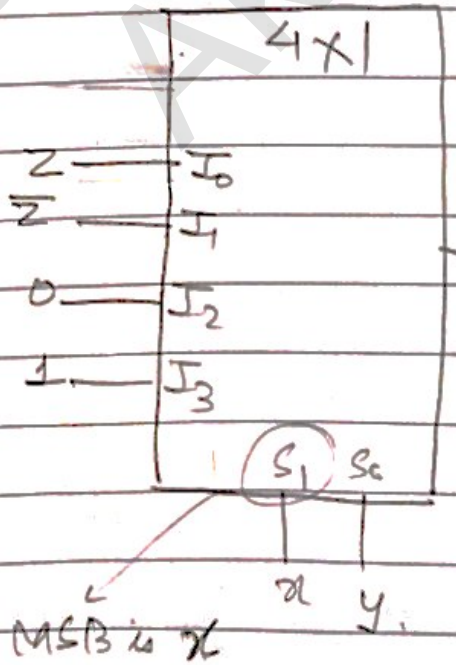
A	B	C	F	
0	0	0	0	
0	0	1	1	C
0	1	0	1	
0	1	1	0	$\bar{C}$
1	0	0	1	
1	0	1	0	$\bar{C}$
1	1	0	0	
1	1	1	1	C



(Sequence matters here)

✡ Implement using ~~8×1~~ 4×1 multiplexer.

$F(x, y, z) = \sum(1, 2, 6, 7)$



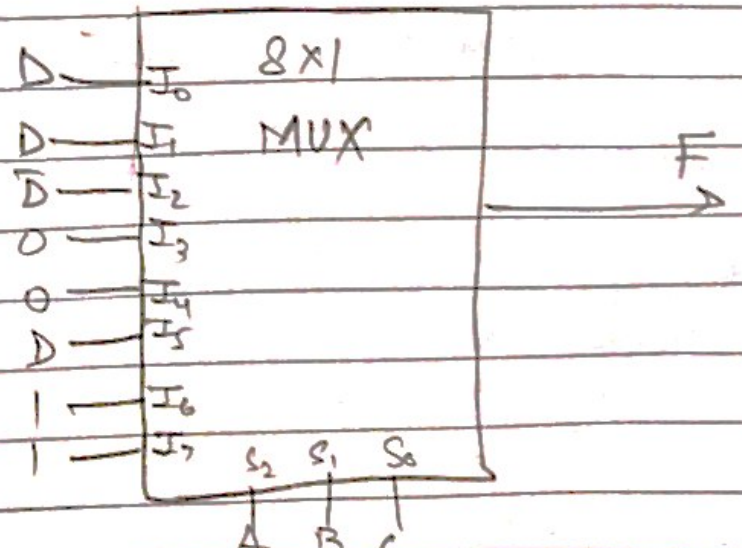
x	y	z	F	
0	0	0	0	z
0	0	1	1	
0	1	0	1	$\bar{z}$
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	

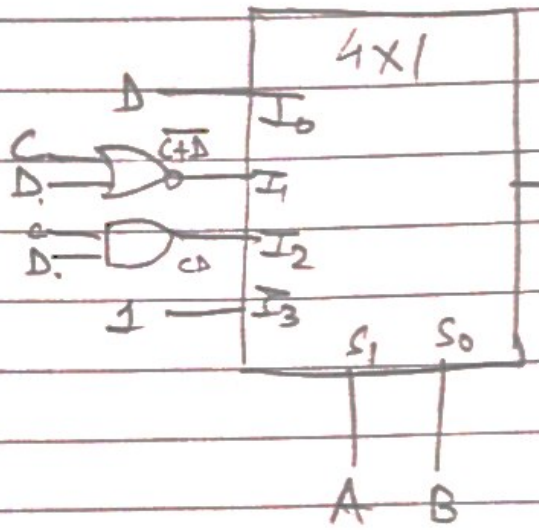
Puffin
Date
Page

✡ Implement using 8×1 multiplexer.

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

A	B	C	D	F	8×1	4×1
0	0	0	0	0	D	D
0	0	0	1	1		
0	0	1	0	0	D	
0	0	1	1	1		
0	1	0	0	1	$\bar{D}$	$\bar{C}\bar{D}$
0	1	0	1	0		
0	1	1	0	0	0	
0	1	1	1	0		
1	0	0	0	0	0	CD.
1	0	0	1	0		
1	0	1	0	0	D	
1	0	1	1	1		
1	1	0	0	1	1	1
1	1	0	1	1		
1	1	1	0	1	1	
1	1	1	1	1		





§ Encoders

↳ Take 2^n i/p, o/p → n.

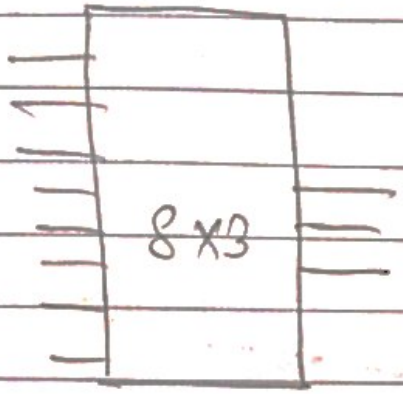
(opposite of decoder)

for n = 3

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

=> If D_4 goes high ⇒ corresponding o/p.

$m_4 \curvearrowright 100$



✱ x, y, z are my o/p fns in terms of D_0 to D_7

$x = D_4 + D_5 + D_6 + D_7$

fn. (∵ x is high when any one of them is high)

$y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$.

4 bit Priority encoder

↳ indicates the bit value corresponding the higher significant high bit in given word (eg $D_3 D_2 D_1 D_0$: D_3 is highest)

eg 0 1 0 1 : 0 is MSB.

But Most high significant bit = D_2 (∵ after D_3 (MSB), next high=1)

If I have 4 bits, I need 2 bits to represent them

eg: suppose → 0 0 D_0

0 1 D_1

1 0 D_2

1 1 D_3

Priority encoder

The code values

D_0	D_1	D_2	D_3	x	y	V (Valid bit)
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

fn table

Valid bit says whether i/p code is valid or not. It's valid when atleast 1 bit is high

→ when $D_3 = 0$, don't look at D_2, D_1 or D_0 & indicate $D_3 = 1$ by $x = 1$, $y = 1$

→ when all bits are low, we get $x = X, y = X$ (don't care)

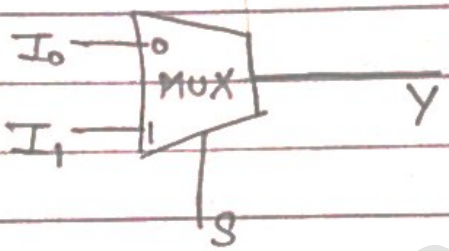
	D_3	D_2	D_1	D_0	X	Y	
d(0)	0	0	0	0	X	X	none high (Invalid o/p)
Σ(1)	0	0	0	1	0	0	D_0 = high
Σ(2,3)	0	0	1	X	0	1	D_1 = high
	0	0	1	X	0	1	
Σ(4,5, 6,7)	0	1	X	X	1	0	D_2 = 1 Others dont matter.
	0	1	X	X	1	0	
	0	1	X	X	1	0	
	0	1	X	X	1	0	
Σ(8,9, 10,11, 12,13, 14,15)	1	X	X	X	1	1	D_3 = 1. So, other terms don't matter. It 'll give o/p as 1 D_3: higher significant high bit
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	
	1	X	X	X	1	1	

Making f^n $X = d(0) + \Sigma(4,5,6\ldots,15)$

$Y = d(0) + \Sigma(2,3,8,9,\ldots 15)$

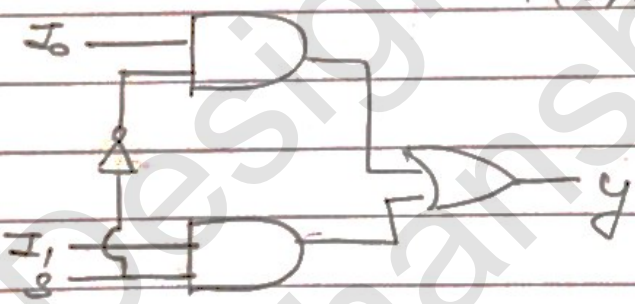
✱ Multiplexer Implementation

Symbolic Representation of Multiplexer

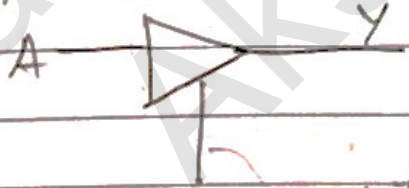


2×1 MUX

≡



✱ Buffer → stores data



(Tristated buffer)

→ activated with high i/p

→ control i/p

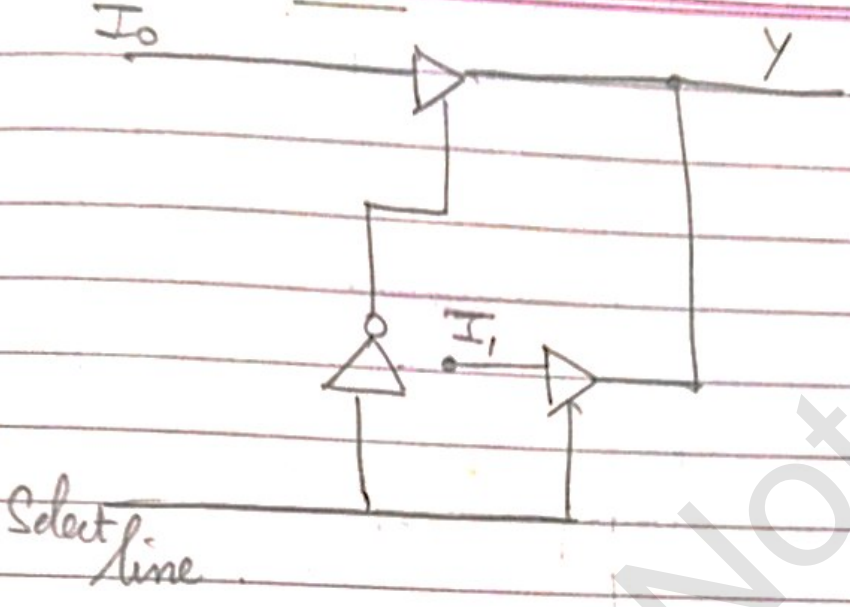
→ $Y = A$ when $C = 1$

$Y = Z$ when $C = 0$

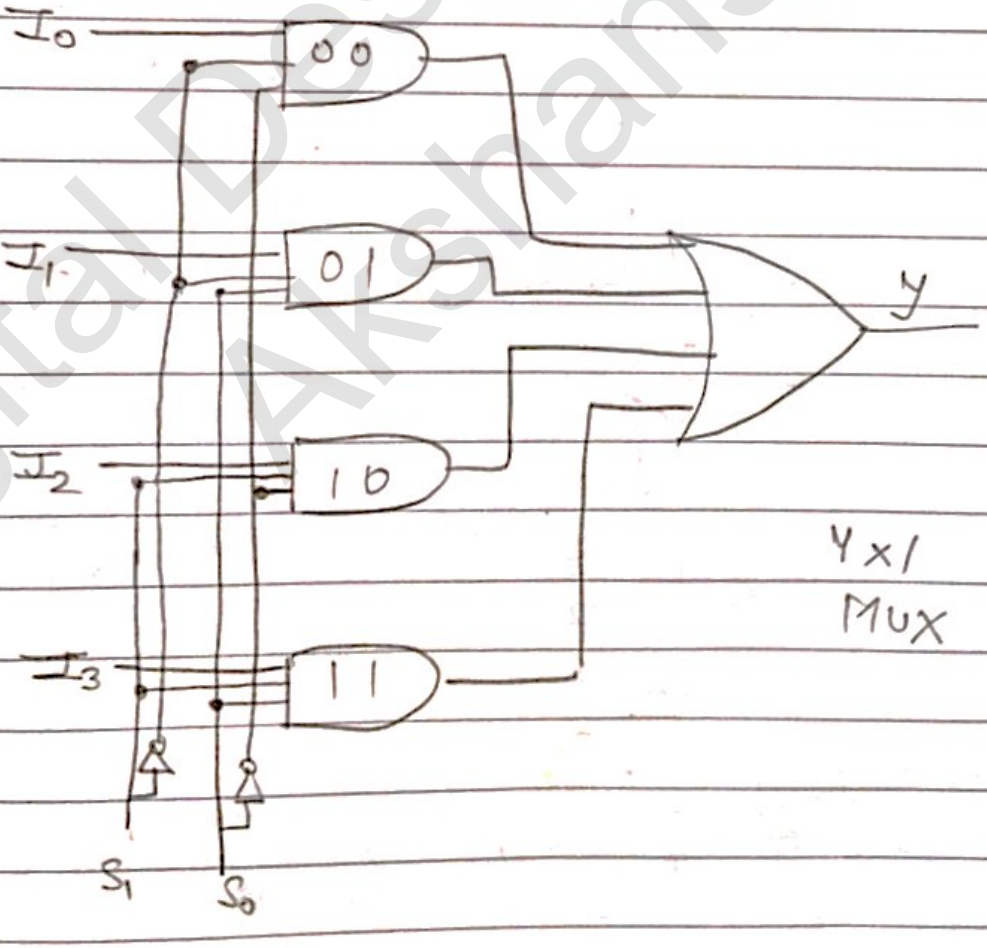
→ basically, the circuit didn't work.

or, a state reached ⇒ high impedance state

TRISTATED LOGIC

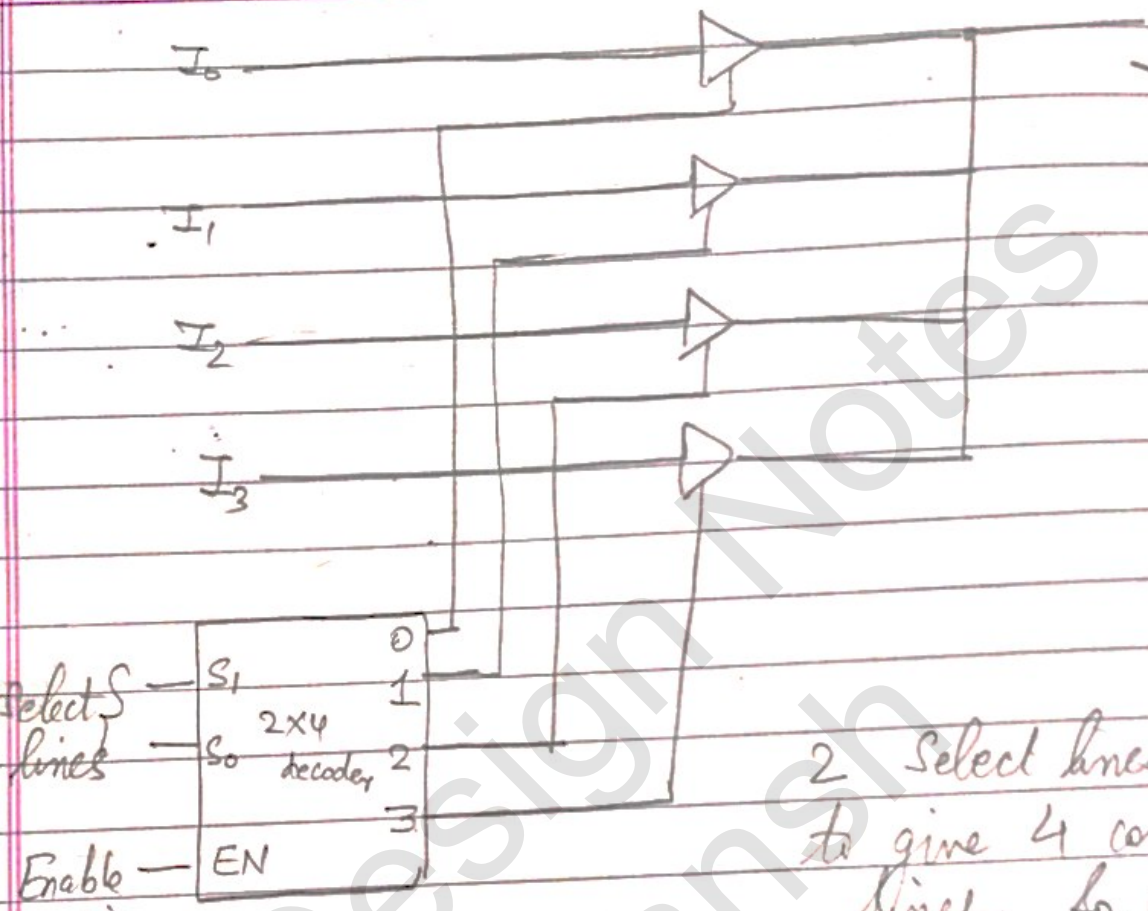


✡ 4×1 multiplexer



4×1 MUX

✡ Tristated logic using buffer & decoder.

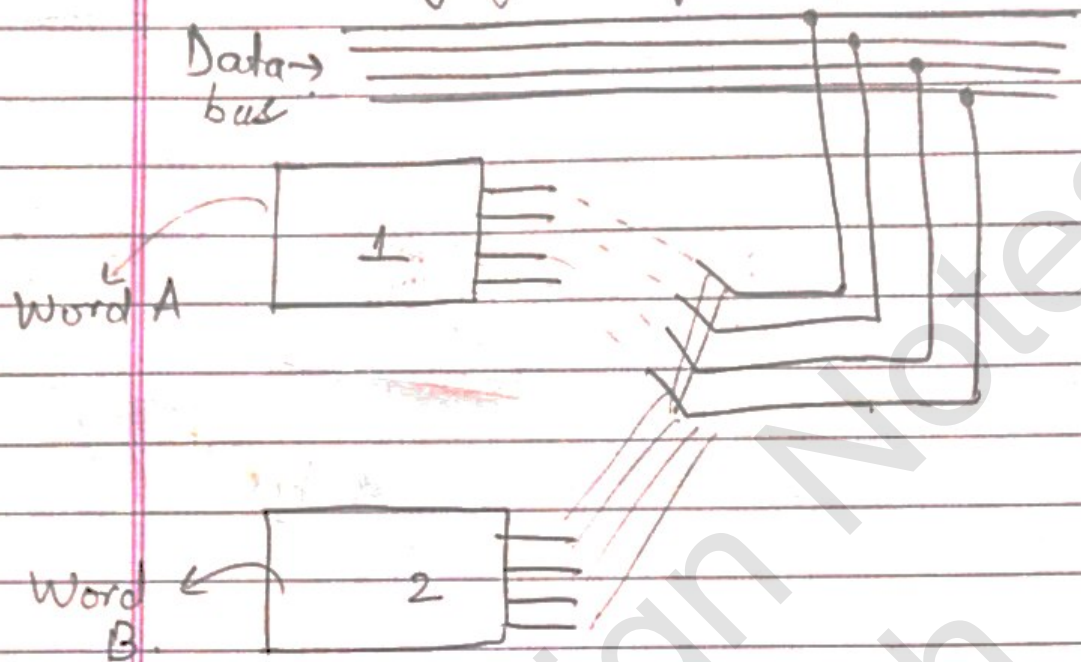


2 Select lines have to give 4 control lines. So, we used decoder.

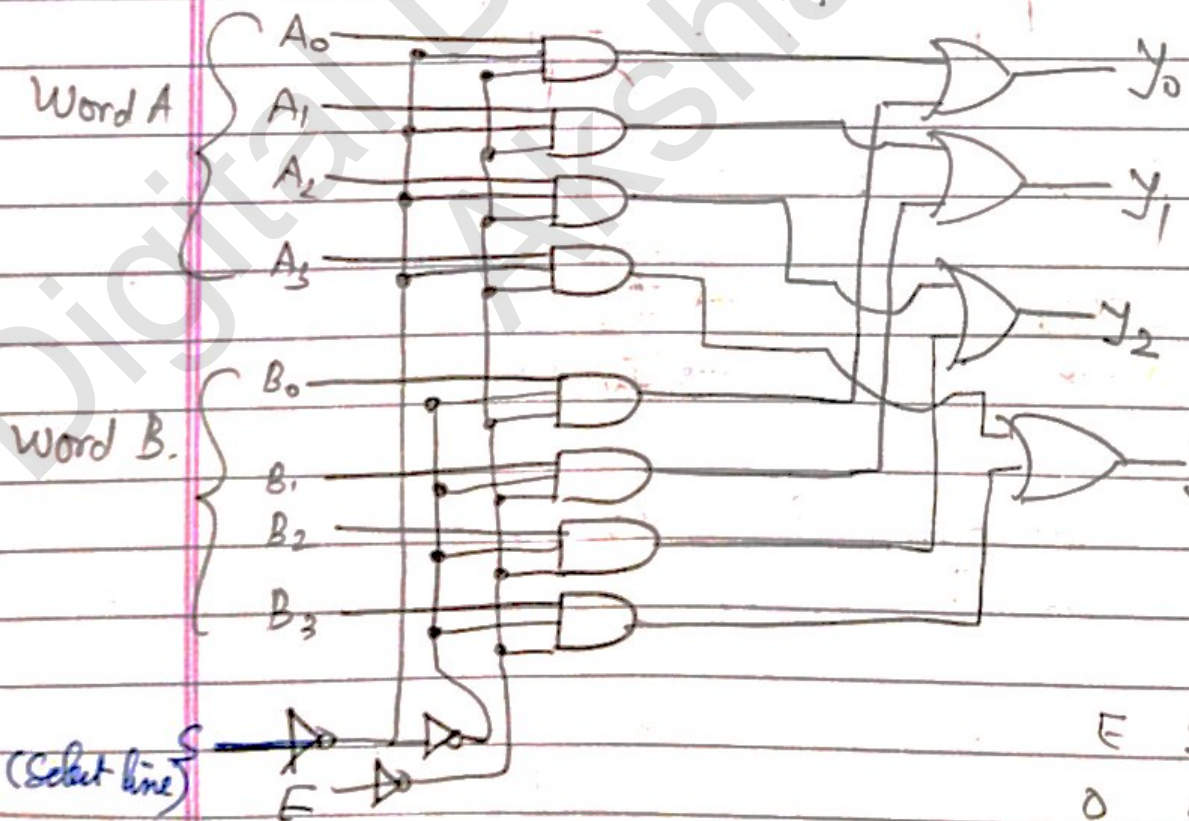
✡ 8/16/32/64 bit sys. (computer)
Size of data (no. of bit) the processor can handle at a time.

✡ For 1 K → 10 address lines
Due to limit^n of address lines, I have limit^n on memory.
✡ For 1 Mega → I need 20 address lines

✱ Selecting from group of conductors

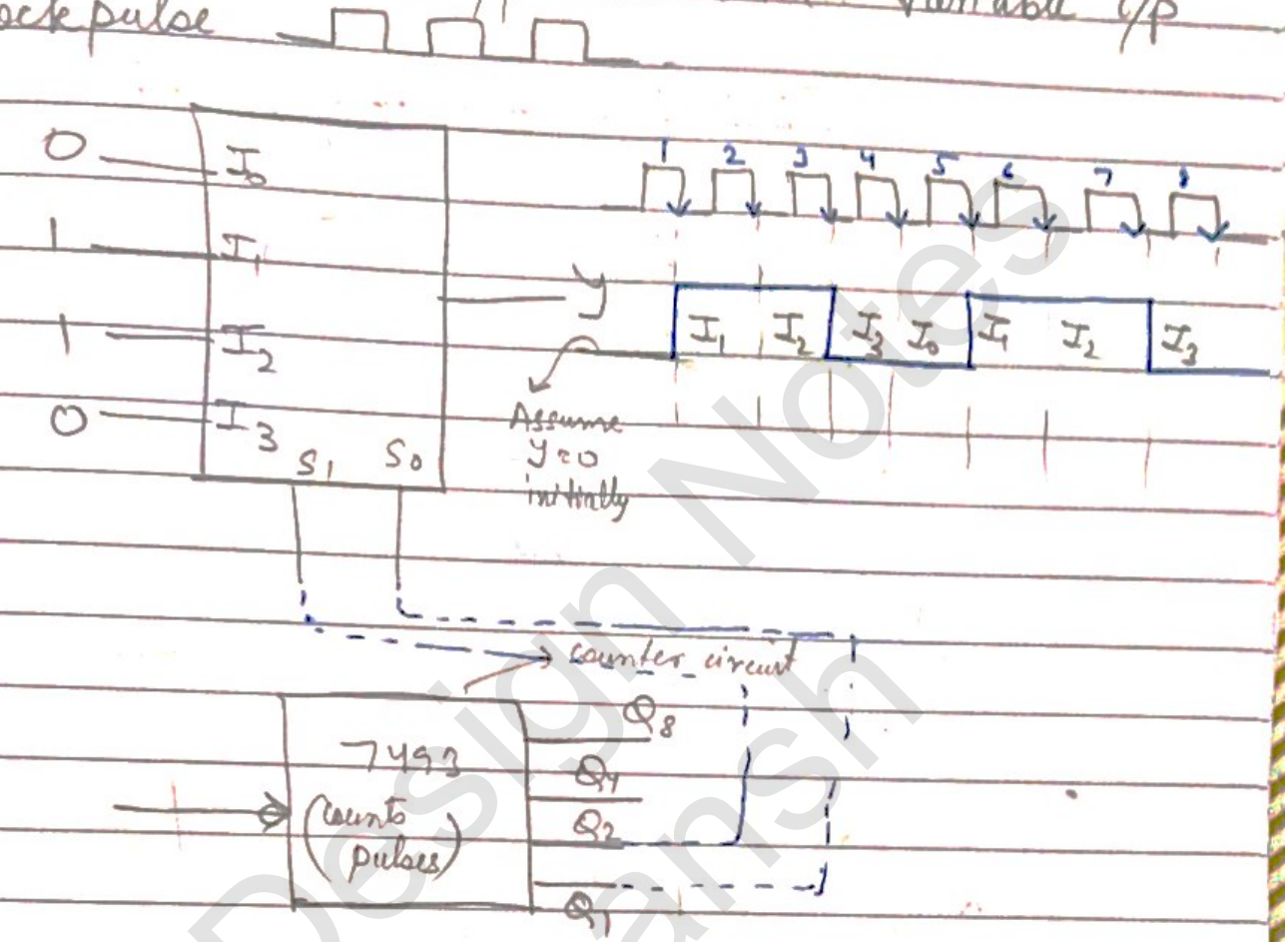


Choose any one of 2 words using multiplexer



E	S	O/P
0	0	A
0	1	B
1	X	0's

Q What will be o/p 'y' w.r.t variable i/p clock pulse



when 1st clock pulse comes, it will give o/p 0 0 0 1 ($Q_4 Q_3 Q_2 Q_1$)

				Q_1	S_1 S_0	
1st clock pulse	→ I_1	1	0 0	0	1	I_1
2nd	→ I_2	1		1	0	I_2
3rd	→ I_3	0		1	1	I_3
4th	→ I_4	0	1	0	0	I_0
			1	0	1	I_1
			1	1	0	I_2
			1	1	1	I_3
			1	0	0	I_0

eg: Let select line i/ps be

$5 = I_1$ (1 0 1) → Q_2 Q_1 = 1 1 ⇒ S_1 S_0 = 0 1 ⇒ I_1

$1 = I_1$

$2 = I_2$

$1 = I_1$

$0 = I_0$

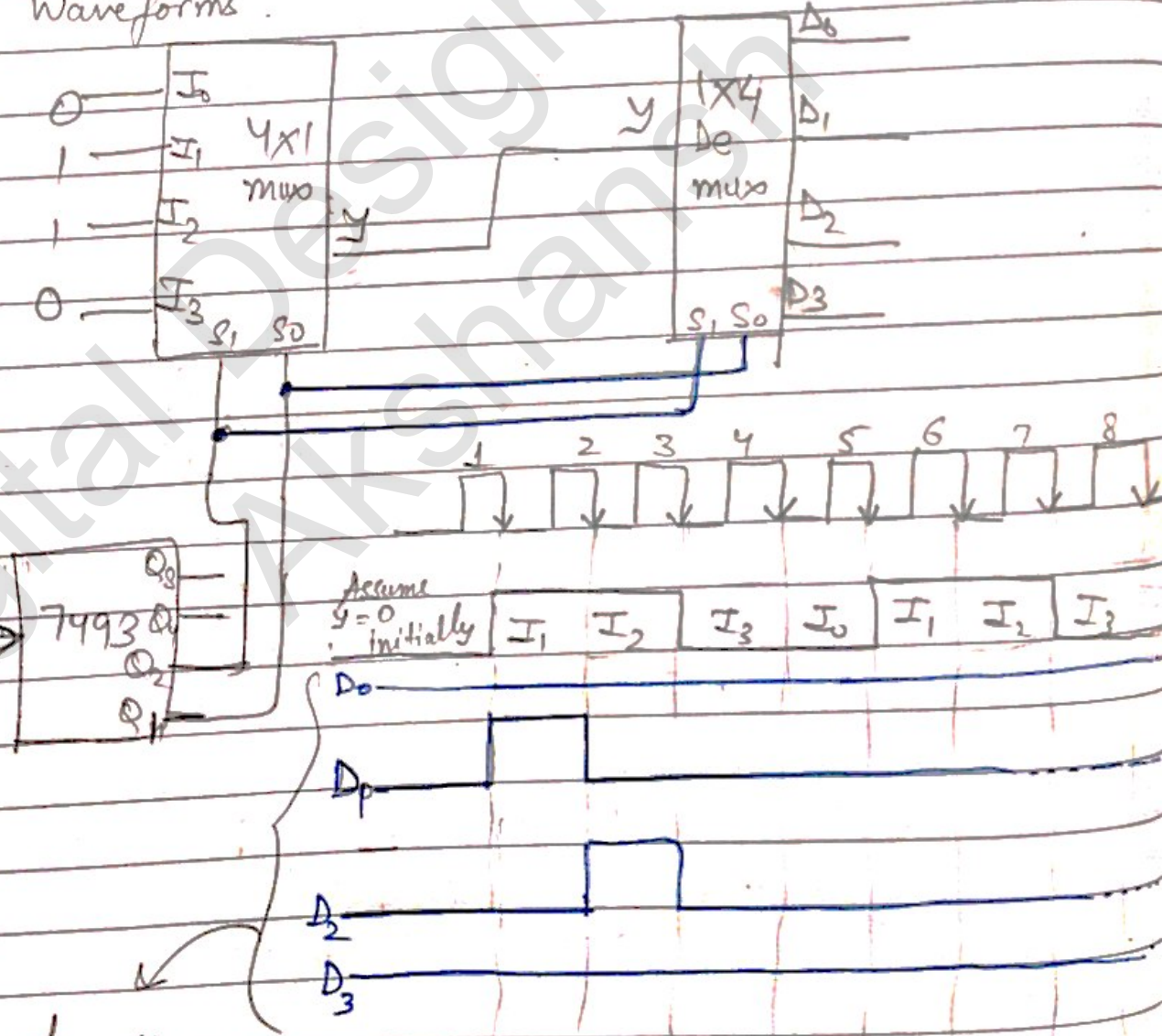
Q Why connect to Q_2 & Q_4. Select line i/ps change.

☆ In previous circuit, i/p of multiplexer is constt.

Select lines are varying ctly with time.

So, time varying multiplexing is being done

☆ Now, consider a demux circuit attached. Find o/p waveforms.



demultiplexing waveforms.

☆ Using decoder as a demultiplexer

In decoder (3×8)

composed of two 2×4 decoders

A (MSB) B C (LSB)

i/p

i/p connected to enable pin

o/p gives out 8 fns s.t each fn is high only at specific i/p

* It has
  - 3 i/p bits :
    - ↳ MSB connected to EN
  - & 1 enable bit (EN)

Change →

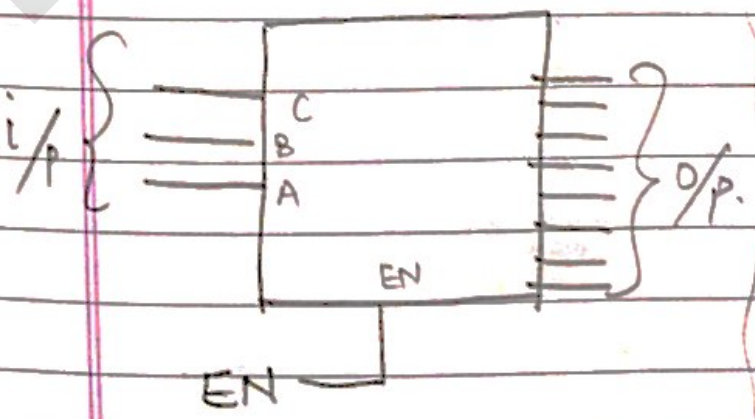
In demux (1×8)

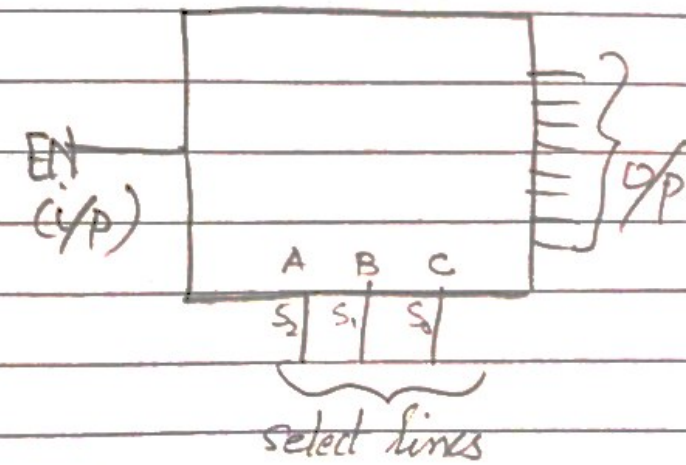
A (MSB) B C (LSB)

↳ select lines

Enable bit : It can be said as a fn which can be high or low. That means an IC giving output as fns (high only at one specific value) can be used as enable bit

* It has
  - 3 select lines
    - ↳ MSB connected to EN
  - & 1 EN is i/p





Puffin
Date 31/10/12
Page

MAGNITUDE COMPARATOR

* MSI → Arithmetic circuits
  - → Adder
  - → Substractor
  - → Multiplier (left)
* → Converter circuits
  - → decoder
  - → encoder
  - → mux
  - → demux

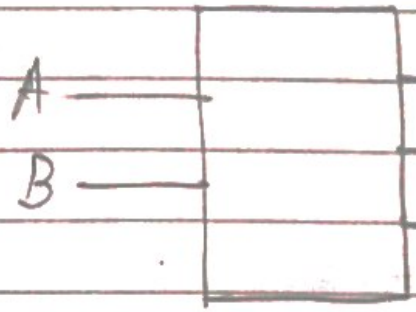
  (done so far)
* → Comparaters
  - → Single bit
  - → Multiple bit

☆ Comparing

3 expected results by comparing a & b

① $a = b$, $a \neq b$

$a > b$ ② $a < b$ ③



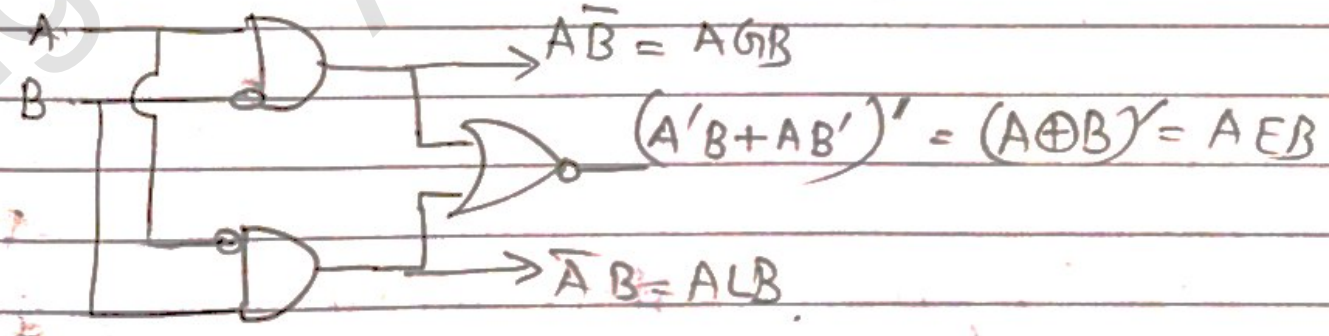
A G B : A > B
A E B : A = B
A L B : A < B

☆ Considering 1 BIT WORDS of A & B

A G B ⇒ A = 1, B = 0
⇒ A B̄

A L B = Ā B

A E B = AB + Ā B̄ = (A ⊕ B)′
= A ⊙ B
= A′ ⊙ B
= A ⊕ B′



Puffin
Date
Page

Considering 2 BIT WORDS

$\Rightarrow A \rightarrow A_1 A_0 \quad B \rightarrow B_1 B_0$

$A > B = A_1 \overline{B_1} + x_1 A_0 \overline{B_0}$

$A < B = \overline{A_1} B_1 + x_1 \overline{A_0} B_0$

$A = B = x_1 \cdot x_0$

Assume, for any bit, $A_i = B_i = x_i$

i.e $A_1 = B_1 = x_1$ - - -

x is equality = 1 : if equal

0 : not equal

4 BIT WORD

$A \rightarrow A_3 A_2 A_1 A_0 \quad B \rightarrow B_3 B_2 B_1 B_0$

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

$A = B = x_3 x_2 x_1 x_0$

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

i.e, comparing MSBs of both words.

If MSB is same then comparing 2nd bits. - - - Ily others

eg: A: 4 3 2 1

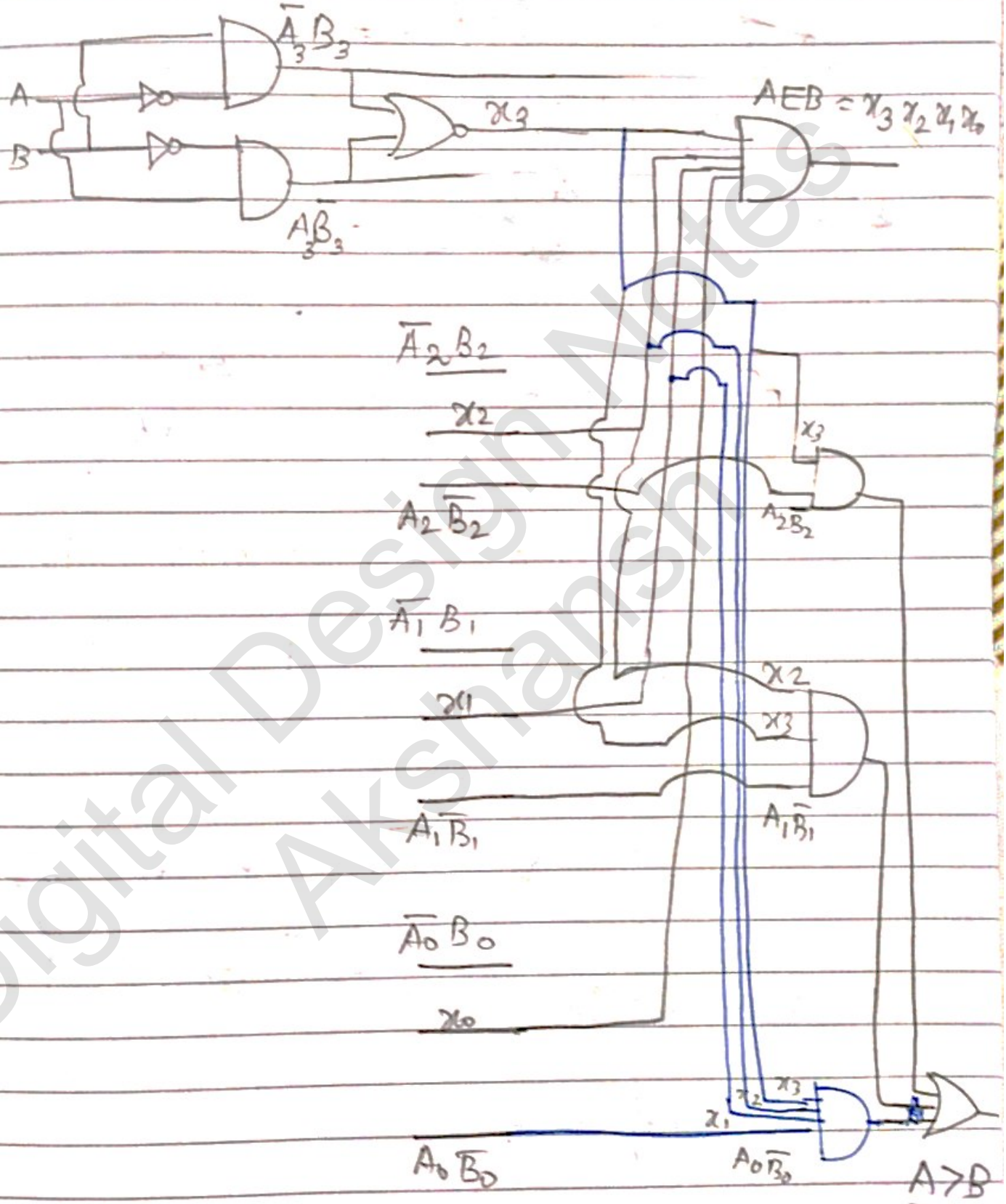
B: 5 6 7 8

$A < B$

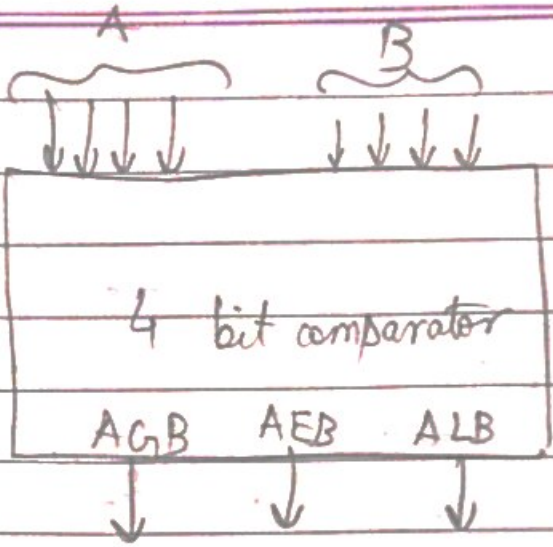
$\Rightarrow A < B$

$\Rightarrow 4 < 5$

Implementing 4 bit



Hy for A L B

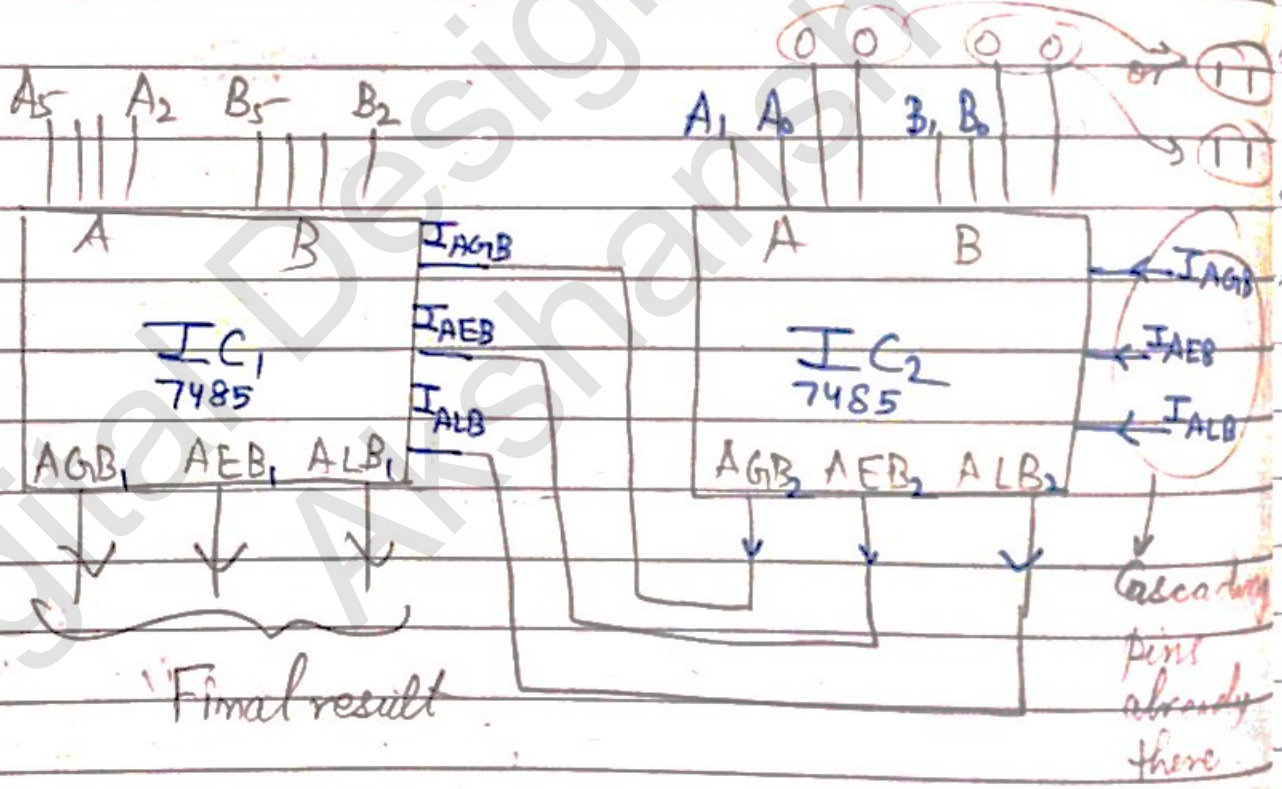


Std. circuit

eg: Comparing 6 bits words

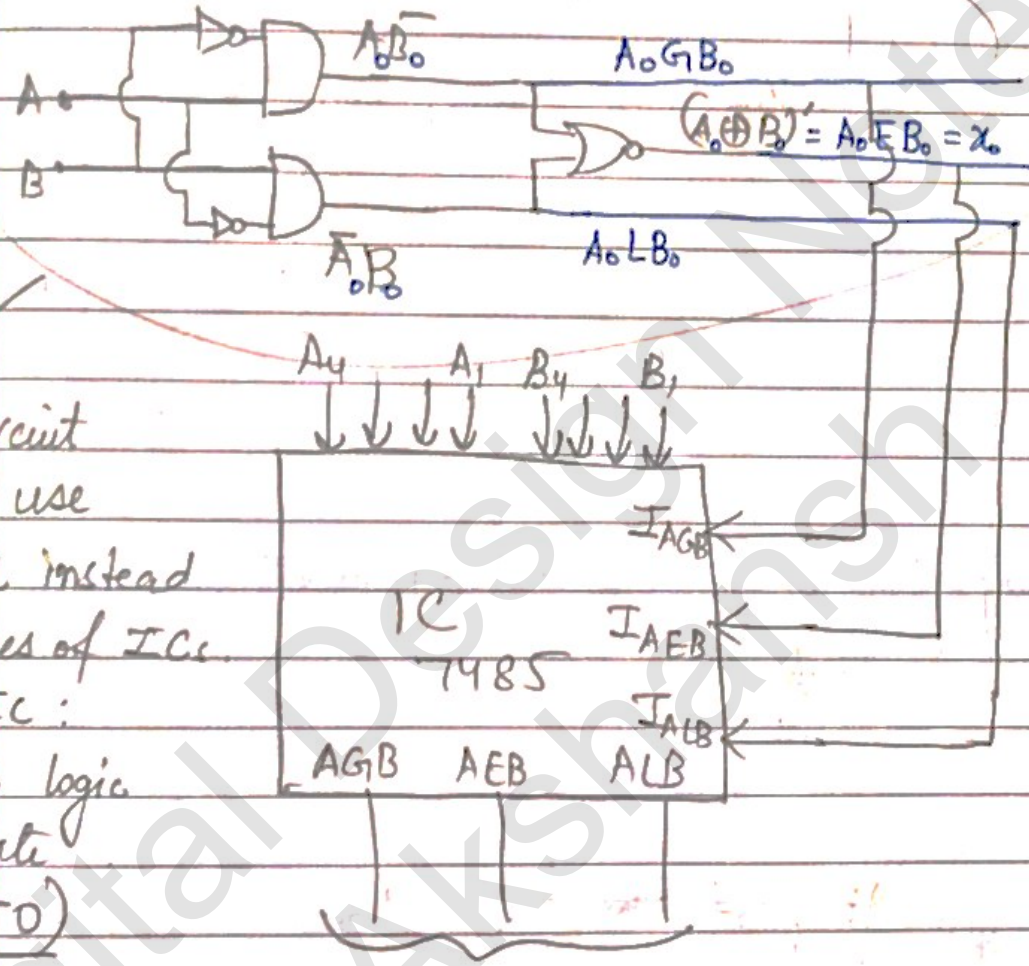
$A \rightarrow A_5 A_4 A_3 A_2 A_1 A_0$

$B \rightarrow B_5 B_4 B_3 B_2 B_1 B_0$



So, firstly IC_1 is being compared.
If result is equal, then IC_2 is used or compared to give its o/p to i/p pins of cascade & that is ANDed with AEB_1.
Finally o/ps are got.
Note: ANDing is internal.

eg Comparing 5 bit words : Only 1 7485 is given.



Make circuit efficient: use single IC instead of 3 types of ICs.
Single IC: XOR logic gate (PTO)

Idea : Compare first 4 MSB bits. Use the circuit of 2 bit comparator for comparing A_0 & B_0. If the o/p $A_4A_3A_2A_1 = B_4B_3B_2B_1$, then, that AEB is ANDed (=D) with the cascading i/ps coming from the 2 bit comparator circuit.

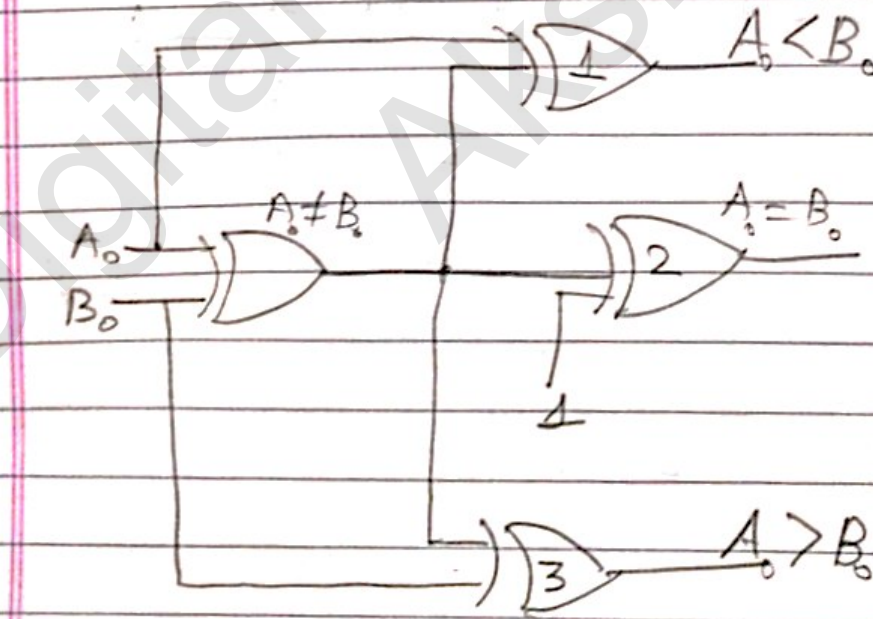
Note :- ANDing is INTERNAL

* lower significant bit has to be cascaded to Higher significant bit IC ∴ o/p comes from Higher IC.

F^n table *

Comparing I/P						Cascading I/Ps			O/Ps		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0			$I_{A_0>B_0}$	$I_{A_0<B_0}$	$I_{A_0=B_0}$	A>B	A<B	A=
① → $A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$			X	X	H	L	L	H
② → $A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$			H	H	L	L	L	L
③ → $A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$			L	L	L	H	H	L

* Single bit comparator EFFICIENT implement^n.
XOR gate



Suppose $A_0 \neq B_0$ is 1 (as o/p), that means A_0 & B_0 are of opp sign. (0 & 1). If IC ① gives 1 as o/p

That means A_0 is 0 & $B_0 = 1$. So, $A_0 < B_0$

Why for $A_0 > B_0$.

Suppose $A_0 \neq B_0$ is 0 (as o/p) & $\overline{I(1)}$ gives 1 as o/p ⇒ $A_0 = 1$ & $B_0 = 1$ → both A_0 & B_0 have 0 or 1

So, $I_{A_0 = B_0}$ is high

$I_{(A_0 > B_0)}$ is high

$I_{(A_0 < B_0)}$ is high

} See ⇒ this should not happen

Now,

Suppose $A_0 \neq B_0$ is 0 (as o/p) [⇒ A_0 & B_0 are both 0 or 1). & $\overline{I(1)}$ gives 0 as o/p. ⇒

$A_0 = 0$ & $B_0 = 0$

⇒ $I_{(A_0 = B_0)}$ is high

$I_{(A_0 > B_0)}$ is low

$I_{(A_0 < B_0)}$ is low

→ $I_{(A_0 = B_0)}$ is high in both cases whatever be the value of $I_{A_0 > B_0}$ or $I_{A_0 < B_0}$

So, See F^n table (↖) point (1).

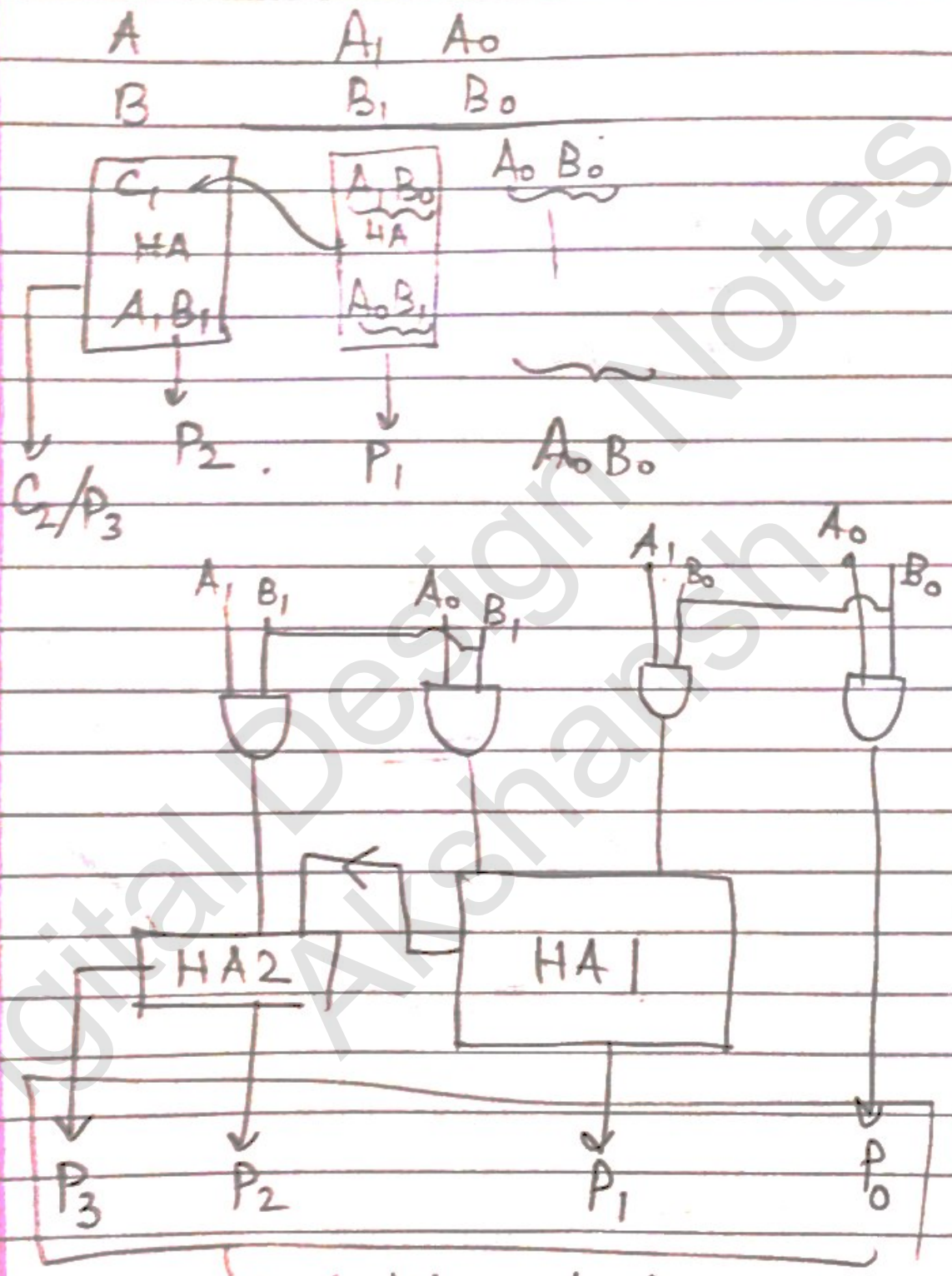
In this case, o/p received is

$\underline{\underline{A = B}}$ = high

$A > B$ } low

$A < B$ }

★ 2 BIT MULTIPLICATION



→ 4 bit word for 2 bit multiplic".

★ For m bit x n bit word, product term will have m+n bits.

★ 4 BIT Parallel adder :- Adds 4 bit words

★ 4 x 3 bit Multiplic^n

$B_3\ B_2\ B_1\ B_0$
$\times A_2\ A_1\ A_0$

$A_0B_3\ \ A_0B_2\ \ A_0B_1\ \ A_0B_0$

$A_1B_3\ \ A_1B_2\ \ A_1B_1\ \ A_1B_0$ → Std. way of multiplying

$A_2B_3\ \ A_2B_2\ \ A_2B_1\ \ A_2B_0$

HA FA HA A_0B_0

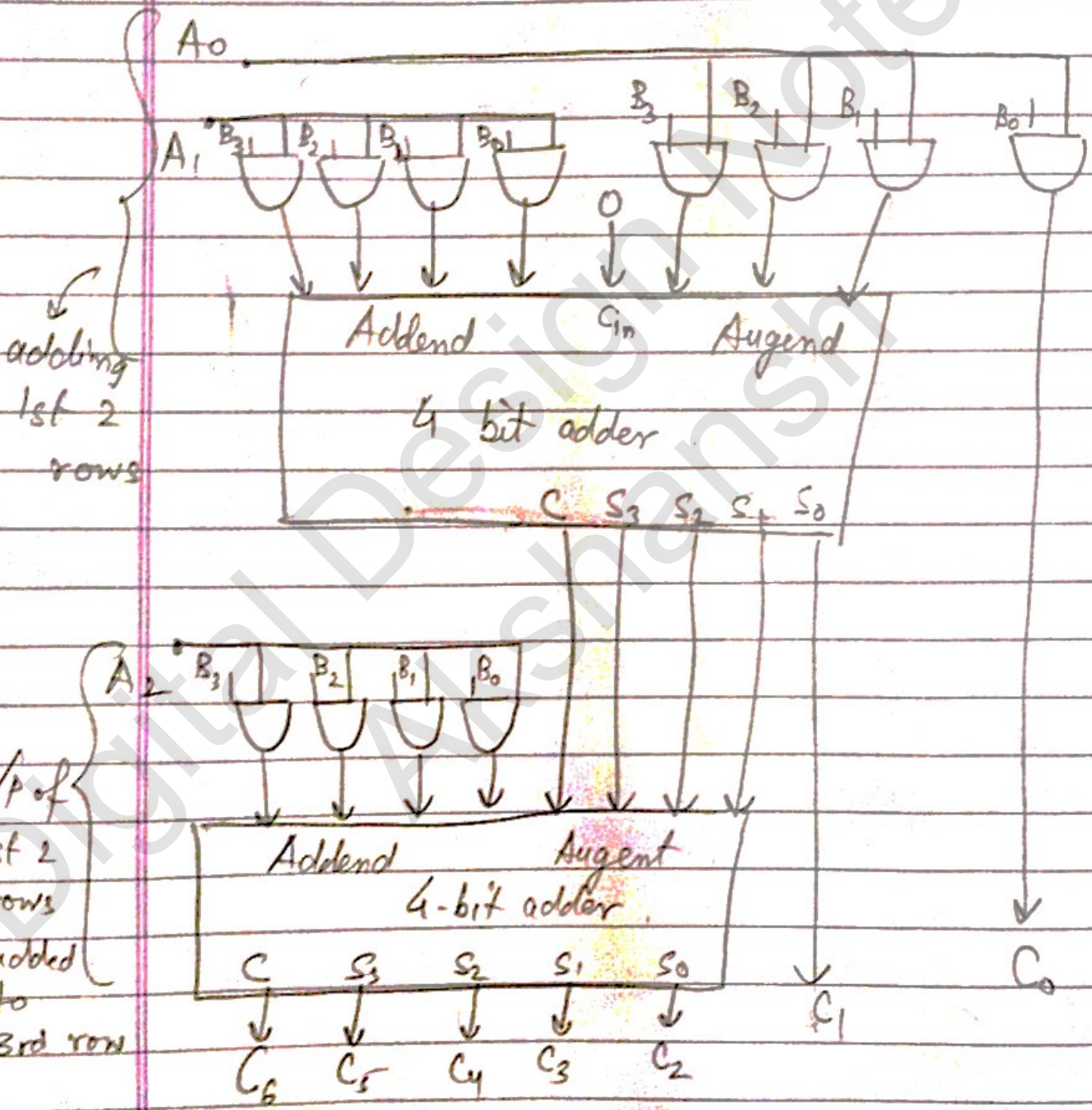
we need to add
3 bits + 1 carry bit
⇓
4 bits.

Now, when multiplic^n of n x m bits is done, then no. of rows will be more. So, say, we need to add m + 1 carry bit, say

So, develop logic for general bit multiplic^n

Logic

Add 1st 2 rows, then proceed



Ily continue if no. of rows are more

SEQUENCIAL CKTs.

Puffin
Date
Page

* Latch : level triggered device → SR, D, T
* Flip-flops :- +ve & -ve edge triggered devices

SR | D | T | JK

* Characteristic tables (Logic done before)

• D flip flop

D	Q(t+1)
0	0
1	1

• T flip flop.

T	Q(t+1)
0	$Q_{(t)}$
1	$\overline{Q_{(t)}}$

• J-K flip flop

		Q(t)
0	0	0
0	1	1
1	0	$\overline{Q_{(t)}}$
1	1	

→ expanding (seeing previous state)

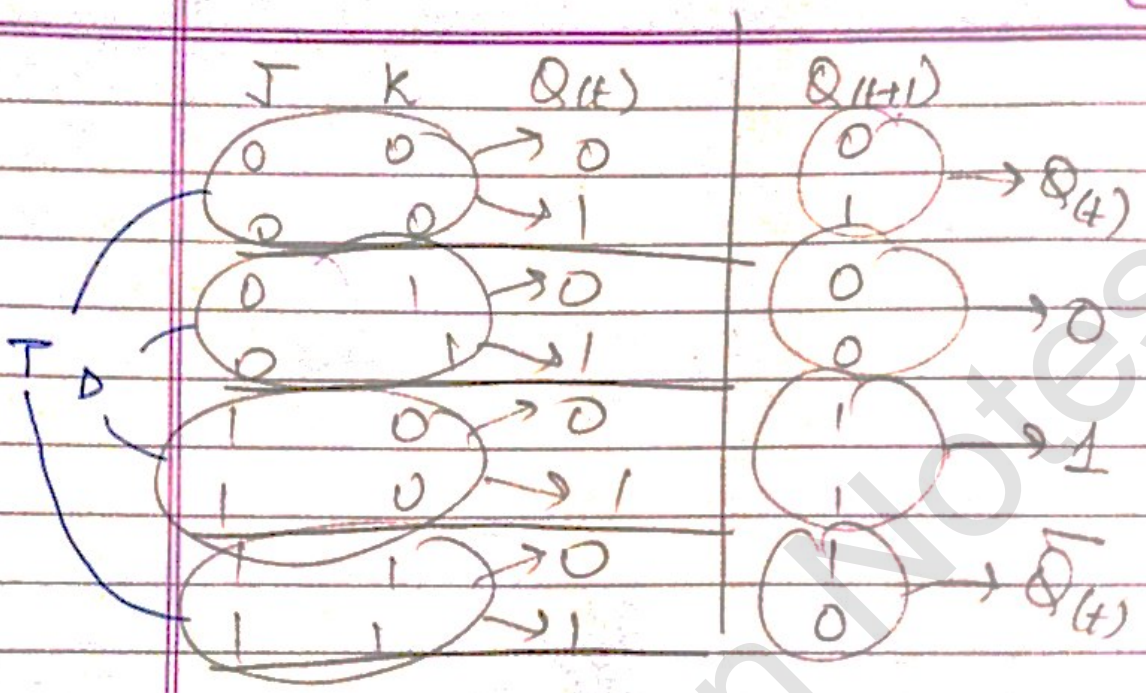
considering previous state

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

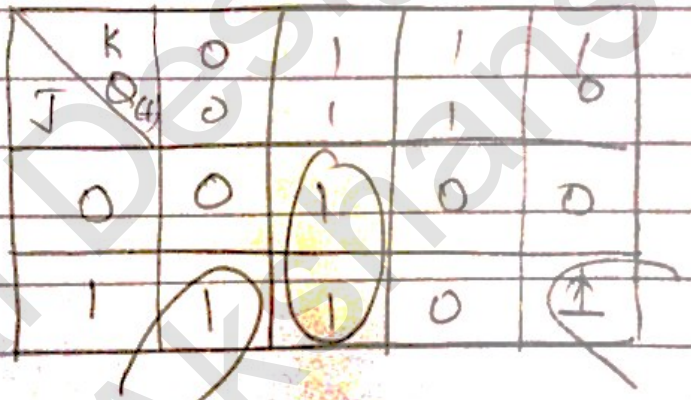
* Characteristic eqⁿ

D flip flop :- $Q_D(t+1) = D$ ——(1),

T " " :- $Q_T(t+1) = T \oplus Q_{(t)}$. ——(2)



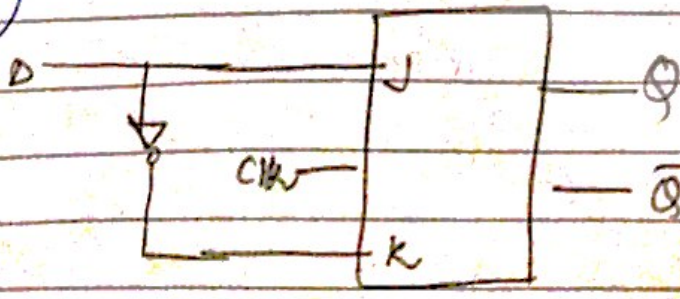
Characteristic eqⁿ for J-K : K-map



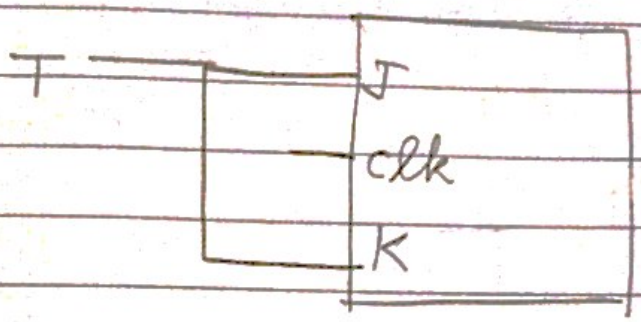
$$Q_{JK}(t+1) = J Q'_{(t)} + K' Q_{(t)} \longrightarrow (3)$$

* Design D & T flip flop : Given JK flip flop.

(D)



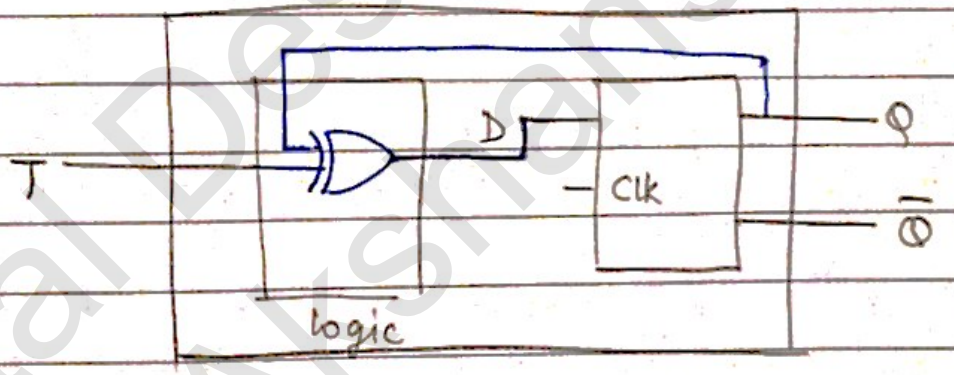
(T)



So, JK flip flop is universal flip flop.

→ Duplicate → Toggle

✱ Given (D) flip flop : Design (T)

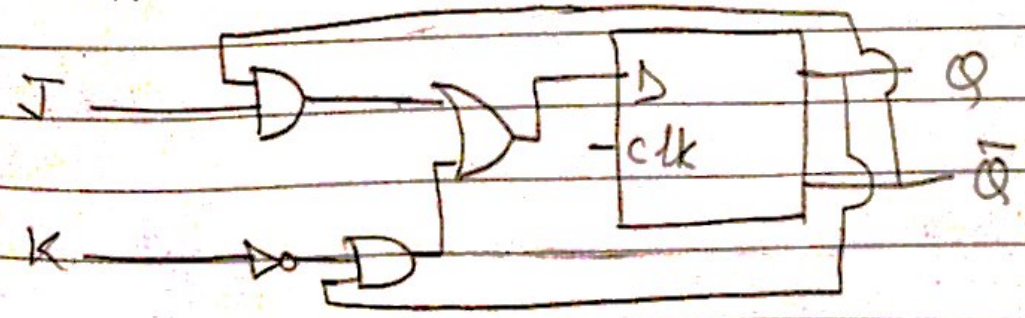


(using logic that $Q_D(t+1) = D$

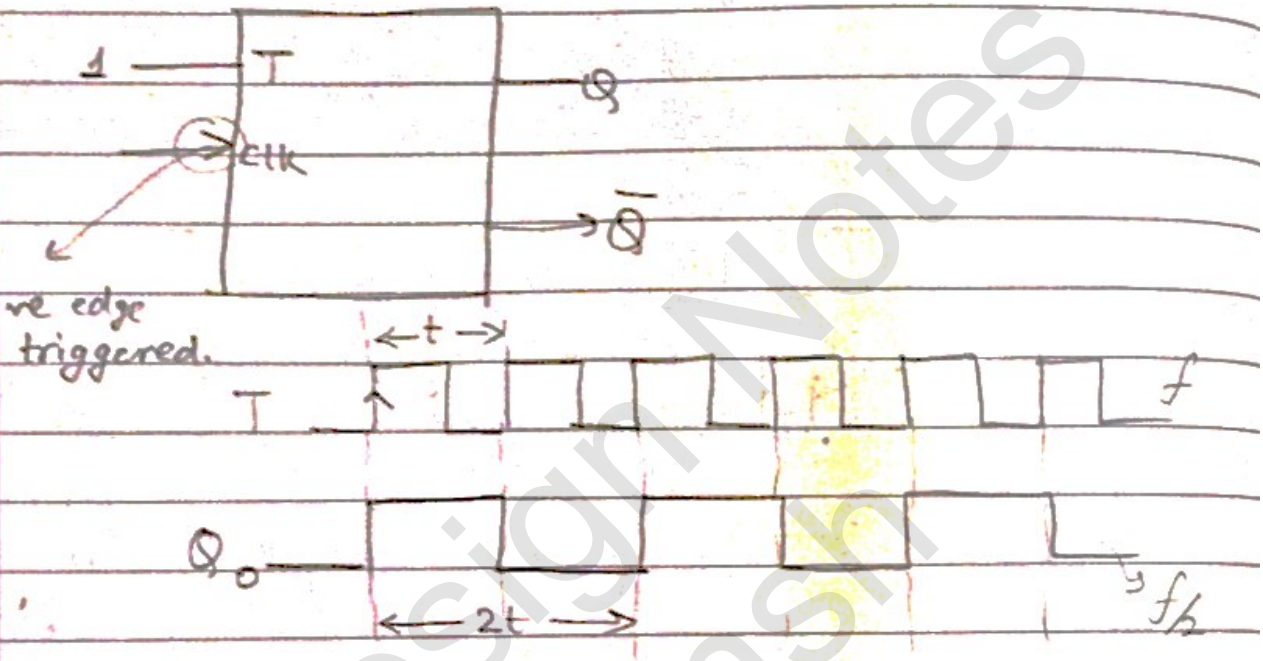
$Q_{T}(t+1) = T \oplus Q_{(t)}$

✱ Given D flip flat : Design JK.

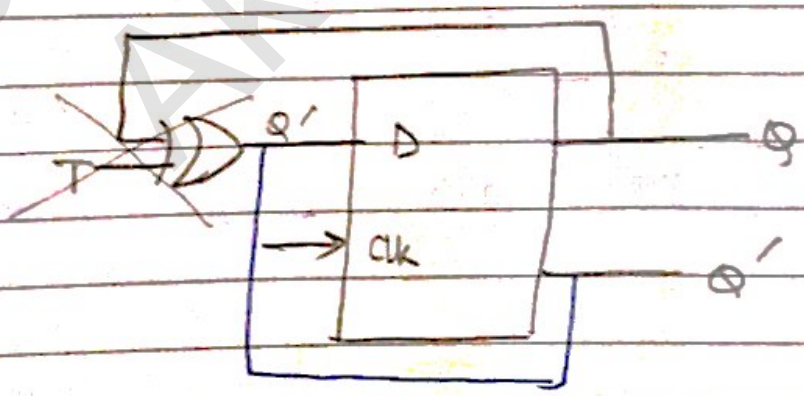
$Q_{JK}(t+1) = JQ' + K'Q$



Q. If T = 1 & time if clk i/p are applied Q = ? (Q = 0 initially)



* T flip flop using D



when T = 1, o/p of XOR gate = $TQ' + T'Q$

$= 1Q' + 0.$

$= Q'$

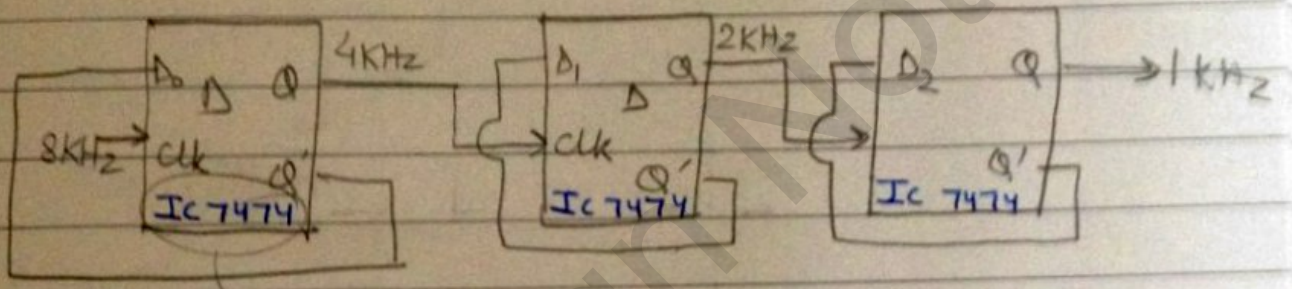
So, connect Q'. No need of XOR.

This is frequency divider circuit. (same clock pulse as above).

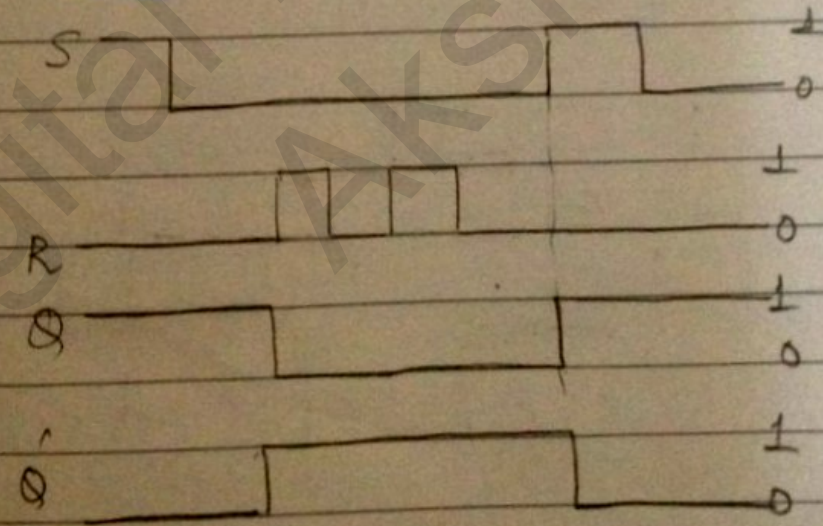
Puffin
Date
Page

* Given 8KHz clock pulse (Pulse generator). How to generate 1 KHz ?

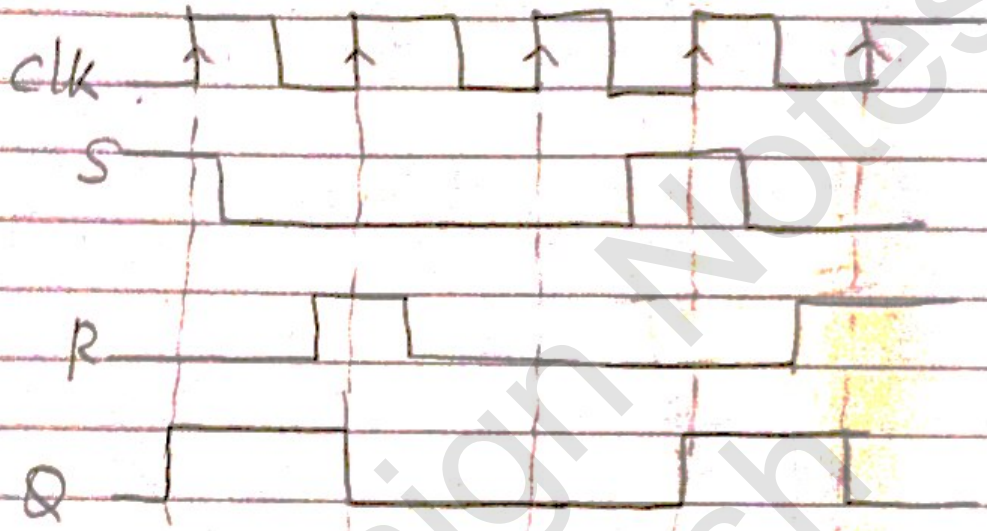
(Use D flip flop)



* Graphical representn of i/p & o/p
* SR Latch



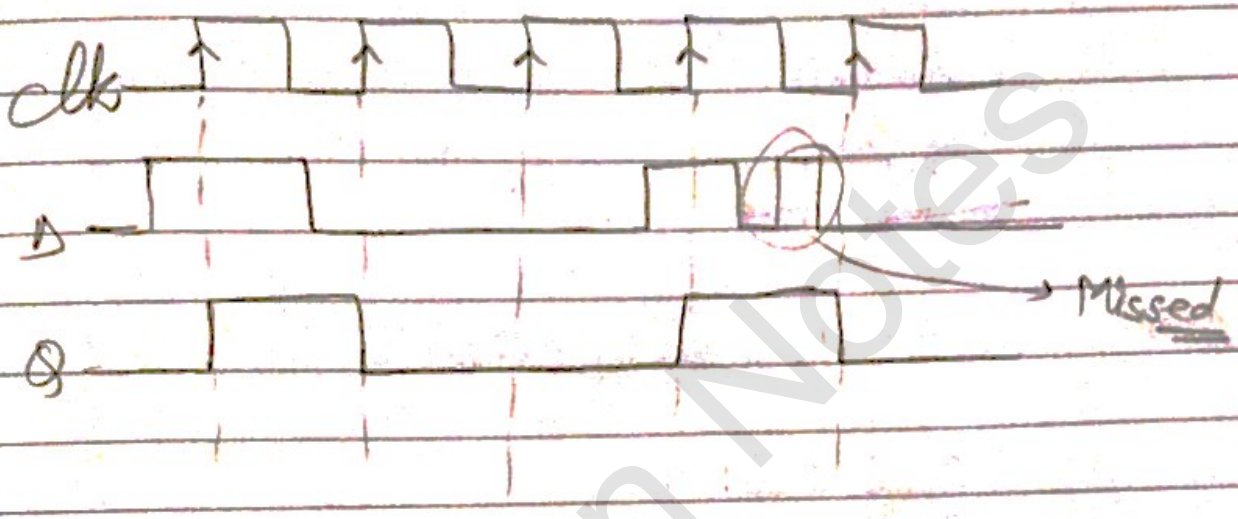
☆ SR flip flop (+ve edge)
O/p won't be affected even if S & R change in b/w clock pulse.



☆ -ve edge

* Only 1 bit data transfer takes place

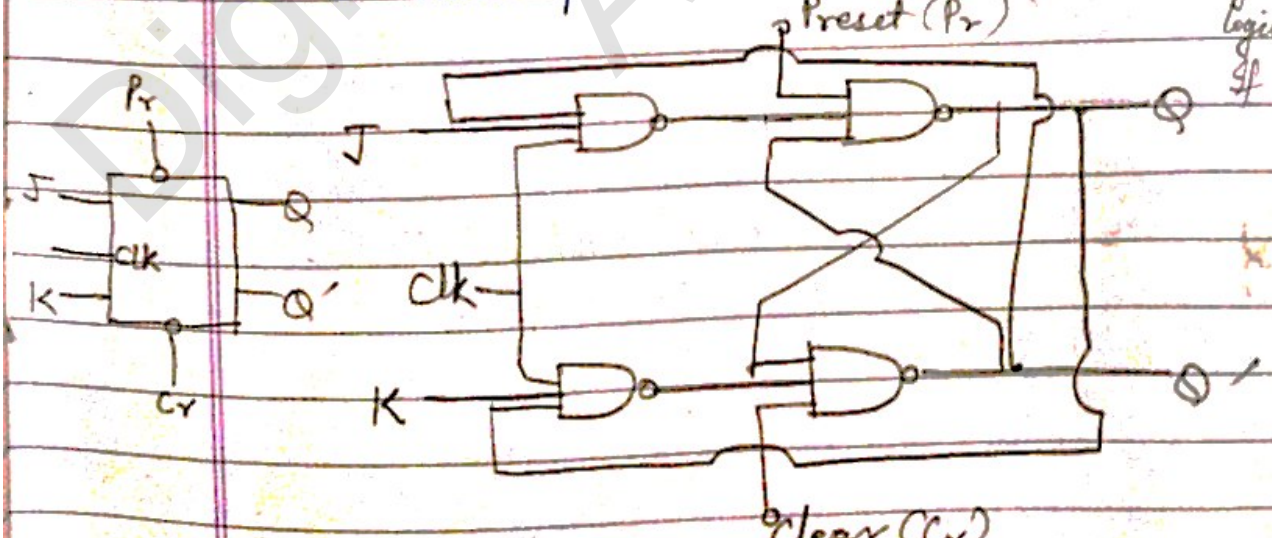
* D flip flop (+ve edge)



* T flip flop

* JK with preset & clear

If $P_r = 0$, $Q = 1$ always (NAND gate logic)
If $Q = 1$ ⇒ obviously $Q' = 0$. So keep $C_r = 1$



Clk	Cr	Pr	Q
↓	0 or 1	0 or 1	—
0	0	1	0
0	1	0	1

} getting o/p by changing Cr & Pr

Puffin
Date
Page

* The present state o/p of circuit = $Q_{(t)}$
When some i/p is marked from clock pulse → o/p is Q_{t+1}
So, basically, both Q_t & Q_{t+1} o/p come from same terminal.

* Due to electrical resistance (≡ mechanical inertia)

What we take → pico second

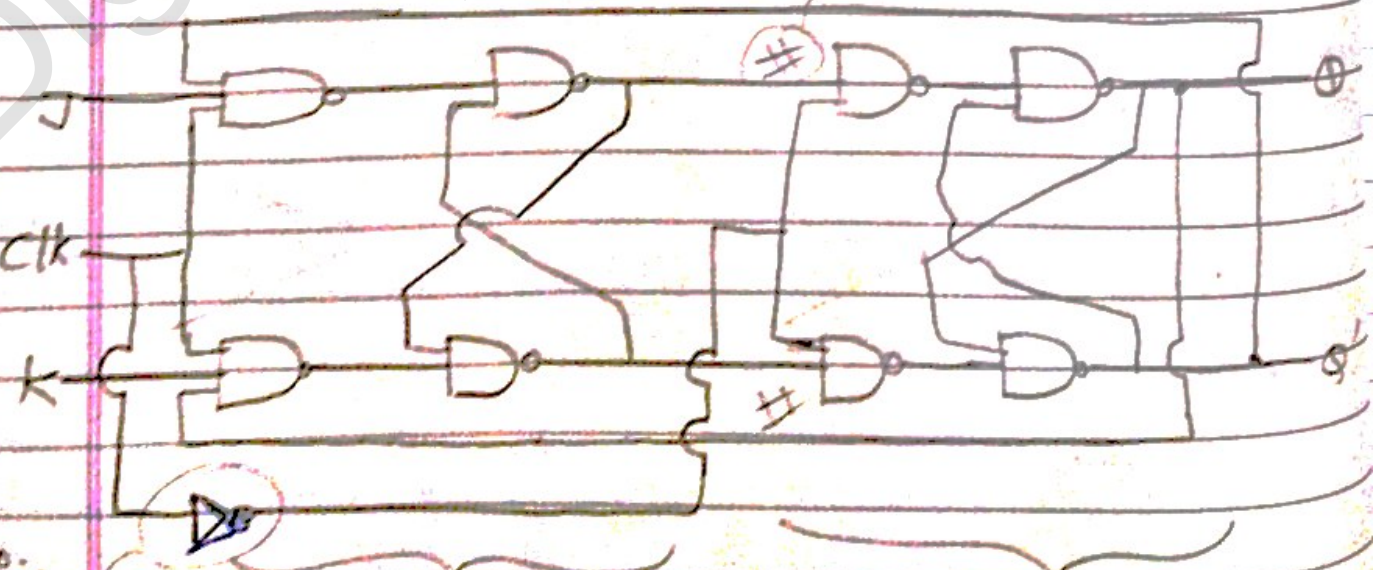
Actual → some (ps) or ns

i/p

→ Suppose i/p comes during trans^n time &, o/p corresponding to this i/p won't come out

To prevent this

Master-slave ⇒ Circuit has to be checked



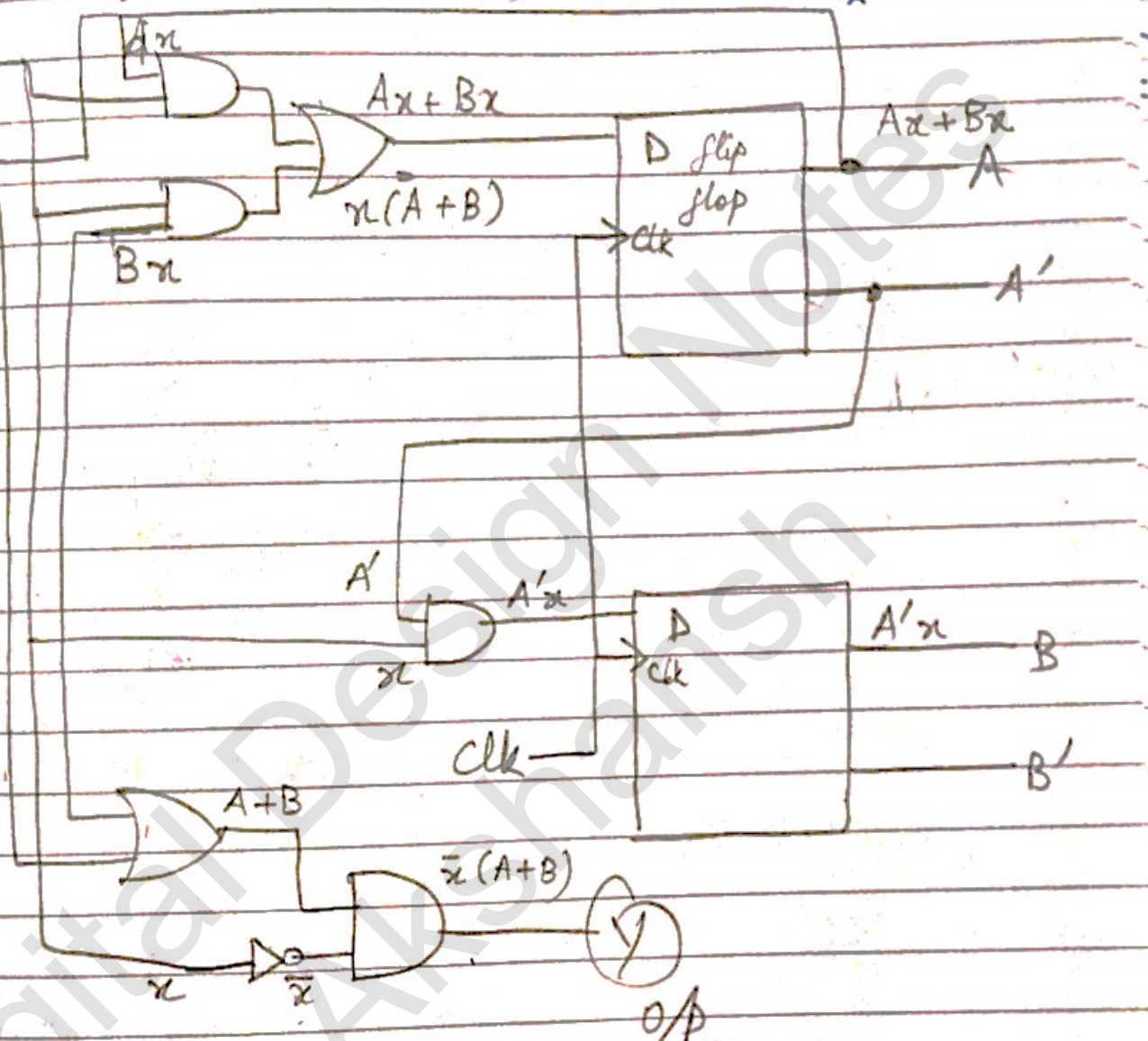
opp. clock pulse given to slave

* Flip flop is +ve edge trigerred always.

Puffin
Date
Page

* Analysis of clocked sequential circuit.
Ex: State eq^ns (trans^n eq^n)

* (logic diagram)



next state interms of i/p & present states.

i/p exit^n eq^ns.

* $A(t+1) = A(t)\,x(t) + B(t)\,x(t) = Ax + Bx = D_A$

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

o/p eq^n $y(t) = [A(t) + B(t)]\,x'(t) = (A+B)\,x' = y$

= Boolean expressions just like that

These eq^ns show:- When i/p state is — & present state is — what will be o/p when clock pulse is applied.

✱ State table (≡ Truth table)

→ (A+B)x → A'x (A+B)x

(i)

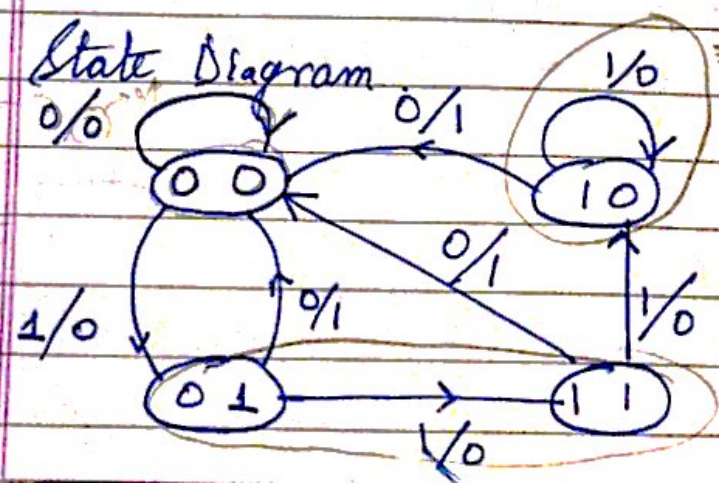
Present State		i/p x	Next State		o/p
A	B	x	D_A	D_B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Splitting above table for simplific" (another way)

(ii)

Present state		Next State				o/p	
		x=0		x=1		x=0	x=1
A	B	D_A	D_B	D_A	D_B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

✱ State Diagram



≡ when i/p 1 is applied to present state 1 0, o/p is 0 & D_A D_B are 1 0 (same)

Lines should not cross each other

≡ when i/p 1 is applied to present state 0 1, o/p is 0 & D_A D_B are 1 1.

☆ Analysis with D flip flop.

≡ Analysis of a clocked sequential D flip flop.

Given :-

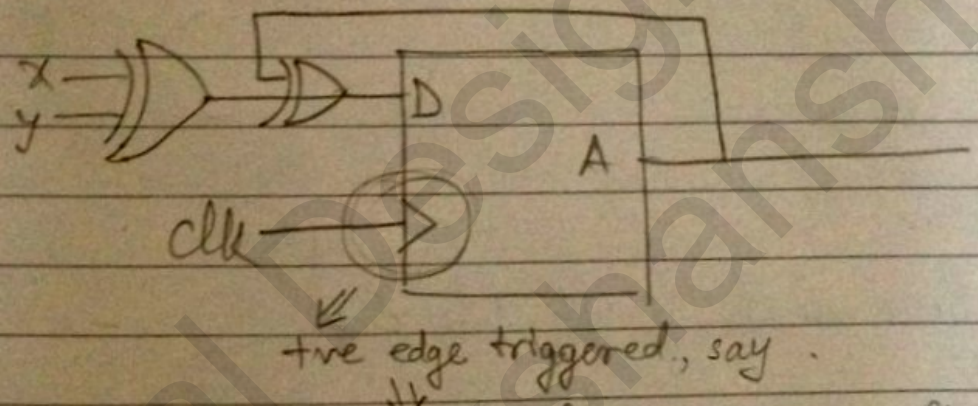
i/p eqⁿ :- $D_A = A \oplus x \oplus y$

D_A : D flip flop with o/p as A.

x, y : i/p

Here, o/p eqⁿ isn't given ⇒ o/p comes from o/p of flip flop.

- Logic diagram:



+ve edge triggered, say.

⇓

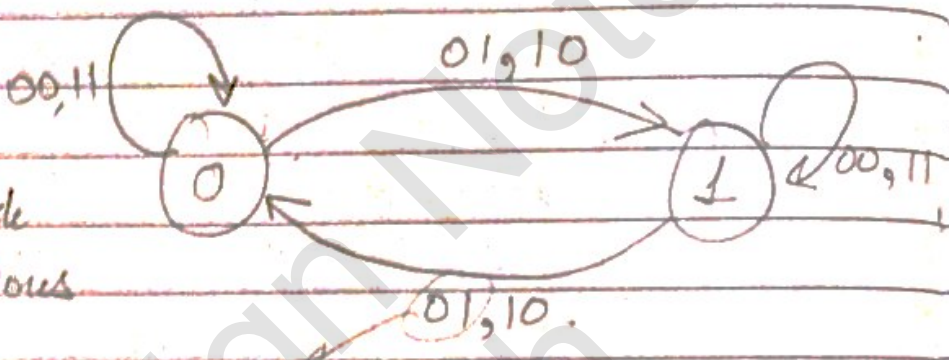
circuit changes its state after 1 +ve edge.

- State table

Present state	i/p		Next state
A	x	y	
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Puffin
Date
Page

* State diagram :
This flip flop has only 2 states → 0 or 1
00 ⇒ x=0 & y=0 } ⇒ same state
11 ⇒ x=1 & y=1 }



* Whatever inside circles :- Previous states
* Whatever near arrows :- Ickle cond^ns that it'll go from one state to another

State changes from 1 to 0 when (prev.) (next)
0 (=x) & 1 (=y) are applied

* Analysis with J-K flip flop.

Eq^n for next state, $Q_{JK(t+1)} = J Q'_{(t)} + K' Q_{(t)}$. = written before

Imp Procedure to derive next state eq^ns of JK or T flip flops (not D : as for it, i/p eq^n & state eq^ns are same)

1. Determine the flip flop i/p eq^ns in terms of present state & i/p variables.
2. Use binary values of each i/p eq^n.
3. Use corresponding flip-flop characteristic table to determine next state values in state table.

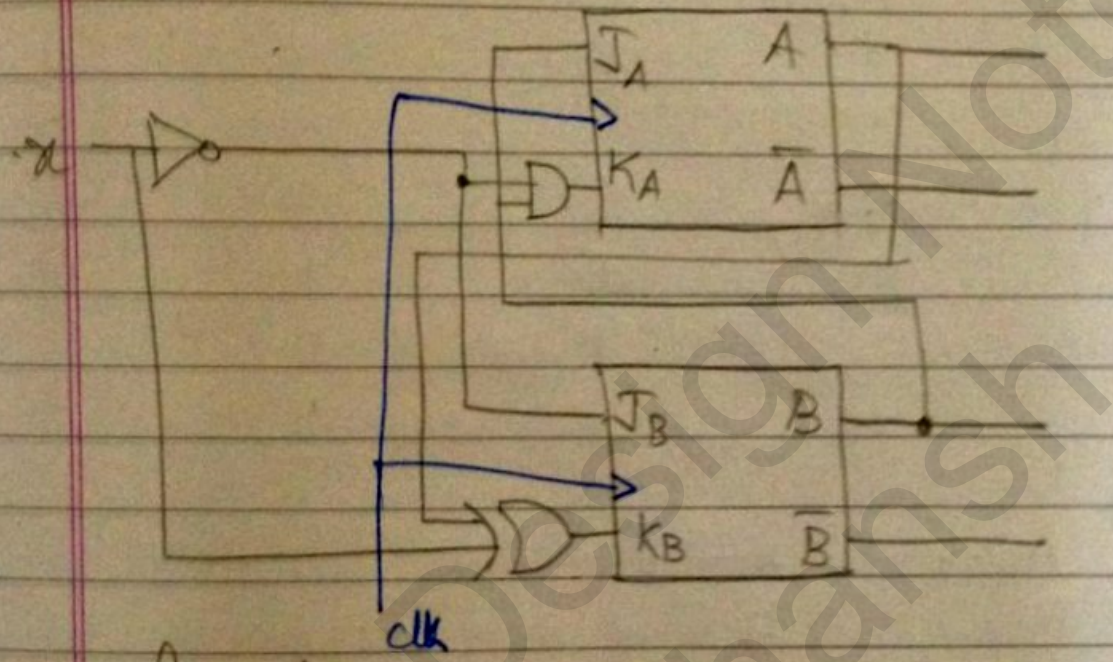
Eg : Consider 2 JK flip flops A & B & one i/p x

Let the i/p eqⁿˢ be

$J_A = B$ $\quad K_A = Bx'$

$J_B = x'$ $\quad K_B = A'x + Ax' = A \oplus x$ → (1)

- Draw circuit accordingly



- Characteristic eqⁿ :-

For A → JK flip flop, replace Q by A

$\Rightarrow A(t+1) = J_A A' + K_A' A$

& $B(t+1) = J_B B' + K_B' B$ } → (2)

Using (1) in (2)

$\Rightarrow A(t+1) = B\bar{A} + (\overline{B \oplus x})A$

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

$\Rightarrow A(t+1) = (B\bar{A} + \overline{BA}) + Ax = (B \oplus A) + Ax$

Similarly, $B(t+1) = \bar{x}\bar{B} + (A \oplus x)B$

$= \bar{x}\bar{B} + (\bar{A}\bar{x} + Ax)B$

$\Rightarrow B(t+1) = \bar{x}\bar{B} + \bar{A}\bar{x}B + AxB$

★★ Note :- · No. of states = $2^{(n)}$ → no. of flip flops.

Puffin
Date
Page

- State table

Present state		i/p	Next state	
A	B	x	A(t+1)	B(t+1)
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1

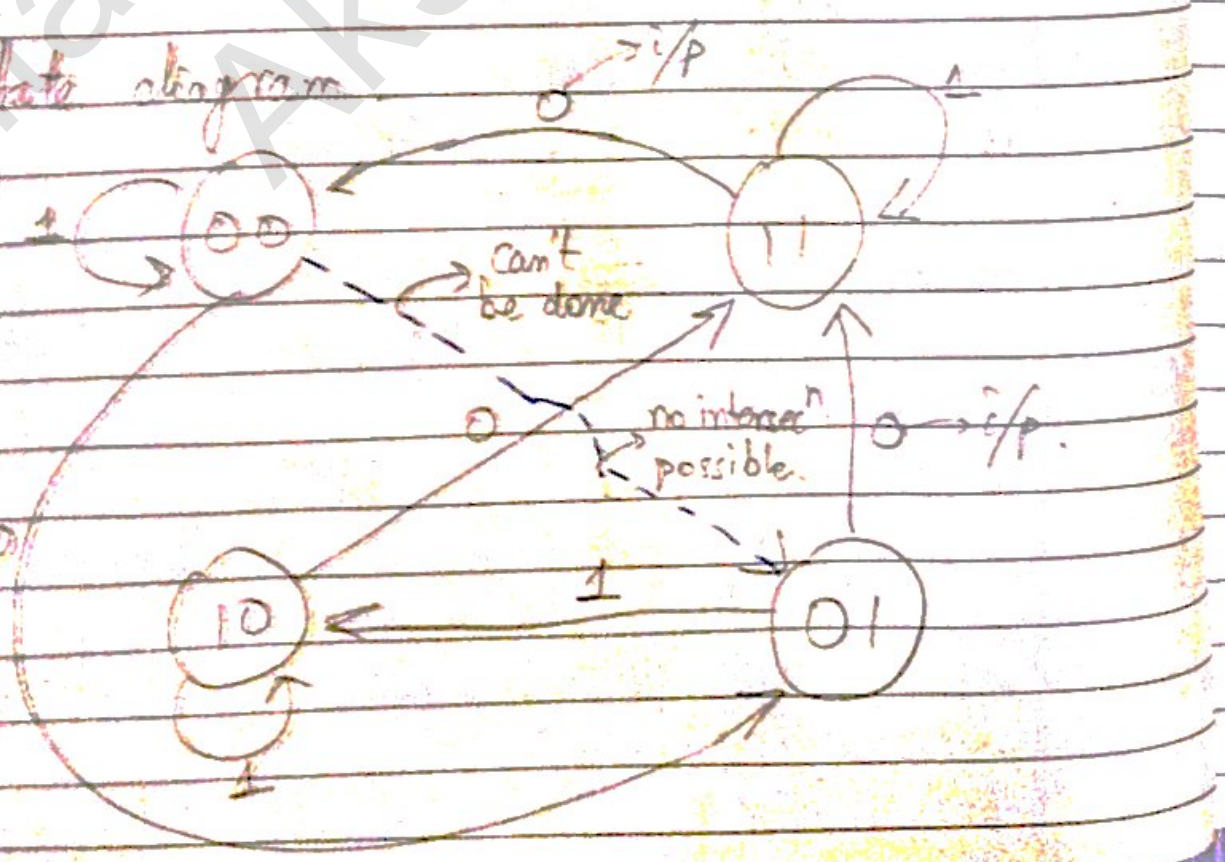
where,

$A(t+1) = A \oplus B + Ax$

& $B(t+1) = \bar{B}\bar{x} + \bar{A}\bar{x}B + AxB$.

Here 2 flip flops ⇒ 4 states

- State diagram.



Note :- ∀ states except 01, state remains same when 1 is i/p.

The states :-

00 ⇒ both flip flops 0

01 ⇒ flip flop (first) = 0 & 2nd = 1

11 ⇒ Both flip flops are in state → 1

10 ⇒ 1st flip flop state 1 & 2nd in state 0.

☆ Analysis with T flip flops.

~~Characteristic~~

Next state eq^n :- $Q_{(t+1)} = T \oplus Q = T'Q + TQ'$ → written proof before.

eg

Consider (2) → A & B T flip flops & 1 i/p → x

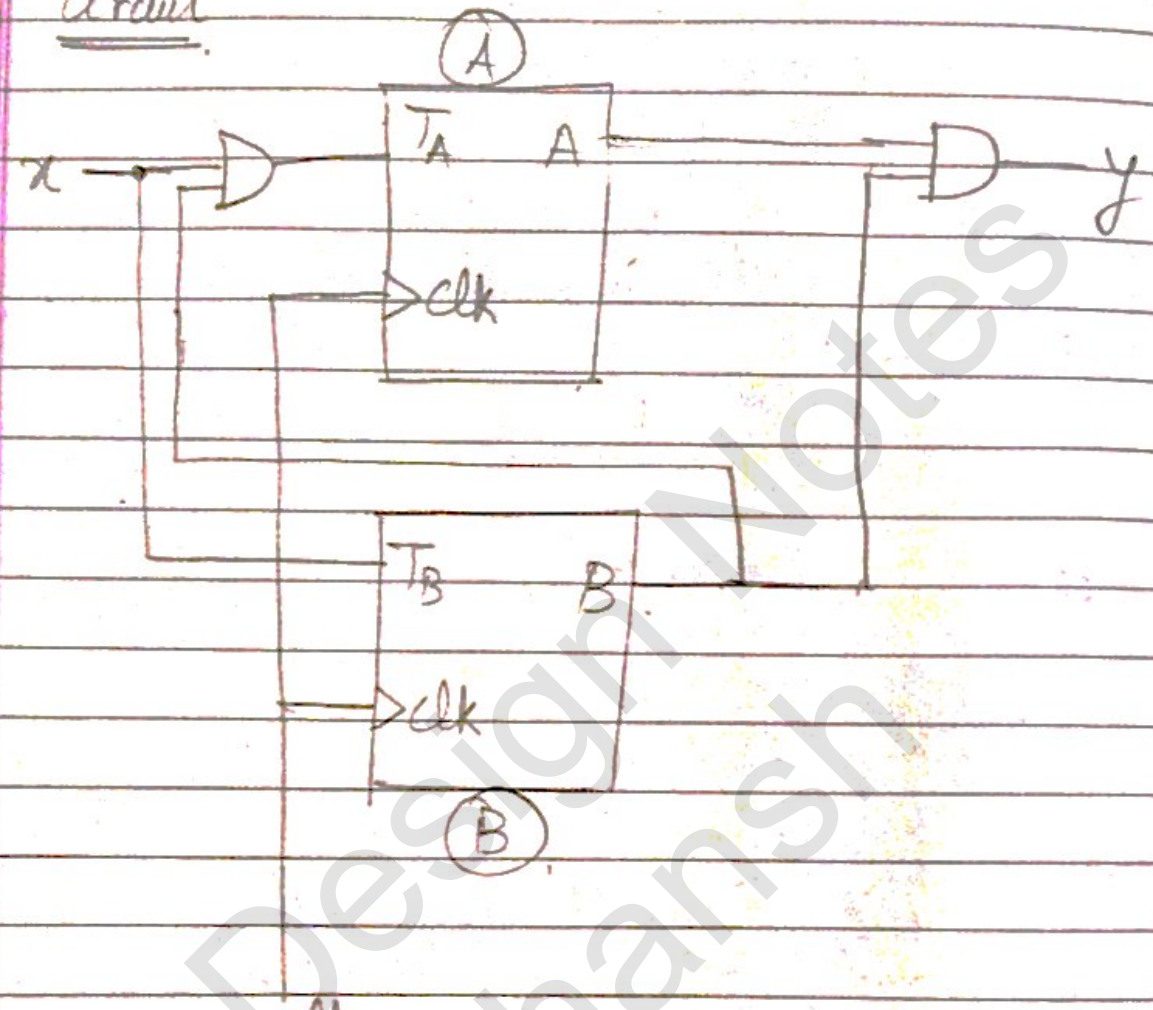
i/p eqns :- $T_A = Bx$

$T_B = x$

o/p eq^n :-

$Y = AB$ → (1)

• Circuit



• Characteristic eqⁿˢ:-

$A_{(t+1)} = T_A \oplus A$
$B_{(t+1)} = T_B \oplus B$ } (Replace Q by A, Q by B) → (2)

Now. Using (1) in (2)

$\Rightarrow A_{(t+1)} = Bx \oplus A$.

$B_{(t+1)} = x \oplus B$.

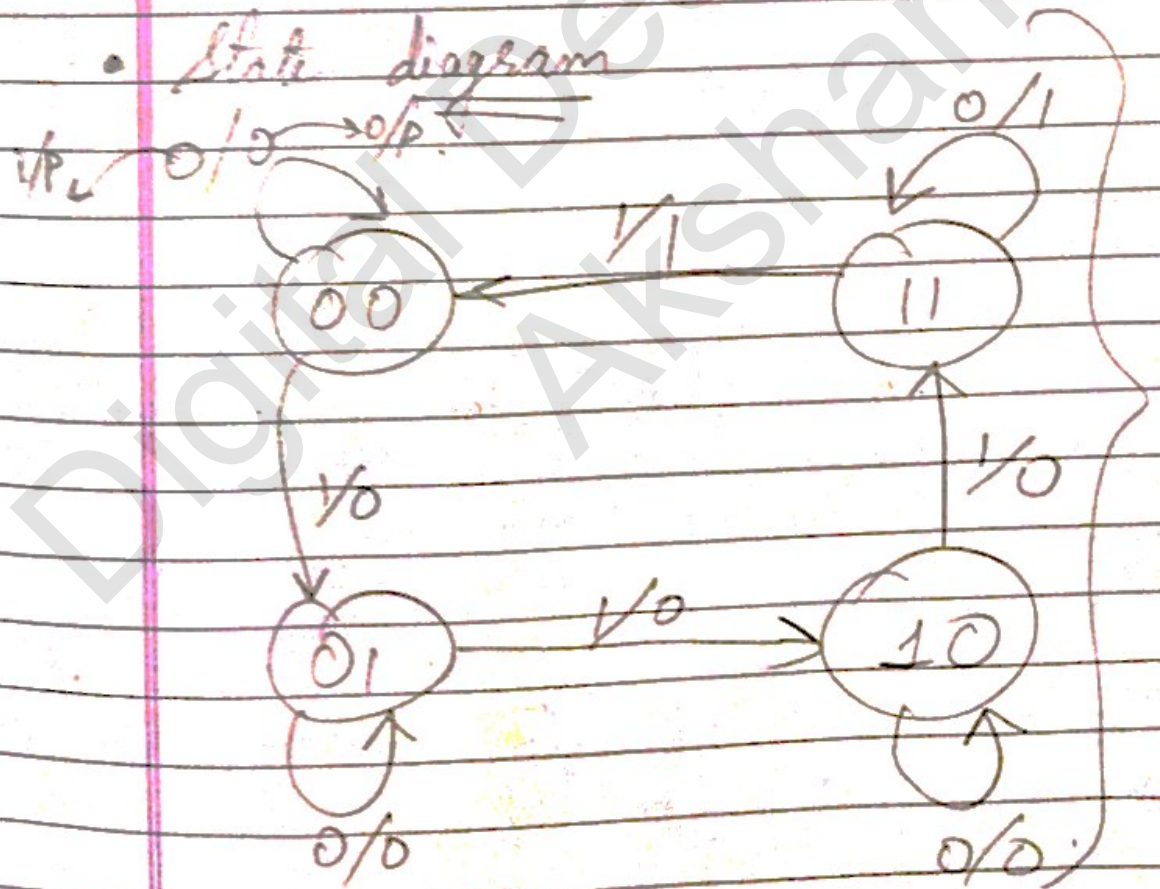
or

$A_{(t+1)} = A \oplus (Bx)$
$B_{(t+1)} = x \oplus B$.

- **State table** → got by putting values of A, B, x & getting next state & o/p.

Present State		i/p	Next state		o/p
A	B	x	A_{t+1}	B_{t+1}	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

- **State diagram**



Wrong.

Go to next page. (#)

Mealy & Moore Model of Finite State Machines (FSM)

They are basically models of sequential circuits. Both of them give o/p.

Difference :- the way in which o/p is generated

Mealy Model : o/p depends on :-
- ✓ Present state
- ✓ i/p

Moore Model :- o/p depends on :-
- ✓ Present state
- → doesn't depend on ext. i/p.

Note :- While making a state diagram having both i/p & o/p, we write above the arrow → i/p / o/p (sth like o/o). Now, in Moore model, it doesn't depend on i/p. So, o/p is written alongwith present state, i.e., inside circles.

(##) Note :-

The previous part (Analyses with T flip flop) had o/p eqn, $Y = AB$. The i/p was x. So, it was independent of i/p. Hence :- previous stage diagram is incorrect. The modified diagram is as shown →

V.Imp ☆ The state diagrams with NO o/p :- If they have present state, i/p & next state, then, as arrow goes from present state → next state (eg :- 00 → 01) So, o/p are taken as the states inside circles

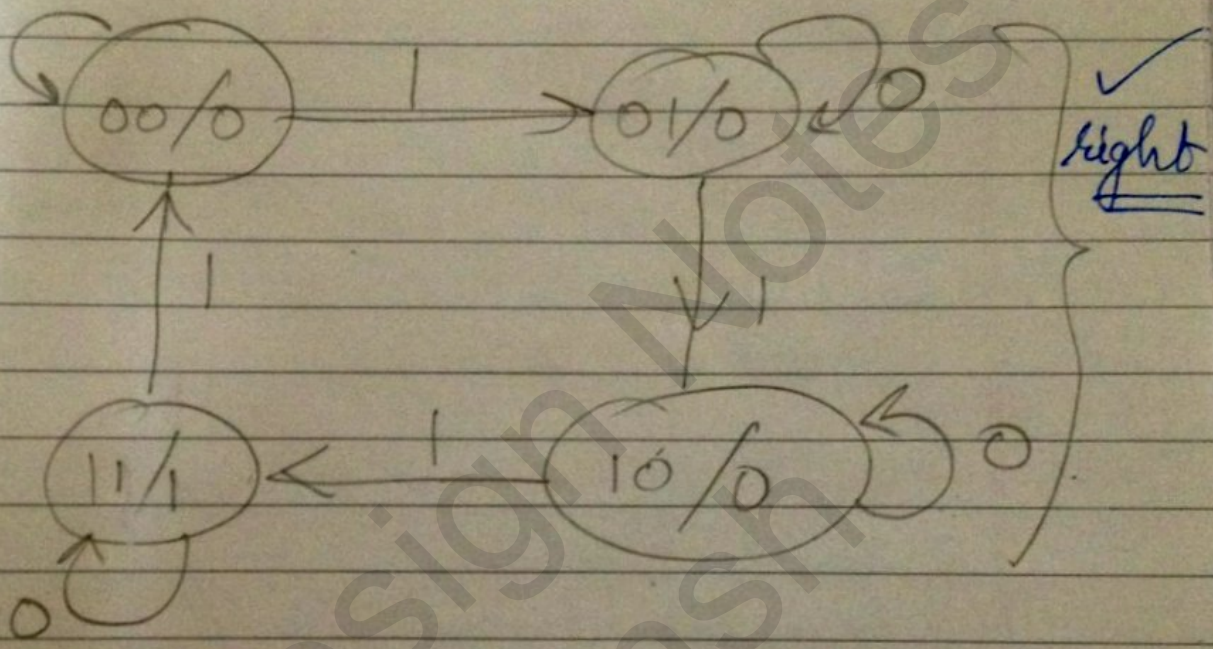
Puffin
Date
Page

o/p f^n of present state only

⇓

Moore Model

eg: State diagram of JK flip flop.



right ✓

☆ Note:— Mealy Model

↳ The state diagrams for this model are the normal state diagrams done before in which basically we can find

(input / output)

above arrows.

Q. 5.10 $J_A = Bx + B'y'$; $K_A = B'xy'$

$J_B = A'x$; $K_B = A + xy'$

$Z = Ax'y' + Bx'y' = (A+B)(x'y')$

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

∃ : 2 JK flip flops & 2 i/p's x & y

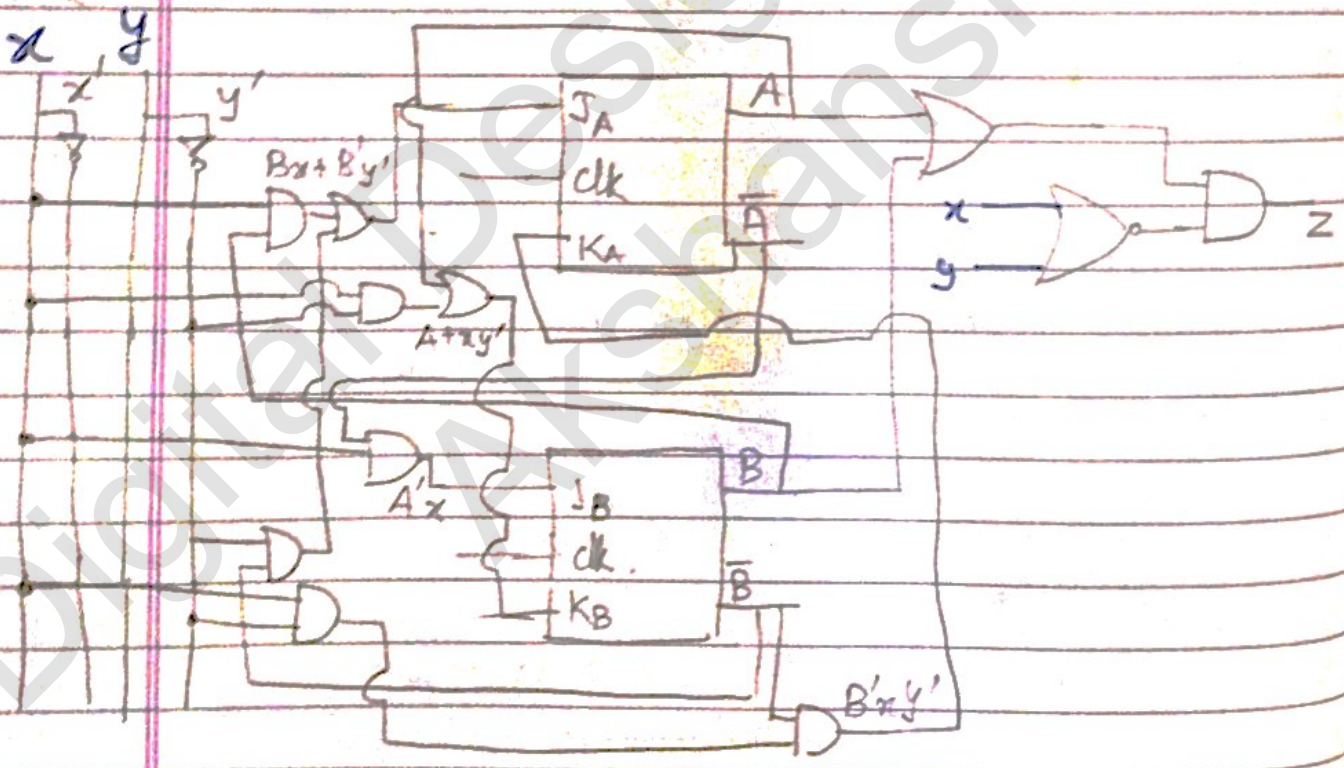
One o/p Z.

(a) Draw Logic diagram of circuit

b) Tabulate State table

(c) Draw state diagram

(a)



State eqⁿ

(b) $Q_{JK}(t+1) = JQ' + K'Q$

$A(t+1) = J_A A' + K'_A A$

→ referring to A flip flop — substitute

substitute their values

$$B(t+1) = J_B B' + K_B' B$$

Now

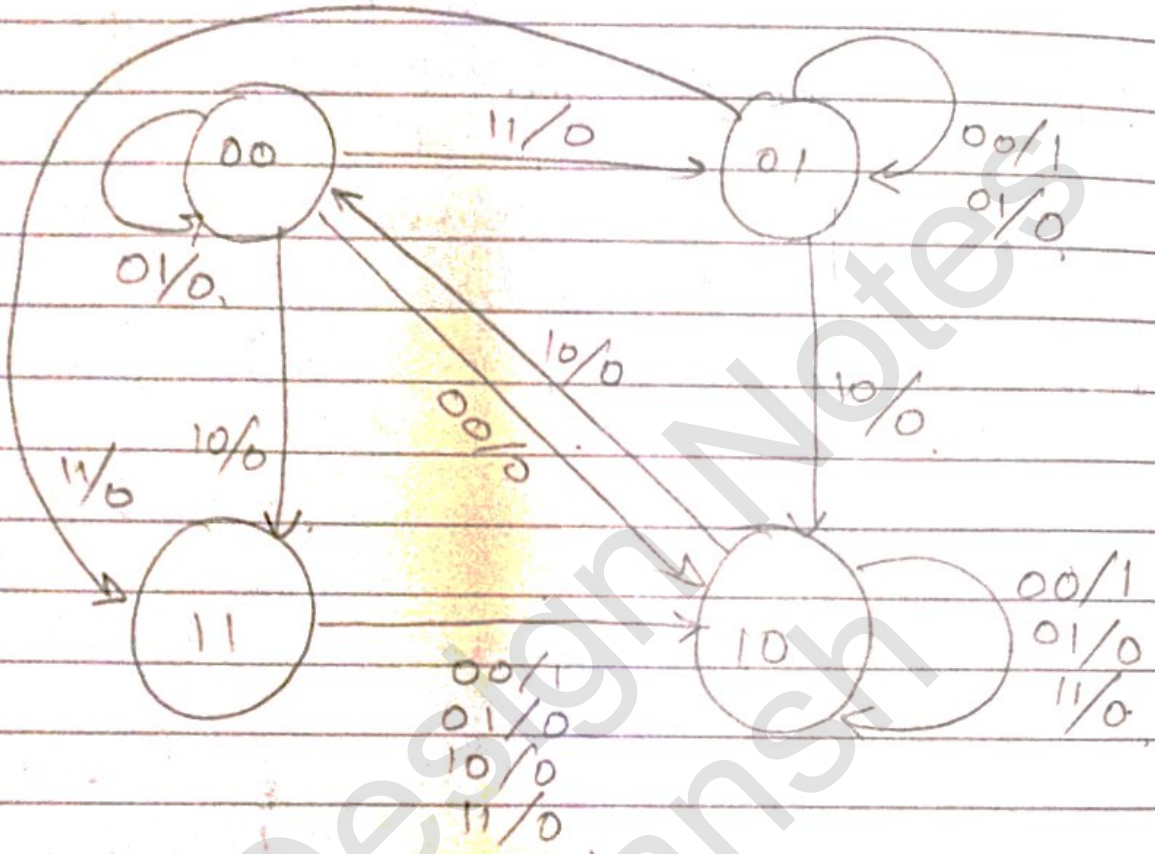
$$A(t+1) = (Bx + B'y')A' + (B'xy')'A$$
$$= (Bx + B'y')A' + (B + x' + y)A$$
$$\Rightarrow A(t+1) = A'Bx + A'B'y' + AB + Ax' + Ay$$

$$B(t+1) = A'xB' + (A + xy')'B$$
$$= A'xB' + (A')(\bar{x} + y)B$$
$$= A'B'x + A'Bx' + A'By$$
$$\Rightarrow B_{(t+1)} = A'(B'x + Bx' + By)$$

State table

Present state		i/p		Next state		o/p
$A(t)$	$B(t)$	x	y	$A(t+1)$	$B(t+1)$	z
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0

(c) State diagram.



✱ State Reduction & Assignment

Given a state diagram.

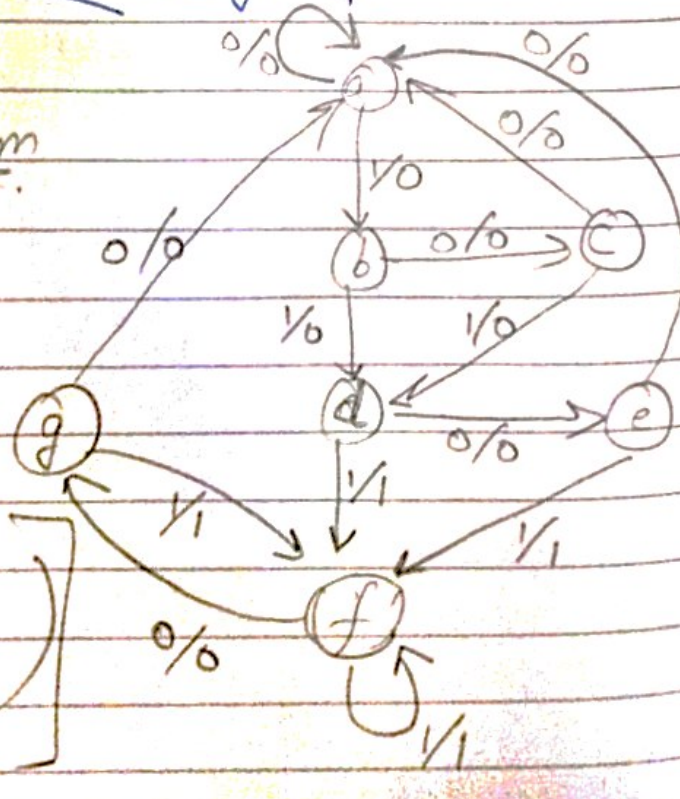
→ What min. no. of flip flops are req^d to implement it ?

Ans → 3.

∴ 3 given 8.

So, (7) ✓

a ⇔ g.



Puffin
Date
Page

Max. no. of flip flops → 8 (optimal)
→ ∞ (otherwise)

Observe: From state diagram
∃ 1 i/p, 1 o/p. So use binary coded flip flops → [like (0)10 (3 bits) → status of 1st flip flop = 0.]

Now, suppose we send an i/p sequence (data)

i/p (0)(1)0 1 0 1 0 1 1 0 1 0 0

Suppose we start from A

Suppose state A has flip flops in 0 0 0 states.

When 0 applied, a goes to (a) (o/p 0)

1 ↓ a goes to b. (o/p 0)

0 → i/p applied

c ← 0 — d ← 1 — c

(o/p 0) (o/p 0) (o/p → 0)

1 ↓

f —1→ f —0→ g —1→ f —0→ g —0→ a

(o/p → 1) (o/p 1) (o/p 0) (o/p 1) (o/p 0) (o/p 0)

So, net

states: a a b c d e f f g f g a } → diff^t forms of represent^n

output: 0 0 0 0 0 0 1 1 0 1 0 0

Equivalency defin^n

* Two diff^t states are same if same i/p. for these states will produce:
Same: next state (&) o/p.
or equivalent next state.

*

* If states are equal (say d & e are equal)
So, d can be replaced by e (or vice versa)
So, no. of flip flops can be reduced.

* State table for previous diagram

Pr. State	Next State x=0	Next State x=1	o/p state x=0	o/p state x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f x d	0	1
e	a	f x d	0	1
f x d	g x e	f x d	0	1
g x e	a	f x d	0	1

Checked again d & f are equal

Part ① equivalent

(cancel (replace with e)

So, 6 states left

↓ check again

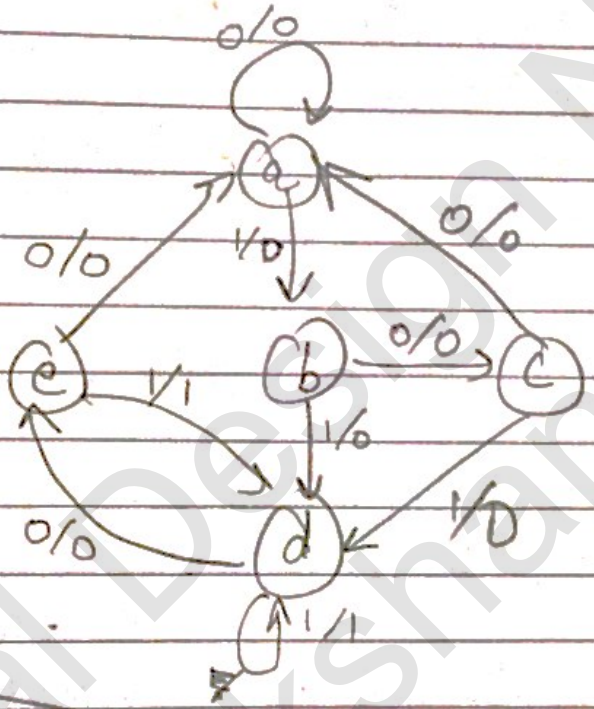
o/p when check done once

II

a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f x d	0	1
e	a	f x d	0	1
f x d	e	f x d	0	1

table when check done second time

III	a	a	b	0 0
	b	c	d	0 0
	c	a	d	0 0
	d	e	d	0 1
	e	a	d	0 1



★ State assignment

State	1. Binary	2. Gray code	3. One hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000.

no. of flip flop = no. of state. only one bit high (= Decoder o/p)

For previous ex, Binary state assignment assumed

Present	Next State		O/P	
	x=0	x=1	x=0	x=1
ABC	ABC	ABC		
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

* 3 flip flops → A, B, C.

What are we seeing?
→ Whats o/p when A=0, B=0, C=0 at x=1 ?

A=0=B=C corresponds to 1st i/p.

Now, for x=1, A → A B → B C → C'

⇒ 0 → 0 0 → 0 0 → 1

⇒ 000 → 001

⇒ next state

Now, for x=1, o/p = 0. (see table)

Ans

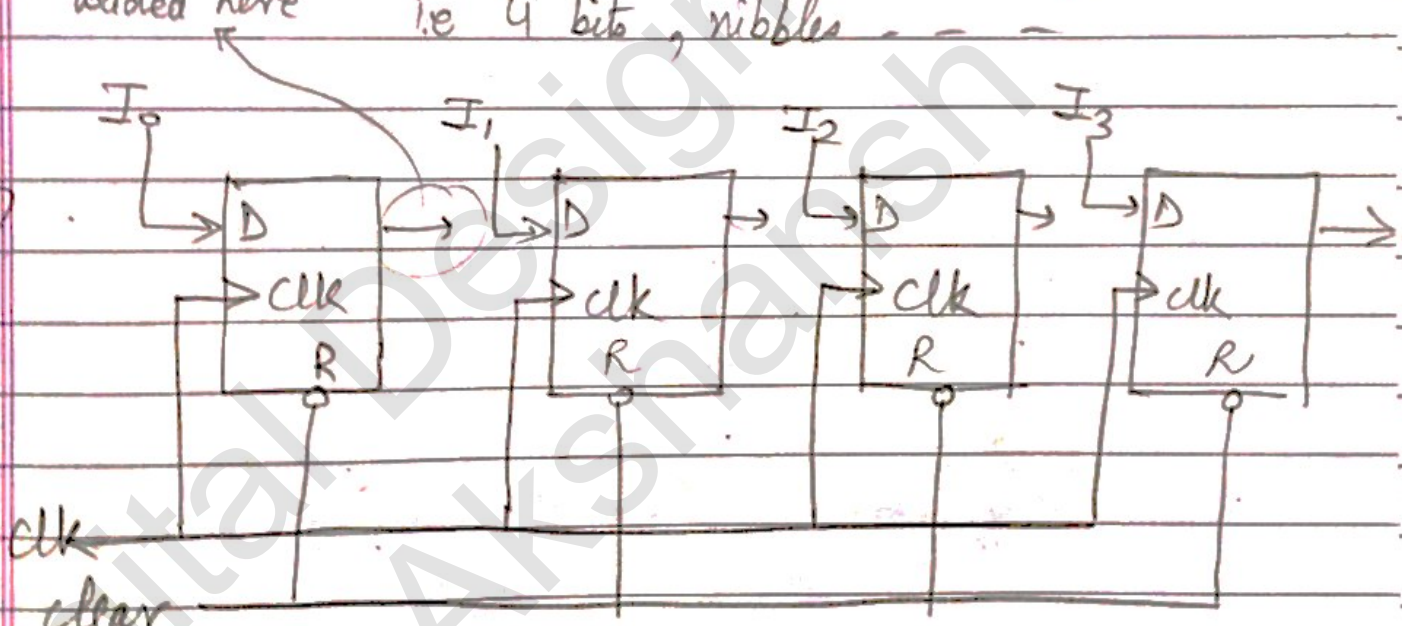
Date 25/11/12
Page

REGISTERS & COUNTERS

Group of flip flops connected in a predefined fashion.

REGISTER

- holds / registers some data
- flip flop stores 1 bit of info
- data is stored in terms of word bits i.e 4 bits, nibbles



→ 4 bit register with D flip flop
- with loading facility i.e loading data
- clear signal applied to R (Reset pins)
  If Clear = 0, all data in D flip flop is cleared to zero
- ✓ when i/p is applied, data at each i/p is loaded to o/p.

• eg: consider 8 bit data using hexadecimal digits 2o, 2 digits req

2 th like :-

2E
AF
AB
12
34
AE

• Data size is 8 bit in size
(4 bit + 4 bit)
1 hexadecimal no. req.

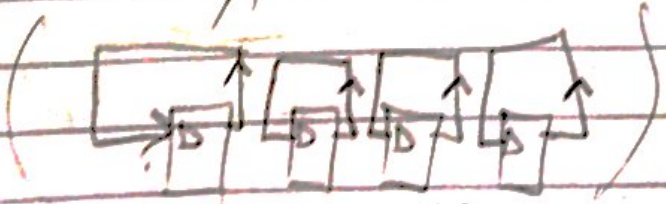
Data I'm providing :- $I_3\ I_2\ I_1\ I_0$

1 0 1 1 → (B)

This data represented by B

Q. how to hold the data (in the register) that is loaded
Clock is running, data is changing at i/p
But once loaded, I want to keep it same.

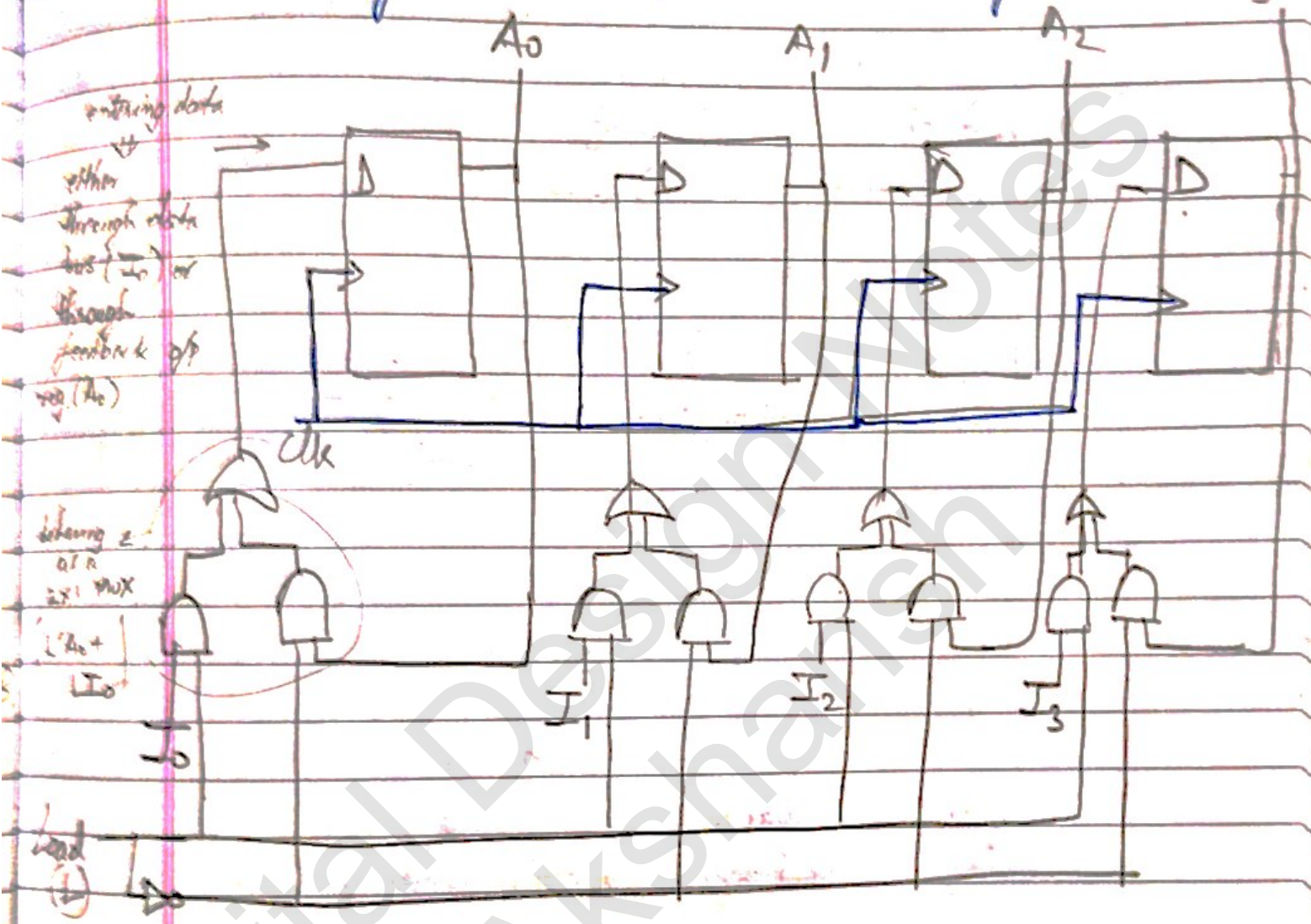
What to do :- Feedback final o/p to initial D i/p



This will make it possible

* Clear & Preset signals are applied directly at o/p Latch :- so, they are asynchronous signals.

- 4 bit register with parallel loading



- load is behaving as select line.

If load = 1, $I_0 = 1$, say, So, data goes to D flip flop i.e loading done & hence data is stored.

To hold data; don't let loading to take place

- Assume initial data is shifted right

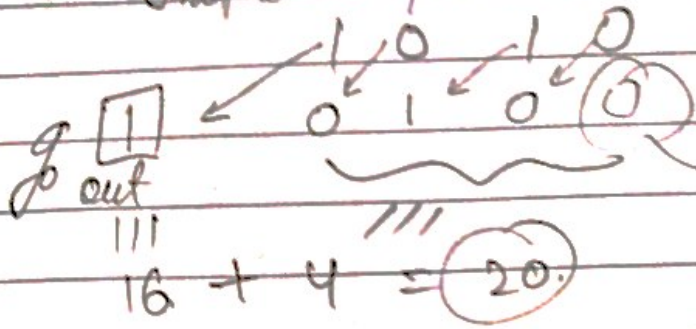
i.e, if $I_3\ I_2\ I_1\ I_0$

$1\ 0\ 1\ 0 \equiv 10$

Shift $0 \to 1\ 0 \to 1\ \boxed{0}$ goes out

$\equiv 5$ So, $\frac{10}{2}$

• Shifted left

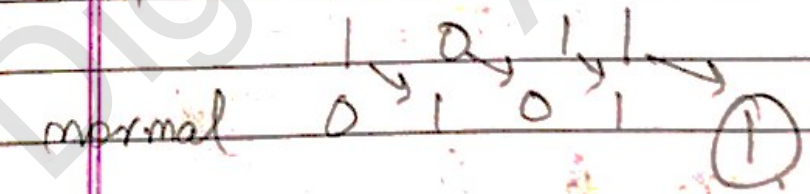


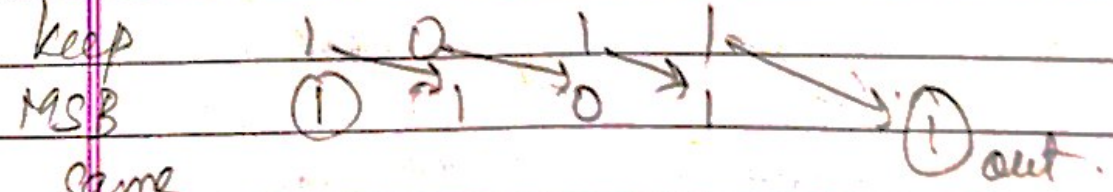
So, shift right ≡ Divide by 2.

bit going out = remainder.

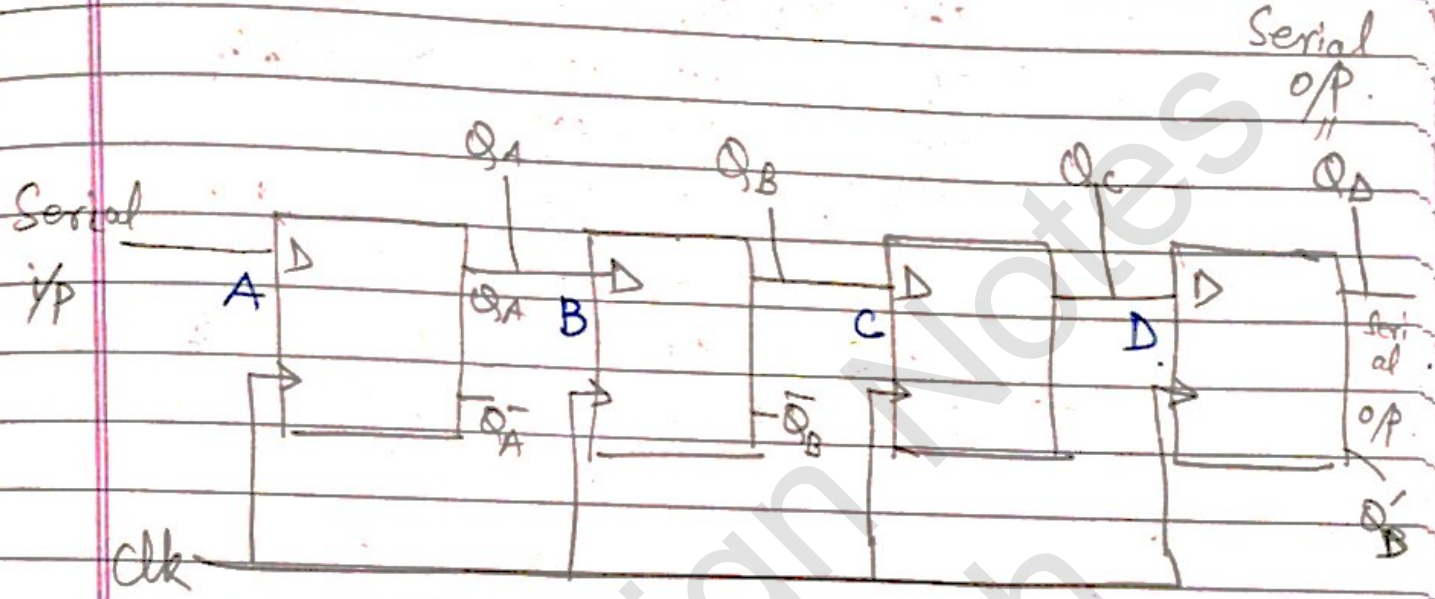
Shift left ≡ Multiply by 2

• Arithmetic shift right
↳ Shifting by keeping MSB same





- Suppose, word :— A B C D

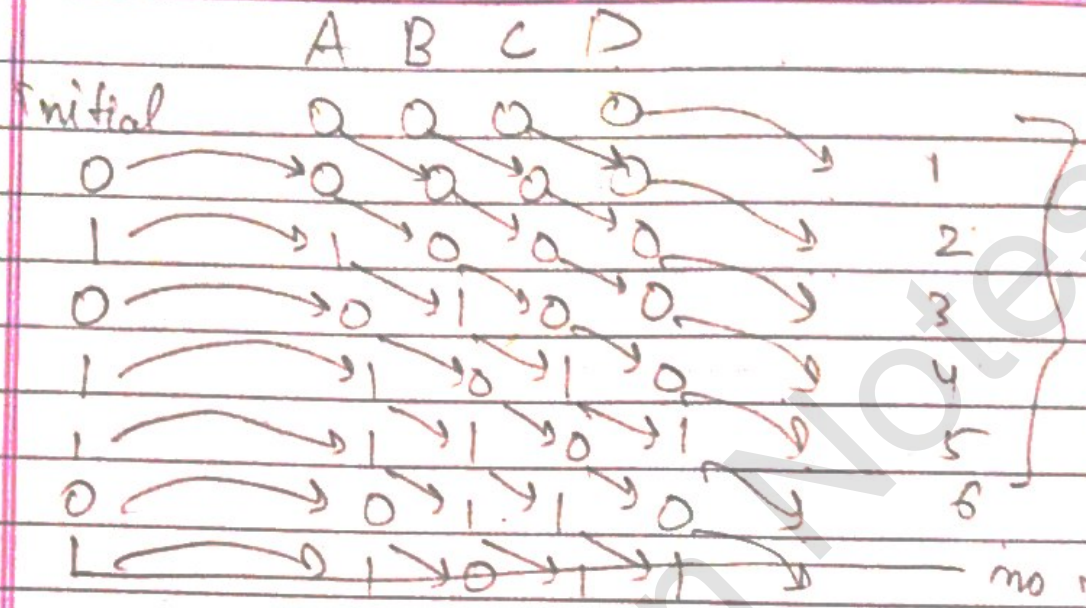


Shifting right from A to D
(See $Q_A \xrightarrow{to} Q_D$)
So $\div 2$

If circuit made like $Q_D \xrightarrow{to} Q_A$
$(\times 2)$

Here, o/p of flip flops (all) are being affected by common clock pulse $\Rightarrow$ synchronous sequential circuit.

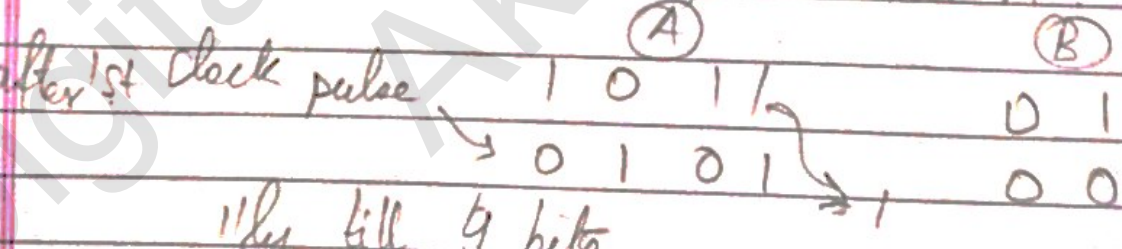
Q. Initially, register is cleared $\Rightarrow$ ABCD $\equiv$ 0000.
If i/p (at every clock pulse) is given as 0101101.
What will be data stored in register after 6 clock pulses.



* Given a shift register : SERIAL TRANSFER
What will be data available in registers A & B after 4 clock pulses.

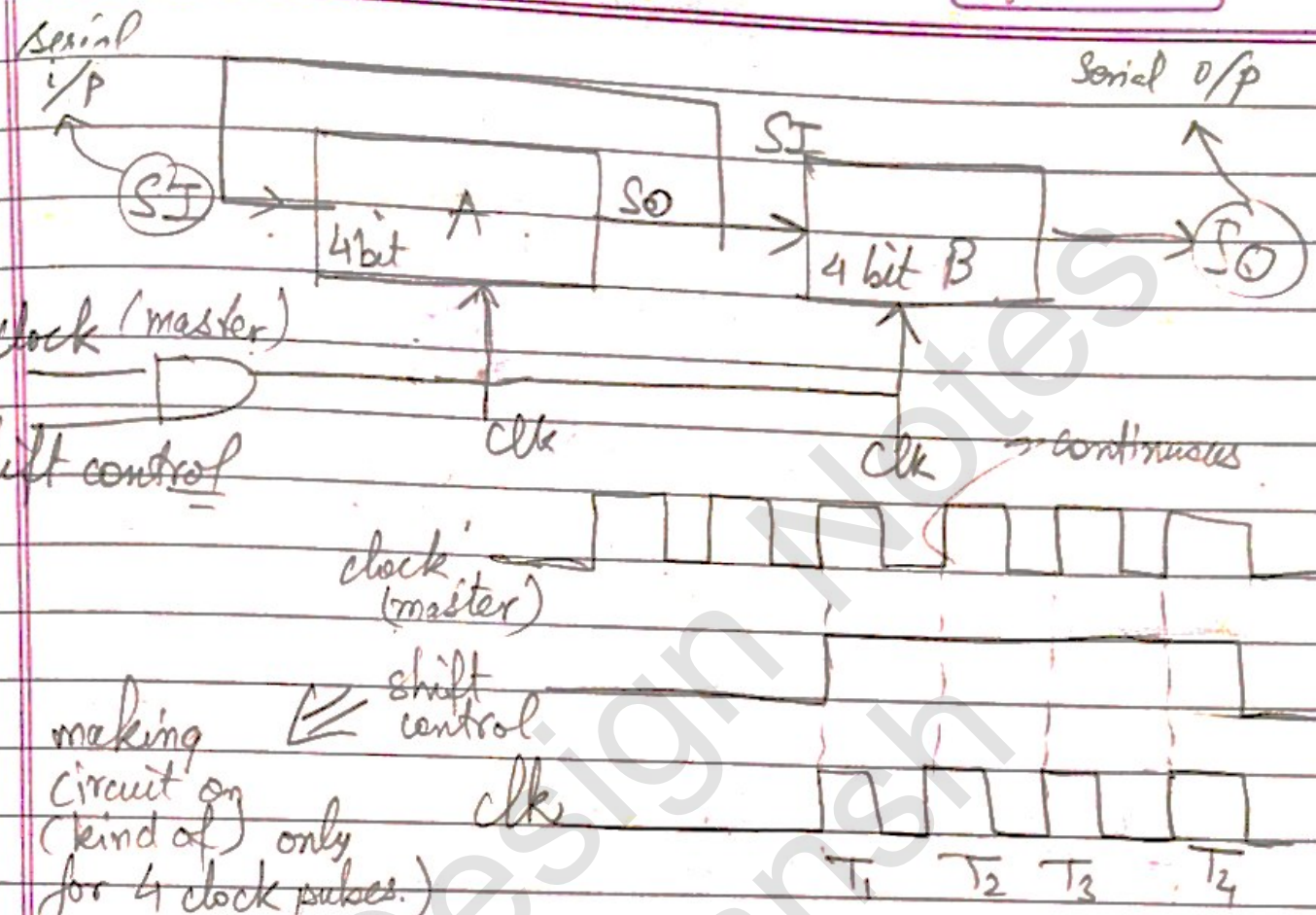
Given, initially A = 1011
B = 0110

Ans :- A remains same B becomes A



Why till 4 bits.

Where used :-
If we want to copy data in a microprocessor
Command used MOV B, A
(destin") (source)
So, such an arrangement is used.



making circuit on (kind of) only for 4 clock pulses.

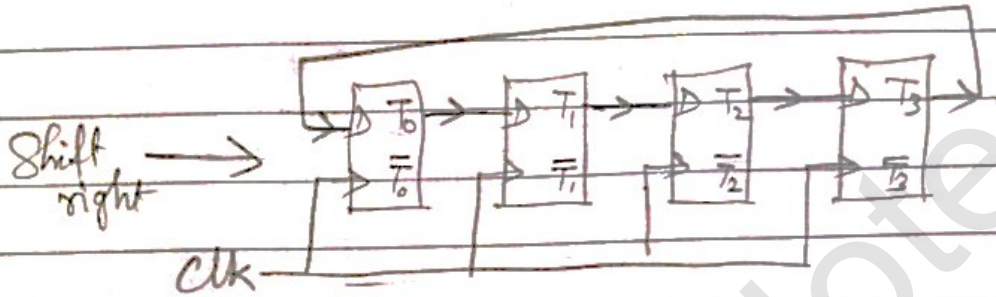
	A				B			
initial	1	0	1	1	0	0	1	0
T_1	1	1	0	1	1	0	0	1
T_2	1	1	1	0	1	1	0	0
T_3	0	1	1	1	0	1	1	0
T_4	1	0	1	1	1	0	1	1

initial A; we get back

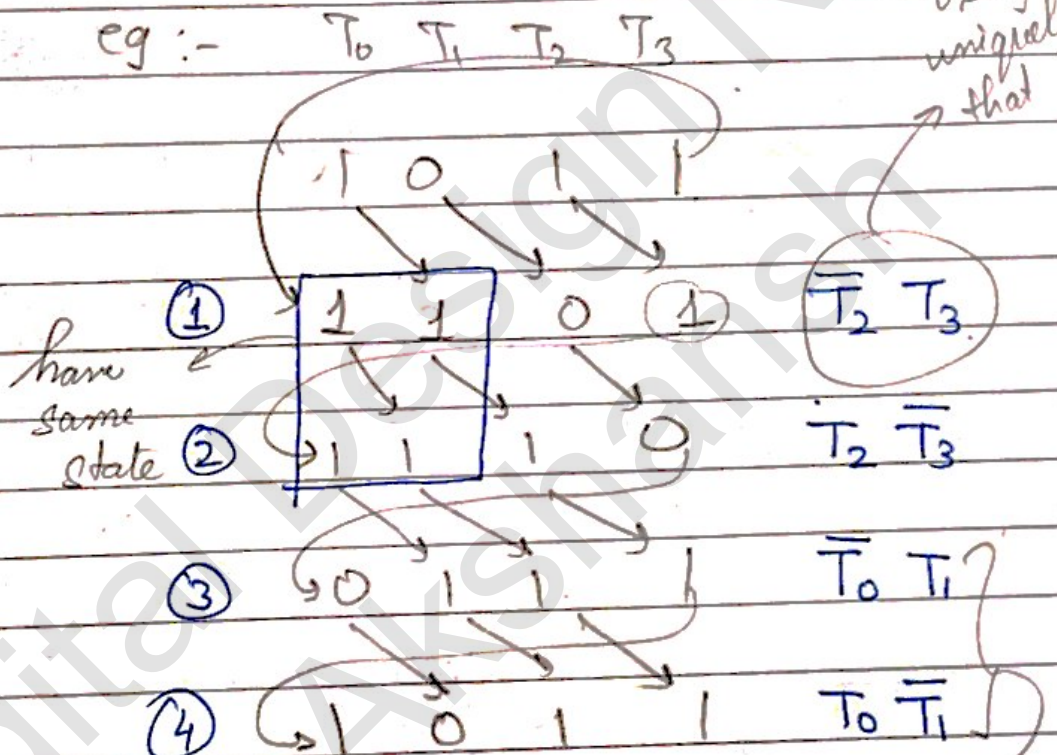
Value of A copied to B after 4 clock pulses.

Ring Counter : Feedback last serial i/p to first

Q) Given 4 bit shift register.
Feedback last bit $T_3 \longrightarrow T_0$.



eg :- T_0 T_1 T_2 T_3



* Feeding last bit to first one : called as RING counter.

in both of them, T_2 & T_3 are 1 & 1. So, to represent ③ & ④ uniquely, we find that T_0 & T_1 are different from all others (① & ②). So,

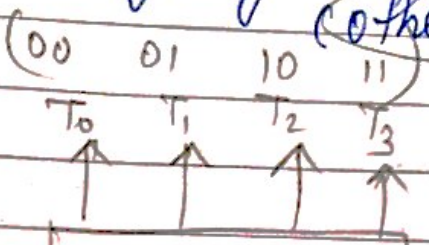
③ → $\overline{T_0}$ T_1 } unique way
& ④ → T_0 $\overline{T_1}$ } (Ily for ① & ②)

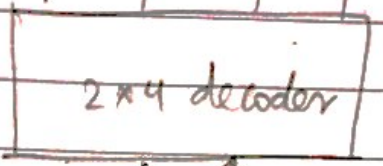
→ activating one flip flop (per clock pulse) & shifting its value to the corresponding flip flop (of next shift register). So, copying 4 bit of data from 1 → 2 shift register in 4 clock pulses.

Puffin
Date
Page

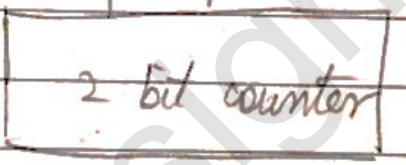
✡ Generate counting signals

(other way)



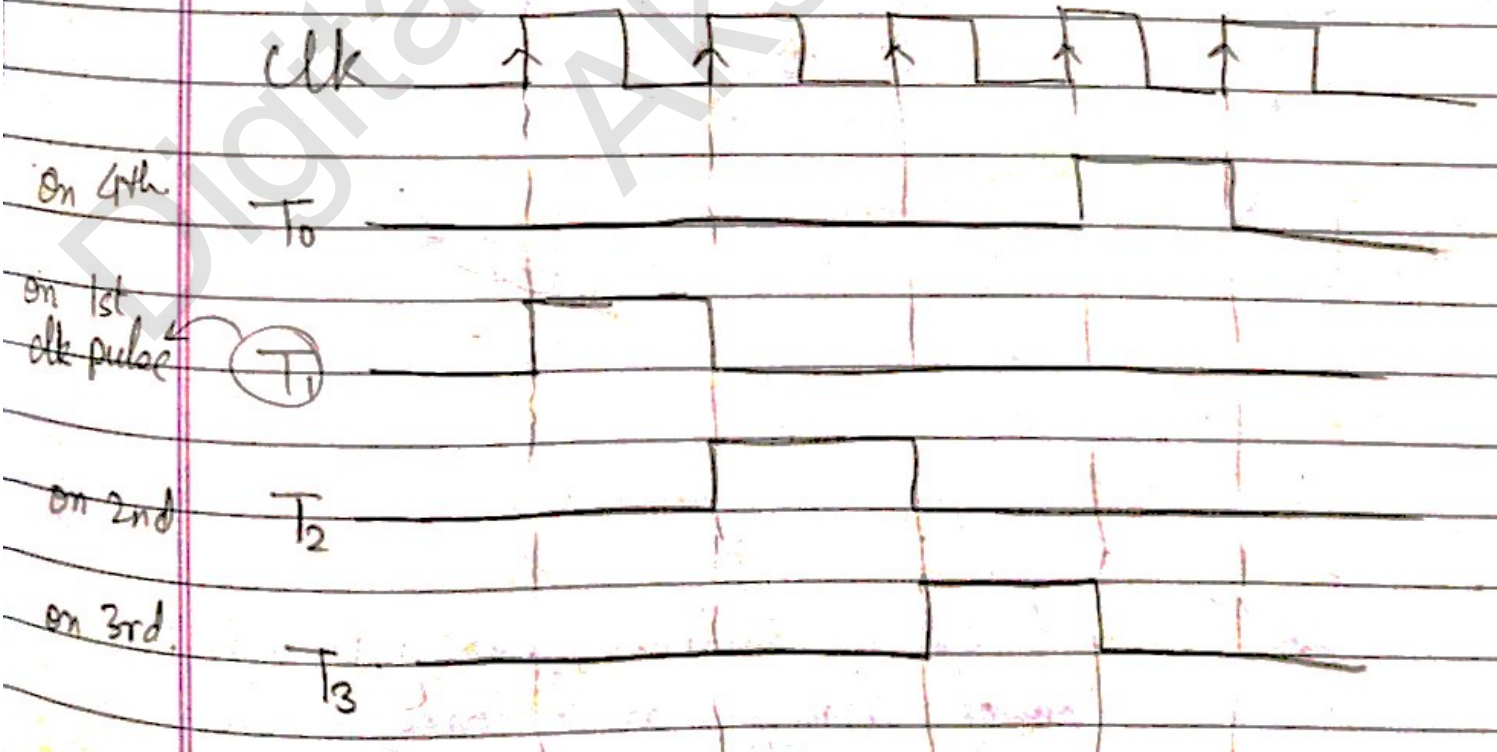


0 1 → T_1 activated
1 0 → T_2 "
1 1 → T_3 "
1 0 0 → T_0 "



on 1st clk pulse
0 1
⇒ T_1 goes high
2nd ⇒ 1 0
⇒ T_2
11 by others.

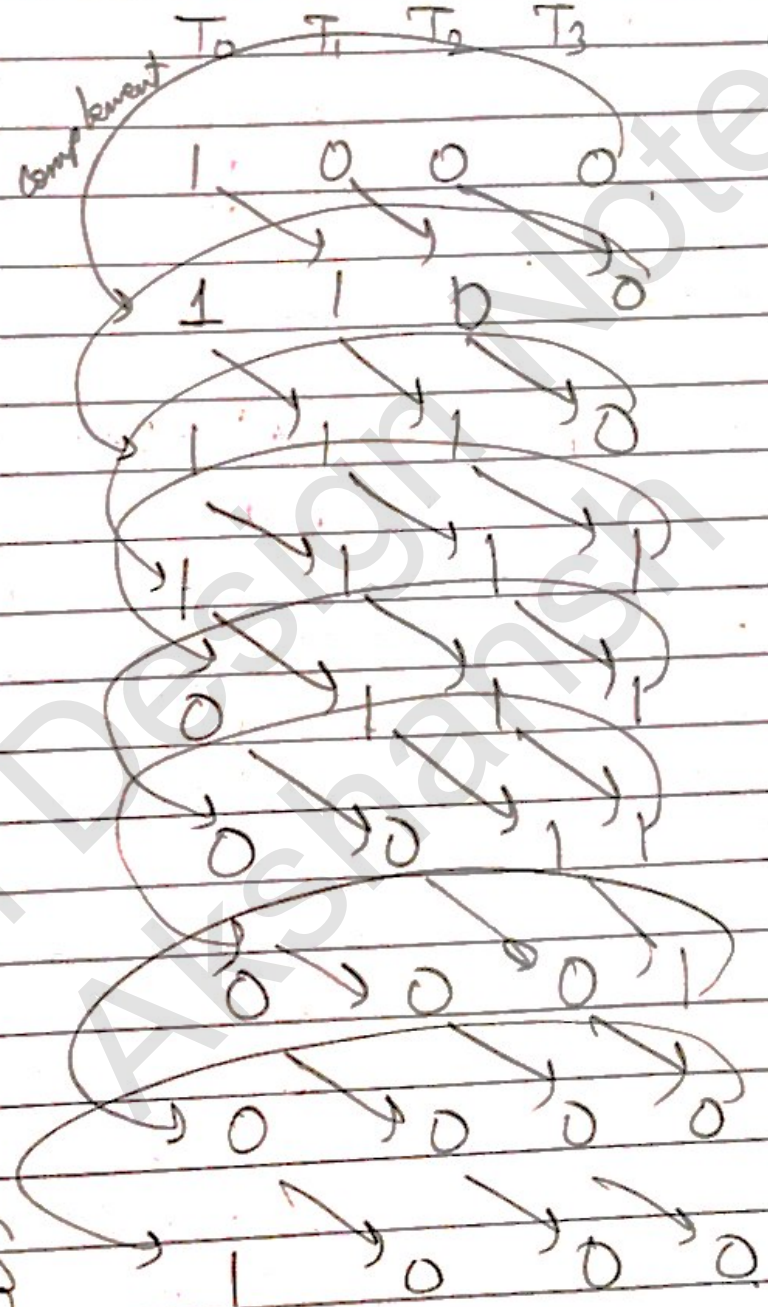
We need diff^t timing signals req^d to activate some part of circuitry at particular time



Johnson's ring counter or Switch tail ring counter :

Q. Feedback complement of last flip flop to first serial i/p.

eg. i/p = 1000.

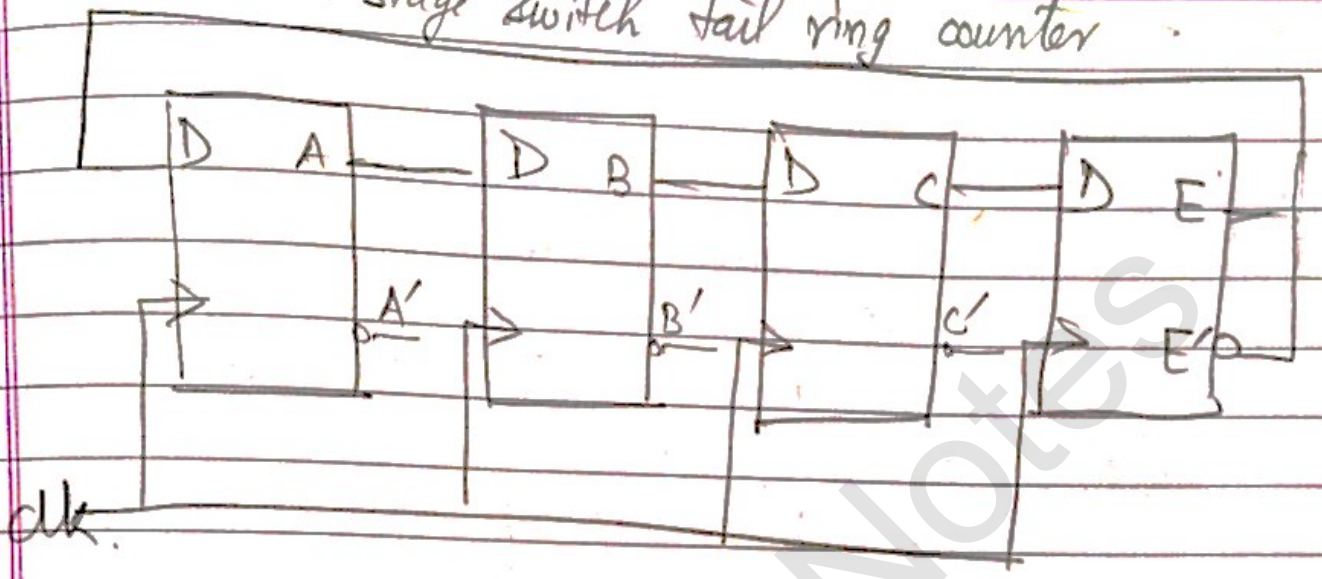


8 diff^t timing signals can be generated (i.e. 8 diff^t pns (see →) that can be used to activate sth else, can be used)

repitition (same as start)

* Given 4 bit ~~re~~ shift register (T_0 T_1 T_2 T_3)
Max. no. of timing signals that can be generated = $2^{\sqrt{4}} = 16$.

★ four stage switch tail ring counter



★ Count sequence & reqd decoding

Sequence no.	Flip flop o/ps A B C E	AND gate reqd for o/p
1	0 0 0 0	A' E'
2	1 0 0 0	A B'
3	1 1 0 0	B C'
4	1 1 1 0	C E'
5	1 1 1 1	A E
6	0 1 1 1	A' B
7	0 0 1 1	B' C
8	0 0 0 1	C' E

Any 2 i/p AND gate

written like before.

★ Where used?

We save hardware (IC's) by increasing no. of clk pulses.
(More time consuming).

Serial : Bit by bit

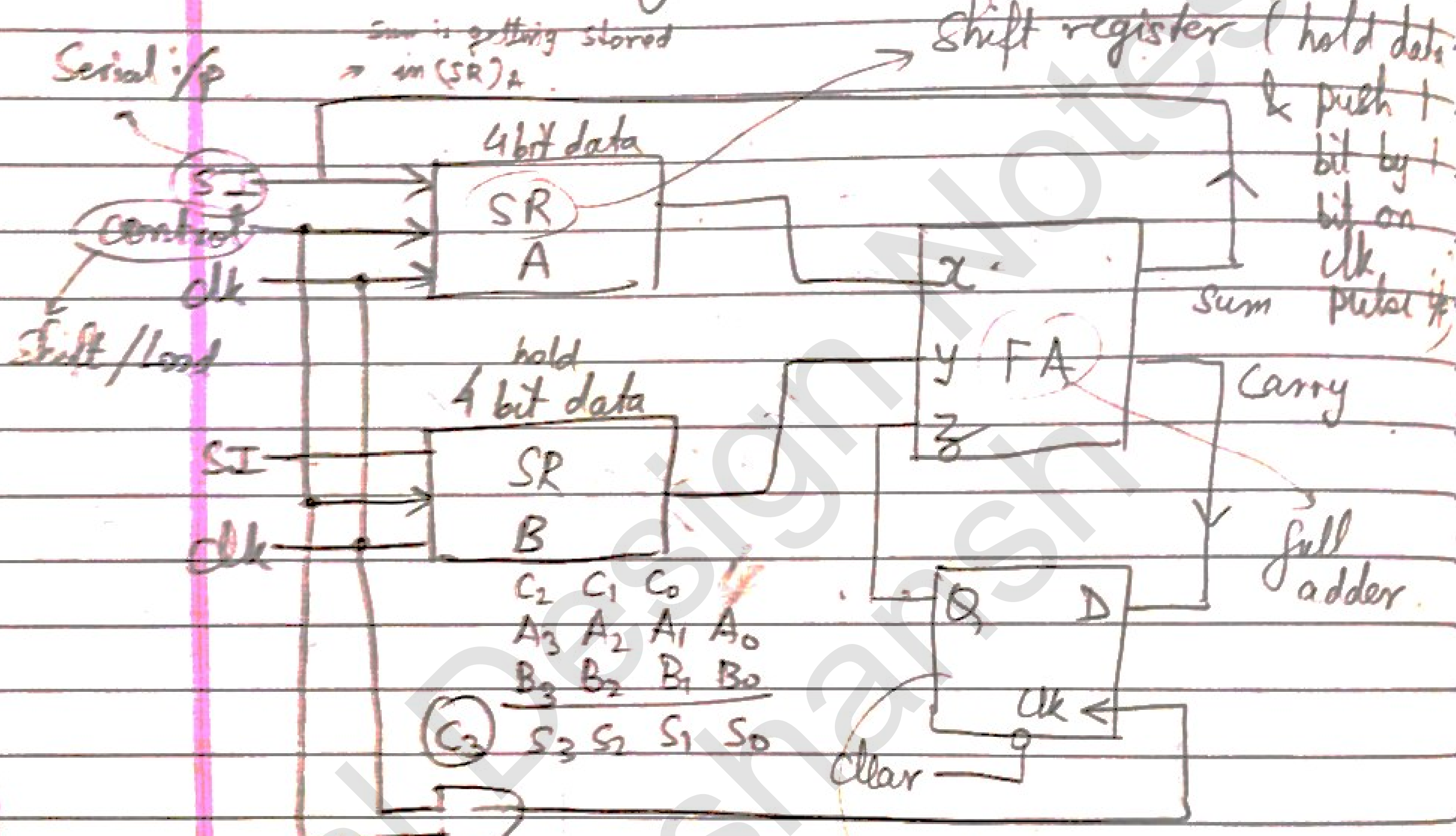
Parallel : All together

SR : Shift register

Suppose :- 1 0 0 1 → Augend
+ 0 1 1 1 → Addend
① 0 0 0 0 → sum + carry

SERIAL ADDER

Q. Add 2 four bit words



SR has many operations
So, control clk pulse to tell SR what to do :- shift or hold

→ D flip flop for storing carry (works on every carry)

★ State table

o/p carry in → D flip flop

written using concept

Present state z	i/p x	i/p y	Next State Q_D	o/p Sum	i/ps to JK: J	K
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

* Sum after through $(SR)_A$. Then, addition happens.

n:- $(SR)_A$ & D flip flop cleared to 0. Augend is loaded in $(SR)_B$ (say, Augend = 1001) SI. Then, with 4 clock pulses, 0000 → 1001 → 1001 gets saved/stored in addend is loaded in $(SR)_B$ through its SI & actual reg'd

Puffin
Date
Page

* Sum after through $(SR)_A$. Then, addition happens. So, $SR(A)$ stores final sum

* Now, we want to replace D flip flop by J-K

→METHOD how to design circuit ?

Idea :- What should be J & K i/p's if my present & next states are given (↙ table)

written using state table of JK

i/p carry	$Q_{(t)}$	$Q_{(t+1)}$ → o/p carry	J	K
0	0	0	0	X
		1	1	
0	1	1	1	X
		0		
1	0	0	X	1
		1		
1	1	1	X	0
		0		

don't care

⇓ net.

$Q_{(t)}$	$Q_{(t+1)}$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

* Note:- If carry goes from Q_t to Q_{t+1}, then J & K i/ps should be 0 & 1, say (done in previous pg.)

Imp. So, if we implement fns for J & K & give i/ps for J & K (in terms of x & y), then, carry must have gone from Q_t to Q_{t+1}. We know that Q_{t+1} was giving fn of o/p carry. So, o/p of JK flip flop = o/p carry

Puffin
Date
Page

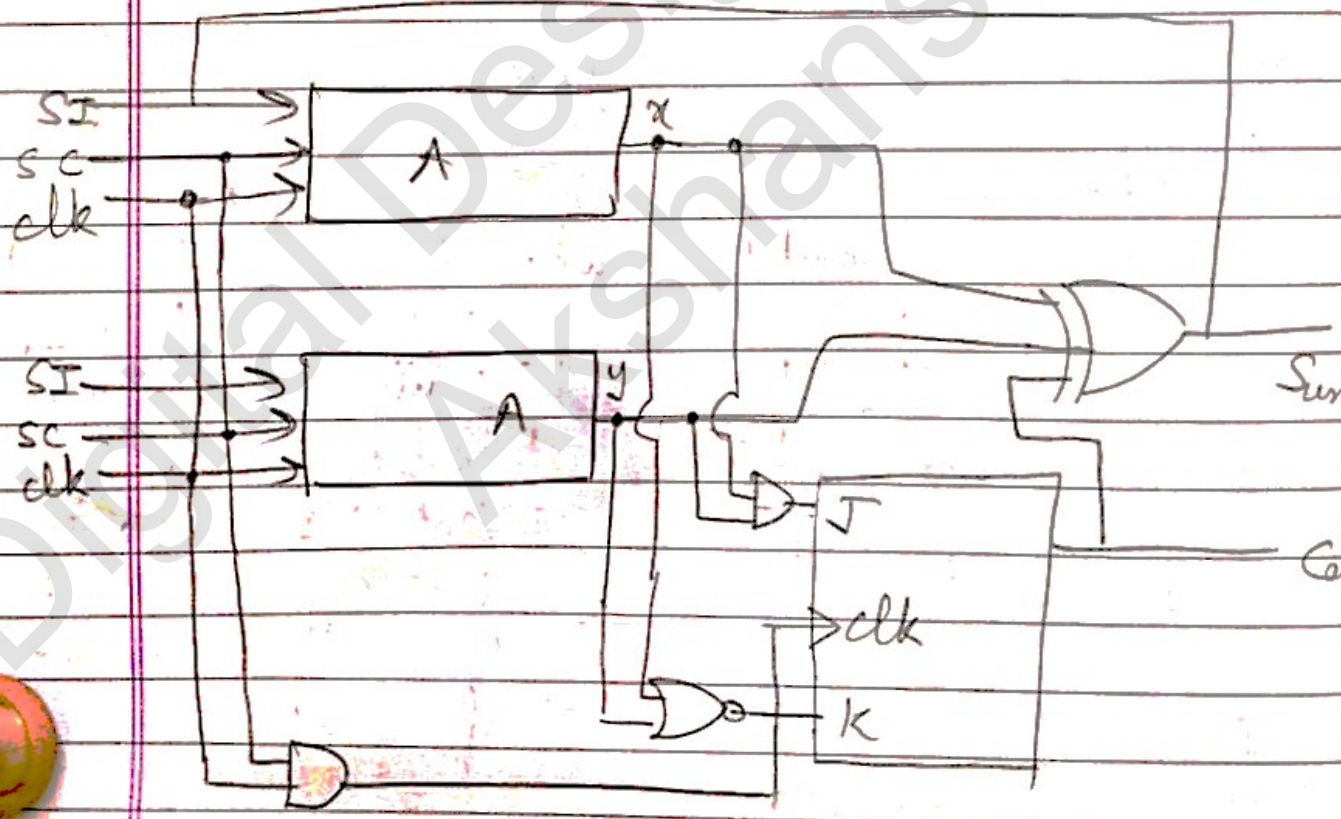
Now, find expression of J & K separately (k-map) in terms of x & y.

$\Rightarrow J = xy$, $K = \bar{x}\bar{y} = (x+y)'$

→ written like minterms.
→ o/p to JK flip flop gives carry fn^. (See previous table)

$S = x \oplus y \oplus \otimes$

↓ Circuit

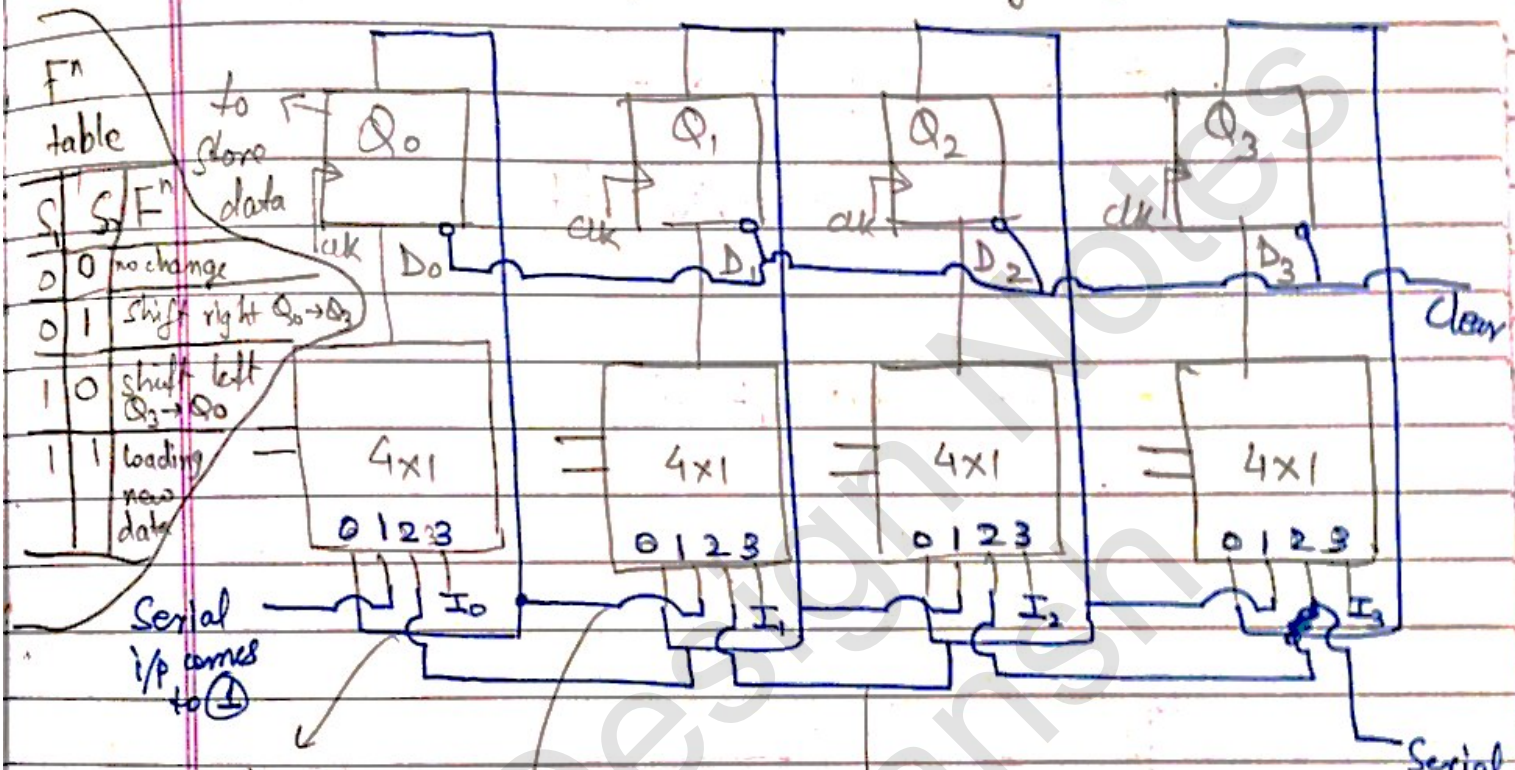


(Last experiment)

* Using J-K flip flop reduces the use of a full adder IC.

Idea * Use of D flip flop good: Use multiplexer to select 1 fn out of 4 (say) fns done by SR.

Q. Design: Universal Shift register.
(right, left everything should do)



when select lines are 00, data shouldn't change. So, Bring Q_0 o/p back to mux i/p.

when Select lines are 01, shift right. So, Q_0 go to Q_1. So, give Q_0 to (1) of Q_1 & by others (1) of Q_0 will get serial i/p

when select line is 10, shift left. So, Q_3 goes to Q_2 So, give Q_3 to (2) of Q_2. (2) of Q_3 will be left out & So, will get serial i/p.

when select lines are 11, new data has to be loaded. So, I_0, I_1, I_2 & I_3 are given to (3)

* All D flip flops are connected to clear pins. so as to clear data (if reqd). ⟶ shown
It can also be connected to preset pin ⟶ not shown

* Asynchronous sequential circuit :-
Doesn't depend on master clock.
(clock pulse not given common to all)

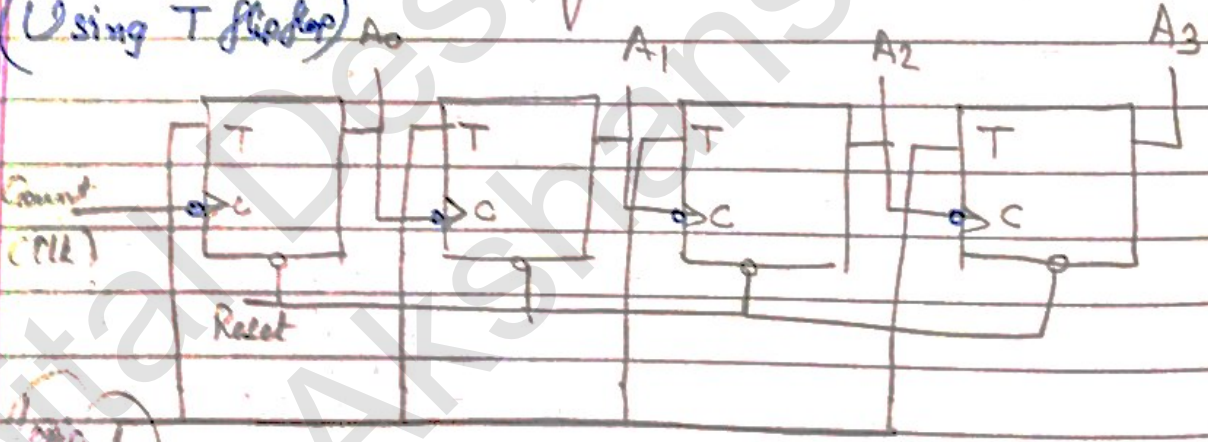
Puffin
Date 28/11/12
Page

COUNTER

Counts the clock pulses in a predefined fashion.
Each count sequence is refused to as a state.

* Synchronous Sequential circuit
State of flip flop changes with a master clock.
→ synchronised with i/p clock pulse (master clock)

* RIPPLE counter → Asynchronous
(Using T flip flop)



Note :- For knowing any flip flop (in), Q is regd as clk pulse i/p. Now, as clk given to → So, on i/p of zero, flip flop will become on in terms of clk pulse, when clk becomes Δ to ↓ flip flop activates

→ T=1 given to all → $\bar{Q}$ o/p.

Suppose $A_0 A_1 A_2 A_3$ = 0 0 0 0 initially
i/ps → permanently 1.

When clk = 1 applied at -ve edge
$A_0 \rightarrow 0 \rightarrow 1$

* In a counter, the clk pulses are responsible for the data/states stored in the flip flop.

☆ An n-bit register has n flip flops which generates 2^n states (from 0 — $2^n - 1$)
A counter is used to count these states (binary counter)

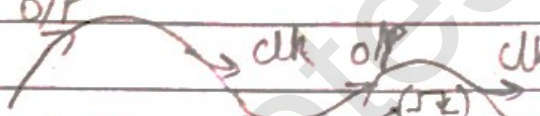
Puffin
Date
Page

clk	A_3 A_2 A_1 A_0
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

note:- A_1 becomes 1 when A_0 goes from 1 to 0.
" A_1, A_2 & A_3.

Circuit will repeat after 15 clk pulses.

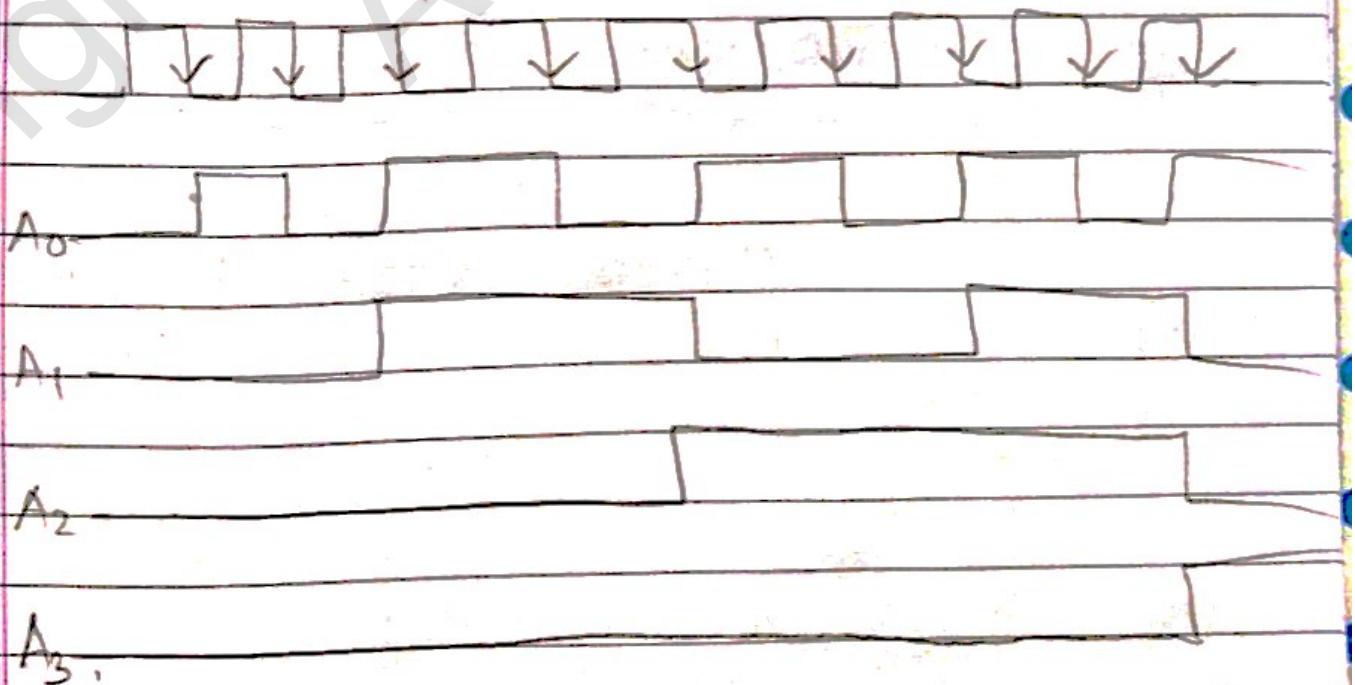
Looks like a ripple.



Logic: When clk i/p goes from 1 to 0 (or, 1 can say that i/p 0 is applied to 1st T flip flop), o/p A_0 changes from 0 → 1 (∵ initially 0000 was stored in all flip flops) So, o/p got is 0001 (meaning, 1st clk pulse is counted). Now, when 2nd clk pulse is applied (↓ t) or 0 applied to 1st T flip flop, A_0 becomes 1 to 0 & 0 gets stored in it correspondingly. A_0 being i/p clk pulse for next flip flop activates it (∵ it will go from 1 to ⓪). So, that will make o/p A_1 = 0 to ①. On applic^n of 3rd clk pulse, A_0 → 0 → ① & ∵ A_0 = 1, So, 2nd flip flop isn't activated & o/p = 0011 (= 3). So, clk pulses are being counted.

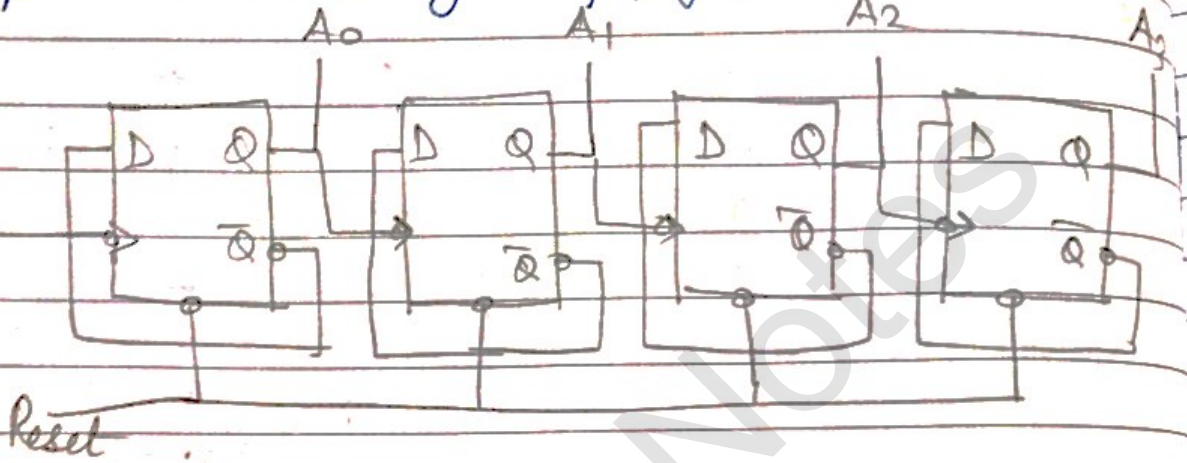
When clk is applied after (1111) then A_0 goes from 1 → ⓪, A_0 becoming 0 will make clk i/p at 2nd flip flop 0 which will activate 2nd flip flop & make A_1 from 1 to ⓪. That will activate 3rd flip flop & make A_2 1 → ⓪ → will activate 4th flip flop & make A_3 → 1 → ⓪. So, A_0 A_1 A_2 A_3 all become 0. Hence, counter starts counting again & this can count only till 15. (1111)

Timing Diagram



✱ Ripple Counter (Using D flip flop)

T=1
$\bar{Q}$ to D.
So, D used to get T

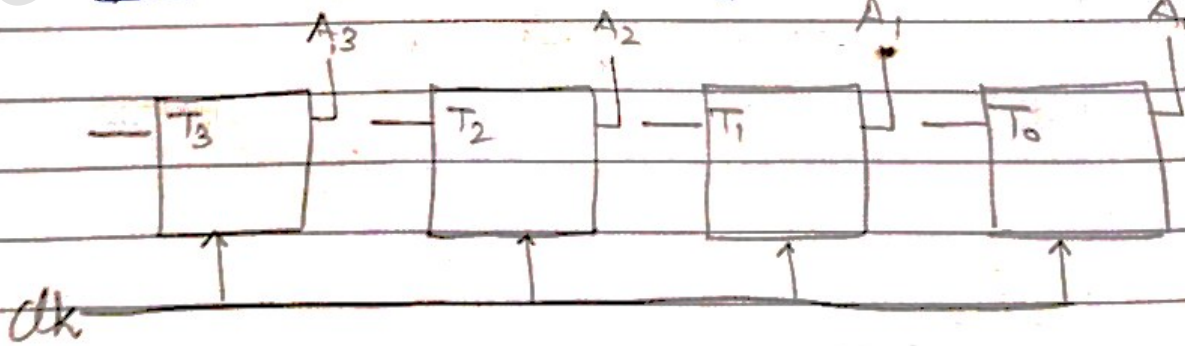


Same thing done like T flip flop.
(repitition after 15th clock pulse)

Q. Why ∃ a bubble before each clock i/p (→○) ?

Ans:- Suppose bubble not there. Then, when 1st clk pulse applied i.e (↑) then, that basically means i/p from clk pulse will go from 0 to 1. i.e, I'm giving 1 as i/p to clk pulse. Giving 1 as i/p, the o/p A_0=1 Now, when next clk is applied (1 is applied), A_0 becomes 1. So, basically, a counter counted the clock pulse (000 (1) A_0) on the 2nd clk pulse. What's it's use then!

✱ SYNCHRONOUS FLIP FLOPS



$T_0 = 1$ ∵ $A_0 \rightarrow A_{0(t+1)}$ is toggled

$T_3 = 0$ ∵ $A_3 \rightarrow A_{t+1}$ remains same.

Present state | Nexst state

A_3	A_2	A_1	A_0	$A_{2(t+1)}$	$A_{2(t+1)}$	$A_{1(t+1)}$	$A_{0(t+1)}$	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	1	0	1	0	0	0	1	1
1	0	1	0	1	0	1	1	0	0	0	1
1	0	1	1	1	1	0	0	0	1	1	1
1	1	0	0	1	1	0	1	0	0	0	1
1	1	0	1	1	1	1	0	0	0	1	1
1	1	1	0	1	1	1	1	0	0	0	1
1	1	1	1	0	0	0	0	1	1	1	1

(carry)

Gw get o/p → $T_0 = 1$

$T_1 = A_0$

$T_2 = A_0 A_1$

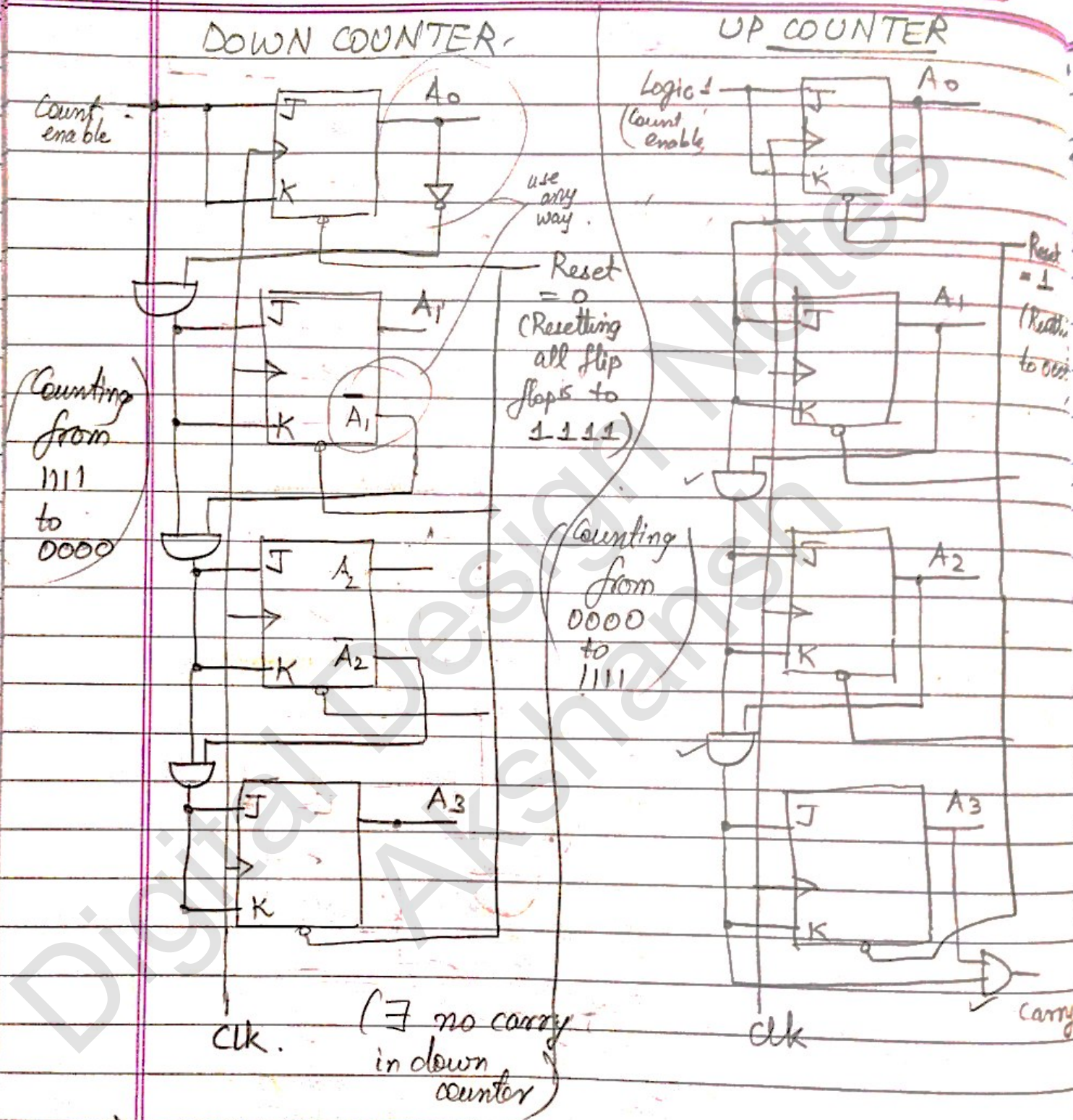
$T_3 = A_0 A_1 A_2$

Carry = $A_0 A_1 A_2 A_3$ → All 4 bits high.

Design procedure

(fn formed by seeing how $T_3 T_2 T_1 T_0$ was varying)

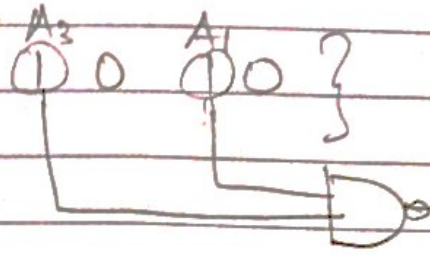
✱ 4 bit binary counter.



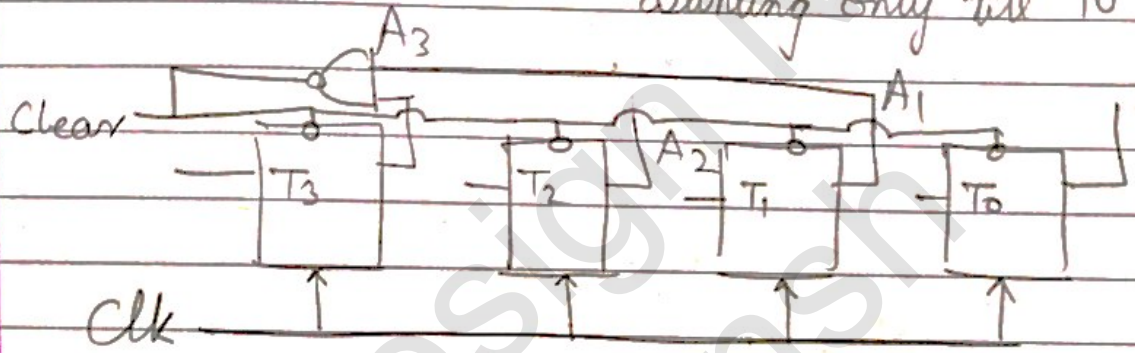
✱ Make 4 bit binary counter (0–15) → Decade counter (0–10)

- Idea: Use clear pins.

10th clock pulse



Use this to clear pins of all gates. That will clear all data. So, counting only till 10.



A_3 & A_1 = 1. So that will be producing a carry & can be given as i/p to clock.

Note

* The present state of previous flip flop goes to the next flip flop on applic^n of clock pulse

Note ✱ V. Imp

While making/implementing any circuit using any flip flop, check for what should be the i/p to the flip flops.

For D flip flop, i/p = next state ⇒ No need to make separate column for i/p.

JK flip flop or T flip flop, i/p is calculated seeing change in present & next state ⇒ Make separate column → implement using K-map.

☆ Down counting

present state to next state

A_3	A_2	A_1	A_0	T_0	T_1	T_2	T_3
1	1	1	1	1	0	0	0
1	1	1	0	1	1	0	0
1	1	0	1	1	0	0	0
1	1	0	0	1	1	1	0
1	0	1	1	1	0	0	0
1	0	1	0	1	1	0	0
1	0	0	1	1	0	0	0
1	0	0	0	1	1	1	1
0	1	1	1	1	0	0	0
0	1	1	0	1	1	0	0
0	1	0	1	1	0	0	0
0	1	0	0	1	1	1	0
0	0	1	1	1	0	0	0
0	0	1	0	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	0	1	1	1	1

A_1 A_0 change every time

Comparing :- $T_0 = 1$

expressions $T_1 = \overline{A_0}$

$T_2 = \overline{A_0}\,\overline{A_1}$

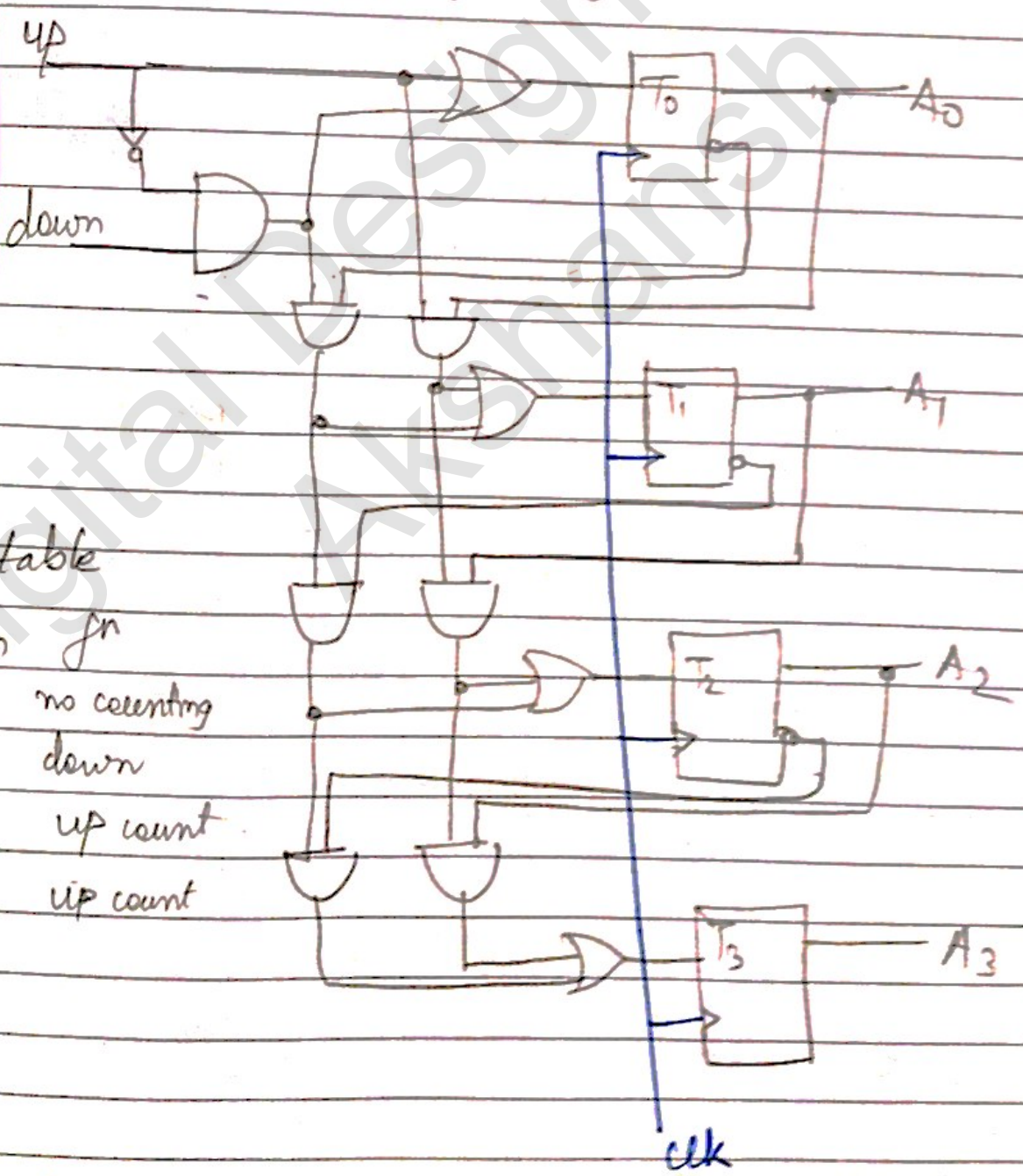
$T_3 = \overline{A_0}\,\overline{A_1}\,\overline{A_2}$

Puffin
Date
Page

* Make same circuit for upcounting & down counting.

Suppose we give only 1 control line.
(like 1 for up & 0 for down)
↳ Possible ✓

Disadvantage: It will be continuously counting, either up/down. We can't stop counting.
So, use 2 signals for up & down.



J^n table

up	down	J^n
0	0	no counting
0	1	down
1	0	up count
1	1	up count

→ Done completely later.

Puffin
Date
Page

IGNORE

★ Design 3 bit binary counter using D flip flop.

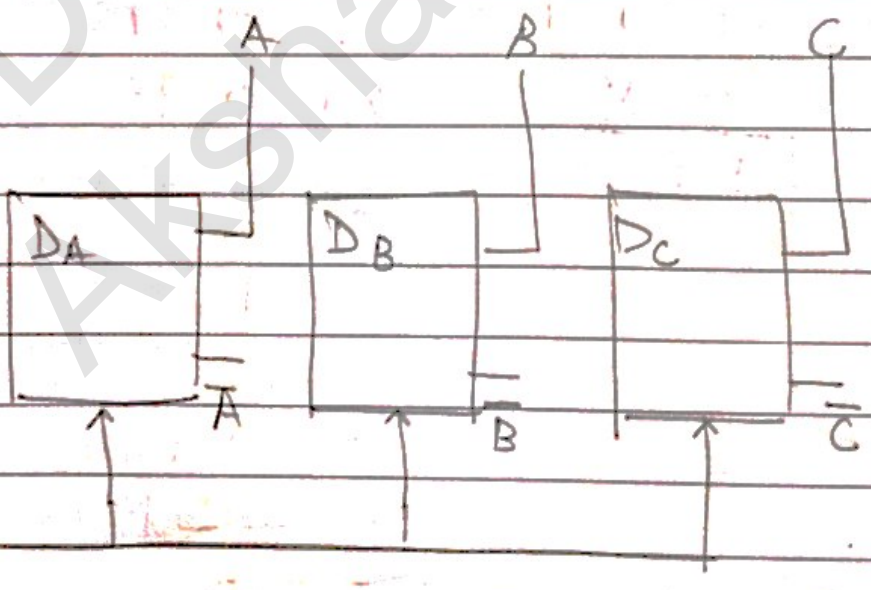
Present state			Next state		
A	B	C	A_{t+1}	B_{t+1}	C_{t+1}
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

$A_{(t+1)} = D_A$

$B_{(t+1)} = D_B$

$C_{(t+1)} = D_C$

found using K-map



Circuit designed by implementing D_A, D_B & D_C on this circuit.

☆ BCD counter using T-flip flop.

Present A_3	A_2	A_1	A_0	Next A_3 $t+1$	A_2 $t+1$	A_1 $t+1$	A_0 $t+1$	T_0	T_1	T_2	T_3
0	0	0	0	0	0	0	1	1	0	0	0
0	0	0	1	0	0	1	0	1	1	0	0
0	0	1	0	0	0	1	1	1	0	0	0
0	0	1	1	0	1	0	0	1	1	1	0
0	1	0	0	0	1	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	1	0	0
0	1	1	0	0	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	1	0	0	0
1	0	0	1	0	0	0	0	1	1	0	1
0	0	0	0								

∵ $A_3 \to A_{3\,t+1}$ as 0 to 0
So, $T_3 = 0$ (no change)

∵ $A_2 \to A_{2\,t+1}$ from 1 to 0
So, $T_2 = 1$ (flip)

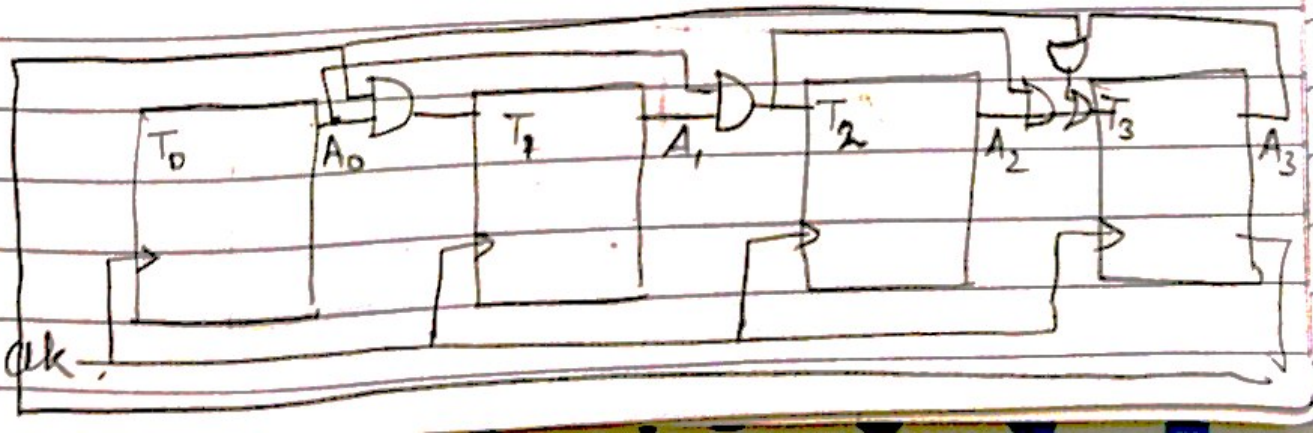
$T_0 = 1$

$T_2 = A_0 A_1$

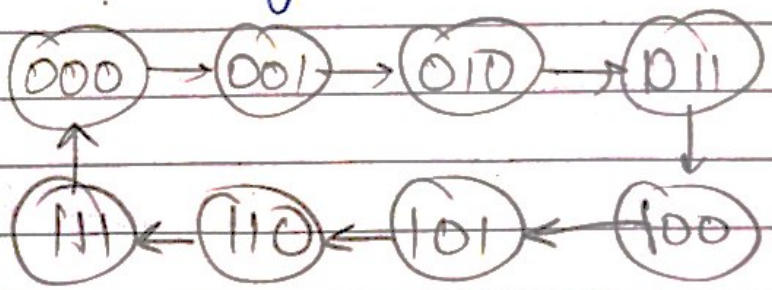
$T_1 = A_0 \overline{A_3}$

$T_3 = A_0 A_1 A_2 + A_0 A_3$

$C = A_0 A_3$



✡ 3 bit binary counter using D flip flop.



Present			Next			C_{t+1}	$D_A = A_+$	$D_B = B_+$	$C_+ = D_C$
A	B	C							
0	0	0	0	0	1				
0	0	1	0	1	0				
0	1	0	0	1	1				
0	1	1	1	0	0				
1	0	0	1	0	1				
1	0	1	1	1	0				
1	1	0	1	1	1				
1	1	1	0	0	0				

∴ D flip flop duplicates

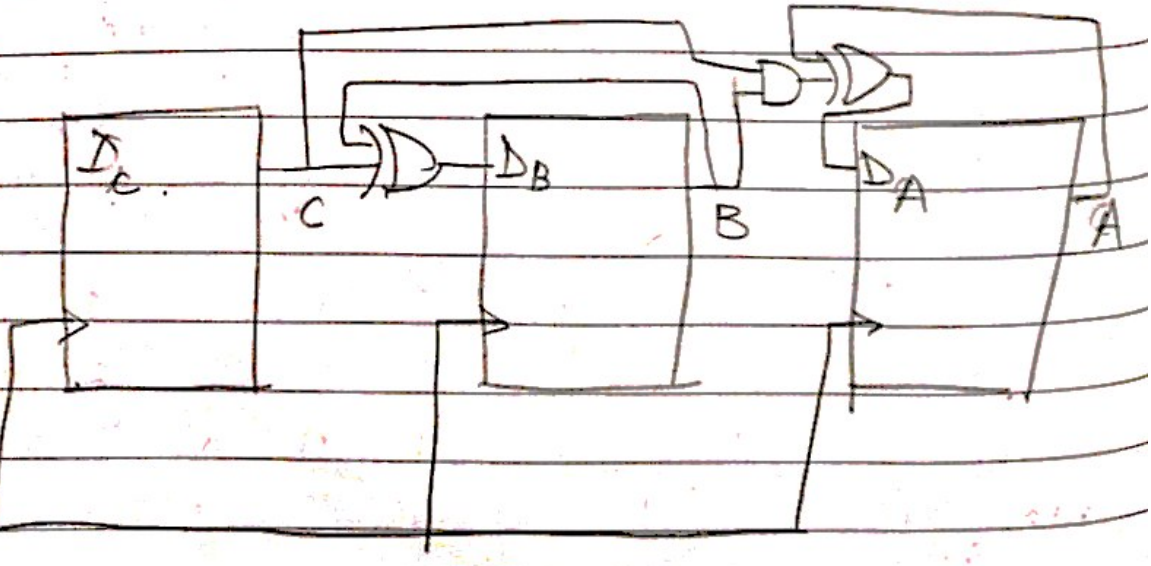
(No need for f/f)

$D_C = C_{t+1} = \overline{C}$

$D_B = B_{t+1} = B \oplus C$

$D_A = A \oplus (BC)$

f^n found using K-map



☆ General thing seen :-

Design Procedure
easy
tougher
toughest

D
T
JK

Circuit
complex
less complex
simple

Puffin
Date
Page

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

$B_{t+1} = D_B = B\bar{C} + \bar{B}C$
$= B \oplus C$

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

$A_{t+1} = D_A = \bar{A}BC + A\bar{C} + A\bar{B}$
$= A(\bar{B}+\bar{C}) + \bar{A}BC$
$= A(\overline{BC}) + \bar{A}(BC)$
$= A \oplus (BC)$

☆ Design of counters using unused states.

Given JK flip flop.

Q_t	Q_{t+1}	J	K
0	0	0	ϕ → 0 or 1 (≡ X)
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

	Present A B C	Next A_+ B_+ C_+	i/p J_A K_A	J_B K_B	J_C K_C
	0 0 0	X X X	X X	X X	X X
1	0 0 1	0 1 0	0 (0 or 1)	1 ϕ	ϕ 1
2	0 1 0	1 0 1	1 (0 or 1)	ϕ 1	1 ϕ
	0 1 1	X X X	X X	X X	X X
	1 0 0	X X X	X X	X X	X X
	1 0 1	1 1 1	ϕ 0	ϕ 1	ϕ 0
	1 1 0	X X X	X X	X X	X X
7	1 1 1	0 0 1	ϕ 1	ϕ 1	ϕ 0

max. count = 7 5

So, min flip flop req^d = 3

1st thing to do.

1, 2, 5, 7 → repeated continuously

Here, I have taken counter only for 1, 2, 5, 7 states. So, when clk is applied when i/p = 2, next state should be 5 (not 3)

Puffin
Date
Page

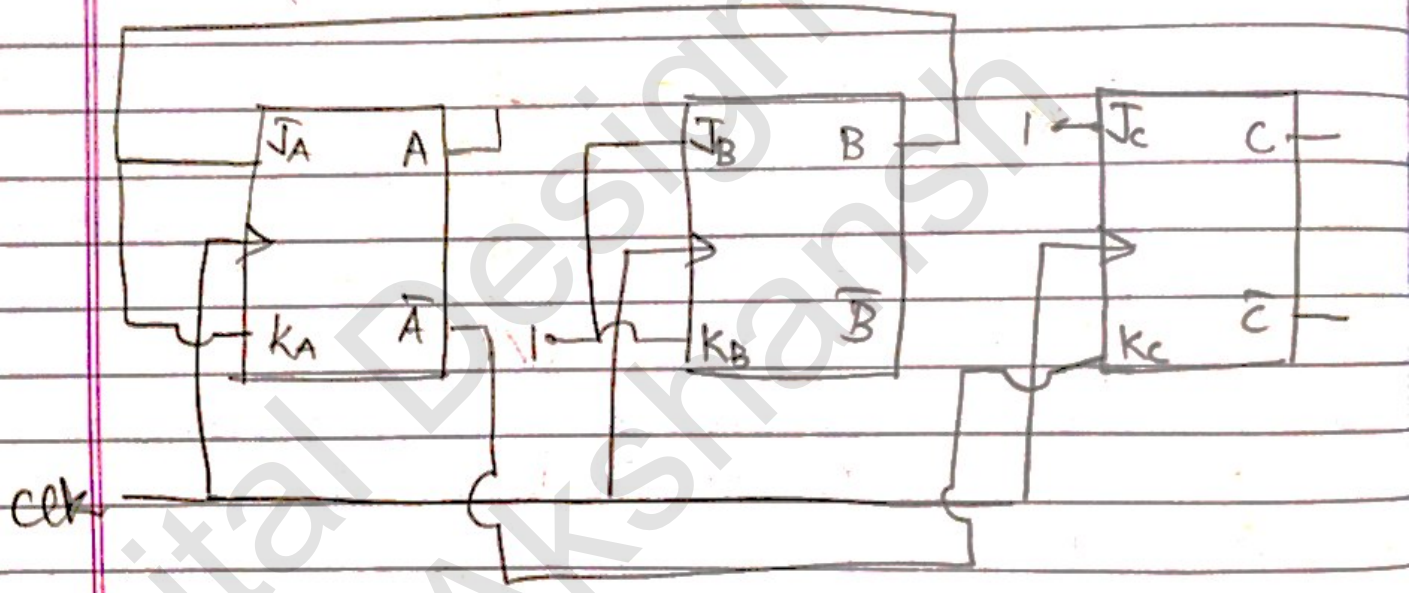
Using i/p's, write eq^ns :-

$J_B = K_B = 1$ (i.e. its either 1 or don't care) So, take 1.

$J_A = K_A = B$

$J_C = 1$, $K_C = \overline{A}$

↓ implementation



(no external logic gate req^d here)

Illy, for any random count sequence, circuit can be designed.

☆ Clock pulse application when present states are unused state (from some other source)
In above circuit, state changes as 1, 2, 5, 7.
If the present states are taken as unused states (0, 3, 4, 6 ⟶ 0, say initially). What will

happen when next clk pulse is applied.

- Unused state analysis:

	A B C	$J_A = K_A = B$	$J_B = K_B = 1$	$J_C = 1, K_C = A'$	$A_+ B_+ C_+$
0 applied	0 0 0	0 0	1 1	1 1	0 1 1
3	0 1 1	1 1	1 1	1 1	1 0 0
4	1 0 0	0 0	1 1	1 0	1 1 1
6	1 1 0	1 1	1 1	1 0	0 0 1

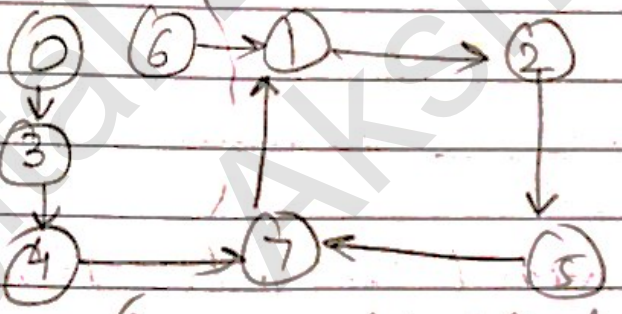
on applic^n of clk pulse on 000, 011 i.e next state comes

written using JK

$Q_{t+1} = JQ_t' + K'Q_t$

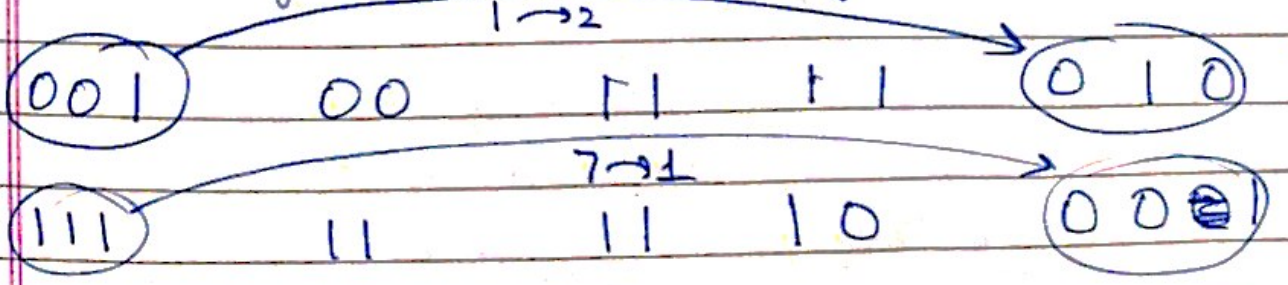
Self correcting counter ← after all unused state come/used, used state sequence comes back.

- State diagram



→ see state table & make this extra part.

- Circuit verific^n : check using any used state



So, verified.

Puffin
Date
Page

* Note :- See in the q question, what is told about the unused states — go to dont care or some used state.
Design table & circuit accordingly.

* Counter with load

Q. Design a counter satisfying following fn.
Counter should have loading facility.

Clear	clk	load	Count	fn :
0	↑	X	X	clear to zero
1	↑	1	X	load i/p I_0 to I_3
1	↑	0	1	Count binary
1	↑	0	0	No change.

We have designed T flip flop sync. circuit counter up ⟶ Use J-K now.
(T i/p same to both J & K)

Now, integrate it with loading.

See back,

$T_0 = 1$

$T_1 = A_0$

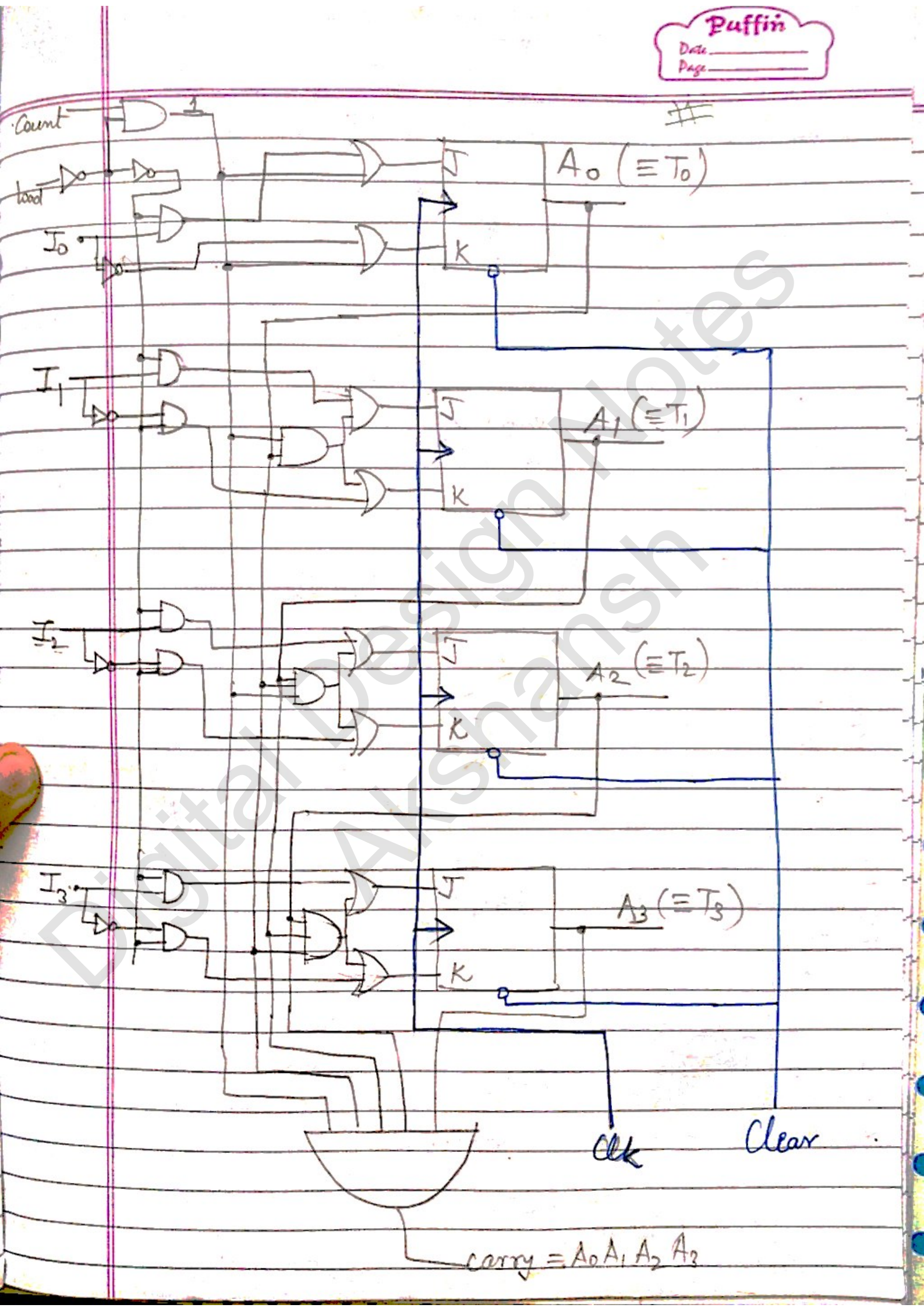
$T_2 = A_0 A_1$

$T_3 = A_0 A_1 A_2$.

Carry : $A_0 A_1 A_2 A_3$.

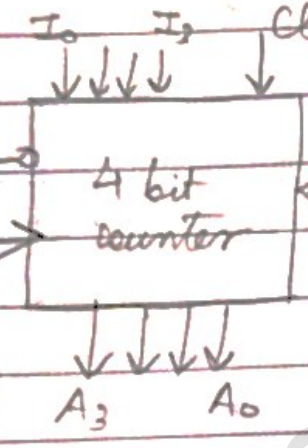
When load 0 ⟶ 0 : J
1 : K,

1 ⟶ 1 → J
0 → K



* mode n counter or divide by n counter
  - other names
  - counts from 0 to (n-1).

* Design BCD counter using f^n table given.
  - i.e., counting only from 0–9

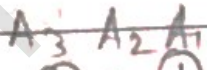


design BCD counter using this IC

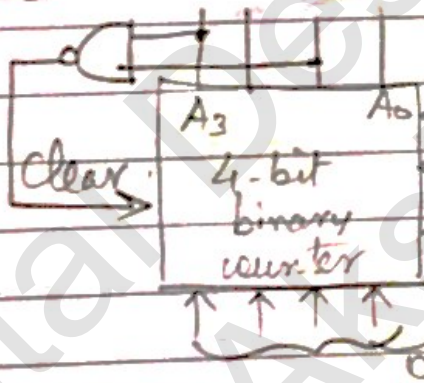
MODE 10 counter or 0 to 9 counter called as

M1

Idea: 10th clock pulse

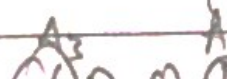


put a nand gate to clear pin. When clr pin becomes 0, it clears (as per table)



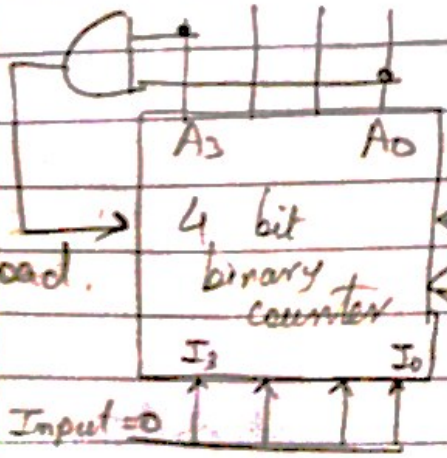
OR

M2 At 9th clock pulse



AND

& load i/p data (0000) to counter circuit.



Puffin
Date
Day

✱ Design of Asynchronous counter

↳ when CLEAR PIN NOT THERE.

✱ Making BCD ripple counter

↳ i.e counting from 0-9

(synchronous done before)

✓ Using JK flip flop

(∵ universal)

A_3	A_2	A_1	A_0	Q_8	Q_4	Q_2	Q_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1
1	0	1	0	0	0	0	0
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

They should have been 1, but I want 0 now.

Note :- whenever A_0 1 → 0, A_1 → Toggles.

Why, Q_1 1 → 0

Q_2 → Toggles → only when $Q_8 = 0$

when $Q_8 = 1$, it shouldn't toggle.

Idea : $K = 1$, $J = 1$ → Toggles when $Q_8 = 0$

$J = 0$ → when $Q_8 = 1$

So, make $J = \overline{Q_8}$

↳ Seeing for Q_2

Seeing for Q_8

↳ Q_2 & Q_4 = 1 1 or 0 0.

Then Q_8 toggles

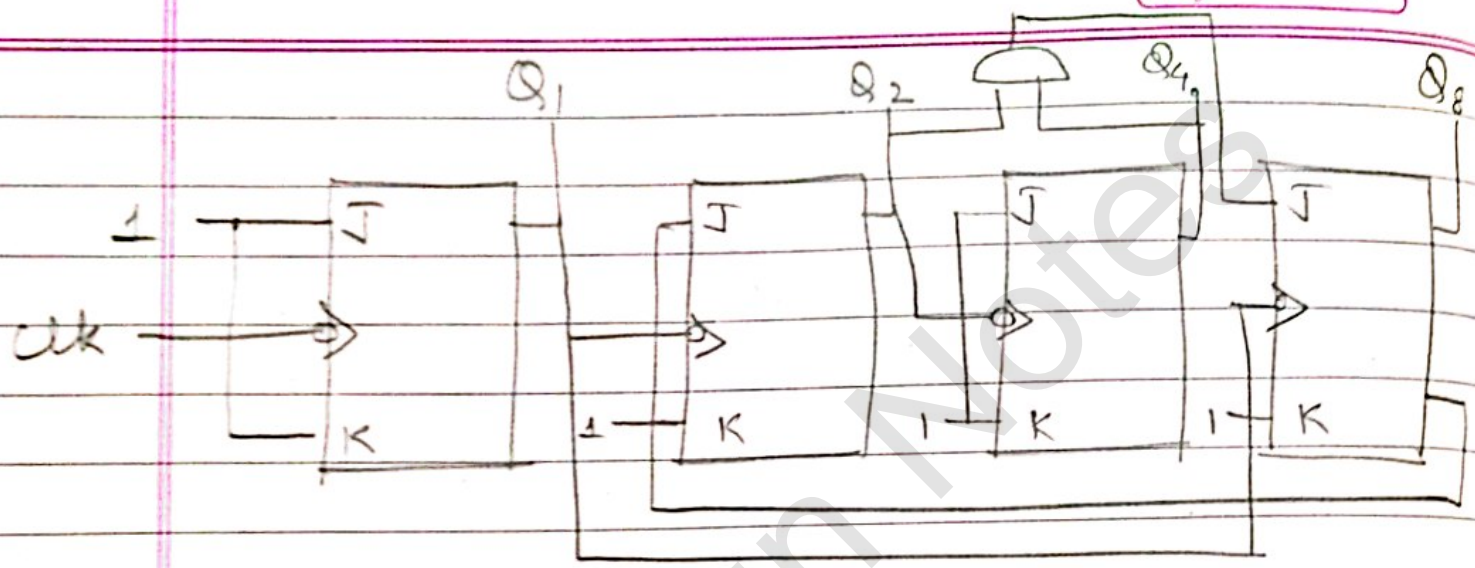
So, give J i/p as AND Q_2 & Q_4

Seeing for Q_4

Q_2 goes from 1 → 0.

Q_4 → toggles

So, give clock i/p of Q_4 as o/p of Q_2. when Q_2 goes from 1 → 0 (active low), circuit starts & Q_4 toggles (J = 1 = K)



BCD ripple counter

☆ PROGRAMMABLE LOGIC DEVICES

↳ Combin^{n} / group of decoders & encoders.
(decoders: has AND & NAND; encoders: has OR & NOR)

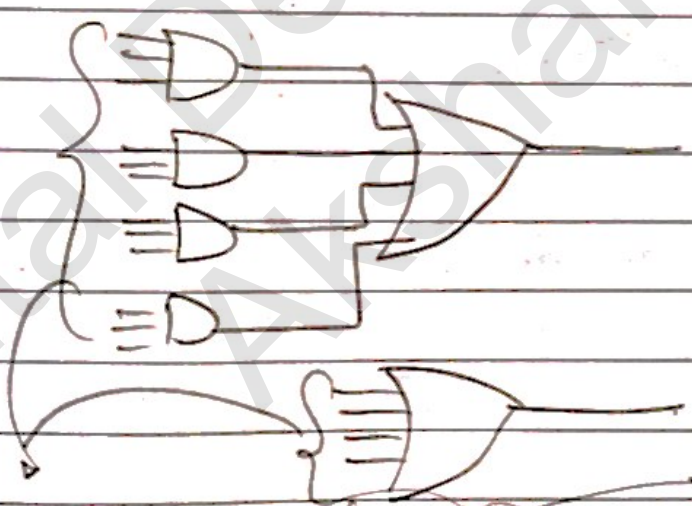
⇓

combin^{n} of AND & OR arrays

eg: Given 3 i/p AND gates (4 nos.) & 4 i/p OR gates (2 nos.) } fabricated in a single IC

which i/p will I put where : Programming.

If I have to implement SOP expression, what will be max. no. of product terms.



Ans = 4

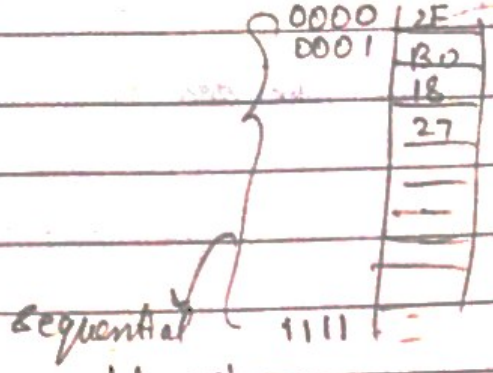
I can program these i/ps (AND or OR) & make my IC

∃ many PLD's (Programmable Logic Devices)

Programmable Read only memory

1. PROM → we can program & read data
2. PLA
3. PAL

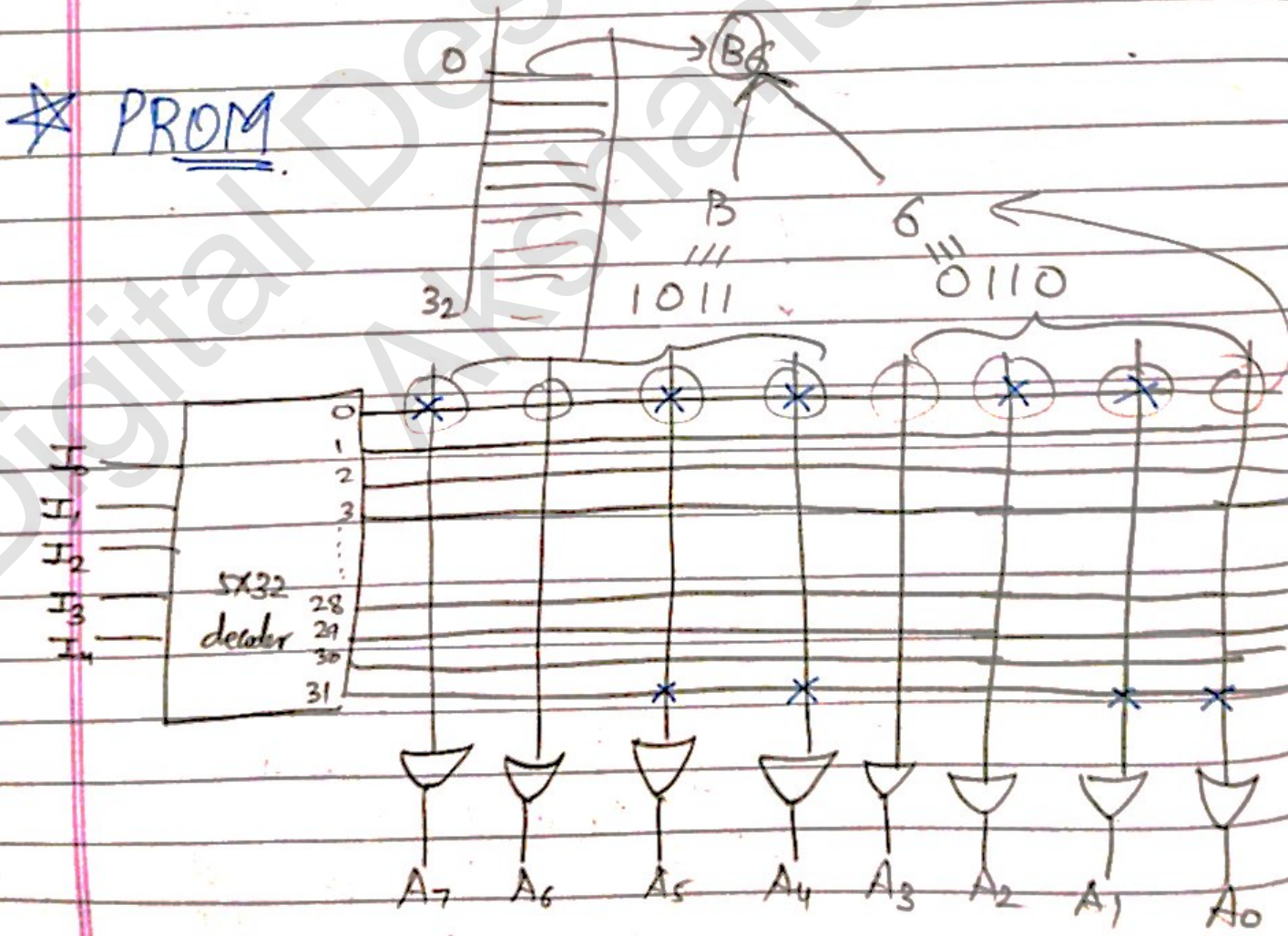
eg: 4 bit memory ⇒ $2^4 = 16$ loc^{ns}



→ Suppose each loc^n is 8 bit in size.

+these 16 combin^{ns} are decoder o/p's, taken into address lines to refer to each memory loc^n.

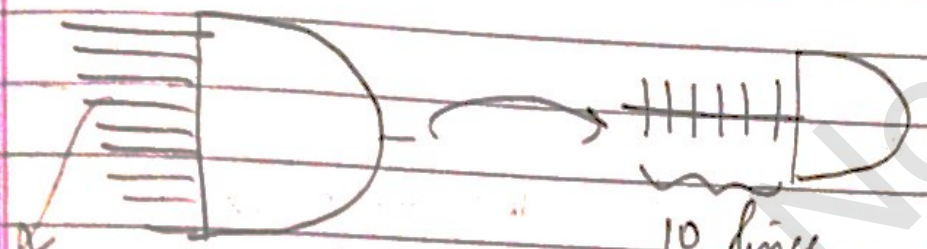
✡ PROM



o/p of decoder are coming from AND gates → (minterms of ^n implement)

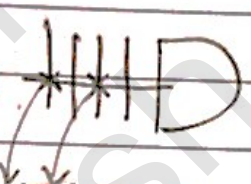
So, here OR gates can be changed (AND-fixed)

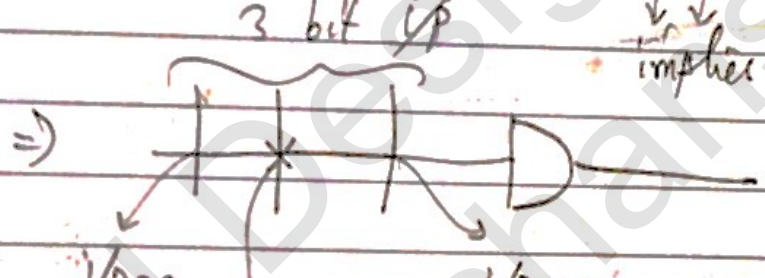
Way to represent.



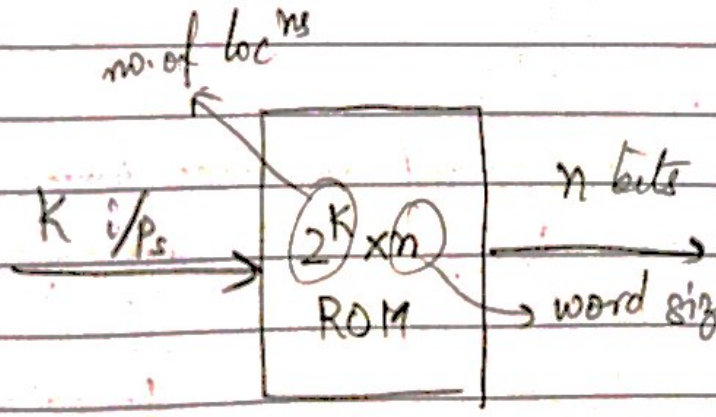
suppose 10 lines.

If any line is (X)





Ily for OR gate 



So, $1K \times 8 \equiv 2^{10}$ locns $= 1KB$

$\frac{bits}{8} = byte$ ← (8) → word size

For finding KB

eg: Given :- 2K X 4

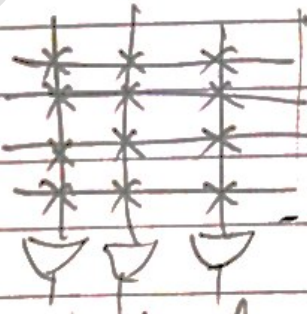
S1) ÷ 8. $\frac{2K \times 4}{8}$

S2) = 1 KB

* Data represent^{n} is in HEXADECIMAL.

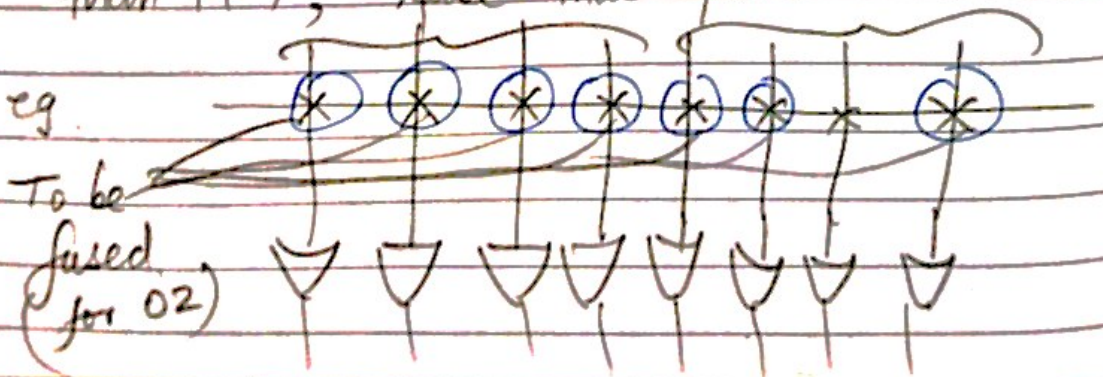
Idea of working of PROM

Initially :- All places are crossed (X)

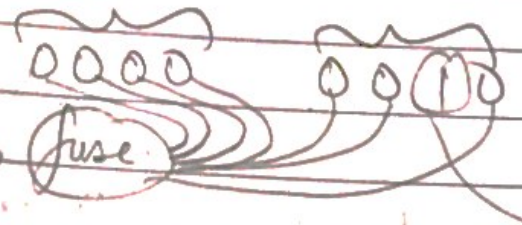
Sth like 

This will make the word stored as FF.

Now, If we want to give any i/p (other than FF), fused that place



If I want 02

⇒ 

So, fuse → leave as it is

Other way :- Initially 0

Put (X) for o/p

Q. Design a EROM which gives sq. of i/p.

⇒ Takes 3 bit binary i/p → gives out sq. of that

size

i/p → 3 bits ⇒ 8 bits.

o/p → square of 111 (7) = 49 → map 2^n = 64

6 bit.

So, 8X6 → size.

PTO

i/Ps			o/Ps						Decimal
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

Why used :- eg: if we have to find sol^n of

$$y = a^2 + 2b + 8$$

a	b	y

i.e directly creates data table to store y o/p.

Otherwise → use multiplier a x a & no adders.] no point

Q Optimise above ROM device (seeing table)

B_1 = Ground.

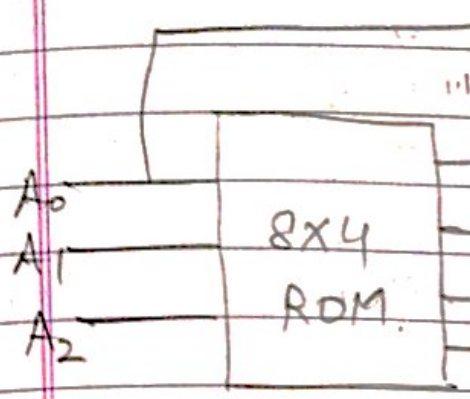
B_0 = A_0.

So, just program B_2 B_3, B_4 & B_5

So, time saved → Size reduces to 8 x (4)

Puffin
Date
Page

Simplified :-



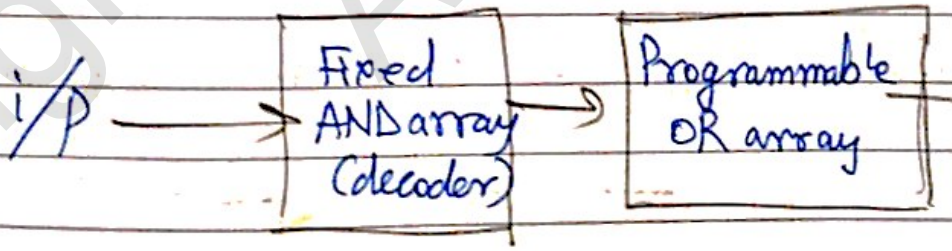
ROM Truth table

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

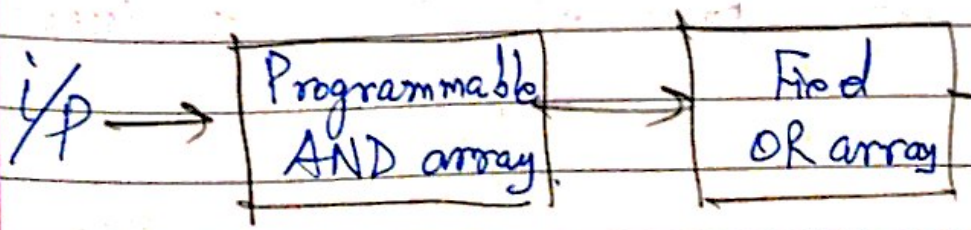
Ily, can be asked for any f^n.

* We can say :- ROM truth table is subset of ↖ truth table.

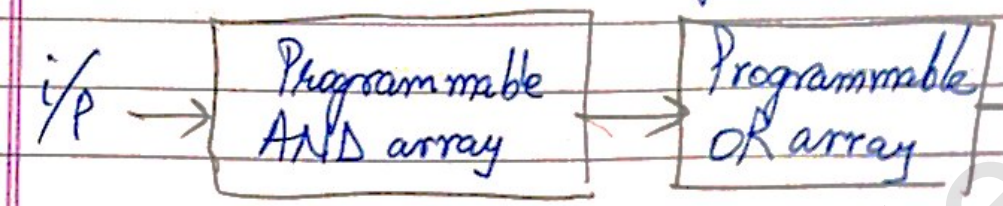
* (a) PROM



(b) PAL (Programmable Array Logic)



(c) PLA (Programmable logic array)



Q. Implement 2 fns F_1 & F_2 using PLA

$F_1 = AB' + AC + A'BC'$

$F_2 = AC + BC$

Optimizⁿ idea
Simplify using K-map c.t
✓ no. of product terms is min.
✓ no. of literals in each term is min.

We can have one thing
↓
product terms from F_1 can match that of F_2'
↓
complement of F_2

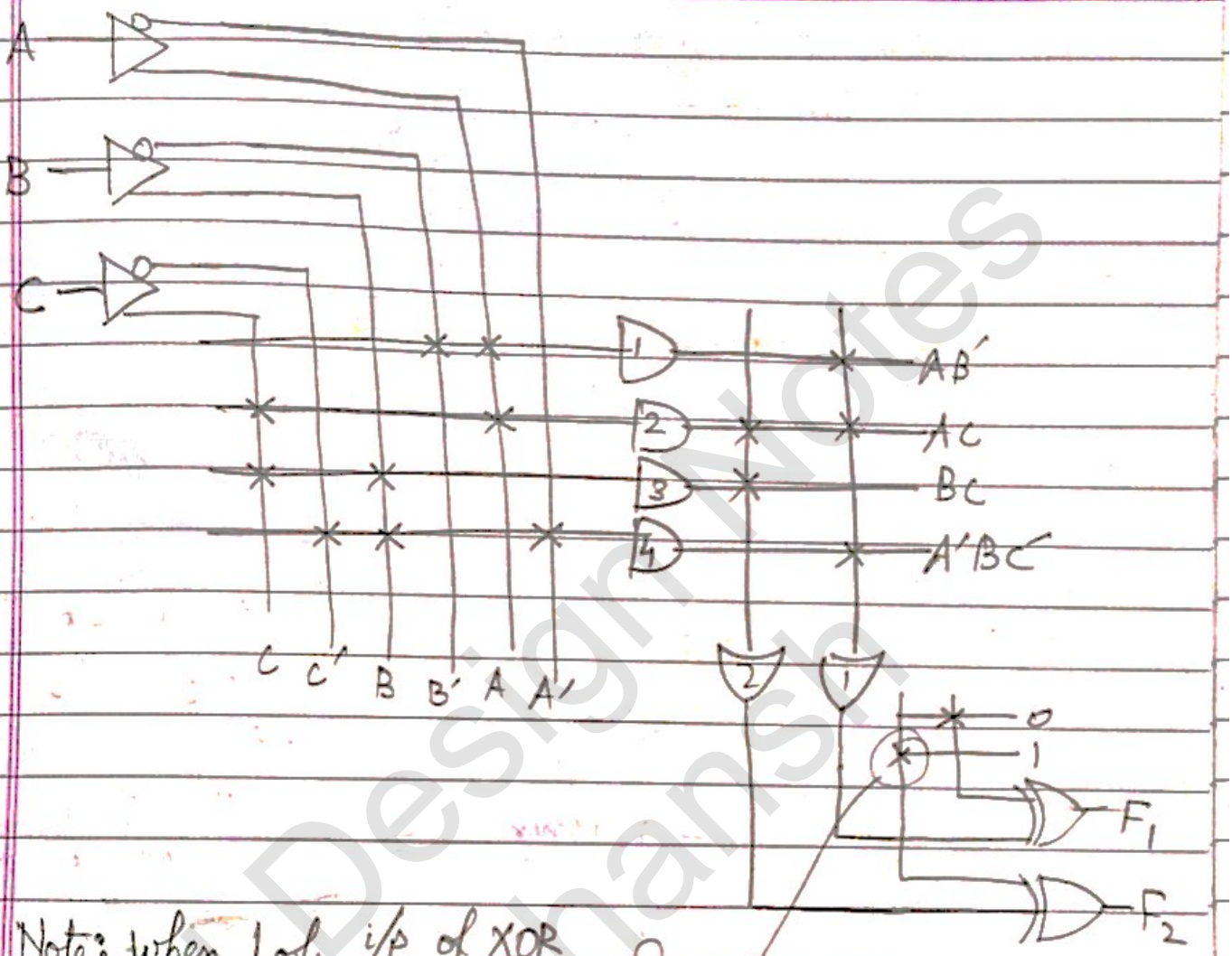
Here, in F_1 & F_2, AC is common in them

See the XOR gate (⟶) in the end.

When F_1 & F_2' are matching in their terms,

F_2 ⟶ connected to 1 (as shown)

F_1 ⟶ 0.



Note: when 1 of i/p of XOR connected to
0: o/p same
1: o/p complemented

not req^d here. We get F_2'
It should be with 0 to get F_2

Q How to give to a vender.

* Programming table

for our reference.

Product term		i/p	o/ps	
		ABC	F_1	F_2
AB'	1	10– (– shouldn't be there)	1	– (AB' should not be there in F_2)
AC	2	1–1	1	1
BC	3	–11	–	1
A'BC'	4	010	1	–

True (T) / Complement (C)

→ Symbol used later

Puffin
Date
Page

Q Implement 2 fns F_1 & F_2, given as minterms using PLA

$F_1(A,B,C) = \Sigma(0,1,2,4)$

$F_2(A,B,C) = \Sigma(0,5,6,7)$

F_1:

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

F_2:

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

SOP $F_1 = A'B' + A'C' + B'C'$ → $F_2 = AB + AC + A'B'C'$

POS $F_1 = (AB + AC + BC)'$ ← $F_2 = (A'C + A'B + ABC')'$

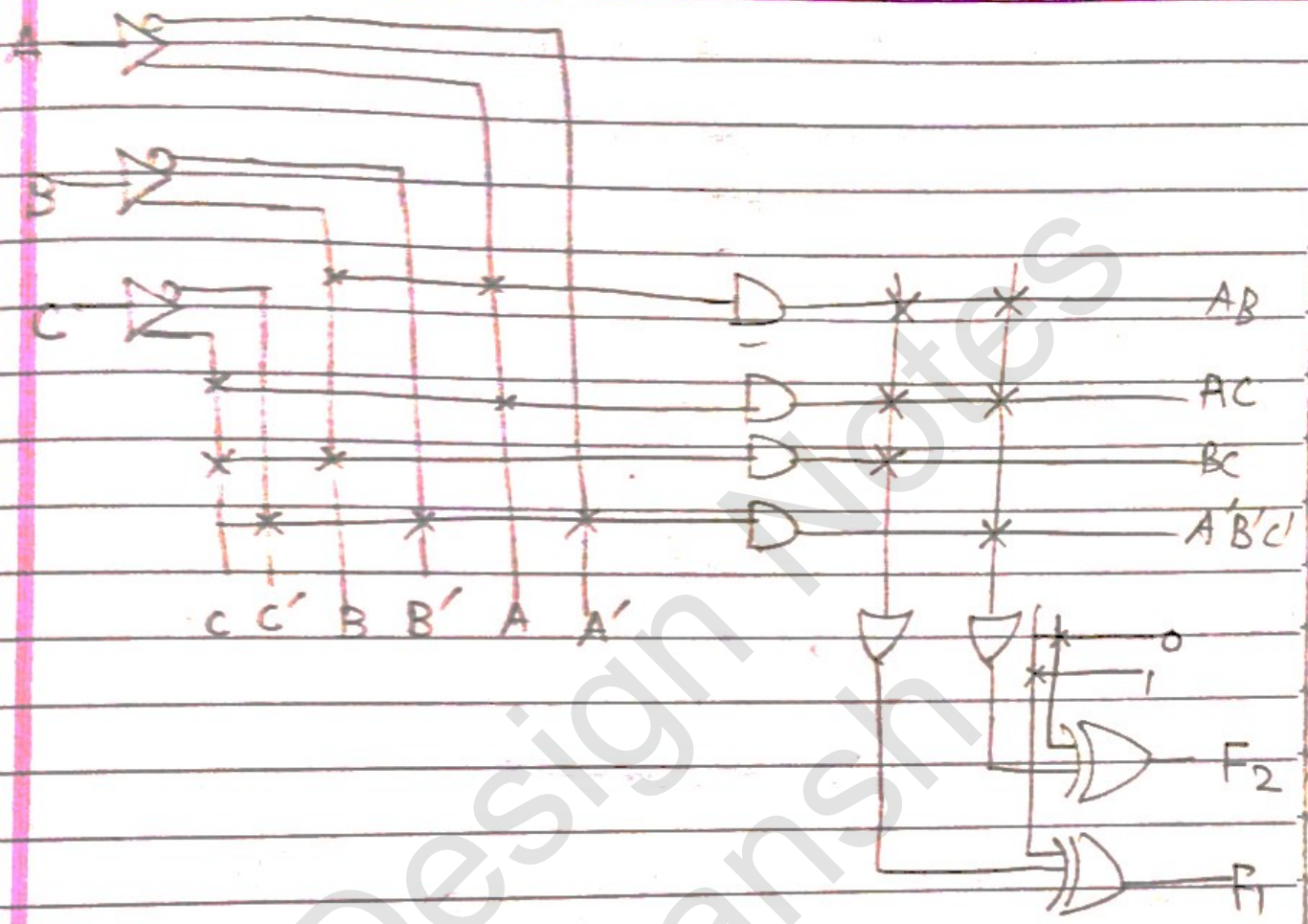
2 matching. So in all, 4 product forms will implement both F_1 & F_2 fns are AB, AC, BC, $A'B'C'$

So, implement F_1 → complement form

F_2 → true form.

Programming table

Product term		i/p A B C	o/p (C) F_1	o/p (T) F_2
AB	1	1 1 –	1	1
AC	2	1 – 1	1	1
BC	3	– 1 1	1	–
$A'B'C'$	4	0 0 0	–	1

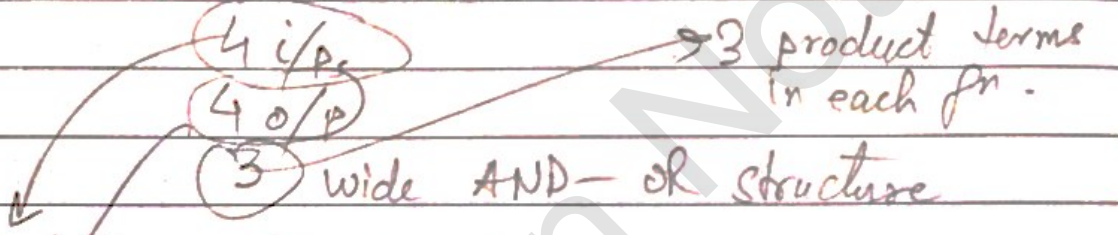


Puffin
Date
Page

3 WIDE-AND OR Structure

⟹ 3 i/p OR gate & each i/p is coming from 3 diff^t AND gate.

Given :- PAL device



Q. Given a PAL device with 4 fns

$w(A,B,C,D) = \Sigma(2, 12, 13)$

$x(A,B,C,D) = \Sigma(7, 8, 9, 10, 11, 12, 13, 14, 15)$

$y(A,B,C,D) = \Sigma(0, 2, 3, 4, 5, 6, 8, 10, 11, 15)$

$z(A,B,C,D) = \Sigma(1, 2, 8, 12, 13)$

Using K-map & solving, we get

$w = ABC' + AB'CD'$

$x = A + BCD$

$y = A'B + CD + B'D'$

$z = ABC' + AB'CD' + AC'D + A'B'C'D$

has 4 product terms

↳ 4 can use only 3.

So, use with w to get 3 product terms.

i.e, $z = w + AC'D' + A'B'C'D$.

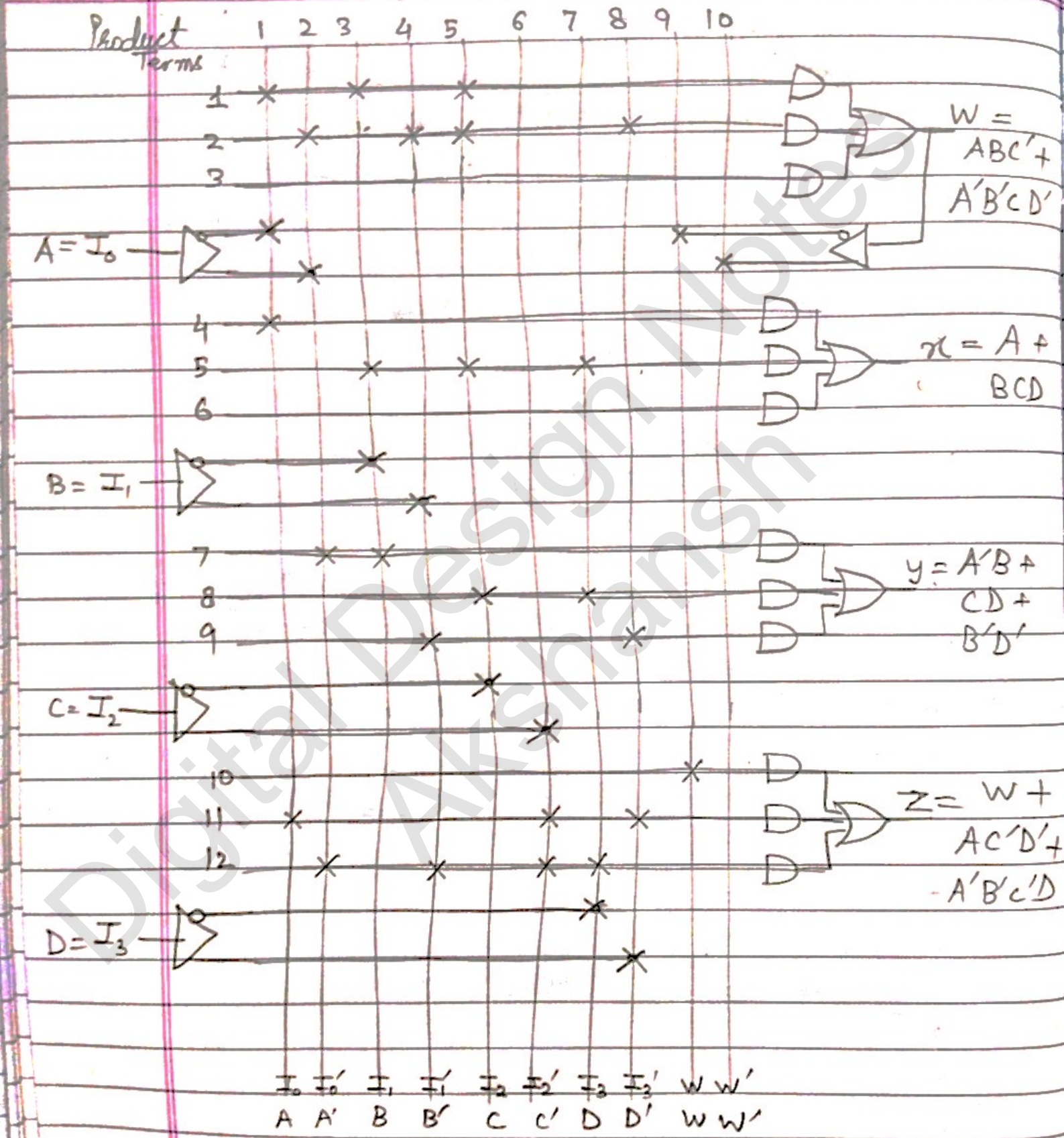
Programming table

W given as i/p ∴ reqd in (Z)

Product terms	AND i/ps A B C D (W)	o/ps
1	1 1 0 - -	W = ABC' +
2	0 0 1 0 -	A'B'C D'
3	- - - - -	
4	1 - - - -	x = A +
5	- 1 1 1 -	BCD
6	- - - - -	
7	0 1 - - -	y = A'B
8	- - 1 1 -	+ C D
9	- 0 - 0 -	+ B'D'
10	- - - - 1	z = W
11	1 - 0 0 -	+ AC'D'
12	0 0 0 1 -	+ A'B'C'D

So, we need 8 + (2) i/p lines

↙ for w & $\bar{w}$



Q. Tabulate truth table for an 8×4 ROM that implements the boolean fn.

$A(x,y,z) = \sum(1,2,4,6)$

$B(x,y,z) = \sum(0,1,6,7)$

$C(x,y,z) = \sum(2,6)$

$D(x,y,z) = \sum(1,2,3,5,7)$

Consider the ROM as a memory, specify the memory contents at addresses 1 & 4.

3 var. fn.

4 product terms posible

i/p x	y	z	o/p A	B	C	D	
0	0	0	0	1	0	0	
0	0	1	1	1	0	1	→ So, memory content in 1 → 1101
0	1	0	0	1	1	1	
0	1	1	0	0	0	1	
1	0	0	1	0	0	0	→ 4 is 1000
1	0	1	0	0	0	1	
1	1	0	1	1	1	0	
1	1	1	0	1	0	1	

Q. List the PLA programming table for the BCD to X-S-3 code converter for which the Boolean fn are given as

$E_0 = D'$

$E_1(A,B,C,D) = CD + C'D'$

$E_2(A,B,C,D) = B'C + B'D + BC'D'$

$E_3(A,B,C,D) = A + BC + BD.$

(what if fns not given ★ like ☆)

Product terms	i/p				o/p.			
	A	B	C	D	E_0	E_1	E_2	E_3
1	–	–	–	0	1	–	–	–
2	–	–	1	1	–	1	–	–
3	–	–	0	0	–	1	–	–
4	–	0	–	1	–	–	1	–
5	–	0	–	1	–	–	1	–
6	–	1	0	0	–	–	1	–
7	1	–	–	–	–	–	–	1
8	–	1	1	–	–	–	–	1
9	–	1	–	1	–	–	–	1

Q. Given: Truth table of 3 i/p, 4 o/p combinational circuit.
Tabulate PAL programming table for circuit & mark the fuse map in PAL diagram.

i/p.			o/p.			
x	y	z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

Step 1) K-maps for fns (o/p's)

(A)

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	0	1

(B)

x \ yz	00	01	11	10
0	1	1	1	0
1	0	0	1	1

Group 1's

- SOP A = $x'y'z + yz' + xz'$ SOP B = $x'y' + yz + xy$

~~POS A = $(xz + yz + x'y'z')'$~~ ~~POS B = $(xy' + x'yz')'$~~

Group 0's.

(C)

x \ yz	00	01	11	10
0	0	1	0	1
1	0	1	1	1

(D)

x \ yz	00	01	11	10
0	0	1	1	1
1	0	1	1	0

- SOP C = $y'z + xy + yz'$ SOP D = $z + x'y$

~~POS C = $(y'z' + x'yz)'$~~ ~~POS D = $(y'z' + xz')'$~~

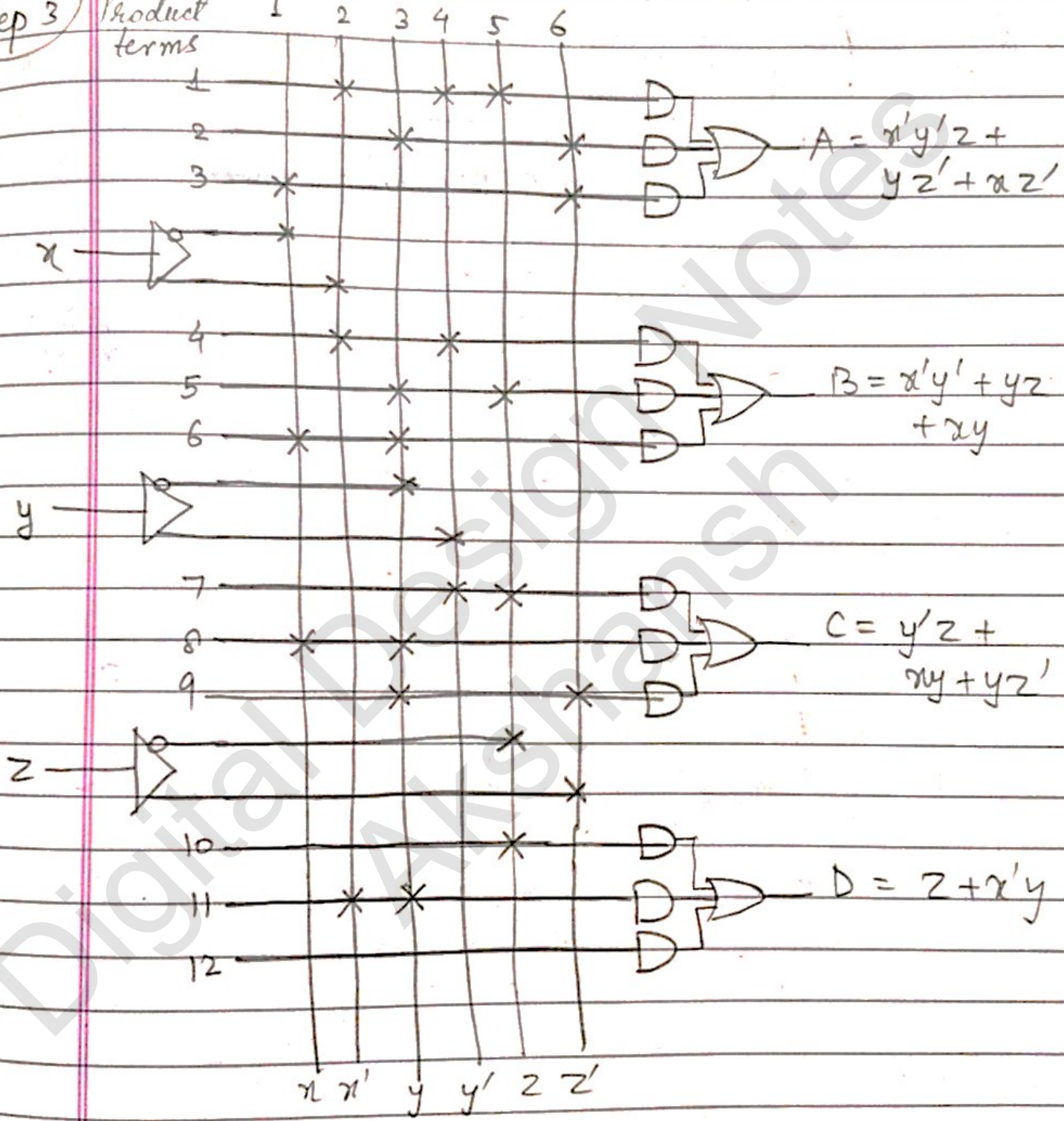
→ no need for POS form. (not reqd for PAL implementn)

Step 2 Programming table

Product term	i/p. x y z	o/p
1	0 0 1	A = x'y'z +
2	- 1 0	yz' +
3	1 - 0	xz'
4	0 0 -	B = x'y' +
5	- 1 1	yz +
6	1 1 -	xy
7	- 0 1	C = y'z +
8	1 1 -	xy +
9	- 1 0	yz'
10	- - 1	D = z +
11	0 1 -	x'y
12	- - -	

Circuit diagram

Step 3



COMPUTER ARITHMETIC

↳ when I want to multiply using (shift register only multiplies 2^n powers.)
Suppose I want to multiply by 3, say

① MULTIPLICATION

name of register

Multiplicand 1 0 1 — M
Multiplier 1 1 0 — Q

How to do this using computers?

0 0 0
1 0 1
1 0 1
1 1 1 1 0.

normal multiplic"

Note :- Each of the partial product is either 0 or the multiplicand ∴ multiplic" done by 0 or 1

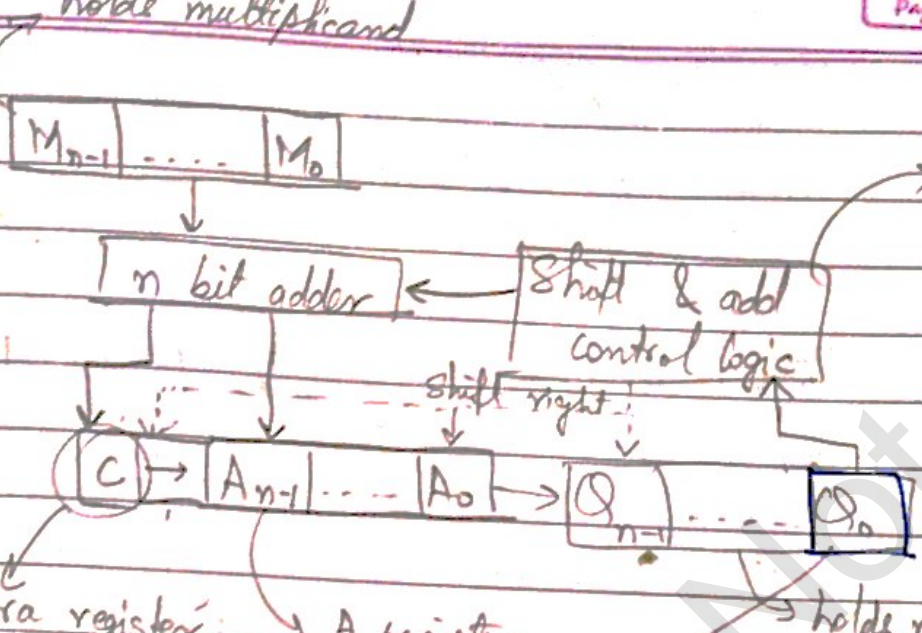
Multiplier 1 1 0 is accessible.
Now, we have to multiply by

0 then 1 then 1

So, use 0 → shift right & use → 1 → shift right & use → 1.

* Partial products are shifted left,
↳ So, how to match both.
Keep lower partial product there (shift the upper right That to would work.

* We need 2 registers to store final data.



If Q_0 => Control logic

$Q_0 = 0$ Only Shift

$Q_0 = 1$ Shift & add.

Data is getting loaded to A & shifted to Q.

This cycle will run till n cycles (no. of multiplier)

eg

C	A	Q	M = 1011 (11)
0	0000	1101 (13)	

We have to do M×Q

S1) If $Q_0 = 1$ => Add M to A & put in A

Shifting : C, A & Q will be shifted by one bit (Q_0 → lost as we used it

0	1011		Add m
0	0101	1110	shift (1st cycle)
0	0010	1111	Add & shift (2nd cycle)

C	A	Q	
0	0000	1101	initial.
0	1011		Add & shift (1st)
0	0101	1110.	
0	0010	1111	shift (2nd cycle)
0	1101	1110	→ Add & shift (3rd)
0	0110	1111	lost
1	0001	1110	Add & shift (4th)
0	1000	1111	lost

→ Result = 143

Note :- Shift : shift Right (C, A & Q)
Add :- Add A to M (1011)

Suppose, signed no. represent" given :-

⇒ M = 1011 ≡ −3
Q = 1101 ≡ −5
Result = +15

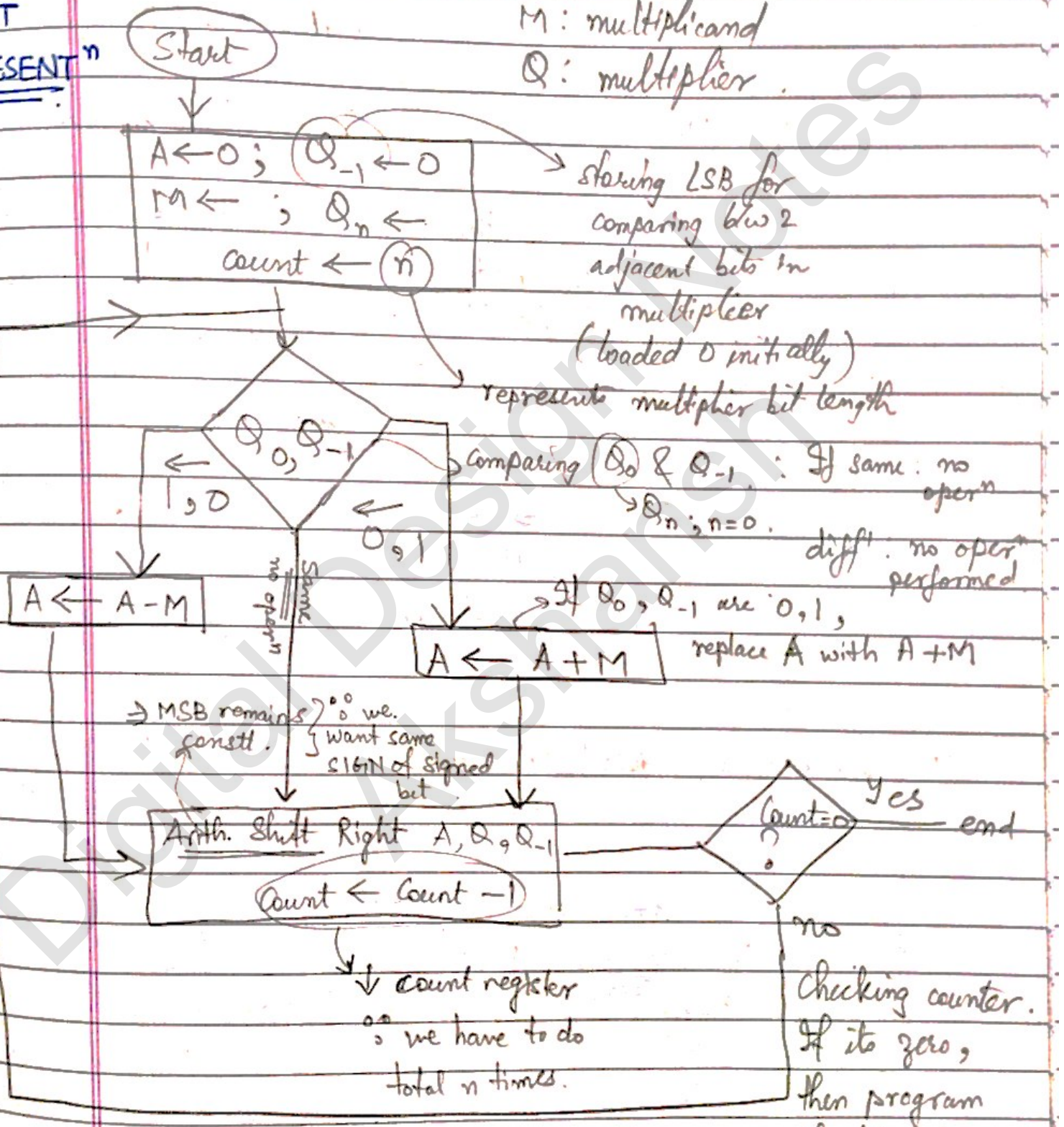
we get 1000 1111 = −15
So, this is not valid for signed nos.

based on 2's complement
(so, neglect carry)

Puffin
Date
Page

BOOTH'S ALGORITHM → for multiplying signed bits

M : multiplicand
Q : multiplier



* Shifting is done 4 times (∵ 4 bit no.)

Booth Algorithm

Ex Multiply (7) × (−3)

2's complement of 1001 (−M)

Multiplicand = 0111 (7)

Multiplier =

A	Q	Q−1	
0000	1101	0	Initial
		1 0	
		A − M	
0000	1101	0	0000 + 1001 (A + (−M))
1000	1110	1	Shift Rt. (Arithmetic)
		0 1	
		A + M	
1100	1110	1	1100 + 0111 → discard
0001	1111	0	Shift Rt. (Arithmetic) — 1st cycle
		1 0	
		A − M	
0001	1111	0	0001 + 1001 = 1010
1010	1111	0	Shift Rt. → discard — 2nd cycle
		1 0	
		A − M	
1010	1111	0	1010 + 1001
1101	0111	1	Shift Rt. — 3rd cycle
1110	1011	1	Shift Rt. — 4th cycle

1's 0011 → 1100 → +1 → 2's 1101 (−3)

4th cycle → −ve no. (MSB = 1) → So, take its 2's complement

A Q Q−1

11101011

1's → 00010100

2's → 00010101

= -21 (7 × (-3))

-ve no. so, put sign

DIVISION

eg (6)(1)2 3 4 5

Idea: use comparator to compare dividend with MSB of divisor & start performing division

logic

look at MSB & check whether divisor is dividing it.

If divisible → divide → write quotient & subtract to get remainder

otherwise → check for next MSB

On subtracting (A - M)

→ Ans = 0... (MSB) : no : +ve ; Put 1 in Quotient

= 1... : no : -ve ; Put 0.

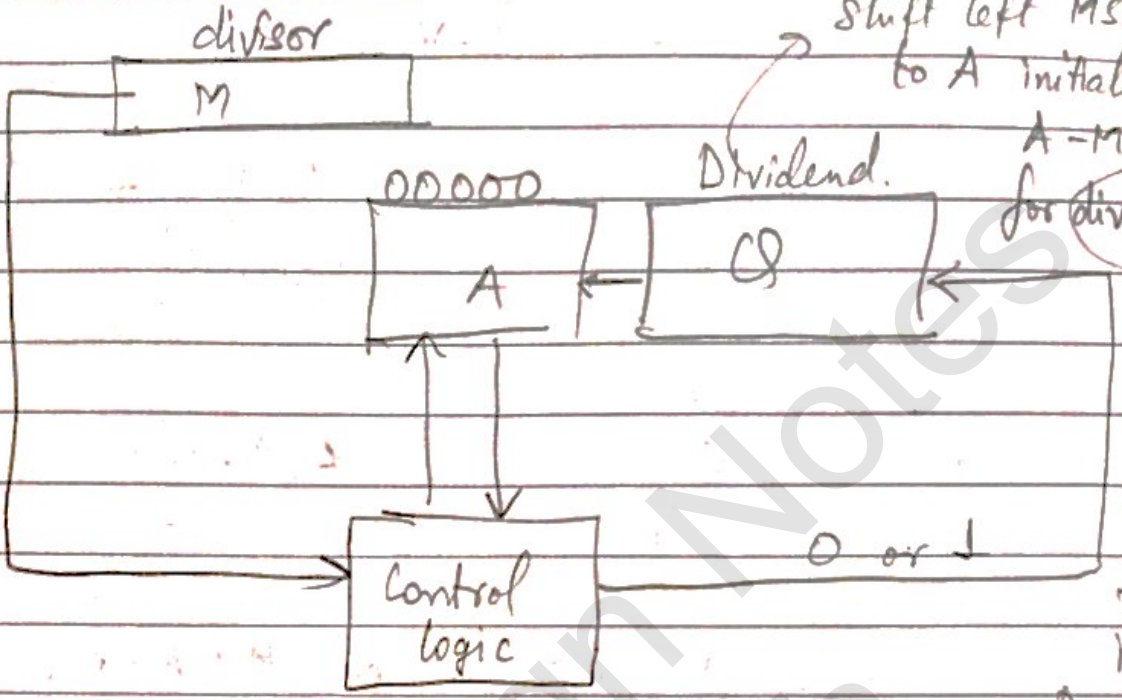
in 2's complement

i.e (MSB) complement

So, for comparing again, convert from 2's to normal (to go to next MSB)

So, add M again (A + M) (new A)

✱ Intel 8085 : 8 bit processor
✱ A register :- Accumulator register (Store result)



Based on its value, we see quotient

MSB of A | Quotient
0 → 1
1 → 0

✱ If A = 0 Push 1 to Q.
✱ If A = 1 Add M to (A)new & Push 0 to Q.

✱ Restoration Method

1. Divisor → Register M
2. Divident → Register Q
3. Register A → Initialise 0.
4. Shift A & Q left by 1 bit.
5. Subtract M from A, result in A
6. If $(MSB)_A$ = 1 → (complement) Set $Q_0 = 0$.
   = 0 → Add M to A → restore A (complement) Set $Q_0 = 1$
7. Repeat 4, 5, 6 n times, where n = no. of bits of divident.
8. The n bit quotient can be obtained from register Q & remainder from register A.

eg Unsigned division :- 7 ÷ 3

dividend 0111 → Q

divisor 0011 (M register)

	A	Q	
	00000	0111	Shift left
	00000	111	→ Blank.
	11101	111	Subtract M ; A = A − M
			(32 − 3) = 29 ; $Q_0 \leftarrow 0$
1st cycle	00000	1110	(Add M) → Restore A
			in 5 bits, ∵ A is in 5 bits
			11101 + 00011
2nd cycle	00001	110	Shift left Q_0, 0000 + (11101)
	11110	110	A ← A − M; 2's complement of 3 (for −3)
			33 − 3 = 30
	00001	1100	(Restore A) (Add M) $Q_0 = 0$
3rd cycle	00011	1000	Shift left
	00000	1001	A ← A − M ; 3 − 3 = 0. $Q_0 = 1$
4th cycle	00001	001	Shift left
	11110	001	A ← A − M : 30 → A $Q_0 = 0$
	00001	0010	Add M

00001 → Remainder.

0010 → Quotient

✱ Non Restoration Method :-

Steps 1, 2 & 3 (same as in restor" method)

4. If sign of A :-

   +ve : shift A & Q left by 1 bit & A ← A − M

   −ve : shift A & Q left by 1 bit & A ← A + M

5. If MSB of A :-

   1 : Q_0 is set to 0

   0 : Q_0 " " 1

6. Repeat steps 4 & 5 n times.

7. At the end, if sign of A is −ve, add M to A for remainder

eg $7 \div 3$ $\quad M = 0011$

A	Q	
00000	0111	Initialise
00000	1110	Shift left & MSB = 0
		Sub M ①
11101	1110	as MSB = 1, $Q_0 = 0$
11011	1100	shift left
+00011	1100	as MSB = 1, add M ②
11110	1100	as MSB = 1, $Q_0 = 0$
11101	1000	shift left as MSB = 1
+00011		add M ③
00000	1001	as MSB = 0, $Q_0 = 1$
00000	0011	Shift left ④
00001	0011	as MSB = 0, Sub M
+11101		as MSB = 1, $Q_0 = 0$
11110		add M to adjust remainder

Remainder → 00001 ; Q → 0011

Q. For feedback shift register, we used IC.
If no IC, what to do?

eg:- generate sequence :- 1011.00
how to design a circuit?
Idea :- use feedback shift register.

feed in data + O/P to LED
(Generⁿ logic) Q_A, Q_B, Q_C, Q_D
↳ give same O/P, just shifted, so, use anyone.

S i/p	A	B	C	D
1	X	X	X	X
0	1	X	X	X
1	0	1	X	X
1	1	0	1	X
0	1	1	0	1
0	0	1	1	0
1	0	0	1	1
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0
0	1	1	0	1
0				

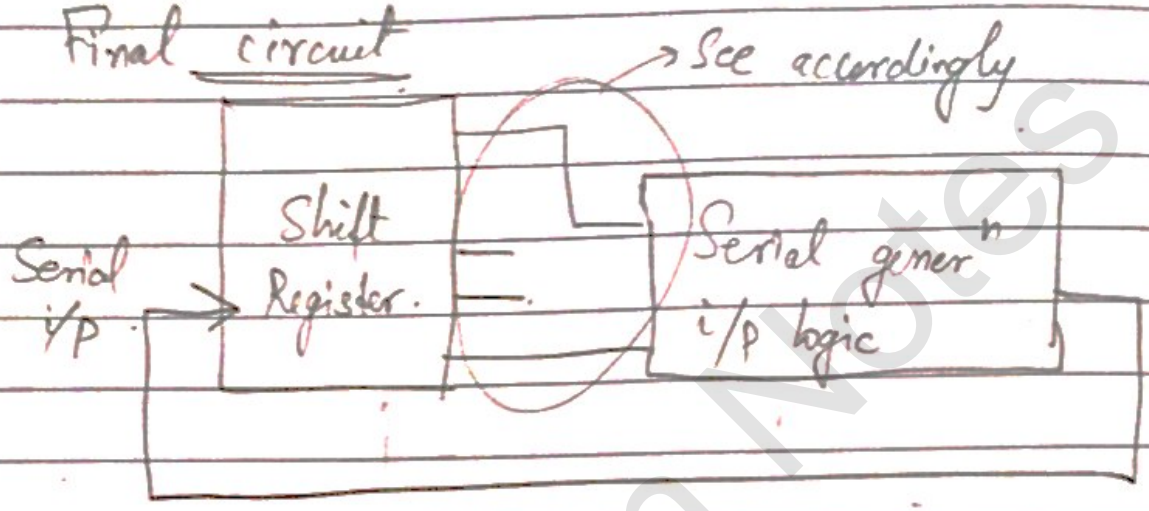
repitition.

→ So, fⁿ implementⁿ (K-map)

PTO

Puffin
Date
Page

Final circuit



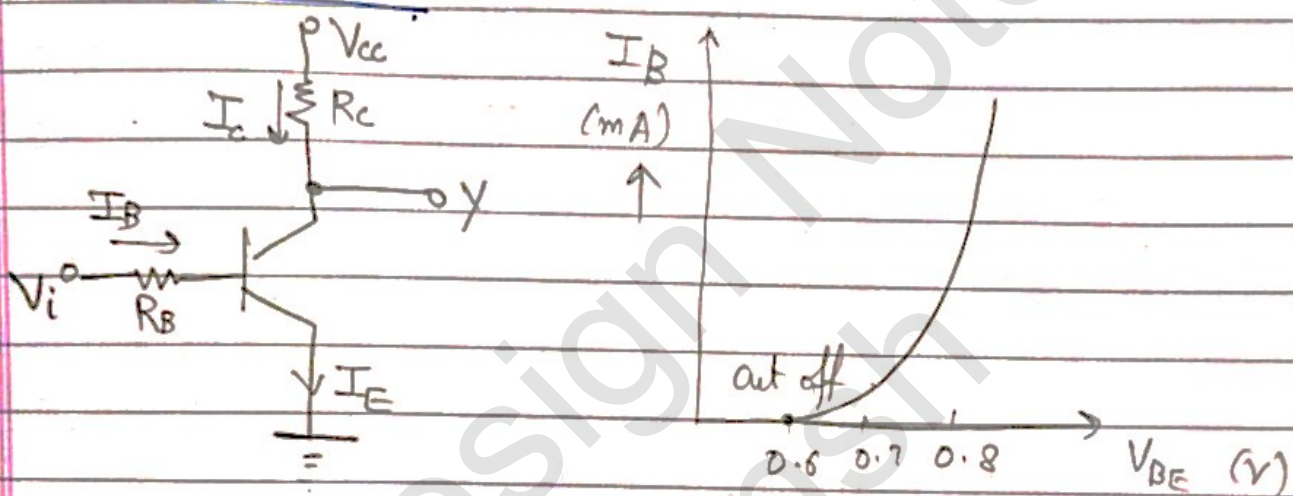
end of Chapter.

$(V_{CE})_{satⁿ}$ for Si NPN transistor = 0.2 V ≡ low mode
(fixed value)

Puffin
Date
Page

Ch: Transistors

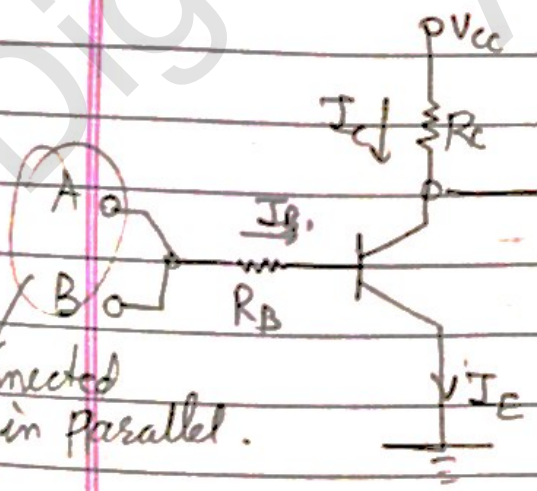
✱ BJT Transistors



(common emitter amplifier)
(say)

When V_i = high, O/p = $V_{cc} - I_c R_c$ ≡ low
V_i = low, O/p = V_{cc} ≡ high.

So, behaving as NOT gate



So, when A = B = high ≡ y = low
otherwise, y ≡ high

NAND gate

η → efficiency in case of power

h_{fe} (= gain) in case of I or V

Puffin

Date

Page

★ Cond^n for Satur^n : $I_B \geq \frac{I_C}{h_{fe}}$ → gain ($\equiv \eta$ in case of power)

npn Si Transistor Parameters.

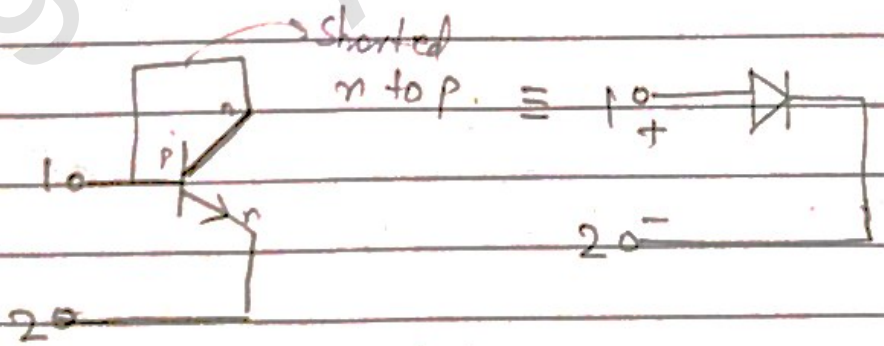
Region	V_{BE}	V_{CE}	current rel^n.
Cut off	< 0.6	open	$I_B = I_C = 0$
Active	0.6 – 0.7	> 0.8	$I_C = h_{fe} I_B$
Satur^n.	0.7 – 0.8	0.2	$I_B \geq I_{CS}/h_{fe}$

I_{CS} → max. current got at satur^n

only these 2 modes used for on & off. (act as switch)

control flow of current

★ Transistor as a diode.



Resistor Transistor Logic

RTL

≡ First alphabet (2nd Alphabet) → o/p taken in this terminal.

i/p applied at this terminal → leads to a lot of power dissipation (loss).

Puffin
Date
Page

✡ Passive :-

- devices not depending on power supply
- depend on physical properties
- eg : (R) R, L, C
  - const. resistance, all time

✡ Active devices

- depend on power supply
- almost all semiconductor supply
- eg : diodes
  - have resistance depending on power supply

So, replace resistor by Diode →

Diode Transistor Logic

DTL

both semiconductor device

Diode can be fabricated in terms of Transistor

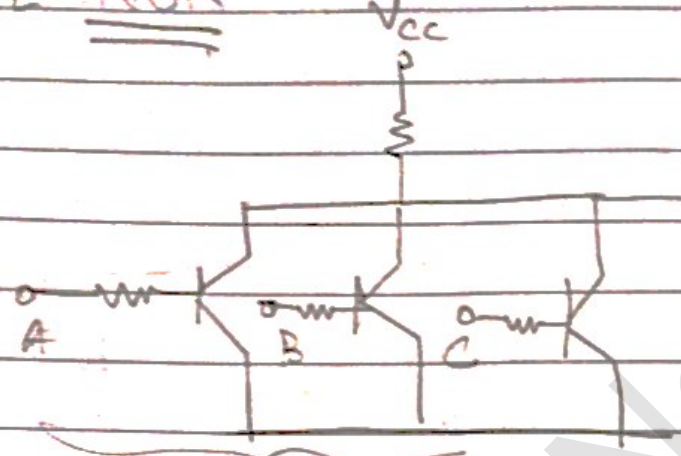
So, we got Transistor Transistor logic

TTL

Next comes **FETs**

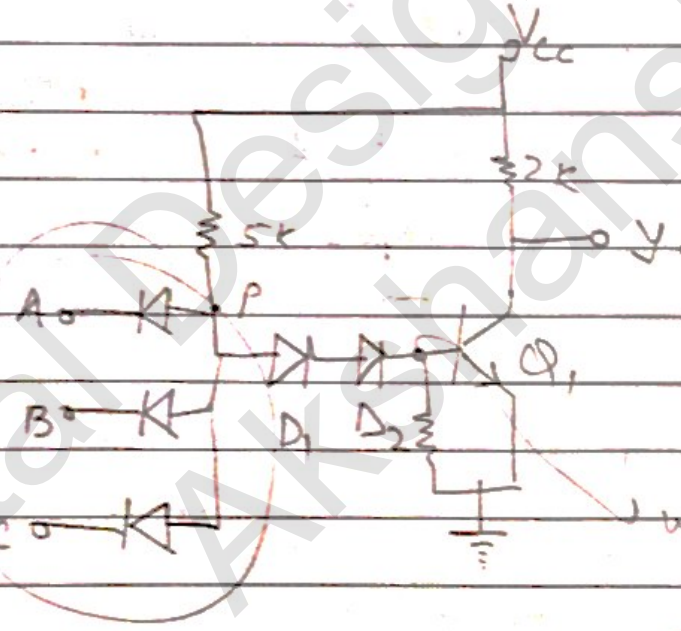
✓ CMOS : Most commonly used low power transistor these days

✡ RTL NOR



all connected in ||.

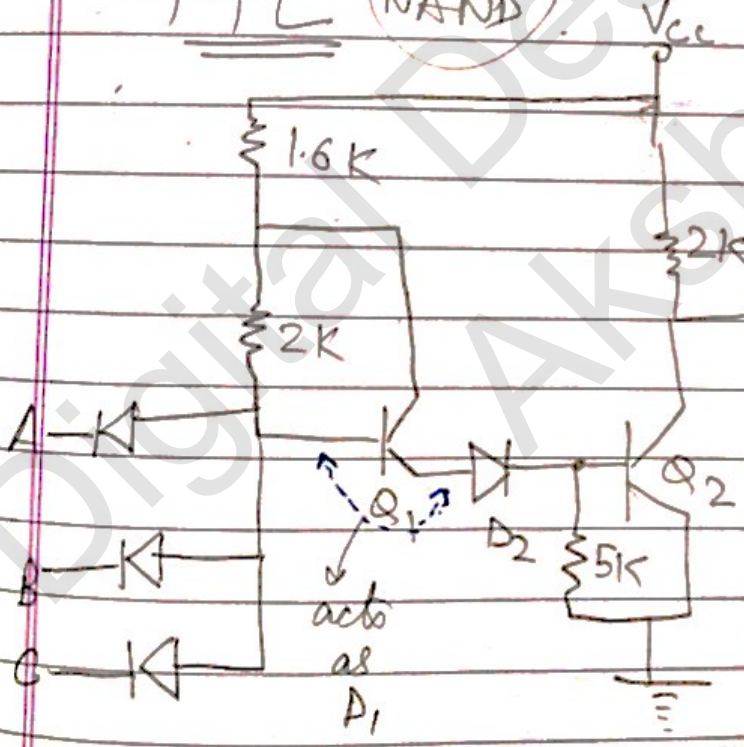
✡ DTL NAND



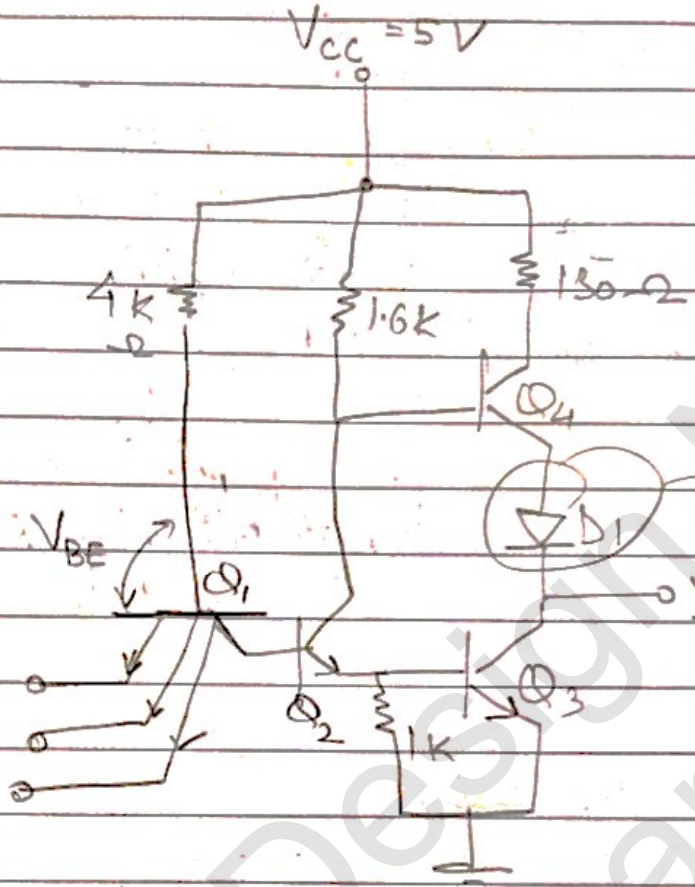
all RB
Anode supplied to Vcc
If HV is applied,
RB
If LV, FB.
When any i/p is low, it conducting. → 0.2 V.
Cut in voltage = 0.6,
So. Potential at P ≥ 0.6+0.2 = 0.9 (say)

This 0.9 V can turn on transistor Q_1,
D_1 & D_2 are supplement diodes (need 0.6 V each = 1.2 V)
So, for Q_1 to conduct 0.8 + 1.2 volt of voltage is req^d for it to conduct.
We only have only 0.9 V. So, no conduction by Q_1. So, o/p = 1 (NAND oper^n)
When all 3 i/ps are high, voltage would increase greater than 1.8 V. So, Q_1 will conduct
$\therefore$ o/p = 0.2 V (low)

TTL NAND → replacing D_1 by transistor



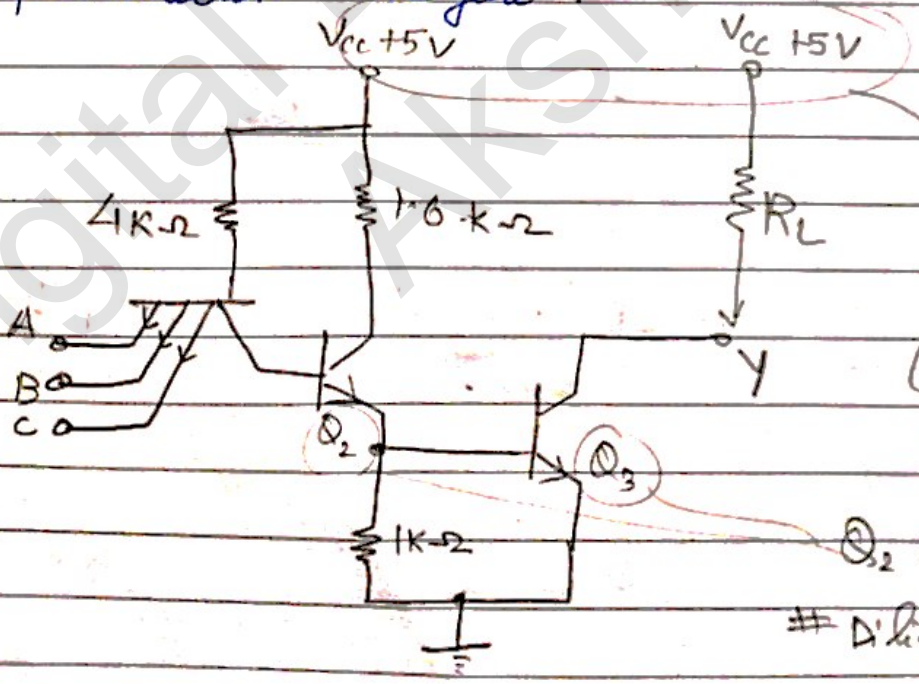
✱ Replacing i/p diodes with transistor.



~~TTL open collector configuration~~

Replacing / removing this diode.

✱ Open collector TTL gate.



→ no load connected (open load transistor)

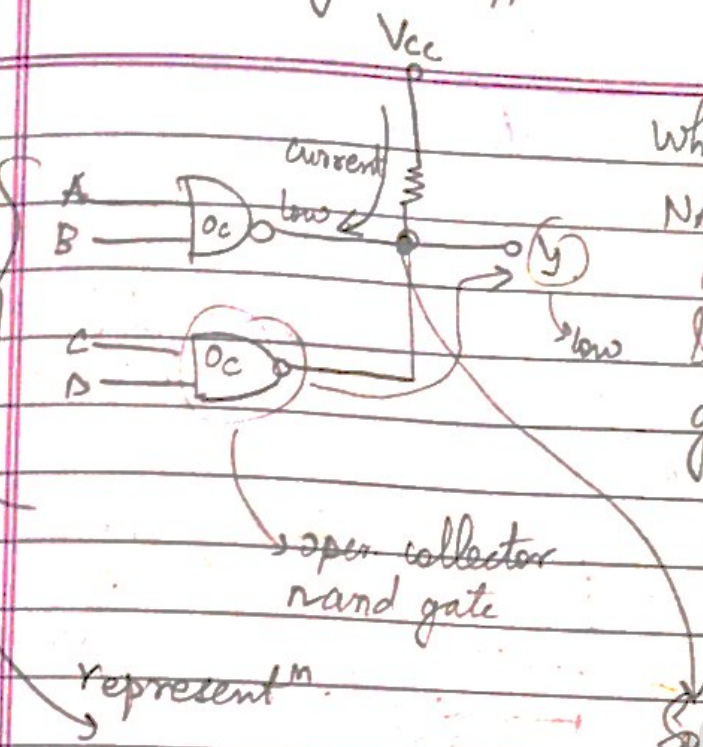
(collector resistance (R_C) → not connected → less heat generated)

Q_2 drives Q_3.

Darlington pair.

→ used for lamp lighting; we can have wired logic gate.

8 WIRED logic Applicns.

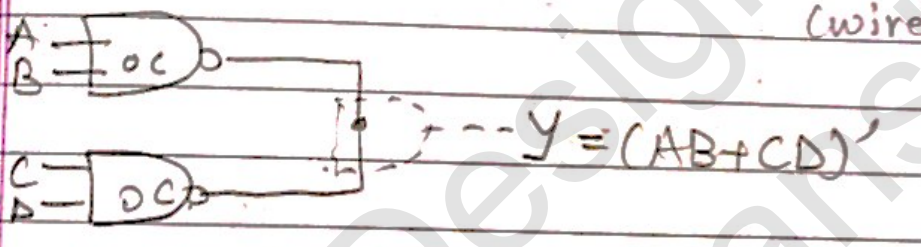


When anyone of the o/pc of NAND gate is low, current will flow as shown & other nand o/p being low gives o/p = low

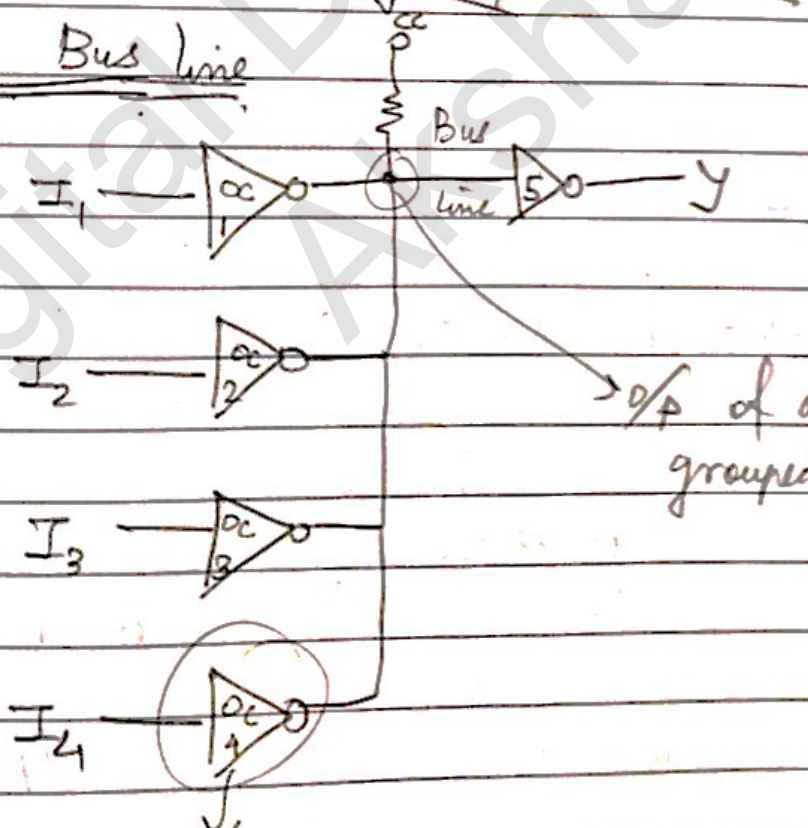
o/p (y) = high only when both o/p of NAND gates = high (current won't flow).

So, its acting as AND gate (wired logic)

represent^n.



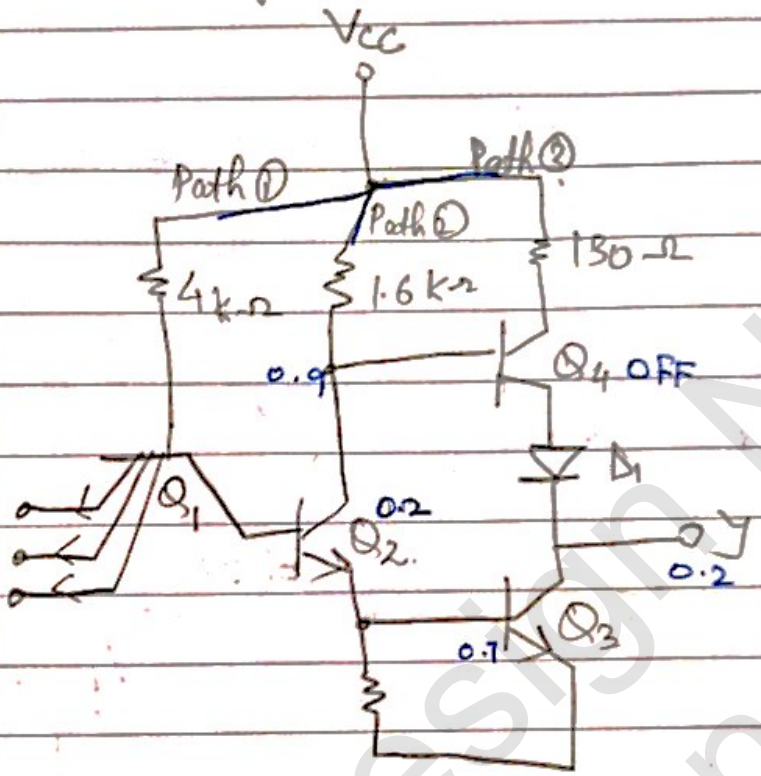
★ OC Bus line



o/p of all data lines grouped together.

open collector inverter IC's.

✱ TTL gate with Totem-Pole o/p.



describing 3 i/p NAND gate

Q_4 is over Q_3 so, configuration is called Totem-Pole

✱ Any one of the Paths only should run at a time.

• Q_4 : not necessary to be in "satur" mode when its conducting. (its in active mode)

See : why diode added ?

✱ PROPAGATION DELAY

✱ Every current carrying conductor has capacitance. But, value is very less.

✱ Transistor junctions & Internal wiring all have capacitance ⟶ but, order is around nF or PF.

But, in millions, ∃ capacitance effect & delay ∃.

In ↙ circuit.

$C = C_{transistor} + C_{fan\ out} + C_{Stray\ (Wiring\ /\ current\ carrying\ conductor)} \simeq 15\,pF$

C → Total capacitance.

Passive Pull up resistance, $R_L \simeq 4\,k\Omega$.

∴ time delay, $t_p = RC \simeq 35\,ns$.

↙ considerable when digital logic switching is concerned.

If Active pull up, $t_p \simeq 10\,ns$. ($R_L = 130\,\Omega$)

* Circuit has inherent delay while Propag^n (for Passive Pull up).

For reducing delay, use active pull up. (time delay reduced).

• Schottky transistor (K)

→ made of 1 junction & 1 metal plate (instead of 2 semiconductor junctions)

→ has cut off voltage less than 0.7.

∴ reduces time delay. (PTO)

Schottky representation.

(replacing diode D_1 by Q_5)