1/09/19

Digital Logic and Design:

(i) Number system and signed data representation (VIMP)

(ii) Boolean algebra and logic gates

(iii) k-maps

(iv) Combinational circuits

(v) Sequential circuits (VVVIMP)

Y.Y.R
8099041357

Modern Digital Electronics
R.P. Jain

1. Boolean Algebra:

→ It is operated with binary variables.

→ Let x, y, and z are different binary variables. different theorems and postulates are:

* $x + 0 = x$
* $x + 1 = 1$
* $x \cdot 1 = x$
* $x \cdot 0 = 0$
* $x + x = x(1+1) = x$
* $x + \bar{x} = 1$
* $x \cdot \bar{x} = 0$
* $x + xy = x(1+y) = x$
* $(x+y) \cdot (x+z) = x + xz + xy + yz$
  $= x(1+z+y) + yz$
  $= x + yz$ (distributive law)

distributive law
$x + yz = (x+y)(x+z)$

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

Scanned by TapScanner

Demorgan's laws:

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

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

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

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

Logic Gates:

→ It is an electronic circuit, delivers only one output by permitting one or more inputs.

→ Except NOT gates all gates are designed with minimum two inputs.

☆ → AND gate, OR gate, NAND gate, NOR gates can have two or more inputs and ~~thge~~ these gates are known as single input signal controlled gates. Hence they are not suitable for staircase operation.

☆ → Generally X-OR gate and X-NOR gates design is available with only two inputs because multiple input X-OR gate / X-NOR gates design cost is too expensive and these gates are perfectly suitable for staircase operation.

Q: A lamp (L.E.D) is operated with 2 switches. First switch is located at first floor and 2nd switch at second floor. Any one of these two switches can be used to control lamp irrespective of other switch position. This logic implies

a) AND b) OR c) NAND d) EXOR ✓

Scanned by TapScanner

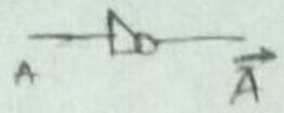
Logic gates

- **Basic**: OR, AND, NOT
- **Universal**: NAND, NOR
- **Derived**: Ex-OR, Ex-NOR

Universal gates:

The gate from which any expression circuit can be achieved

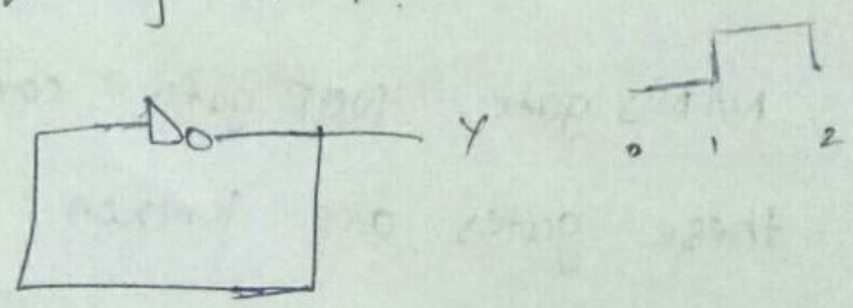
NOT gate (Invertor)



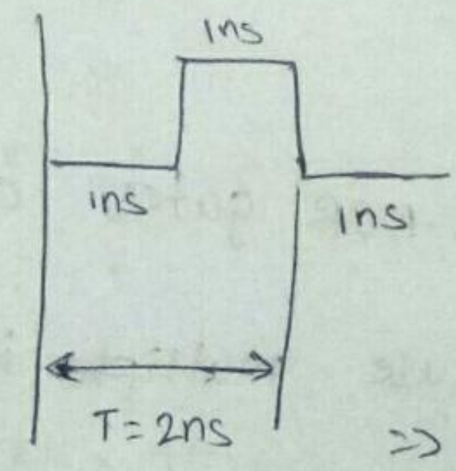
t_{pd} (Propagation delay)

Time taken by a ~~not~~ gate to produce O/P from I/P

Q: In the below logic diagram t_{pd} of NOT gate is 1ns what the signal frequency at Y?

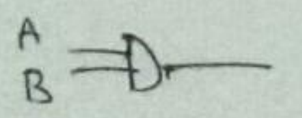


a) 1 GHz b) 2 GHz c) 0.5 GHz ✓ d) None



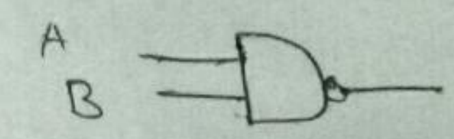
$\Rightarrow f = 0.5\,GHz$

AND gate:



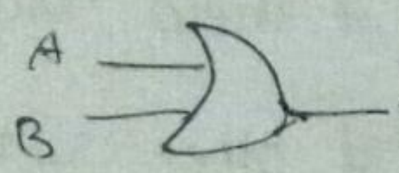
$Y = A \cdot B = A \cap B$

NAND gate:



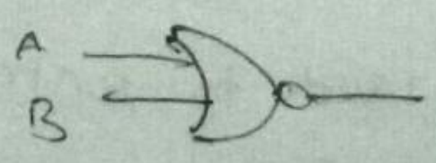
$Y = \overline{A \cdot B}$

OR gate:



$Y = A + B = A \cup B$

NOR gate



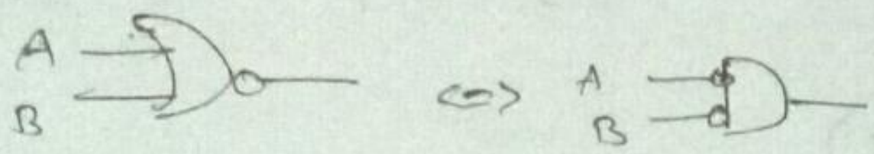
$Y = \overline{A + B}$

Scanned by TapScanner

By De morgan's 1st law

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

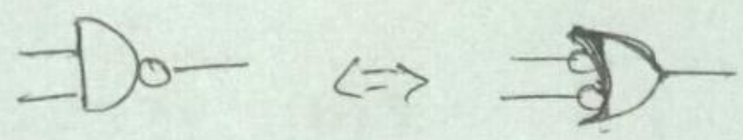
* NOR gate = bubbled AND gate.



By De morgan's 2nd law

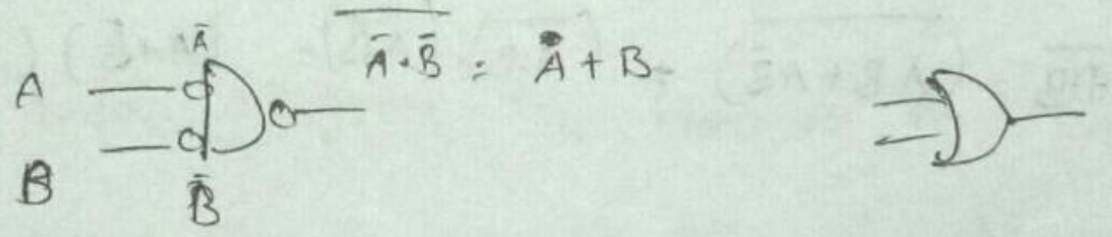
$$\overline{A\cdot B} = \bar{A}+\bar{B}$$

* NAND gate = bubbled OR gate

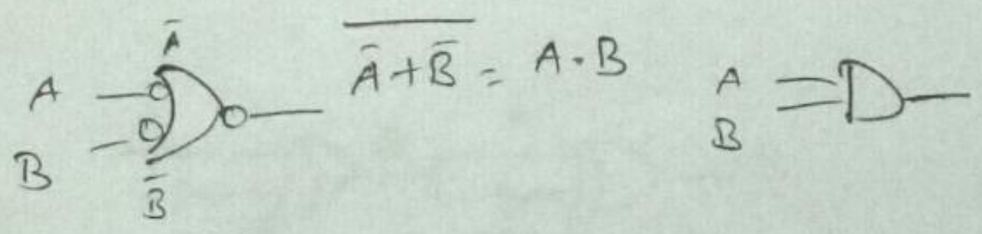


Similarly

* Bubbled NAND gate ~ to OR gate



* Bubble NOR gate ~ to AND gate



* Any of AND, OR, NAND, NOR gates o/p can be sometimes decided with single input ~~when~~. Hence they are known as <u>single input signal controlled</u> gates which makes unusefull for staircase operation

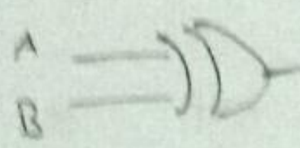
* <u>Stair case operation:</u> It is process of controlling a device by either of two switches irrespective of position (state) of other switch.

Scanned by TapScanner

* **EXOR gate**: o/p is high when it's input is having odd no of 1's.

* **EXNOR**: o/p is high when its i/p is having even no of 1's.

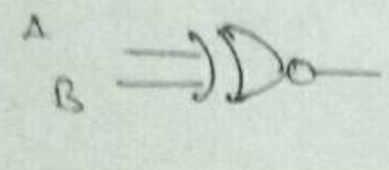
EXOR (Non-coincidence gate)



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

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

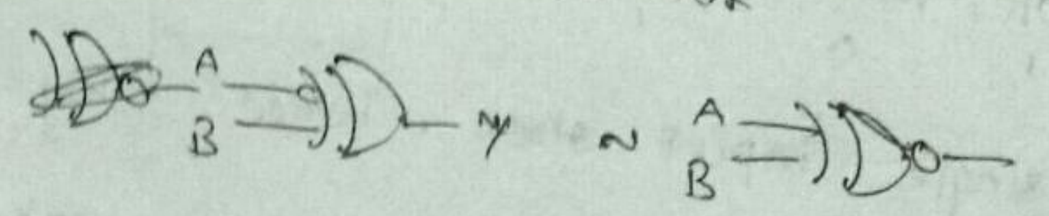
Ex-NOR (Coincidence gate)



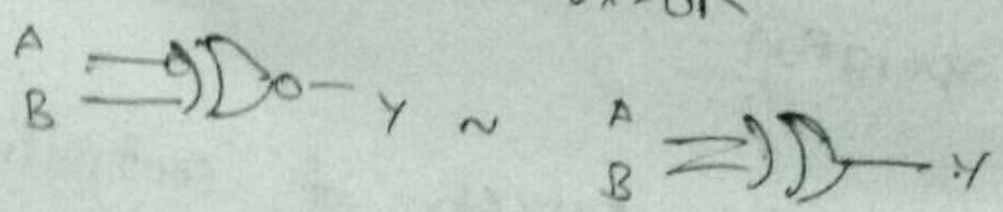
$Y = \overline{A \oplus B} = \overline{(\bar{A}B + A\bar{B})} = (\overline{\bar{A}.B}).(\overline{A\bar{B}}) = (A+\bar{B})(\bar{A}+B) = \bar{A}\bar{B} + AB$

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

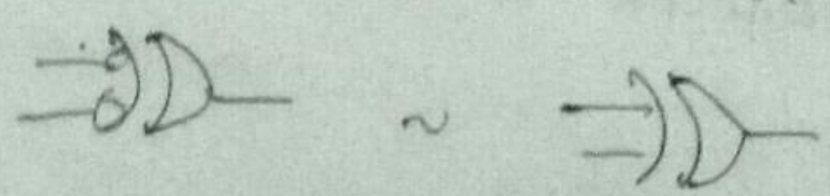
* Single bubbled EXOR ~ to Ex-NOR



* Single bubbled Ex-NOR ~ to Ex-OR



* Bubble EXOR is EXOR itself (So is Ex-NOR)



Scanned by TapScanner

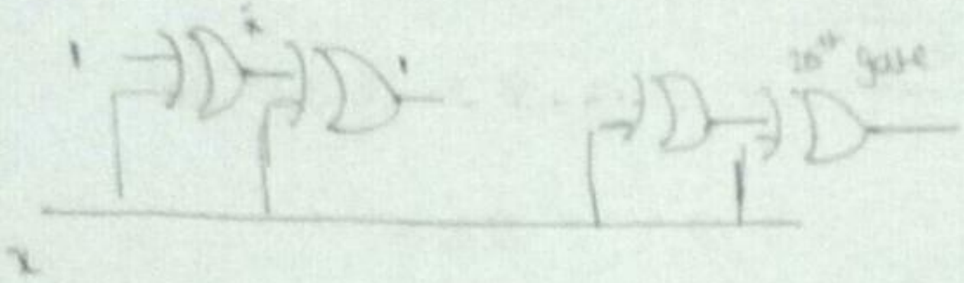
→ $x \oplus x = 0$ $\quad x \oplus \bar{x} = 1$

→ $x \odot x = 1$ $\quad x \odot \bar{x} = 0$

→ $x \oplus 0 = x$ $\quad x \oplus 1 = \bar{x}$

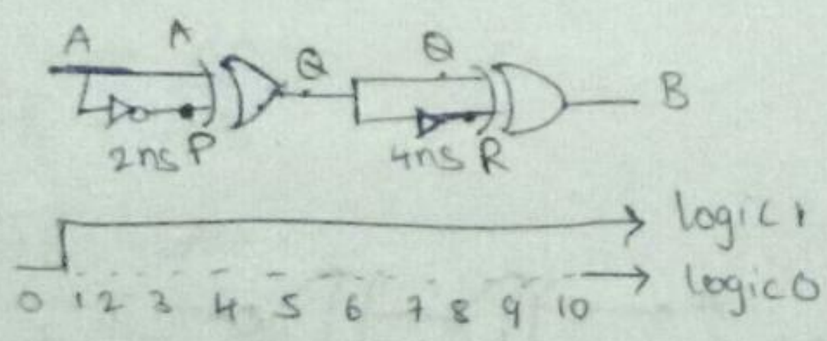
→ $x \odot 0 = \bar{x}$ $\quad x \odot 1 = x$

Q: In the below diagram, what is the o/p at 20th gate?



a) x b) $\bar{x}$ c) 1 ✓ d) 0

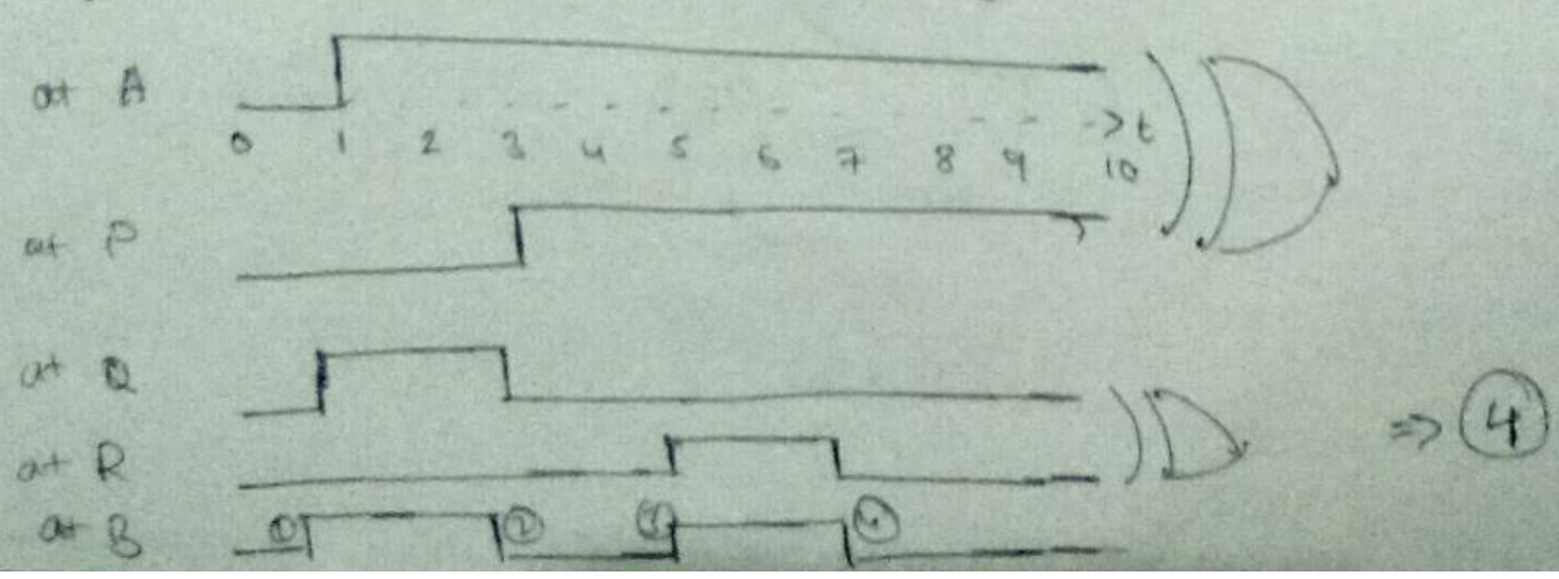
Q: In the below diagram t_{pd} there is no t_{pd} for XOR gates and all gates initial i/p & o/p are 0. First buffer non-inverting buffer → o/p is same as i/p except it provides some delay. takes 2ns and second buffer takes 4ns.



The above is applied at 'A'. The no of transitions (changes) occured at B from 0 to 10 ns time is _____

a) 1 b) 2 c) 3 d) 4 ✓

Sol: Method 1:



⇒ 4

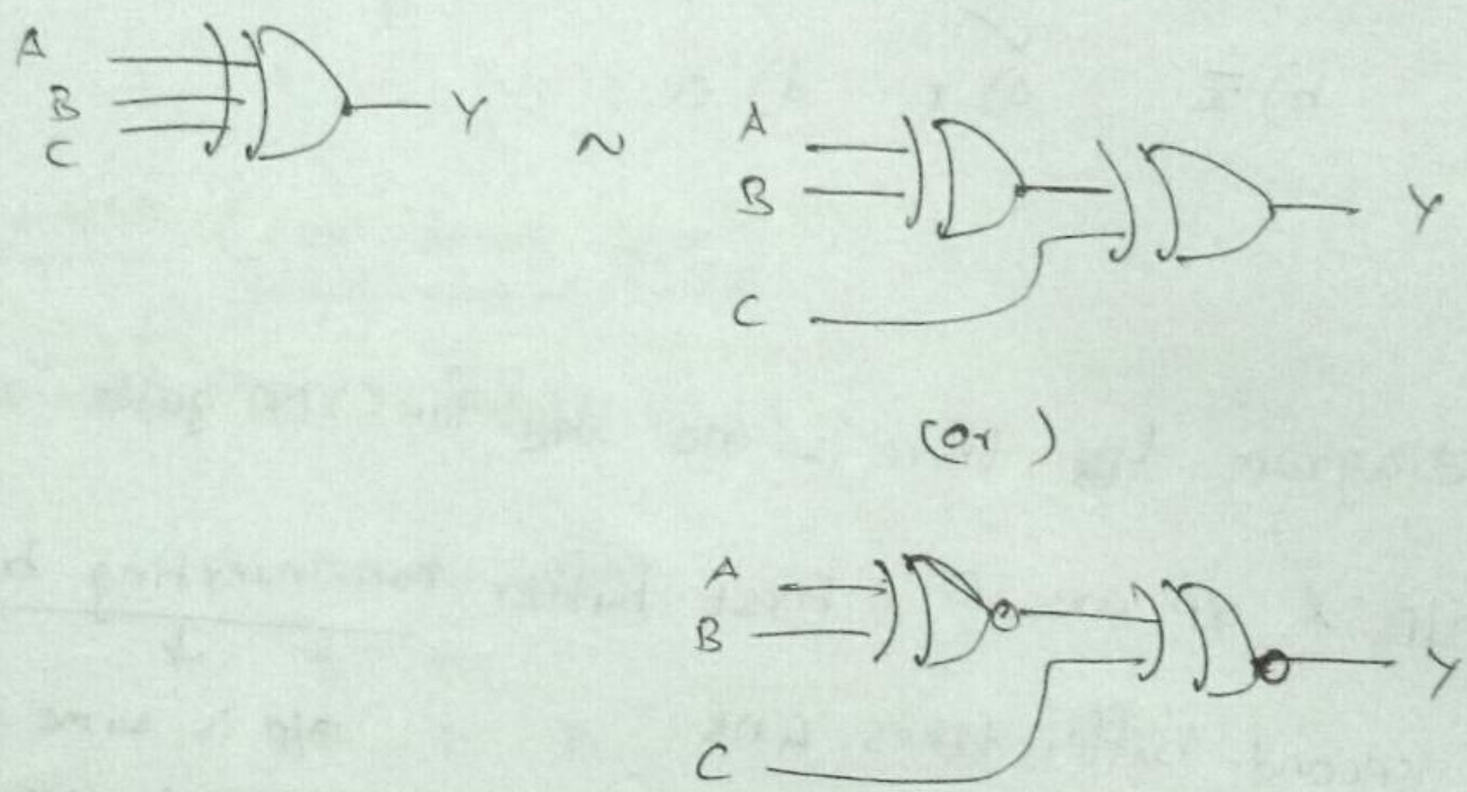
Scanned by TapScanner

Method 2 :

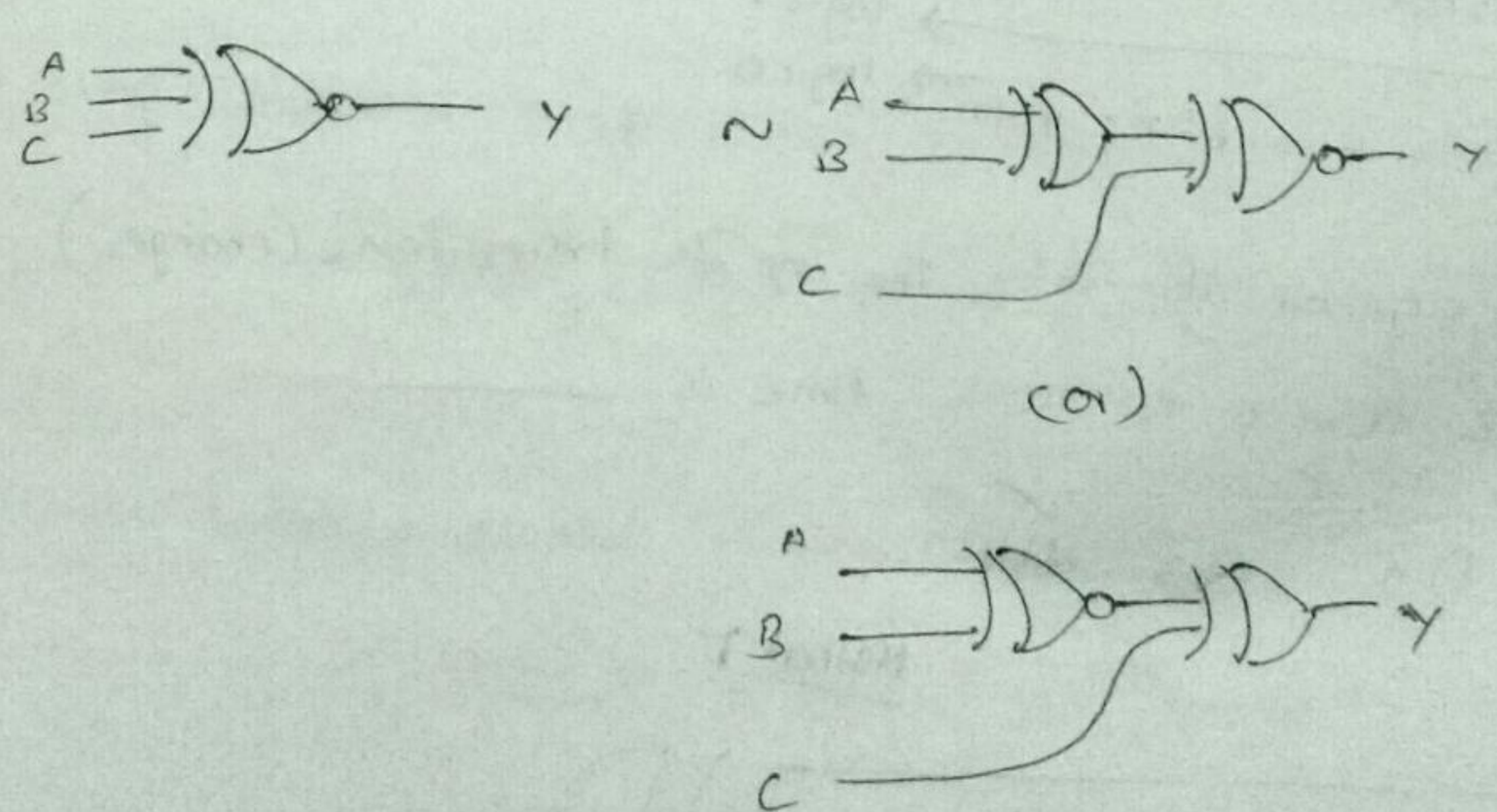
time	A	P	Q	R	B
0-1	0	0	0	0	0
1-2	1	0	1	0	1
2-3	1	0	1	0	1
3-4	1	1	0	0	0
4-5	1	1	0	0	0
5-6	1	1	0	1	1
6-7	1	1	0	1	1
7-8	1	1	0	0	0
8-9	1	1	0	0	0
9-10	1	1	0	0	0

① ② ③ ④

3-input Ex-OR



3-input Ex-NOR



Scanned by TapScanner

Exclusive gates special properties:

Let $A \oplus B = Y$

$\Rightarrow A \oplus B = 0 \oplus Y$

$A \oplus B \oplus Y = 0$

$\Rightarrow A = B \oplus Y$

$B = A \oplus Y$

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

Let $A \oplus B \oplus C = Y$

$A \oplus B = Y \oplus C$

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

$\bar{A} \oplus \bar{B} = Y \oplus C$

$A \oplus B = \bar{Y} \oplus \bar{C}$

$\bar{A} \oplus B = \bar{Y} \oplus C$

$\bar{A} \oplus B = Y \oplus \bar{C}$

(A variable can be shifted from L.H.S to R.H.S (or) R.H.S to L.H.S along with operator. This is a special property of exclusive gates)

Problems

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

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

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

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

6) $A \oplus B = 0 \Leftrightarrow \cancel{A \neq B}\ A = B$

$\overline{A+B} = 0 \Leftrightarrow A = 1$ (or) $B = 1$

$\bar{A} \cdot B = 0 \Leftrightarrow A = 1$ (or) $B = 0$

$A \oplus B = 1 \Leftrightarrow A \neq B$

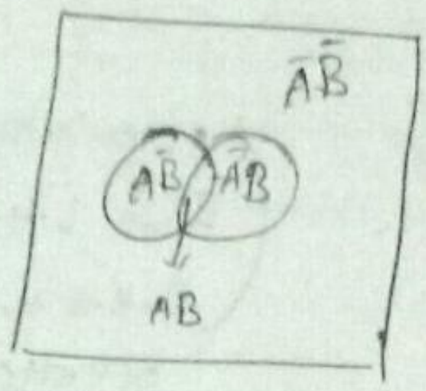
Scanned by TapScanner

Venn diagrams for logic gates:

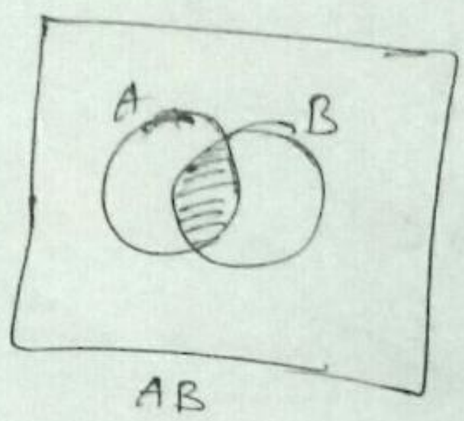
→ venn diagram is collection of circles.

→ one circle is required for one input.

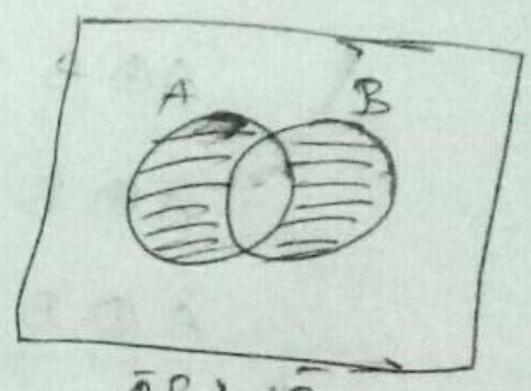
→ In venn diagram shaded area represents High output.



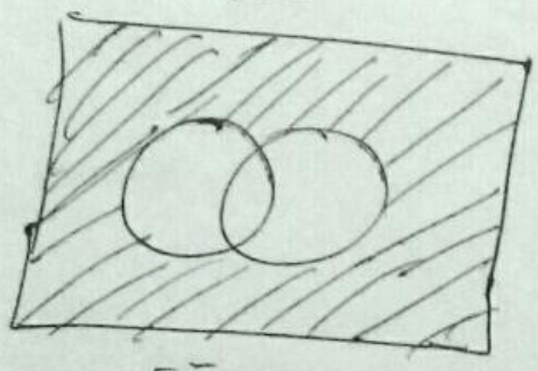
venn diagram for AND gate:

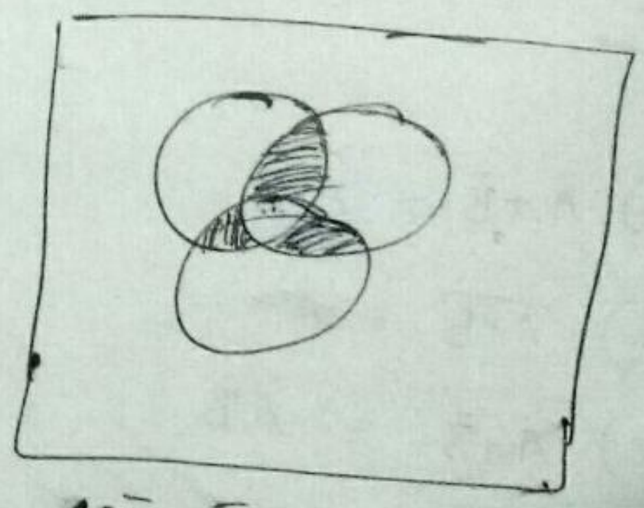


Ex-OR



NOR





Different methods used to obtain expression from given truth table:

(i) Apply S.O.P (minterm)

(ii) Apply P.O.S (maxterm)

(iii) Apply k-map

(iv) Apply tabulation method

Note: Result is same from any method

→ k-map technique is used when given function is having don't care(x).

→ k-map range is 2 to 5 variables

→ when the given function is having more than 5 variables, apply tabulation technique.

→

	AB	Y	
0	00	0	~~minterm~~ maxterm
1	01	0	~~maxterm~~ maxterm
2	10	1	~~maxterm~~ minterm
3	11	1	~~maxterm~~ minterm

For above table

(i) <u>SOP</u>: (minterm)

$= m_2 + m_3$

$A\bar{B} + AB$

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

(ii) <u>POS</u> (maxterm)

$M_0 . M_1$

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

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

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

$= A(1)$

$= A$

	I/P	Term Designation			
	x y	Σ	S.O.P '1'	Π	P.O.S '0'
0	0 0	$\bar{x}\bar{y}$	m_0	$x+y$	M_0
1	0 1	$\bar{x}y$	m_1	$x+\bar{y}$	M_1
2	1 0	$x\bar{y}$	m_2	$\bar{x}+y$	M_2
3	1 1	xy	m_3	$\bar{x}+\bar{y}$	M_3

Scanned by TapScanner

→ Sum of all minterms is high (1)

$m_0 + m_1 + m_2 + m_3$

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

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

$= x + \bar{x}$

$= 1$

→ Product of all marterms is low (0)

$M_0 . M_1 . M_2 . M_3$

$\underline{(x+y)(x+\bar{y})}\,\underline{(\bar{x}+y)(\bar{x}+\bar{y})}$

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

$(x+0)(\bar{x}+0)$

$(x\bar{x}) = \underline{0}$

<u>Canonical form</u> :

→ It is also known as standard form

→ An SOP/POS expression is said to be in canonical form, when given terms are represented as functional variables only.

→ Cannonical form is used to prepare the truth table for the given minimized expression

Eg:

Prepare truth table for

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

Scanned by TapScanner

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

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

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

101 100 110 010

$m_5 + m_4 + m_6 + m_2$

$\Sigma_m(2,4,5,6) = \Pi M(0,1,3,7)$

ABC	Y
000	0
001	0
010	1
011	0
100	1
101	1
110	1
111	0

Shortcut :

	ABC
$A\bar{B}$	10X
	100 (4)
	101 (5)
$B\bar{C}$	X10
	110 (6)
	010 (2)

$\Rightarrow \Sigma m(2,4,5,6)$

Eg:2 $f(ABC) = A + \bar{B}C$

	ABC
A	1XX : (4,5,6,7)
$\bar{B}C$	X01 : (5,1)

$\Rightarrow \Sigma m(1,4,5,6,7)$

Scanned by TapScanner

Redundant term:

→ It is a repeated term.

→ It's removal from expression does not disturb it's original form

Eg:3 Find redundant term in

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

a) $A\bar{B}$ b) $\bar{A}C$ c) $\bar{B}C$ ✓ d) None

	ABC
$A\bar{B}$	10x : (4,5)
$\bar{A}C$	0x1 : (1,3)
$\bar{B}C$	x01 : (1,5)

$\Sigma_m(1,3,4,5)$

If we remove $\bar{B}C$ still m_1 & m_5 exists. So $\bar{B}C$ is the redundant terms

Problems

(2)

	VWXYZ
X	001 ~~1xxxx~~ : (16 to 31)
Y	

	VWXYZ	
X	xx1xx	→ 16
$\bar{Y}Z$	xxx01	→ 8

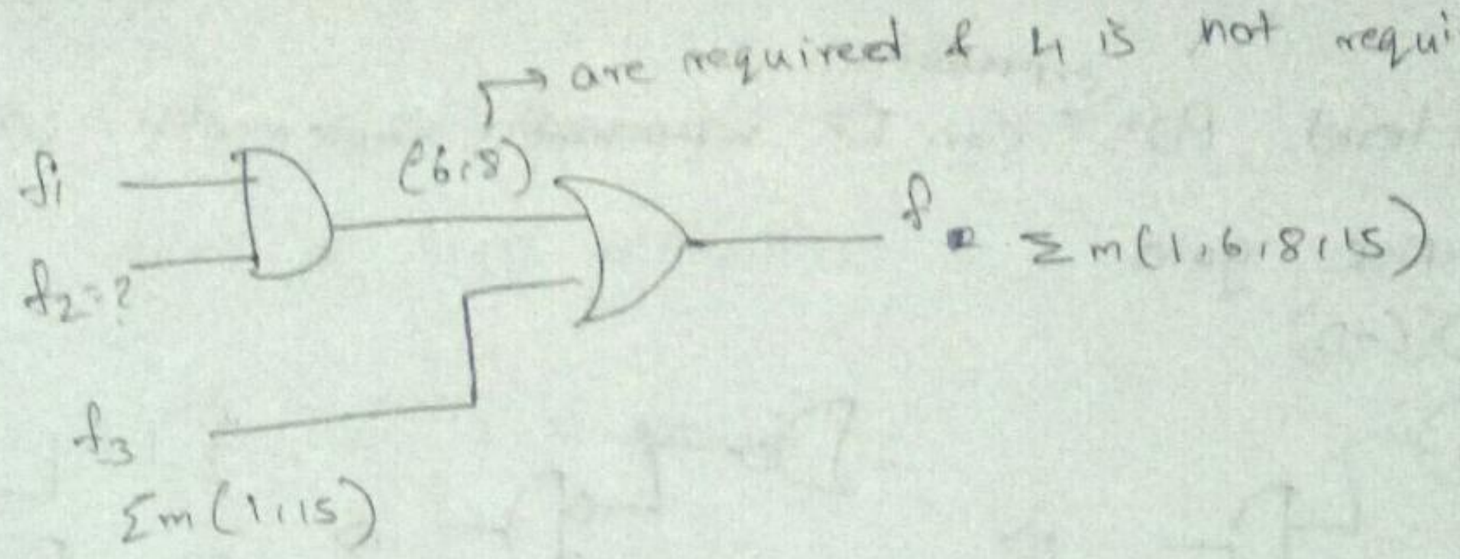
} common: xx101 ⇒ 4 ↓ redundant terms

∴ 16+8-4 = 20 (minterms)

no of maxterms = 32-20 = 12

Scanned by TapScanner

③



(2007)

Q: Max no of distinct boolean functions that can be implemented with 'n' binary variables is

a) 2^n b) 2^{2^n} ✓ c) $2^{2^{n-1}}$ d) None

Consider

	xy	F_0	F_1	F_2	- - - -	F_{15}
0	00	0	0	0		1
1	01	0	0	0		1
2	10	0	0	1		1
3	11	0	1	0		1

2^n

0000

with 2 variables, we can generate 2^2 combinations

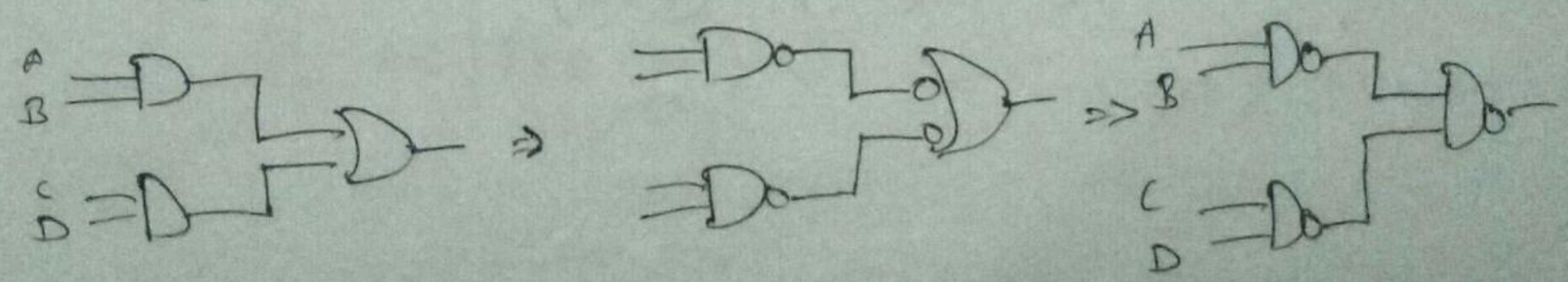
with 2^2 combinations, max no of distinct functions $= 2^{2^2} = 16$

with n variables, we can generate 2^n combinations

with 2^n combinations, max no of distinct functions $= 2^{2^n}$

→ Two level SOP ~~con~~ expression can be implemented with 2level NAND NAND gates

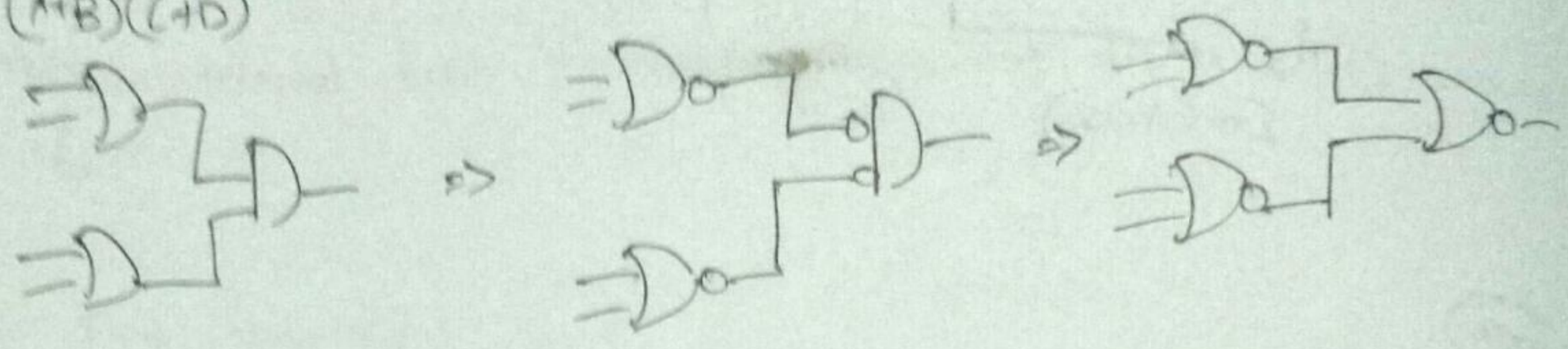
AB+CD



Scanned by TapScanner

→ To Two level POS expression can be ~~represented~~ implemented with 2 level NOR-NOR gates:

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



Dual function (F^D):

In the given function if '+' is replaced by '.' and '.' is replaced by '+' one new function will be generated.

→ This new function is dual function for the given input fn.

→ If all minterms/Maxterms produced by the input fn and its dual function are same then input function is known as self dual function (SDF)

→ Maximum no of self dual functions that can be implemented with n binary variables $= 2^{2^{n-1}}$

Note:

→ To become SDF, in the given function the no of minterms must be equal to no of maxterms (It doesn't mean that a function is SDF if no of minterms & no of maxterms are equal. Its just a pre-requirement)

Eg: $F(ABCD) = AB+CD$ (S.O.P)

$F^D(ABCD) = (A+B).(C+D)$ (P.O.S)

Scanned by TapScanner

Eg 1: Check if $F(ABC) = AB + BC + AC$ is SDF

$F(ABC) = AB + BC + AC$

	ABC	
AB	11x	(6,7)
BC	x11	(3,7)
AC	1x1	(5,7)

$\Sigma m(3,5,6,7) = \Pi M(0,1,2,4)$

no of minterms = no of maxterms

$F^D(ABC) = (A+B)(B+C)(A+C)$

	ABC	
A+B	00x	(M_0, M_1)
B+C	x00	(M_0, M_4)
A+C	0x0	(M_0, M_2)

$\Pi M(0,1,2,4) = \Sigma m(3,5,6,7)$

equal

$\therefore$ SDF

Eg 2: $F(ABC) = \bar{A}B + BC + AC$ is SDF ~~are~~ or not?

	ABC	
$\bar{A}B$	01x	(2,3)
BC	x11	(3,7)
AC	1x1	(5,7)

$\Rightarrow \Sigma m(2,3,5,7) = \Pi M(0,1,4,7)$

no of minterms = no of maxterms

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

	ABC	
$\bar{A}+B$	10x	(4,5)
B+C	x00	(0,4)
A+C	0x0	~~(5,7)~~ (0,2)

$\Pi M(\cancel{0,4,5,7}$

$\Pi M(0,4,5,2) = \Sigma m(1,3,6,7)$

not equal

$\therefore$ Not SDF

Scanned by TapScanner

Eg3: $F(ABC) = A + \bar{B}C$ is SDF or not

	ABC
A	1 x x (4,5,6,7)
$\bar{B}C$	x 0 1 (1,5)

$\Rightarrow \Sigma_m(1,4,5,6,7) = \Pi M(0,2,3)$

no of minterms $\neq$ no of maxterms

$\therefore$ not SDF

Consider

	x y	F_0	F_1	F_2	F_3	F_4	F_5	F_6 (EXOR)	F_7	F_8	F_9 (Ex-NOR)	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		x	x	x	✓	x	✓	x	x	x	x	✓	x	✓	x	x	x

→ The rounded ones contain equal no of minterms & maxterms.

→ Here no of variables $n = 2$.

→ $\therefore$ no of SDFs possible $= 2^{2^{n-1}} = 2^{2^1} = \underline{\underline{4}}$

→ Of above 6 rounded ones 4 are SDF. The EXOR & Ex-NOR are not SDF. Rest are SDF.

Note:

In 3 variable, it is not necessary that EXOR & Ex-NOR functions are not SDF.

→ The Rounded & ticked (✓) Ones are SDFs.

Scanned by TapScanner

Universal gate applications: ~~Any target~~

→ Any expression can be implemented by universal gates.

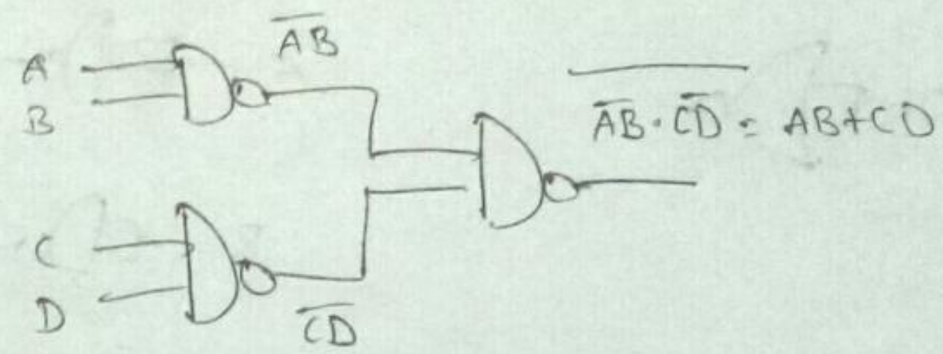
→ while implementing expression with NAND gates, first bring entire expression in SOP form then apply Demorgan's law.

→ while implementing expression with NOR gates, first bring entire expression in POS form then apply demorgan's law.

Consider

$f(A,B,C,D) = AB + CD$ (SOP form)

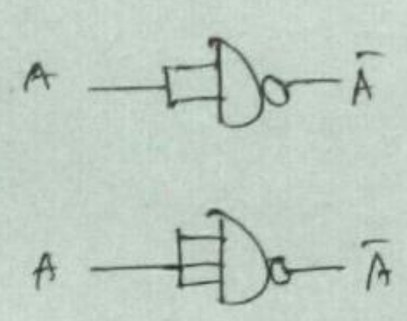
$AB + CD = \overline{(\overline{AB + CD})} = \overline{\overline{AB} \cdot \overline{CD}}$



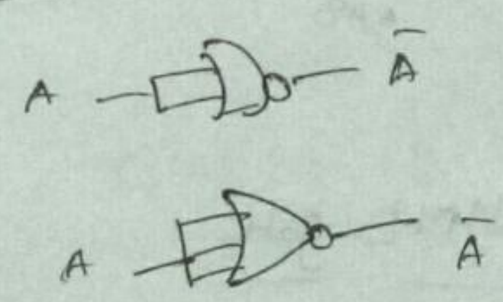
NOT gate with universal gates:

I/P: A target $\bar{A}$

with NAND



with NOR



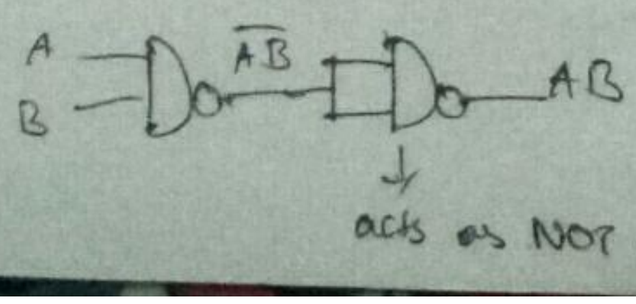
AND gate with universal gates:

I/P: A, B target : AB available : $\overline{A \cdot B}$

with NAND

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

⇒

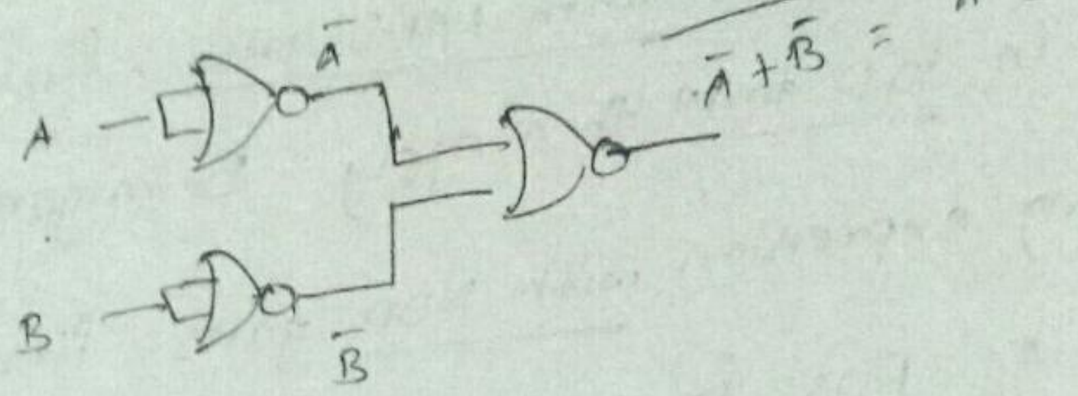


Scanned by TapScanner

with NOR

available: $\overline{A+B}$

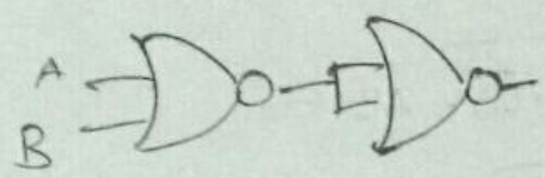
$\bar{A}$

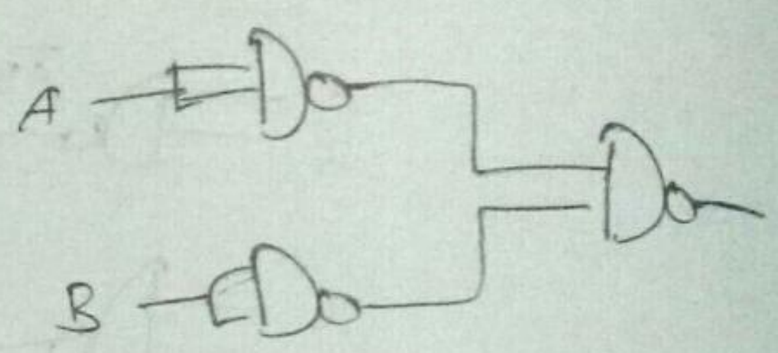


OR gate with Universal gates

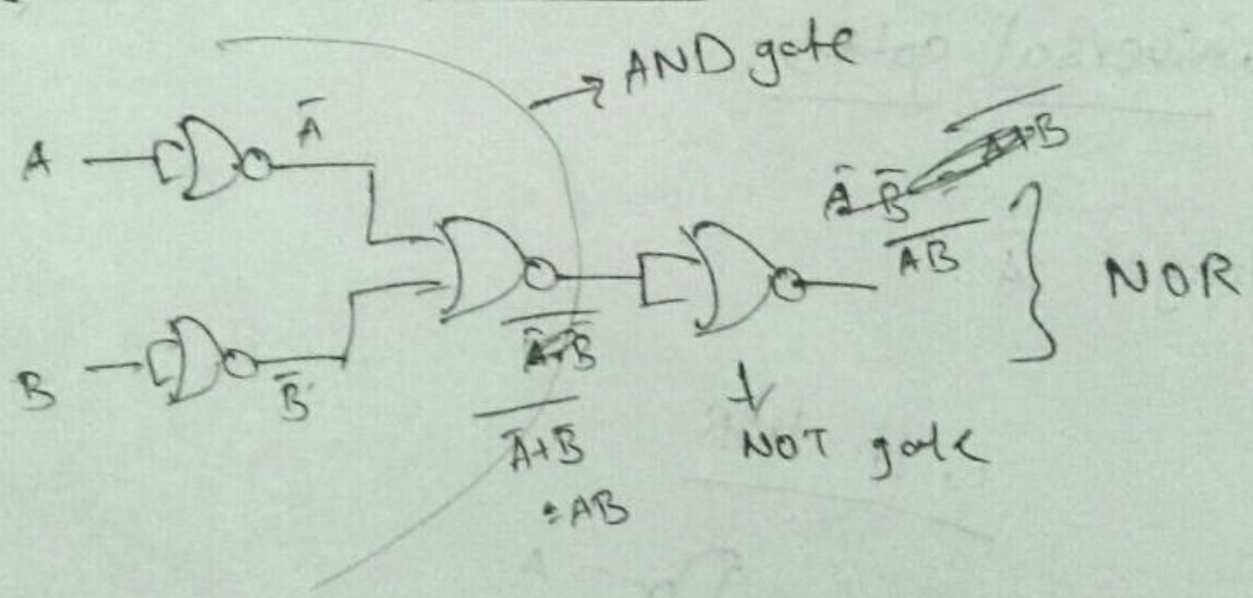
with NOR ~~NAND~~ gate

with NAND ~~NOR~~ gate

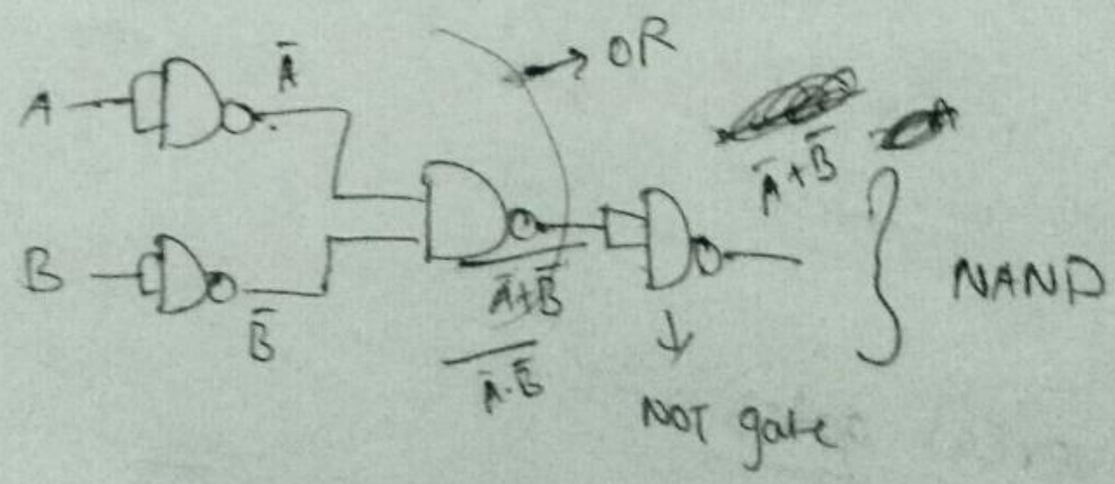




NAND gate with NOR gate:



NOR gate with NAND gate



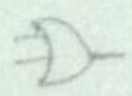
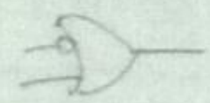
$A+B$

$\overline{AB}$

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

Scanned by TapScanner

Q: (Previous EC) Universal gate is used to implement any target expression and we are permitted to use the required no of gates given below gate

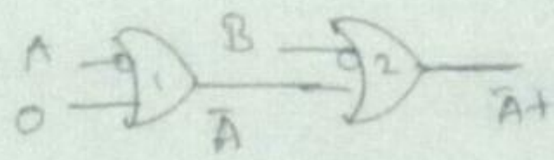
(i) (ii)  (iii) 

which one of the above is universal

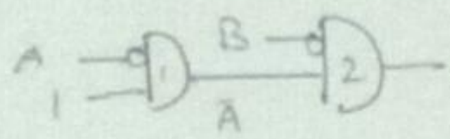
a) i b) ii c) iii ✓ d) None of the above

* It is clear that AND & OR can't be used

Consider

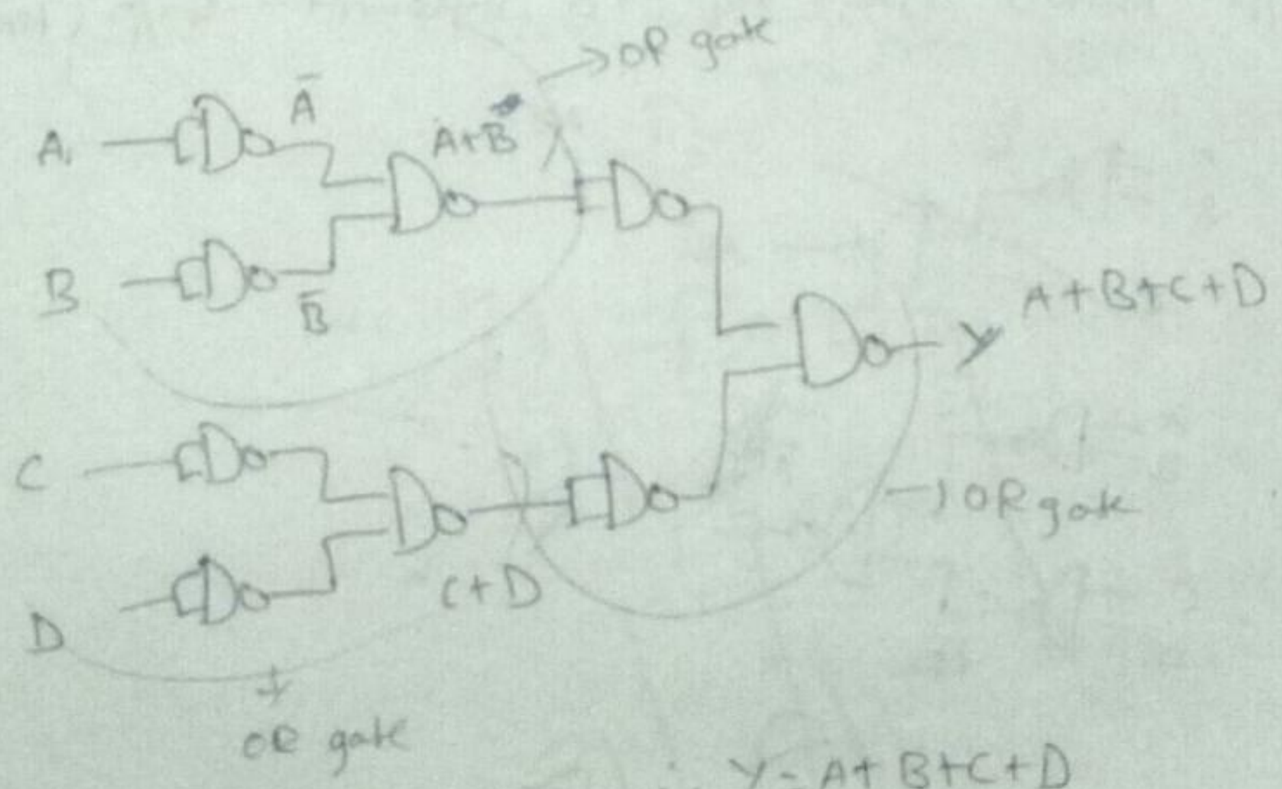


$\bar{A}+\bar{B} = \overline{AB}$ = NAND



$\bar{A}\cdot\bar{B} = \overline{A+B}$ = NOR } This is just shown for example to show this is also possible.

Q: what is expression at Y



$\therefore Y = A+B+C+D$

Q: Min no of 2 i/p NAND gates to implement

$A + A\bar{B} + A\bar{B}C + ABC\bar{D}$ is _____

$\Rightarrow A(1+\bar{B}+\bar{B}C+BC\bar{D}) = A(1)$

$= A$

$\therefore$ no gates are required

Ans: 0

Scanned by TapScanner

Q: Min no of 2 i/p NOR gates required to implement $(A+BC)$

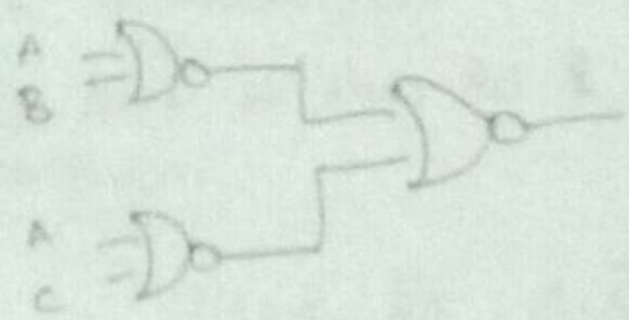
is _____

$A+BC$ ↓ convert into POS

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

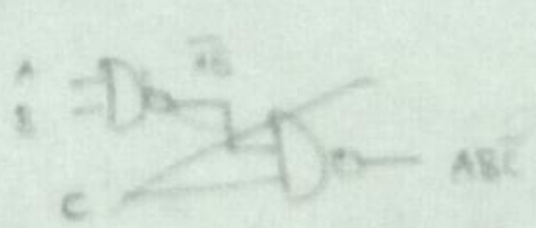
$$= \overline{\overline{(A+B)(A+C)}}$$

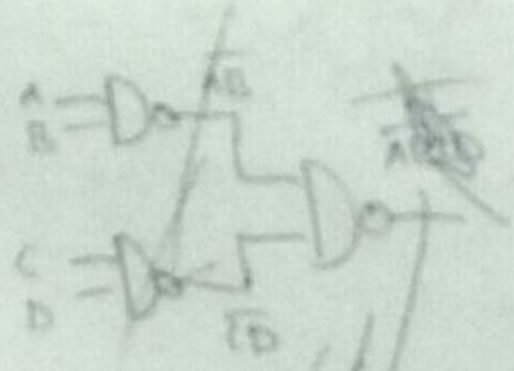
$$= \overline{\overline{(A+B)} + \overline{(A+C)}}$$



Ans: 3

Q: Min no of 2 i/p NAND gates req to implement 4 i/p NAND gate





Target: $\overline{ABCD}$

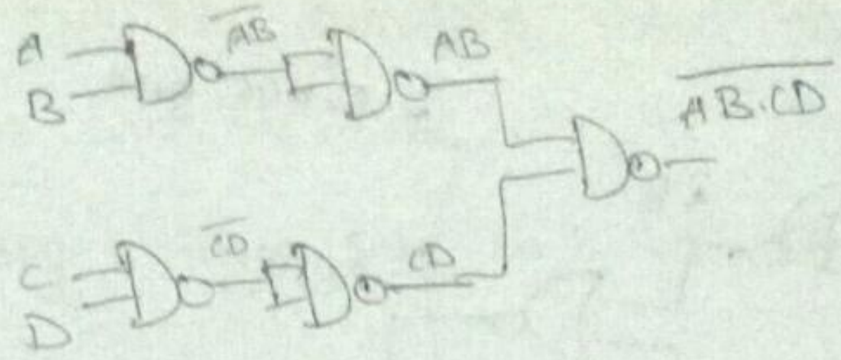
Method:

1st: $\overline{AB}$ 2nd: $\overline{AB} \rightarrow AB$

3rd: $\overline{CD}$ 4th: $\overline{CD} \rightarrow CD$

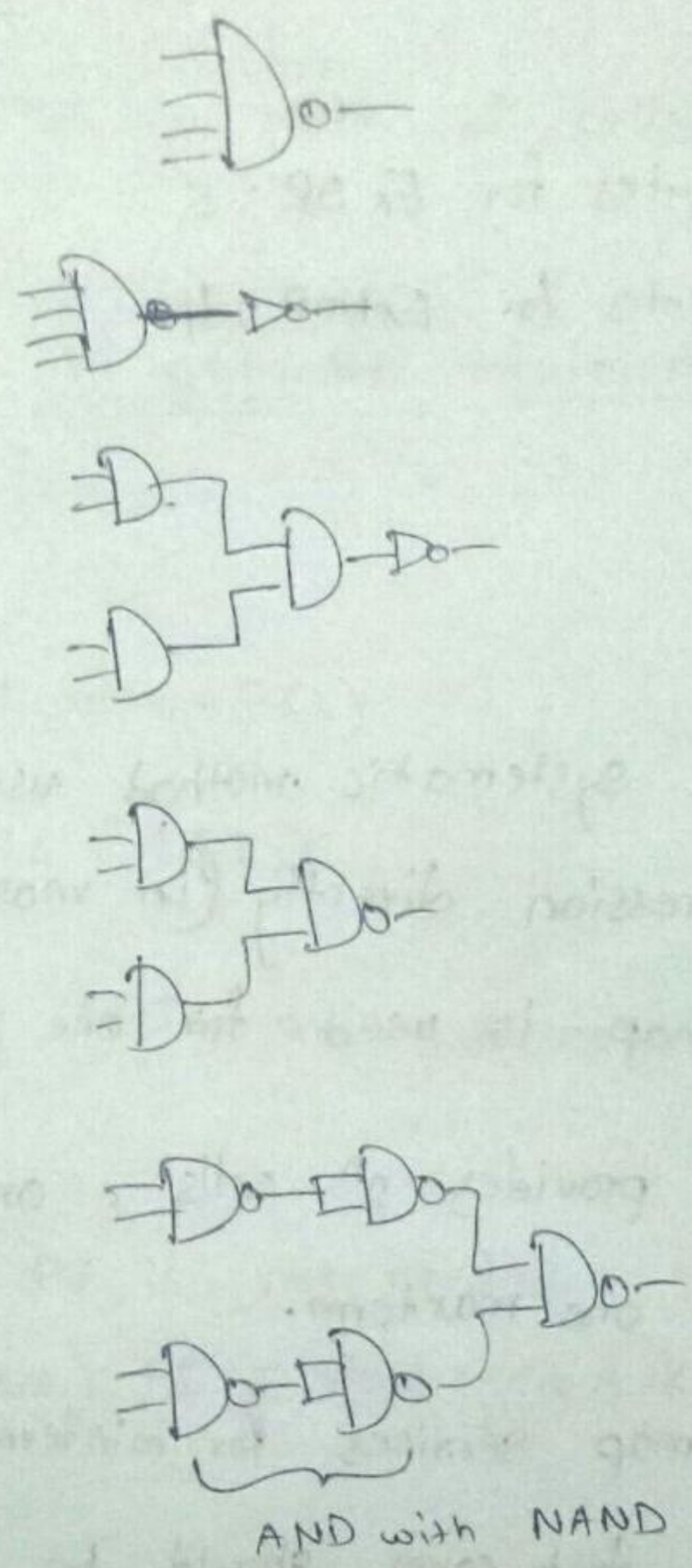
5th: $AB \,\&\, CD \rightarrow \overline{ABCD}$

Scanned by TapScanner



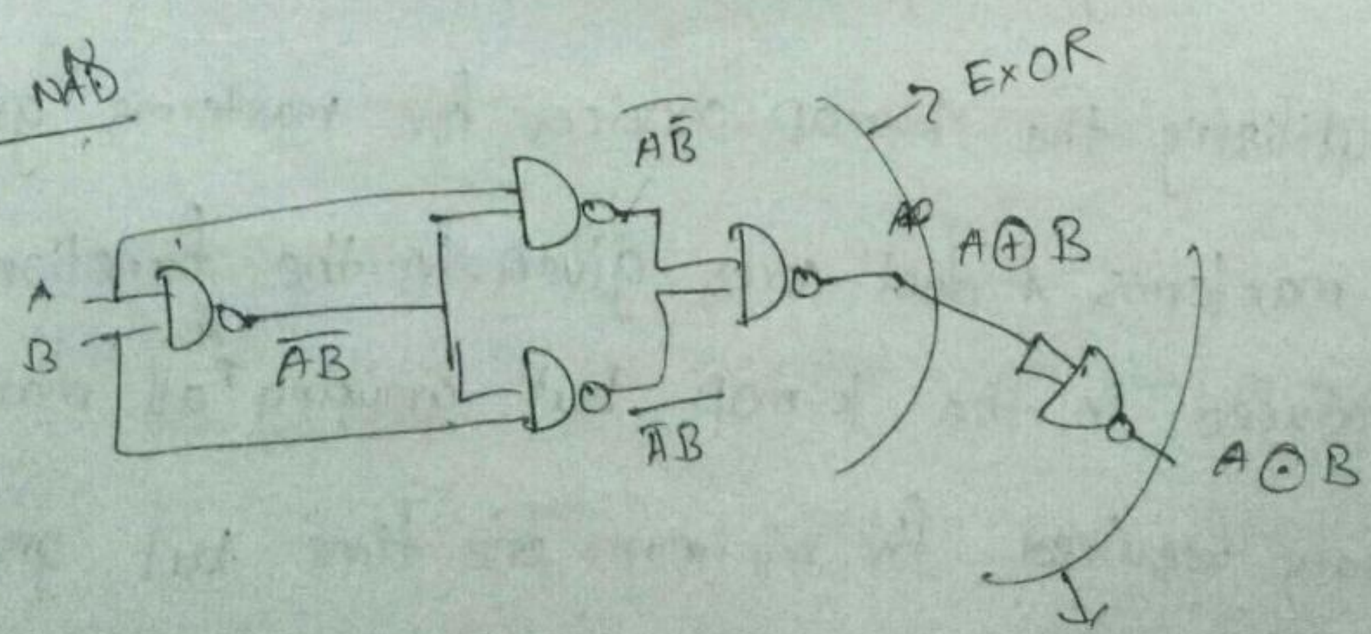
Method2 :

Target :



Exclusive gates with NAND/NOR gates :

with NAND

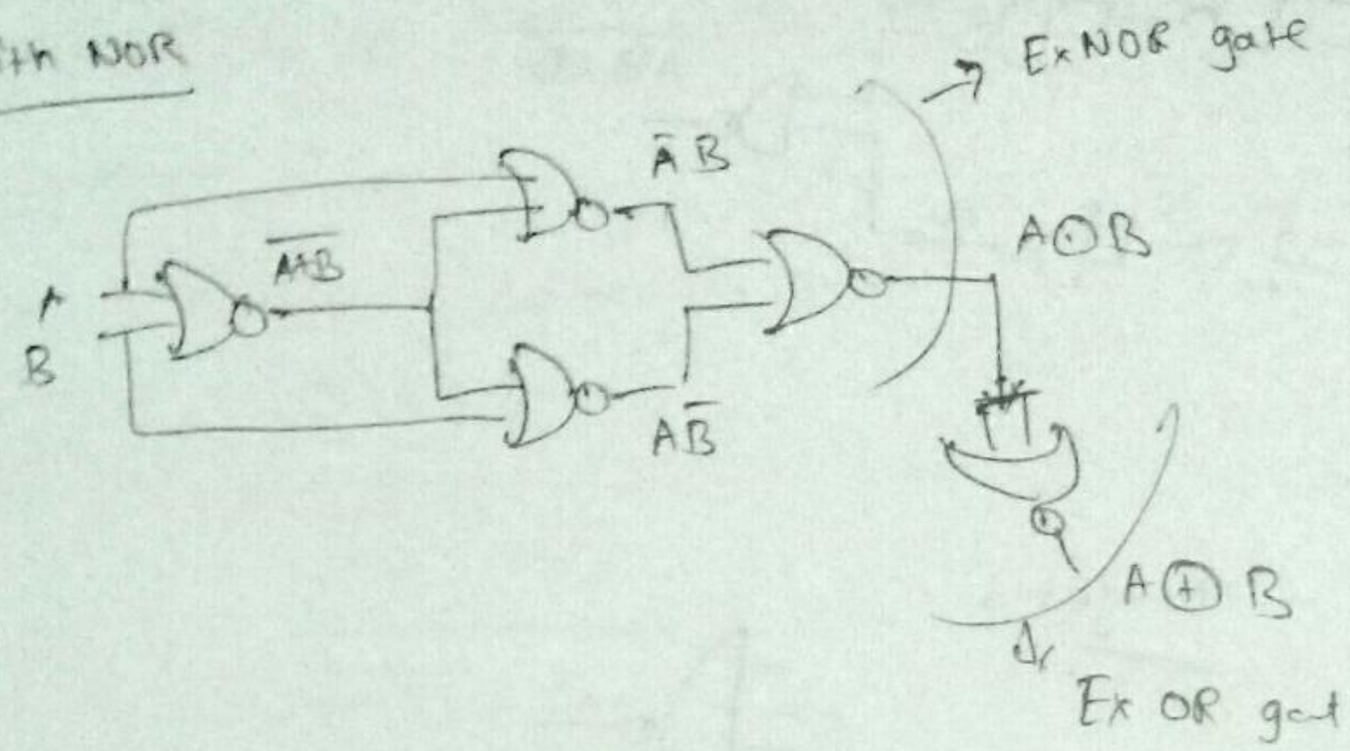


NAND gates for ExOR : 4

NAND gates for ExNOR : 5

Scanned by TapScanner

with NOR



No of NOR gates for Ex OR : 5

NOR gates for ExNOR : 4

2. K-Maps :

It is a systematic method used to get the ~~minimum~~ minimized expression directly (in most cases). Generally one n-variable k-map is used for one n variable function.

→ n variable k-map provides 2^n cells, one cell is used to represent one minterm or one maxterm.

→ While using k-map services for minterms ~~groups~~ grouping, all the minterms, dont cares should be produced in the k-map.

→ Grouping of all min-terms is compulsary required for minimum one time but grouping of don't care cell is not compulsory

→ while utilizing the k-map services for maxterms grouping all the maxterms & dont cares given in the function should be produced in the k-map but grouping of all maxterms is compulsary required for minimum one time but grouping of don't care cell is not compulsary

Scanned by TapScanner

→ A cell can be grouped more than one time when its services are required.

→ The target term size will be reduced by grouping maximum no. of cells.

→ In n-variable, k-map each cell is having 'n' chances for grouping with other cells.

→ Single grouping is possible ~~with~~ for 2^x cells only

Standard method used to get the minimized expression with k-map:

Step I: get prime implicants (PIs)

Step II: get essential PIs (EPIs)

Step III: check, if only EPIs are sufficient to cover all minterms or ~~not~~ NOT

If Yes: one more PI is not needed

Step IV: If No: one (or more) PI is ~~used~~ needed known as selective PI (SPI)

Prime Implicant: It is the minimized term ~~maintained~~ obtained by grouping the max no of cells

Essential PI: It's presence is essential in final minimized expression basically it is a PI.

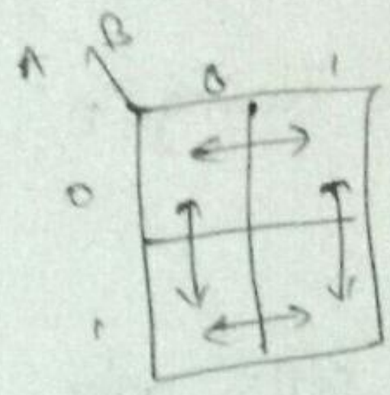
→ A PI is said to be essential when one (or more) cell is grouped only one time

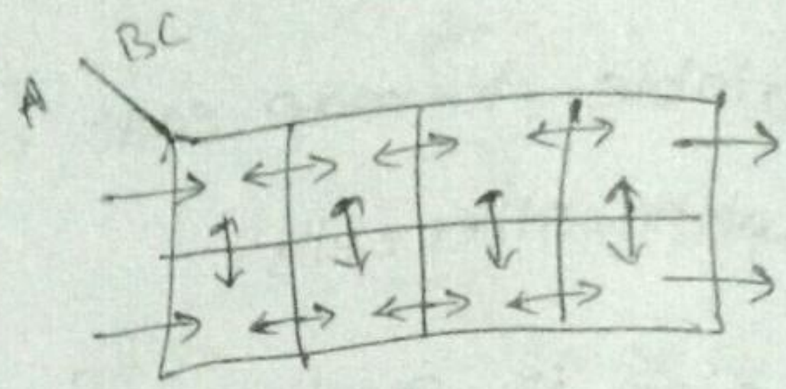
Selective PI: It is a PI, which is used to cover the minterms which are not covered by EPIs.

Scanned by TapScanner

Grouping: It is process of finding common bits.

Grouping chances:

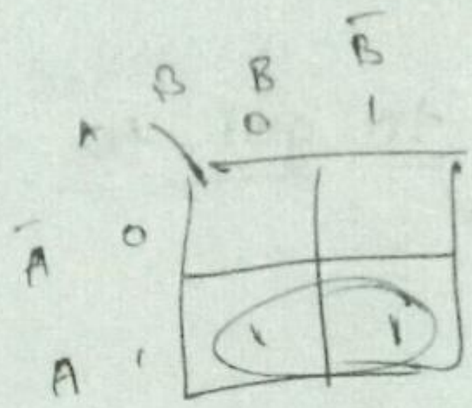




Consider

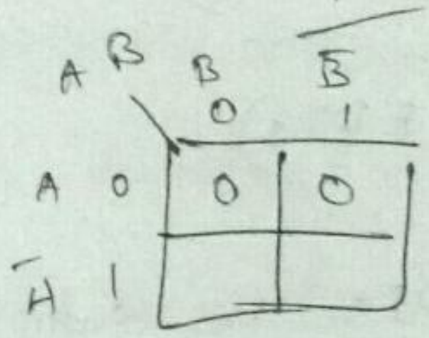
A	B	Y
0	0	0
0	1	0
1	0	1
1	1	1

using SOP



=> A

POS

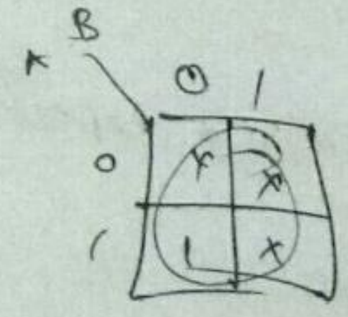


=> A

Note:

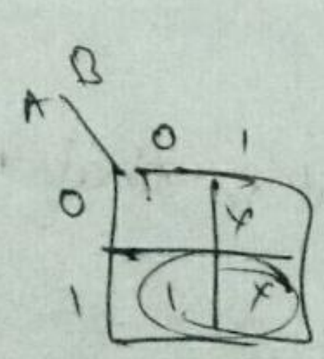
→ Don't care (x) grouping is used to reduce H/W logic circ

Find no of useful x's and term:



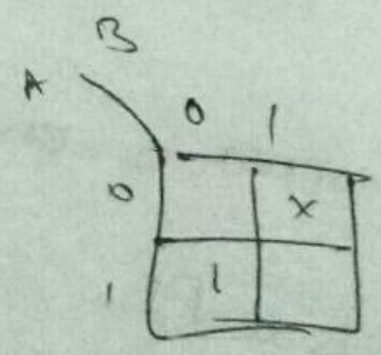
3 useful x's

term = '1'



1 useful

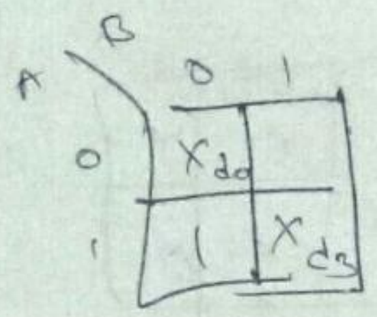
term = A



$A\bar{B}$

Scanned by TapScanner

Q: Find the most useful 'x' in given below for S.O.P grouping



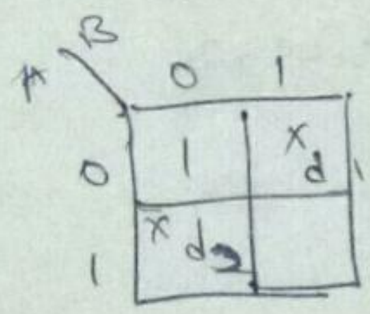
a) d_0 b) d_3 ✓ c) d_0 (or) d_3 d) none

grouping with d_0 : $\bar{B}$ (requires 1 NOT gate)

grouping with d_3 : A (requires no gates)

$\therefore d_3$

Q: Find most useful 'x' in given below for S.O.P grouping



a) d_1 b) d_2 c) d_1 (or) d_2 ✓ d) none

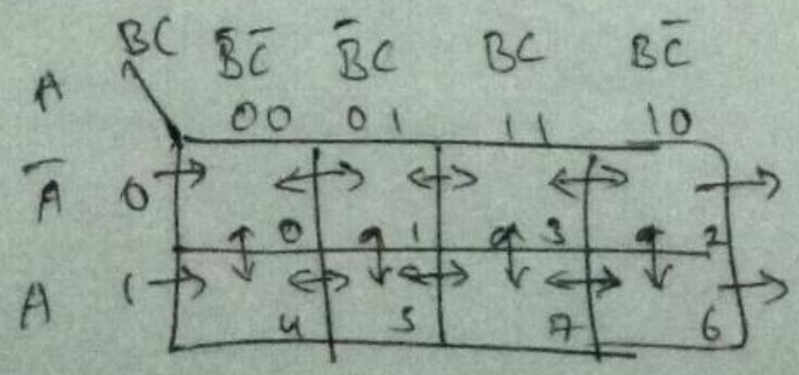
grouping with d_1 : $\bar{A}$ (req 1 NOT gate)

grouping with d_2 : $\bar{B}$ (req 1 NOT gate)

$\therefore d_1$ (or) d_2

Term size: No of variable available in the term.

Consider 3 variable k-map. It is drawn as below



Scanned by TapScanner

Find term & termsize in given k-maps:

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

term : -

termsize : 0

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

term : A

size : 1

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

term : $A\bar{B}$

size : 2

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

term $A\bar{B}\bar{C}$

size : 3

Expression for finding the term size in 'n' variable k-map:

$$\boxed{\log_2 \frac{N}{D}}$$

Here N = max no of cells given by the in base 2

D = Max no of cells under single group

Get result in base 2

→ only exponent gives term size

Scanned by TapScanner

Q: In n variable k-map, what is the term-size when '8' cells are grouped?

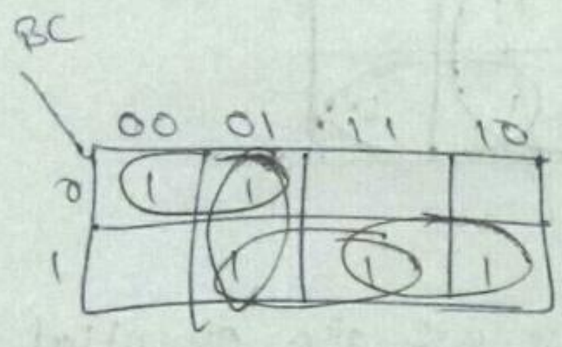
a) 3 b) 0 c) 8 d) n-3 ✓

Here $N = 2^n$

$D = 8 = 2^3$

term size $= \log_2 \frac{2^n}{2^3} = \log_2 2^{n-3} = \underline{n-3}$

Q:

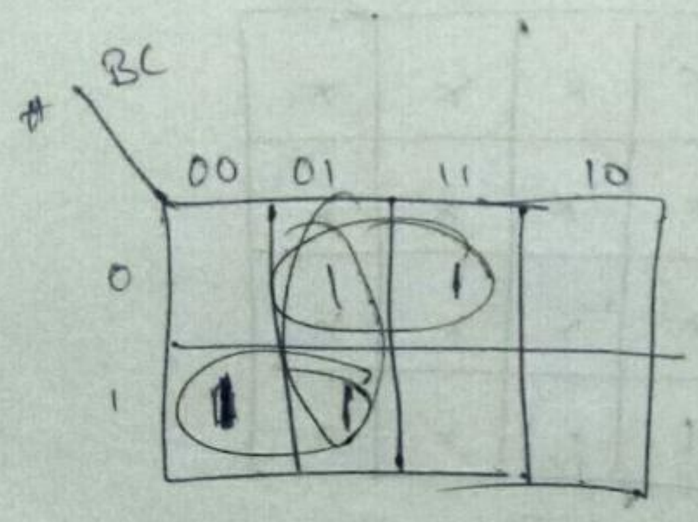


Chances (PI)	EPIs
$\bar{A}\bar{B}$	$\bar{A}\bar{B}$
$\bar{B}C$	AB
AC	
AB	

AC is selected

$f(ABC) = \underbrace{\bar{A}\bar{B} + AB}_{\text{not sufficient to cover all min-terms}} + \underset{\text{SPI}}{AC}$

Q:



Chances (PI)	EPIs
$A\bar{B}$	$A\bar{B}$
$\bar{B}C$	$\bar{A}C$
$\bar{A}C$	

$f(ABC) = \underbrace{A\bar{B} + \bar{A}C}_{\text{sufficent to cover all min terms. So SPI is not needed}}$

Scanned by TapScanner

Q: No of EPIs in given below k-map is

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

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

a) 1 b) 2 c) 3 d) None

Checks:

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

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

AB

EPIs

C

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

AB

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

These are also essential

∴ C is also essential.

4-variable k-map:

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

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

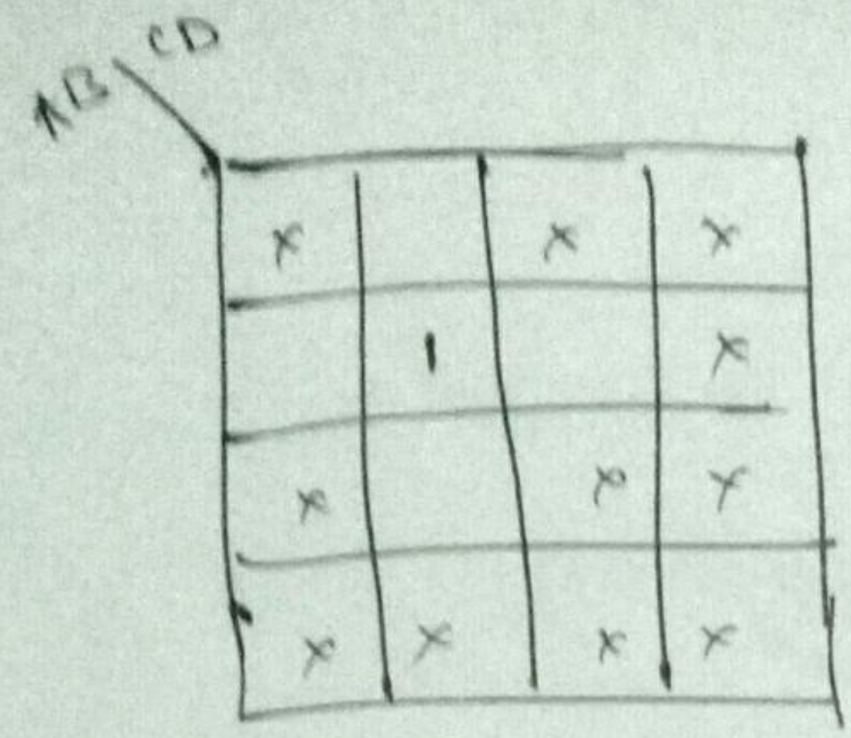
term: 1

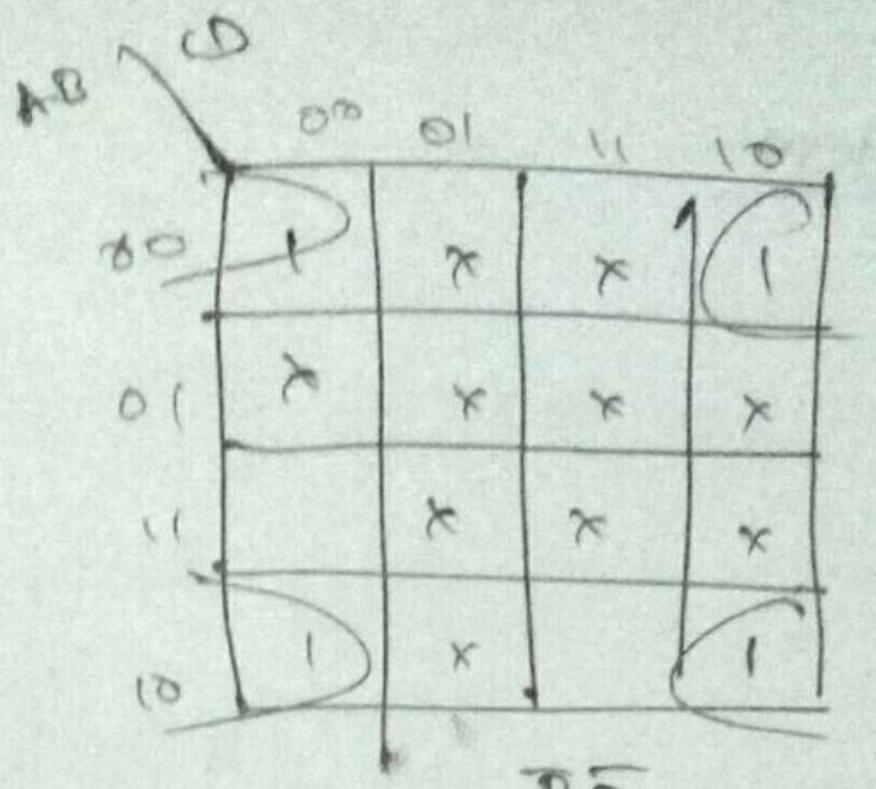
AB \ CD	00	01	11	10
00		x	x	x
01	x	x	x	x
11	x	x	x	x
10	1	x	x	x

term: A

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

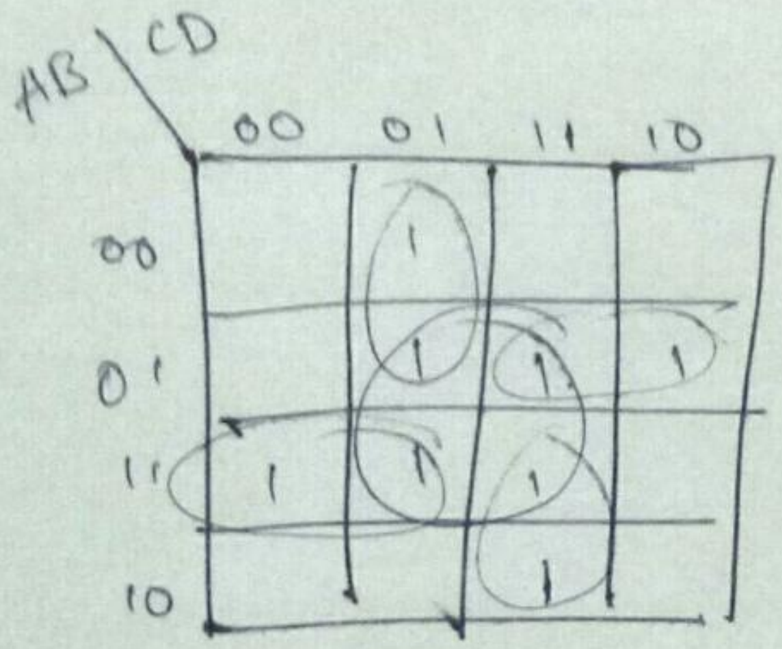
Scanned by TapScanner





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

Q: Consider below K-Map. The min no of terms appeared in final minimized expressions is _____



Chamas:

BD

$AB\bar{C}$ ✓

$\bar{A}\bar{C}D$ ✓

$\bar{A}BC$ –

ACD –

EPIs

$AB\bar{C}$

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

$\bar{A}BC$

ACD

(Here BD is called Redundant PI)

$$f(ABCD) = AB\bar{C} + \bar{A}\bar{C}D + \bar{A}BC + ACD$$

$\therefore$ Ans: 4

Scanned by TapScanner

Conversion b/w base r and r^n systems:

→ Direct conversion is possible

→ Each symbol in base r^n system is directly represented with 'n' bits

→ Direct conversion is possible if the largest symbol in that systems consists consists of all 1's in binary form.

Eg1: $(24)_8$ =

$(010100)_2$

Eg2: $(201A)_{16} = (0010000000011010)_2$

Eg3: $(10\,111\,111\,011\,011\,101)_2 = (?)_8$

010 | 111 | 111 | 011 | 011 | 101
2 | 7 | 7 | 3 | 3 | 5 $= (277335)_8$

2008
Eg4: $(567)_8 = (?)_{16}$

$(567)_8 = (101\,110\,111)_2$

0001 | 0111 | 0111
1 | 7 | 7 $= (177)_{16}$

Fraction data Conversions:

→ It requires inverse operations when compared to Integer conversions.

→ Consider $(a_{n-1} \ldots a_2\, a_1\, a_0 \,.\, b_1 b_2 \ldots b_n)_r = \sum^{n-1} a_i \times r^i + \sum^{n} b_i \times r^{-i}$

Scanned by TapScanner

Eg: $(26.8125)_{10} = (11010.1101)_2$

$= (32.64)_8$

$= (1A.D)_{16}$

base 2

$2 \,|\, 26$ 1010

$2 \,|\, 13 - 0$

$2 \,|\, 6 - 1$

$2 \,|\, 3 - 0$

$1 - 1$

LIFO

~~11010.~~

FIFO

$.8125 \times 2$

1.625×2

1.250×2

0.50×2

1.0

1101

i.e. (11010.1101)

In decimal form also base r to r^n conversion is allowed. But after decimal zeroes are appended at end

Consider

$(11010.1101)_2 = (?)_8$

$(11010)_2 = (\underline{011}\,\underline{010})_2 = (32)_8$

$(.1101)_2 = (.\underline{110}\,\underline{100})_2 = (64)_8$

$(11010.1101)_2 = (32.64)_8$

base 8

$(26)_{10} = (32)_8$

$.8125 \times 8$

6.5000×8

4.0

64

$\Rightarrow (26.8125)_{10} = (32.64)_8$

base 16 :

$(26)_{10} = (1A)_{16}$

$.8125 \times 16$

13.0000

D

$\Rightarrow (26.8125)_{10} = (1A.D)_{16}$

Now Consider

$(11010.1101)_2 \Rightarrow (11010)_2 = (26)_{10}$

$2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$

$.1101 \Rightarrow 1 \times \frac{1}{2} + 1 \times \frac{1}{4} + 0 \times \frac{1}{8} + 1 \times \frac{1}{16}$

$= \frac{8+4+0+1}{16} = \frac{13}{16} = 0.8125$

$\Rightarrow \Rightarrow 26.8125$

Scanned by TapScanner

$\Rightarrow (32.64)_8 \quad (32)_8 = (26)_{10}$

64

$\cancel{68}$ $\frac{6}{8} + \frac{4}{64} = \frac{52}{64} = \frac{13}{16} = 0.8125$

$\Rightarrow (26.8125)_{10}$

$\rightarrow (1A.D)_{16}$

$(1A)_{16} = (26)_{10}$

16^{-1}

$D \Rightarrow \frac{13}{16} = 0.8125$

Q: 3

Q: Let $\frac{(312)_x}{(20)_x} = (13.1)_x$. Find x.

$\Rightarrow$ Convert into base 10

$$\frac{3x^2+x+2}{2x} = x+3+\frac{1}{x}$$

$$\frac{3x^2+x+2}{2\cancel{x}} = \frac{x^2+3x+1}{\cancel{x}}$$

$3x^2+x+2-2x^2-6x-2=0$

$x^2-5x=0$

$x(x-5)=0$

$x=5$ (or) $x=0$

Scanned by TapScanner

Signed data representation

→ In unsigned binary data all the bits are used to provide magnitude (value), but in signed binary data MSB is reserved for representing sign and remaining bits are used for providing value.

→ 0 is occupied in MSB position for representing +ve data. otherwise it is 1

→ There are 3 methods to represent signed data

(i) signed magnitude representation

(ii) signed 1's complement representation

(iii) signed 2's complement representation.

→ In all above representations the +ve data is represented in normal form i.e; no complement is required but -ve data is represented in the complement given in the notation only.

→ Result for the given data is same if 1's complement operation or 2's complement operation is performed twice

Consider below table

Representation	+5	-5
normal method	0101 (0x)	1101 (1x)
1's Complement	0101 (0x)	1010 (1$\bar{x}$)
2's complement	0101 (0x)	(1011) (1$\bar{x}$+1)

→ 1's of x is $\begin{cases} 0x & \text{if } x \geq 0 \\ 1\bar{x} & \text{if } x < 0 \end{cases}$

→ 2's of x is $\begin{cases} 0x & \text{if } x \geq 0 \\ 1\bar{x}+1 & \text{if } x < 0 \end{cases}$

Scanned by TapScanner

Q: 1011 is signed 2's complement notation. its decimal value?

Given 2's comp

Since any of 1's or 2's comp is applied on same number, we get original value.

So apply 2's comp

let $x = 1011$

$\bar{x} = 1100$

$\bar{x}+1 = 1101$ → original value

i.e: -5

→ for unsigned number with size n ranges from 0 to $2^n - 1$

Shortcut to convert decimal to binary:

→ $(26)_{10}$

~~perfect~~ ~~posers~~

~~hei~~ ~~het~~ hieghest power of 2 below 26 is 16

	16	8	4	2	1
26 =	1	1	0	1	0

i) Signed magnitude notation:

+ve { 0000 to 0111 } +0 to +7

-ve { 1000 to 1111 } -0 to -7

Range

$-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$

→ zero has two notation of which '-0' is invalid

Scanned by TapScanner

(ii) Signed 1's Complement Notation:

+ve { 0000 to 0111 } +0 to +7

-ve { 1000 to 1111 } -7 to -0

Range

$-(2^{n-1}-1)$ to $(2^{n-1}-1)$

→ Zero has 2 notations of which '-0' is invalid

(iii) Signed 2's Complement Notation:

→ It is free from negative zero.

→ Provides more range for negative data than other notations.

→ It permits 8421 weighted code ~~for~~ while finding its decimal value.

In this process only MSB weight sign is -ve.

→ It permits sign bit expression for converting the smaller size signed data to larger size ~~size sign~~.

→ The additional padded bits are known as dummy bits/Guard bits.

+ve { 0000 to 0111 } to +7

-ve { 1000 to 1111 } -8 to -1

-8 to +7

Range

-2^{n-1} to $2^{n-1}-1$

2012

Q: The smallest -ve data that can be represented in signed 2's complement rotatation for 8 bit size is _____

a) 127 b) 128 ✓ c) 255 d) 256

Scanned by TapScanner

The following data are in signed 2's complement notation and their decimal values are:

→ $101 = (-)\overline{01} + 1 = -3$

<u>shortcut</u>

$\overset{-4\ 2\ 1}{101} = -4+1 = -3$

→ $1000 = (-)\overline{000} + 1 = -8$

<u>shortcut</u>

$\overset{-8\ 4\ 2\ 1}{1000} = -8$

→ $\overset{-16\ 8\ 4\ 2\ 1}{10001}$

$-16+1 = -15$

→ $\overset{-32\ 16\ 8\ 4\ 2\ 1}{101110}$

$\Rightarrow -32+8+4+2 = -18$

→ $\overset{-256\ 128\ 64\ 32\ 16\ 8\ 4\ 2\ 1}{100000001} = -256+1 = -255$

→ $\overset{-512\ \cdots\ 2\ 1}{1111\ 111111\ 11} = -1$

(If all digit are 1's in 2's complement representation the value is 1)

i.e, It permits 8421 weighted code with MSB weight is -ve.

Now consider +2

010

Here sign bit is 0

Now 010 ~ 0010 ~ 00010 . . .

Scanned by TapScanner

Now consider -2

$10 = (-2)_{10}$

$\cancel{(110)_2}$ 110 = -2

1110 ~ 11110 ~ . . . .

All the padded extra bits are known as dummy bits.

i.e; It permits sign bit extension.

Q: Find no of dummy bits in

110010

111010

↓

Dummy bits

Q: The 16 bit data $(FFFF)_{16}$ is in signed 2's Complement notation, its decimal value is _____

a) -1 b) -2 c) 10 d) none

1111 1111 1111 1111

Since all are ones value is -1

we have 15 dummy bits => original value is $\overset{-2^0}{1}$ => -1

MSD

Problems

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

539 = 1000011011

-539 -

1's comp = 0100100100

2's comp = 0100011100

1C

Problems:

2) -127

$127 = 1111111$

$\overline{127} = 0000000$

$\overline{127} + 1 = 0000001$

sign bit

1's comp = 10000000

$-127 \Rightarrow 10000001$ (2's comp)

$n \geq 1$

$m = 2$

~~$n = 6$~~

~~$m:n = 2:6 = 1:$~~

$m:n = 2:1$

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

$539 = 1000011011$

$\overline{539} = 0111100100$

$\overline{539} + 1 = 0111100101$

$-539 \Rightarrow 1101111100101$

D E 5

sign bit extension

2017 7)

x = no of distinct 16 bit integers in 2's compt

range: -2^{15} to $2^{15} - 1$

~~$\Rightarrow 2^{15} + 2^{15} + 1 \Rightarrow 2^{15}$~~

i.e; 12^{16}

Y = no of distinct 16 bit integers in sign magnitude

$Y = 2^{16} - 1$

$\Rightarrow X - Y = 1$

(This can be answered directly)

Scanned by TapScanner

Overflow (OVFL) in signed 2's complement Notation:

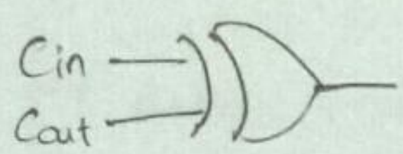
→ OVFL is known as signed CARRY

→ while adding 2 n-bit data the OVFL occurs when 'n' bits are not sufficient to store the result.

→ OVFL does not occur after adding one +ve data and one -ve data.

→ Occurance of OVFL changes (disturbs) the target sign bit in the result.

→ OVFL detection



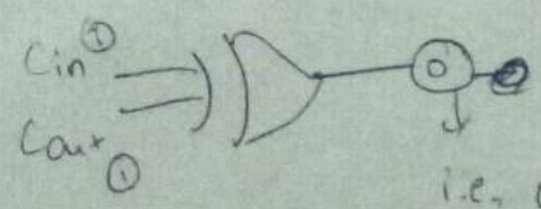
Cin = carry generated by adding data that enters in sign bit position

Cout = carry generated by adding both sign bits.

→ Presence of both Cin and Cout are required to detect OVFL.

Note:
* Cout participates in final result when OVFL occured only.

Eg: 1111 (-1)
1111 (-1)
Cout ← 1 1110 (-2) → overflow should'nt occur
this is answer (1110) ⇒ -8+4+2 = -2



i.e. overflow does not occur

Scanned by TapScanner

Eg 2:

```
  0111   7
  1110  -2
 (1)
 (1)0101
```

→ 1 +ve & 1 -ve

So overflow does not occur

↳ if no overflow this 4 bits gives answers

0 → overflow does not occur

Eg 3:

```
  0111   +7
  0110   +6
 (1)
 (0)1101
```

1 → overflow occurs

Eg 5:

```
  0110   +6
  1011   -5
 (1)
(1)0001
```

↳ result

∴ OVFL doesn't occur

Eg 4:

```
  1000   -8
  1001   -7
 (0)
(1)0001
```

⇒ overflow occurs

Conclusion: To determine if OVFL occurs we need to know abt both C_{in} & C_{out}

Q: After adding 2 signed 2's comp ~~numbers~~, notation data, OVFL

a) occurs if there is C_{in}

b) occurs if there is C_{out}

c) occurs if $C_{in} = C_{out}$

✓ d) doesn't occur after adding one +ve data & one -ve data

Eg 1:

```
  1011  (-5)
  1000  (-8)
 (0)
(1)0011  (-13)
```

⇒ overflow occurs

∴ C_{out} participates in final result

⇒ final result 10011 i.e: $-16+2+1 = -13$

Scanned by TapScanner

Eg:2 :

```
  0111   +7
  1111   -1
 10110   +6
```

=> OVFL does not occur

Cout do not participate in result

∴ result : 0110

Eg3 :

```
  0111   +7
  0110   +6
 01101   +13
```

overflow occurs

final result : 01101 i.e. 13

Q: Let 'x' is the sign bit of 1st data, Y is sign bit of 2nd data and 'z' is target sign bit in the result. The expression for the occurance of OVFL is

options

a) xyz

b) $xy + yz + xz$

c) $\bar{x}\bar{y}z + xy\bar{z}$

d) $x\bar{y}\bar{z} + xyz$

x	y	Cin	~~OVFL~~ OVFL
0	0	0	0
0	0	1	1 ✓
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1 ✓
1	1	1	0

$S.OP = xy\bar{C}in + \bar{x}\bar{y}Cin$

$x=0, y=0, Cin=1 \Rightarrow z=1 \Rightarrow \bar{x}\bar{y}z$

~~x=0, y=~~ $x=1, y=1, Cin=0 \Rightarrow z=0 \Rightarrow xy\bar{z}$

∴ expr is $\bar{x}\bar{y}z + xy\bar{z}$

Method 2 :

xyz	OVFL	
000	0	→ x=0, y=0, z=0 (+ve & +ve gave +ve which is fine)
001	1	→ x=0, y=0, z=1 (adding two +ve gave -ve ∴ OVFL occured)
010	0	1 +ve & 1 -ve ∴ no OVFL
011	0	
100	0	
101	0	
110	1	→ x=0 y=0 z=1 (adding 2 -ve gave +ve ∴ OVFL)
111	0	→ x=1 y=1 z=1 (-ve & -ve gave -ve which is fine)

Scanned by TapScanner

Now S.O.P is :

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

Conclusion

S_1 x x x x . . .

S_2 x x x x . . .

S x x x x . . .

S_1 – sign bit of 1st data

S_2 – sign bit of 2nd data

S – sign bit of sum of 1st & 2nd data

~~if no~~ if S_1 & S_2 are same sign and

- if overflow not occurs S is same as S_1 & S_2
- if overflow occurs S is opposite of S_1 & S_2

4. Combinational Circuits

→ These are faster.

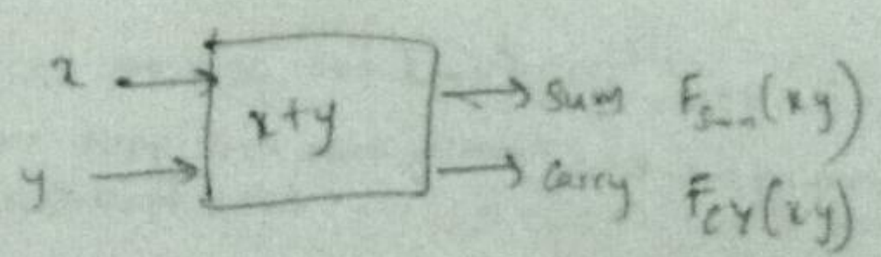
→ Present o/p depends on i/p only.

→ No memory & no clock.

→ Different combinational circuits are Adders, Subtractors, MUX, DeMux, encoder, decoder etc.

→ MUX is known as universal combinational circuit.

Half Adder

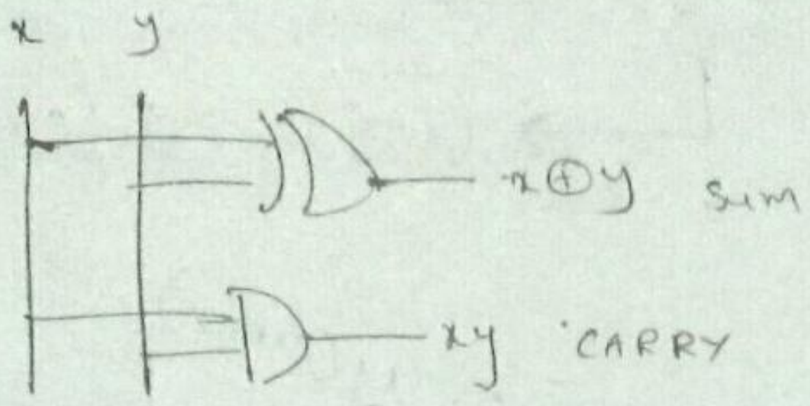


Scanned by TapScanner

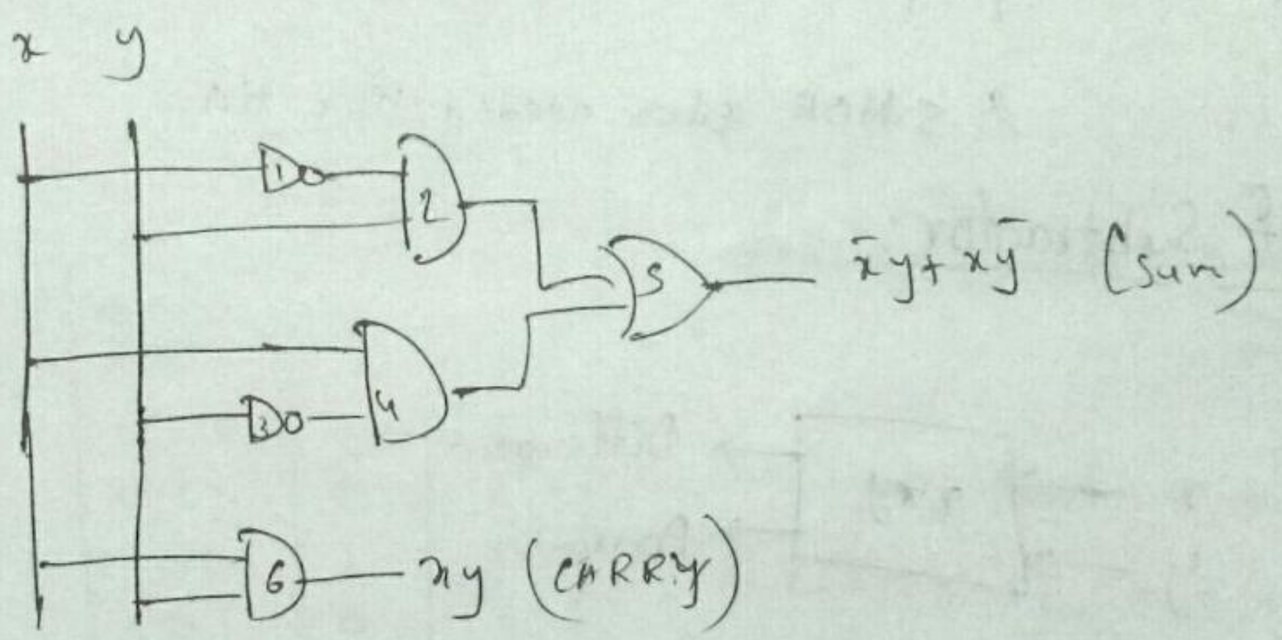
x	y	S	CY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$F_{sum}(xy) = \bar{x}y + x\bar{y} = x \oplus y$

$F_{cy}(xy) = xy$

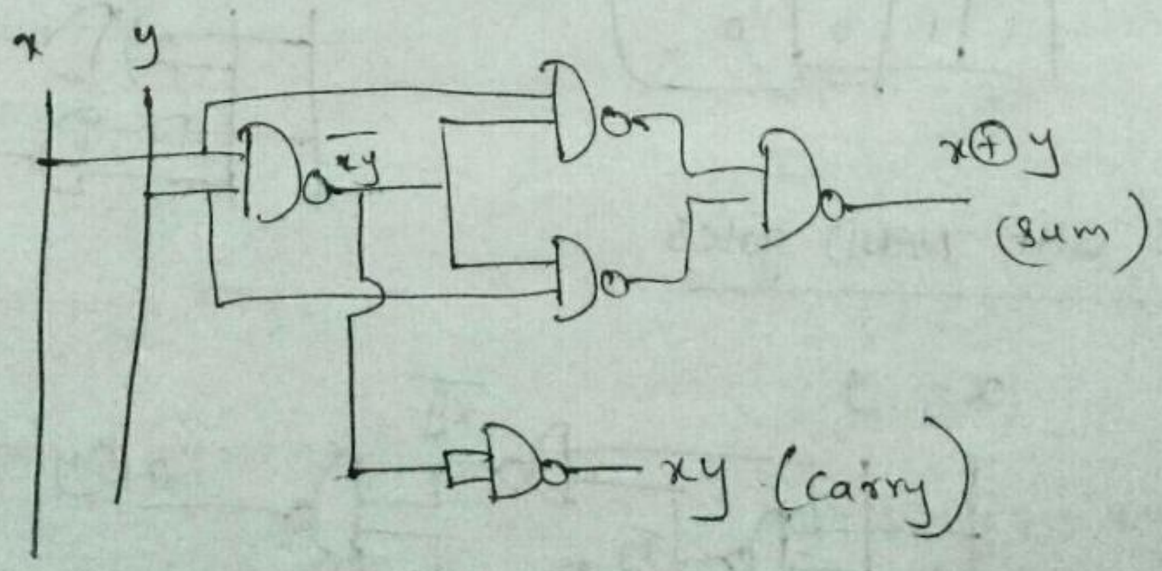


* <u>HA with basic gates</u>



<u>Here we need 6 basic gates</u>

<u>HA with NAND gates</u>



∴ <u>Here we need 5 NAND gates</u>

Scanned by TapScanner

HA with NOR gates

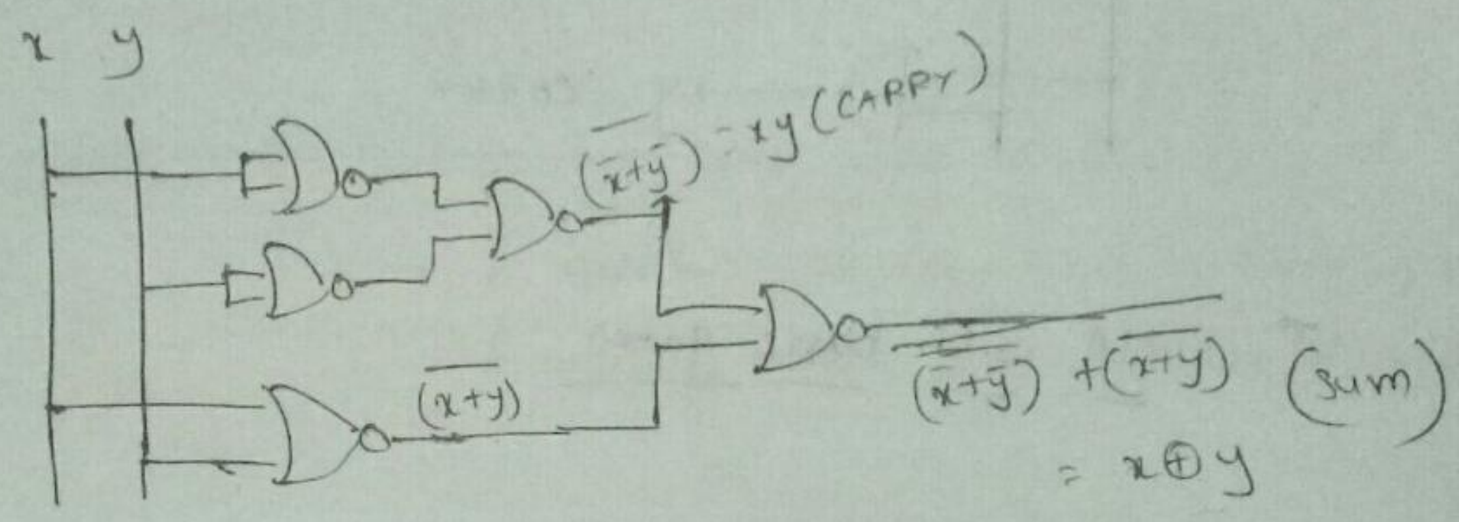
$$\text{Sum} = x \oplus y = \bar{x}y + x\bar{y}$$

$$= \overline{\overline{xy + \bar{x}\bar{y}}}$$

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

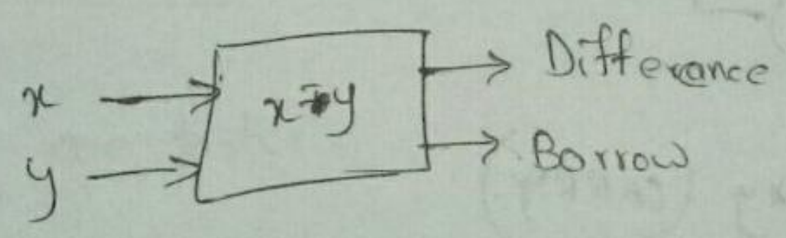
$$= \overline{(\overline{\bar{x}+\bar{y}})} + \overline{(\overline{x+y})} \longrightarrow \text{Sum}$$

$$\longrightarrow \overline{(\bar{x}+\bar{y})} = xy \ (\text{C.y})$$



∴ 5 NOR gates needed for HA

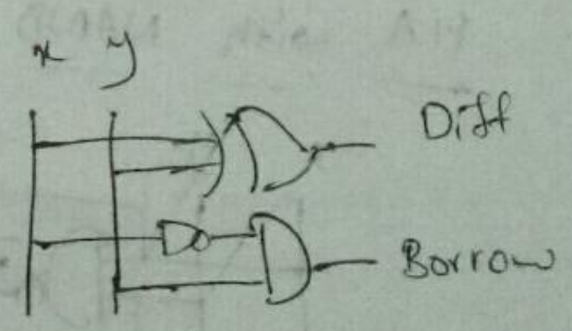
Half Subtractor:



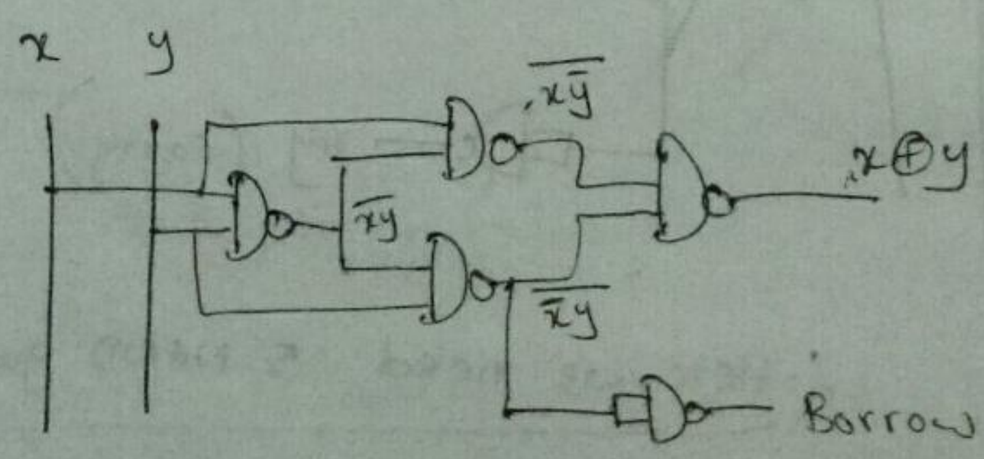
x	y	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$\text{Diff} = x \oplus y$

$\text{Borrow} = \bar{x}y$



HS with NAND gates



∴ 5 NAND gates for HS

$\overline{\bar{x}y} \cdot x$

$xy + \bar{x}$

$\bar{x} + y$

$\overline{x\bar{y}}$

$\overline{\bar{x}y} \cdot y$

$xy + \bar{y}$

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

$x\bar{y} + y$

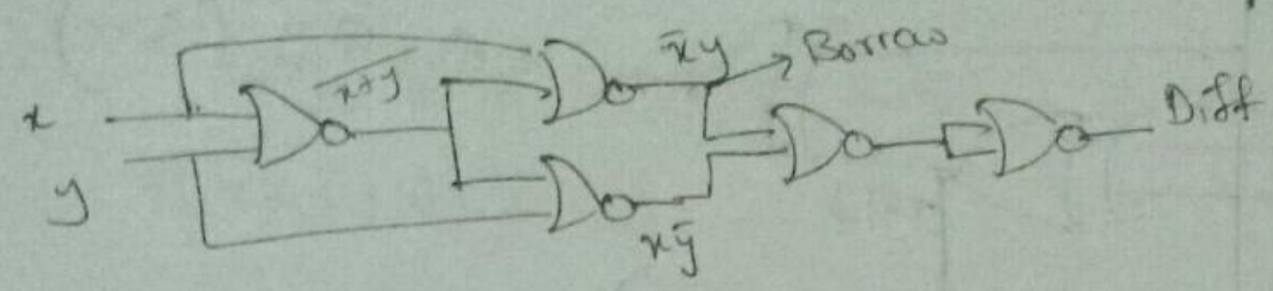
Scanned by TapScanner

HS with NOR gates

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

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

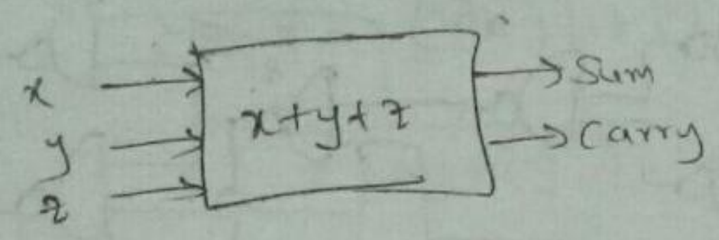
$\bar{x}y$



Note:

Min no of NAND/NOR gates required for Half adder or Half Subtractor is 5.

Full Adder:



xyz	Sum	Carry
000	0	0
001	1	0
010	1	0
011	0	1
100	1	0
101	0	1
110	0	1
111	1	1

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

$\cancel{Sum(xyz) = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z}}$

$\cancel{+ xyz}$

$Sum(xyz) = \sum m(1,2,4,7)$

$C(xyz) = \sum m(3,5,6,7)$

Sum

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

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

$\bar{x}(\bar{y}z + y\bar{z}) + x(\bar{y}\bar{z} + yz)$

$\bar{x}(y \oplus z) + x(y \odot z)$

$\bar{x}(y \oplus z) + x\overline{(y \oplus z)}$

$x \oplus y \oplus z$

CARRY

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

$yz + xz + xy$

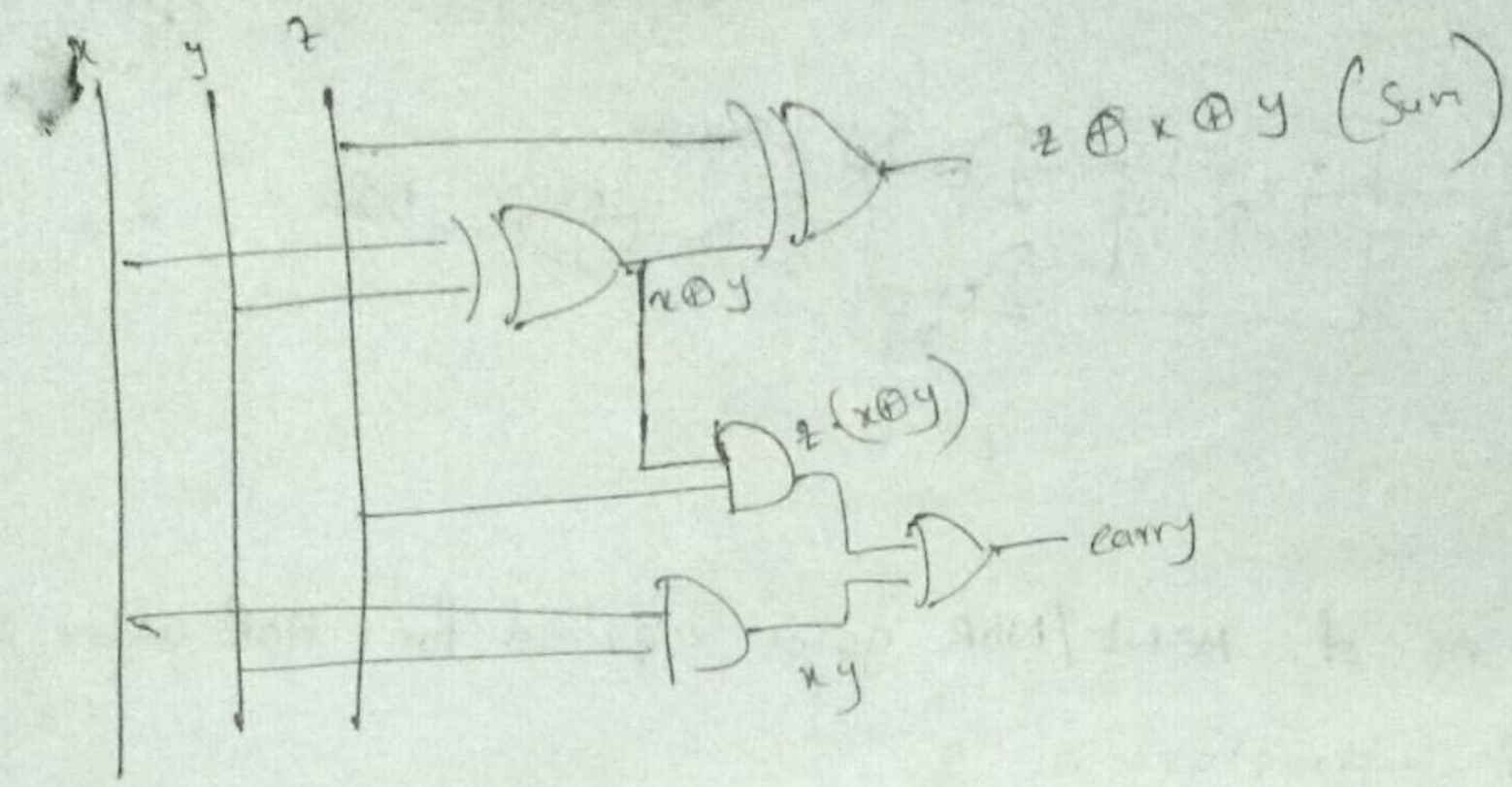
(or)

$\bar{x}yz + x\bar{y}z + xy$ (This is written directly from minterms)

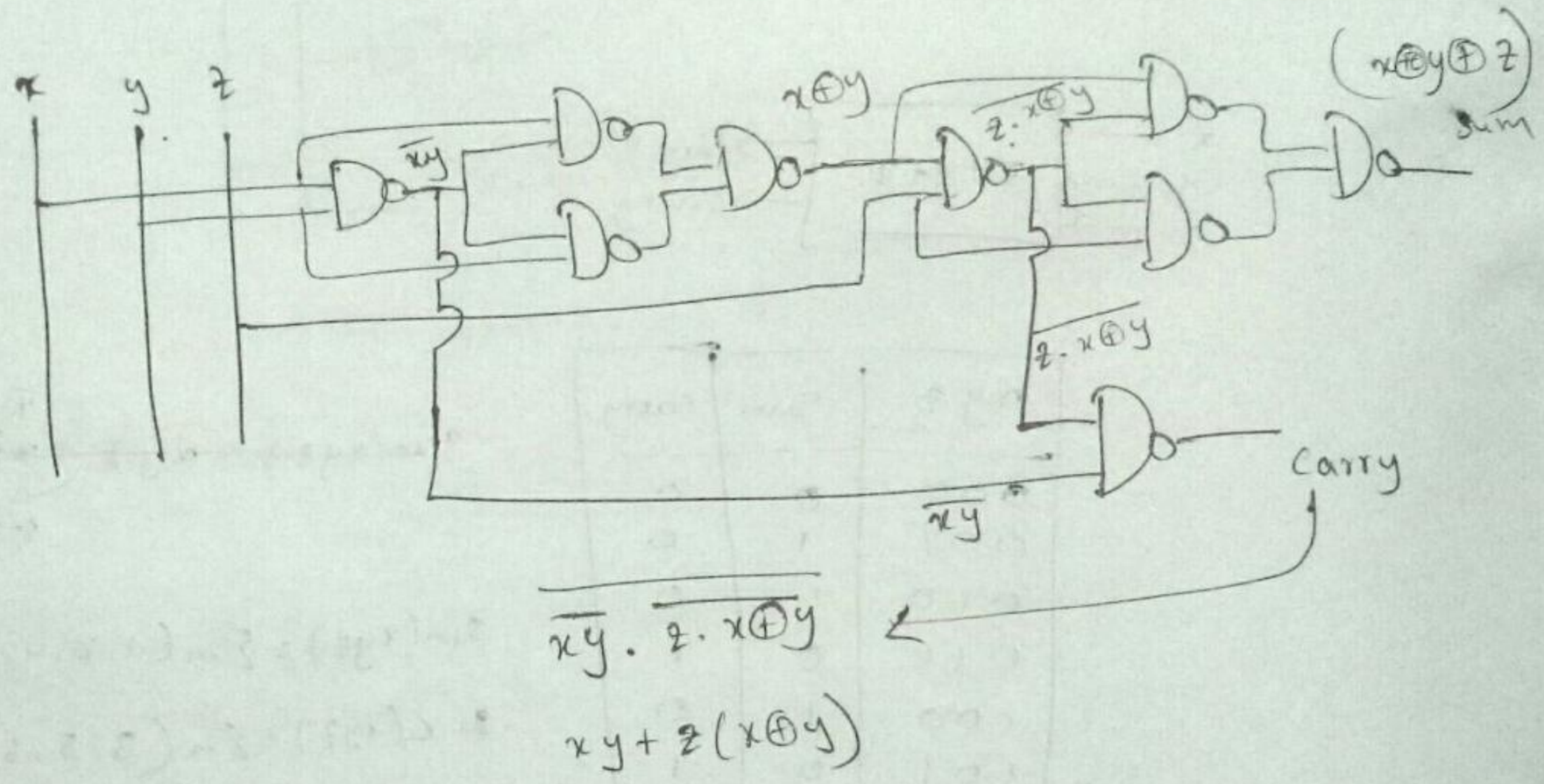
$z(x \oplus y) + x.y$

Here $z(x \oplus y) + xy$ is better to use.

Scanned by TapScanner



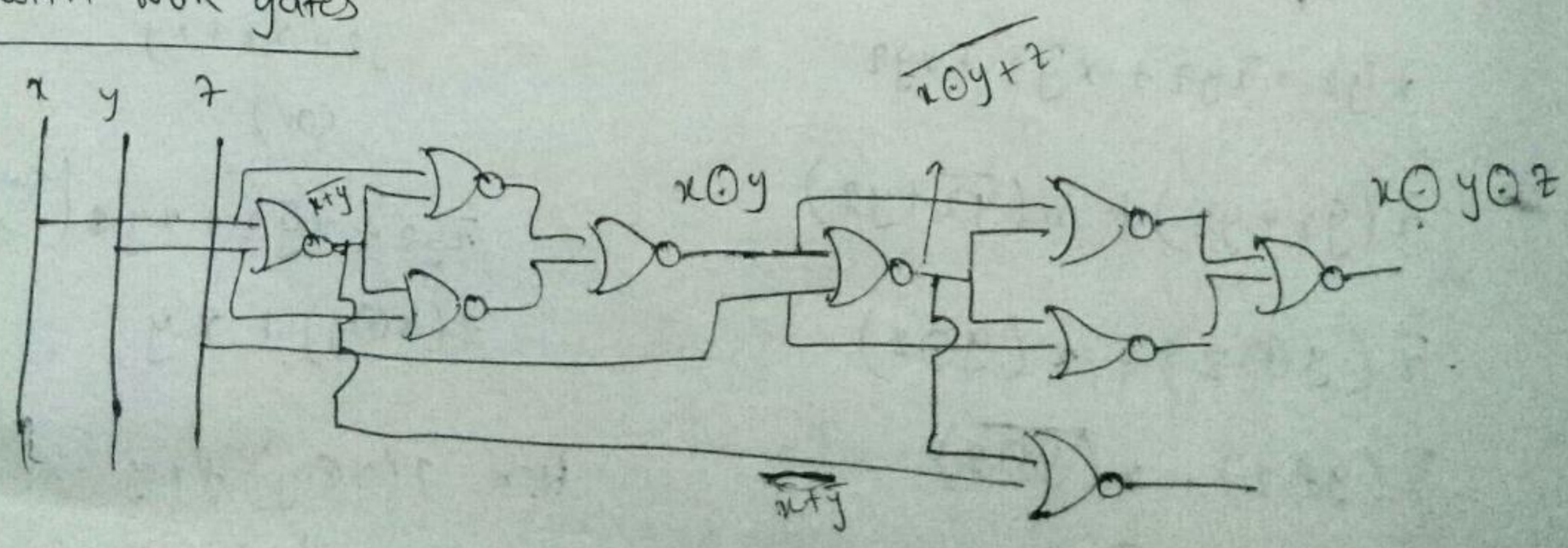
Full Adder with NAND gates



Note :

Above circuit produces same functional o/p even if all NAND gates are replaced with NOR gates.

FA with NOR gates



Scanned by TapScanner

Sum = $x \odot y \odot z$

$= (x \odot y) \oplus \bar{z}$ (Single bubbled ExNOR is ExOR)

$= (\overline{x \oplus y}) \oplus \bar{z}$

$= x \oplus y \oplus z$ (double bubbled ExOR is normal ExOR)

At 5th gate

o/p : $\overline{x \odot y + z}$

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

At 9th gate

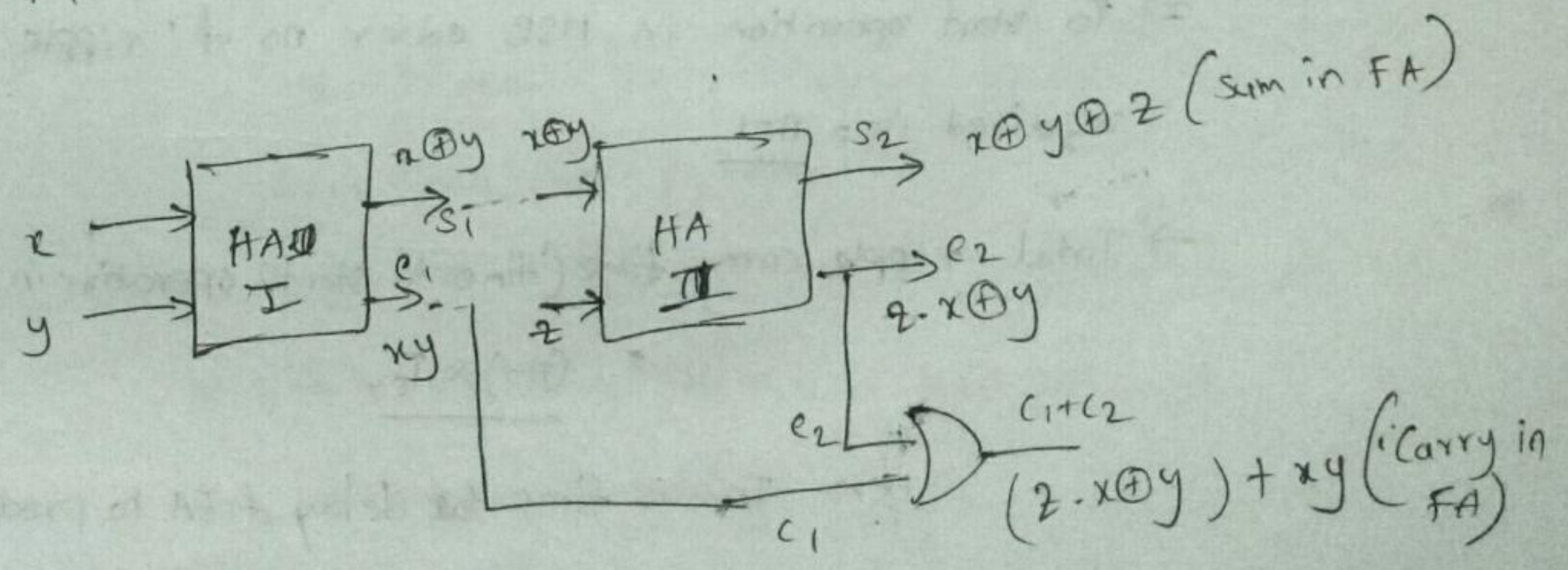
$$\overline{\overline{(\overline{z + xy + \bar{x}\bar{y}}) + (\overline{x+y})}}$$

$$\overline{(\overline{z + xy + \bar{x}\bar{y}}) \cdot (x+y)}$$

$= yz + xz + xy$ (expression obtained from carry k-map)

Full Adder with half adder:

* Here we use 2 half adder & 1 OR gate to obtain 1 full adder



n bit adder:

→ It is used to add 2 n bit data.

→ FA is known as one full bit adder.

→ HA is known as one half bit adder.

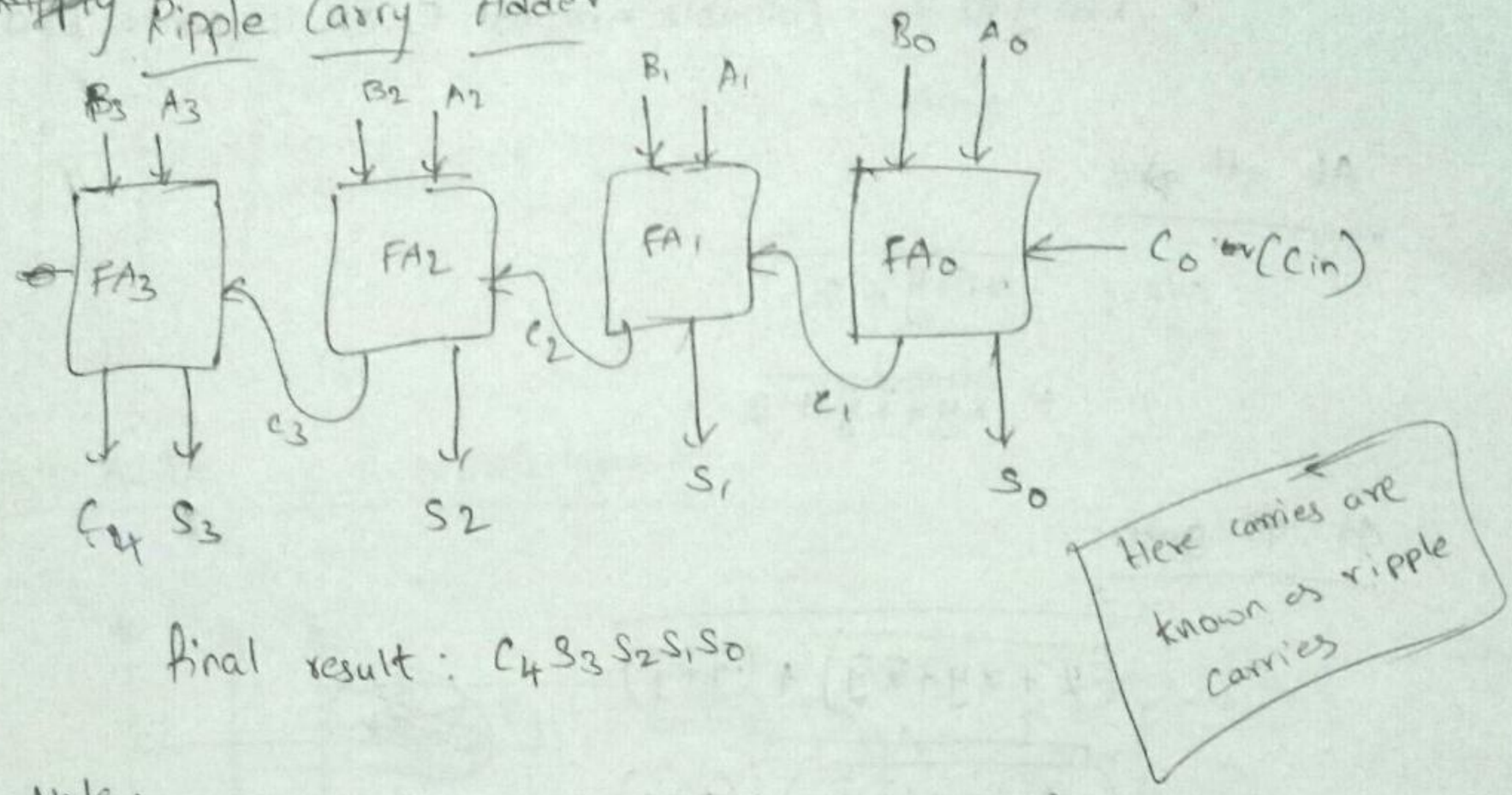
Scanned by TapScanner

(2003)

Q To add 2 n bit data (without Cin) min no of HA required is <u>1</u> and min no of FA req is <u>n-1</u>.

→ Here Cin refers to initial carry.

~~Ripply~~ <u>Ripple Carry Adder</u>



final result : $C_4 S_3 S_2 S_1 S_0$

<u>Note</u>:

In above diagram all A_i, B_i & C_0 are available directly (i.e. no delay)

for n bit Ripple carry adder:

→ After completion of operation in MSB adder (FA_{n-1}) the final result will be stabilized

→ To start operation in MSB adder no of ripple carries required is = <u>n-1</u>

→ Total ripple carry time (time to start operation in MSB)

$$= \underline{(n-1) \times T_{CY}}$$

Here T_{CY} is time ~~tak~~ delay of FA to produce carry

→ $T_{worst} = ((n-1) \times T_{CY}) + \max(T_{sum}, T_{CY})$

Here T_{worst} is max time to produce stabilized output

T_{sum} is time taken by FA to produce sum.

Scanned by TapScanner

Problems

5) $(16-1) \times 12 + 15 = 195$ ns → worst case delay

$(16-1)$ → total carries to produce start operation in MSBFA

12 → Time to start operation in MSB

15 → $Max(T_{sum}, T_{cy})$

2016

4) $A = (1)_{10}$

$i_{cy} = (00000001)_2$ (latency means time delay)

* longest latency is required when CARRY is generated in MSB adder (i.e when longest data i/p is applied)

A = 00000001

B = 11111111
1 1 1 1 1 1 1 1

$\therefore B = (-1)_{10}$

2) HA with XOR & AND

FA with 2HA & 1OR

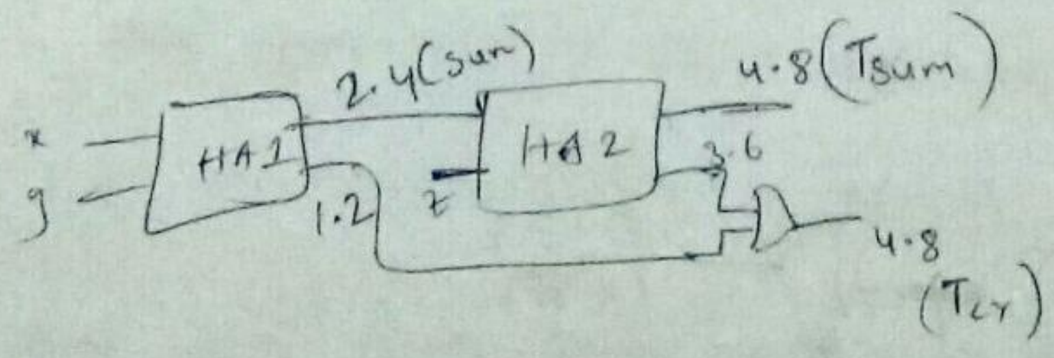
$t_{XOR} = 2t_{AND} = 2t_{OR}$

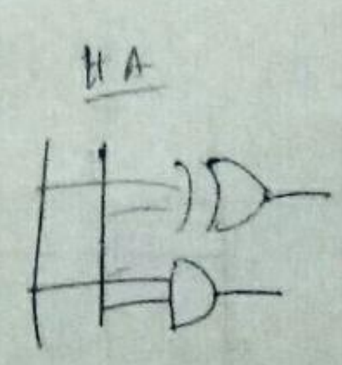
$t_{AND} = t_{OR} = 1.2\,\mu s$

$t_{XOR} = 2.4\,\mu s$

4-bit ripple carry adder with 4FA

Top Total propagation delay = ? μs





~~Since we are using 4 FA~~

Here both carry & sum are generated exactly after $4.8\,\mu s$

we are using 4FA ⇒ ~~4 × 4.8 = 19.2 μs~~ $(4-1) \times 4.8 + 4.8 = 19.2\,\mu s$

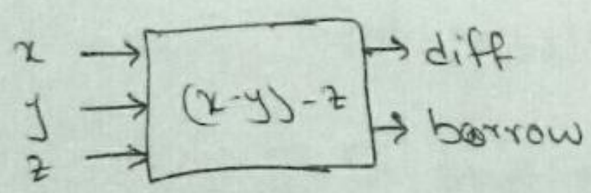
Scanned by TapScanner

Method 2 :

FA3 FA2 FA1 FA0

07/09/19

Full Subtractor :



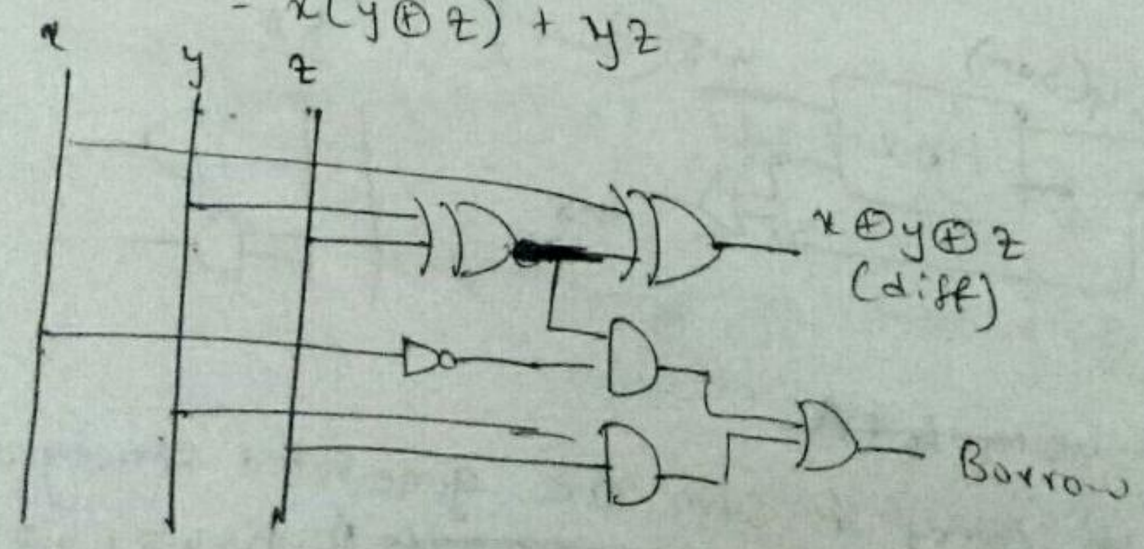
x y z	D	B
0 0 0	0	0
0 0 1	1	1
0 1 0	1	1
0 1 1	0	1
1 0 0	1	0
1 0 1	0	0
1 1 0	0	0
1 1 1	1	1

$F_{Diff}(xyz) = \Sigma_m(1,2,4,7)$

$= x \oplus y \oplus z$

$F_{Borrow}(xyz) = \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + xyz$

$= \bar{x}(y \oplus z) + yz$

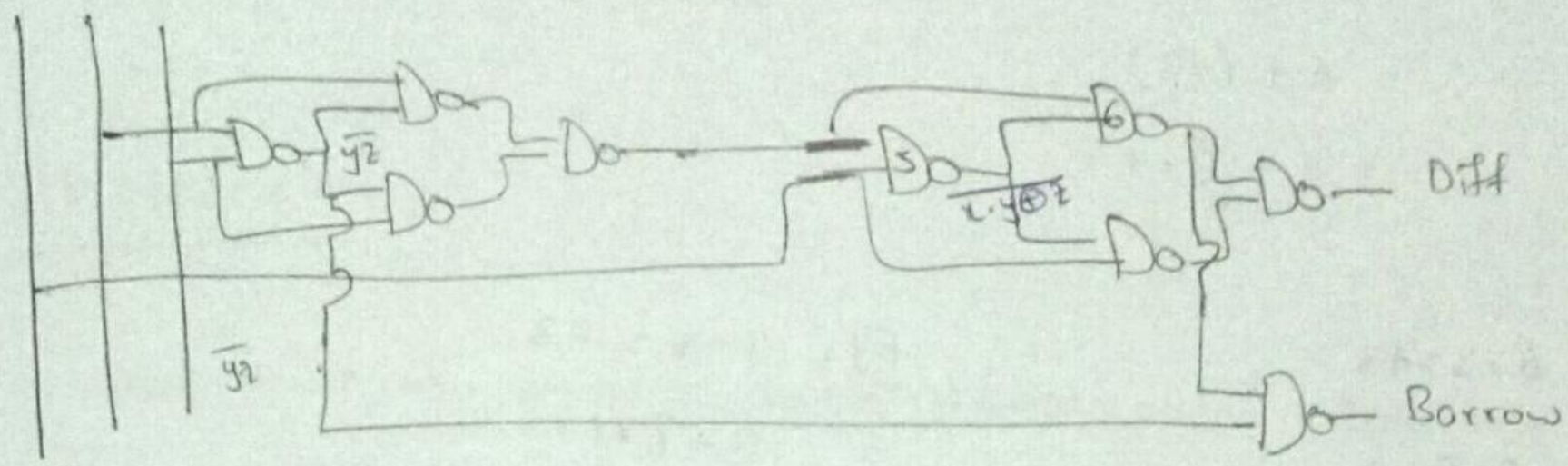


Scanned by TapScanner

Full Subtracter with NAND gates:

Borrow $= \bar{x}(y \oplus z) + yz$

consider it as $\overline{\overline{\bar{x}(y \oplus z) + yz}} = \overline{\overline{\bar{x}(y \oplus z)} \cdot \overline{yz}}$



At 6th gate:

$$\overline{(y \oplus z) \cdot \overline{x \cdot y \oplus z}}$$

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

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

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

Note:

The above diagram produces same functioning even if all NAND gates are replaced with NOR gates

Binary Substraction:

→ It is performed with 2's complement Addition

i.e, $A - B = A + \bar{B} + 1$

→ During this operation if there is a carry then result sign is +ve and +ve result is available in normal form.

→ If there is no carry then result sign is -ve and -ve result is available in 2's complement form.

Scanned by TapScanner

Q: Min no of 2 i/p NAND / NOR gates req to implement full Adder/ Full Subtractor is 9

Here $A - B$

$= A + (-B)$

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

Eg1 $6 - 3 = +3$

$6 + \bar{3} + 1$

$6 : 110$

$\bar{3} : 100$
$\quad\quad 1$

1011

3

(No Carry generated

$\therefore$ 011 is answer)

Eg: $3 - 6 = -3$

$3 + \bar{6} + 1$

$3 : 011$

$\bar{6} : 001$
$\quad\quad 1$

$1001 \rightarrow$ This 2's comp form ($\because$ no carry)

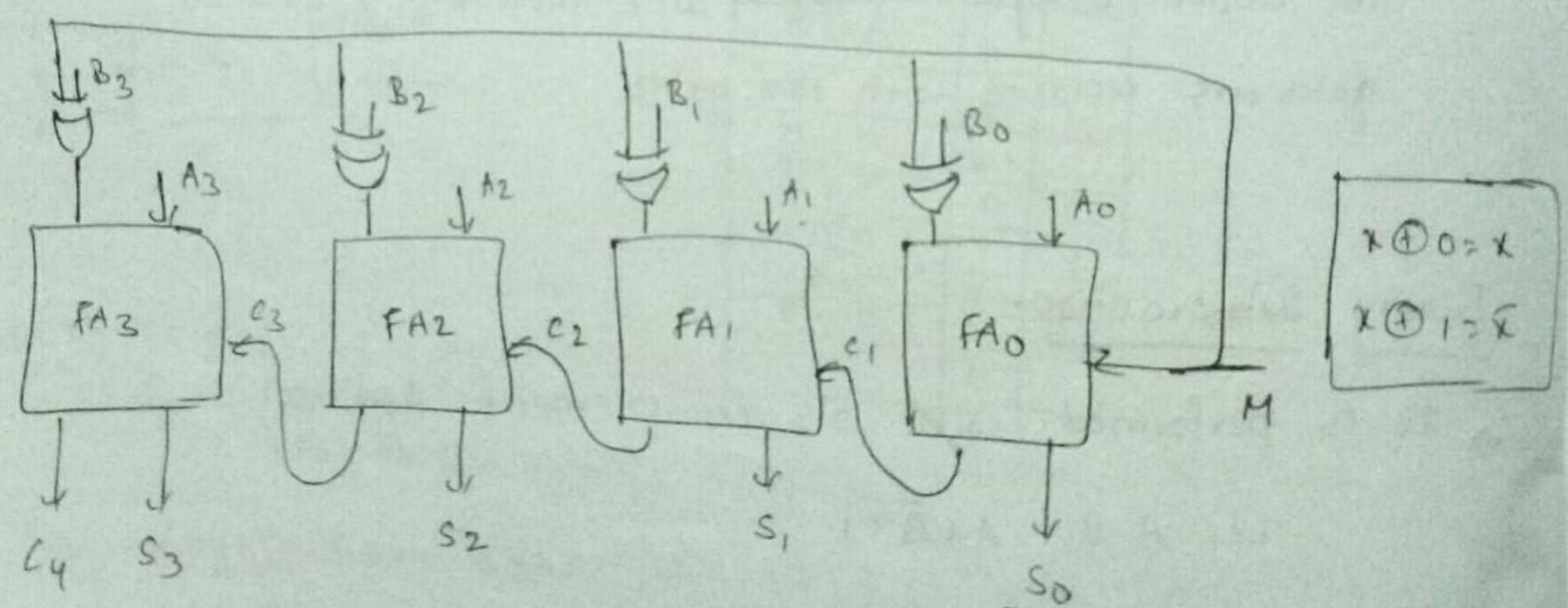
$\therefore 101 = -4 + 0 + 1 = -3$

(or) $101 = 110$ (2's comp)

$= 111$ (original)

$\Rightarrow$ i.e. -3

4 bit Adder/ Subtractor :



$\rightarrow$ Here M is called Mode control

$\rightarrow$ If M=0 the above circuit acts as Full Adder

$A_3\ A_2\ A_1\ A_0$
$B_3\ B_2\ B_1\ B_0$ } adder

→ If M=1 the circuit acts as full subtractor

$$\left.\begin{array}{l} A: A_3A_2A_1A_0 \\ (1's)\ \bar{B}: \bar{B}_3\bar{B}_2\bar{B}_1\bar{B}_0 \\ \qquad\qquad\qquad\quad 1 \end{array}\right\} \text{Subtractor}$$

(2's comp)

Multiplexer

→ It is also known as universal combinational circuit, Data selector, Many to one selector and parallel to serial converter.

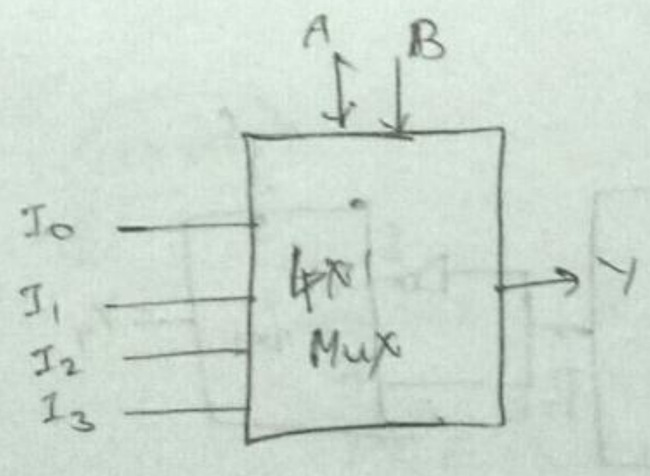
Standard size is $2^n \times 1$, 2^n = no of data inputs with n selection lines

→ Selection lines are used to select only one input to ~~out~~ output line.

→ Different sizes are $2\times1, 4\times1, 8\times1, \ldots 2^n\times1$

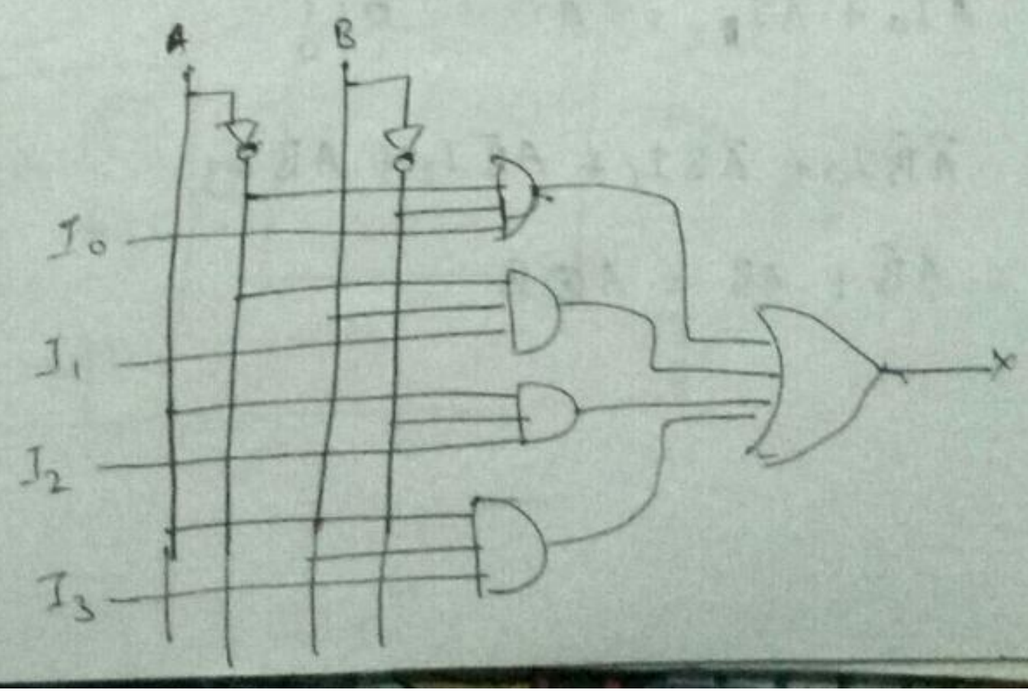
Q: No of selection lines required for 1024×1 MUX is 10

Q: No of selection lines require for $N\times1$ MUX is $\log_2 N$



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

$Y = \cancel{\bar{A}\bar{B}}\ \bar{A}\bar{B}I_0 + \bar{A}BI_1 + A\bar{B}I_2 + ABI_3$



Scanned by TapScanner

For designing 4x1 Mux

No of AND gates = 4

No of OR gates = 1

No of NOT = 2

~~No of 0~~

Size of OR gate = 4

Size of each AND gate = 2 + 1
S I

For designing $2^n \times 1$ Mux

No of AND gates = 2^n

No of OR gates = 1

No of NOT gates = n

Size of OR gates = 2^n

~~No of~~

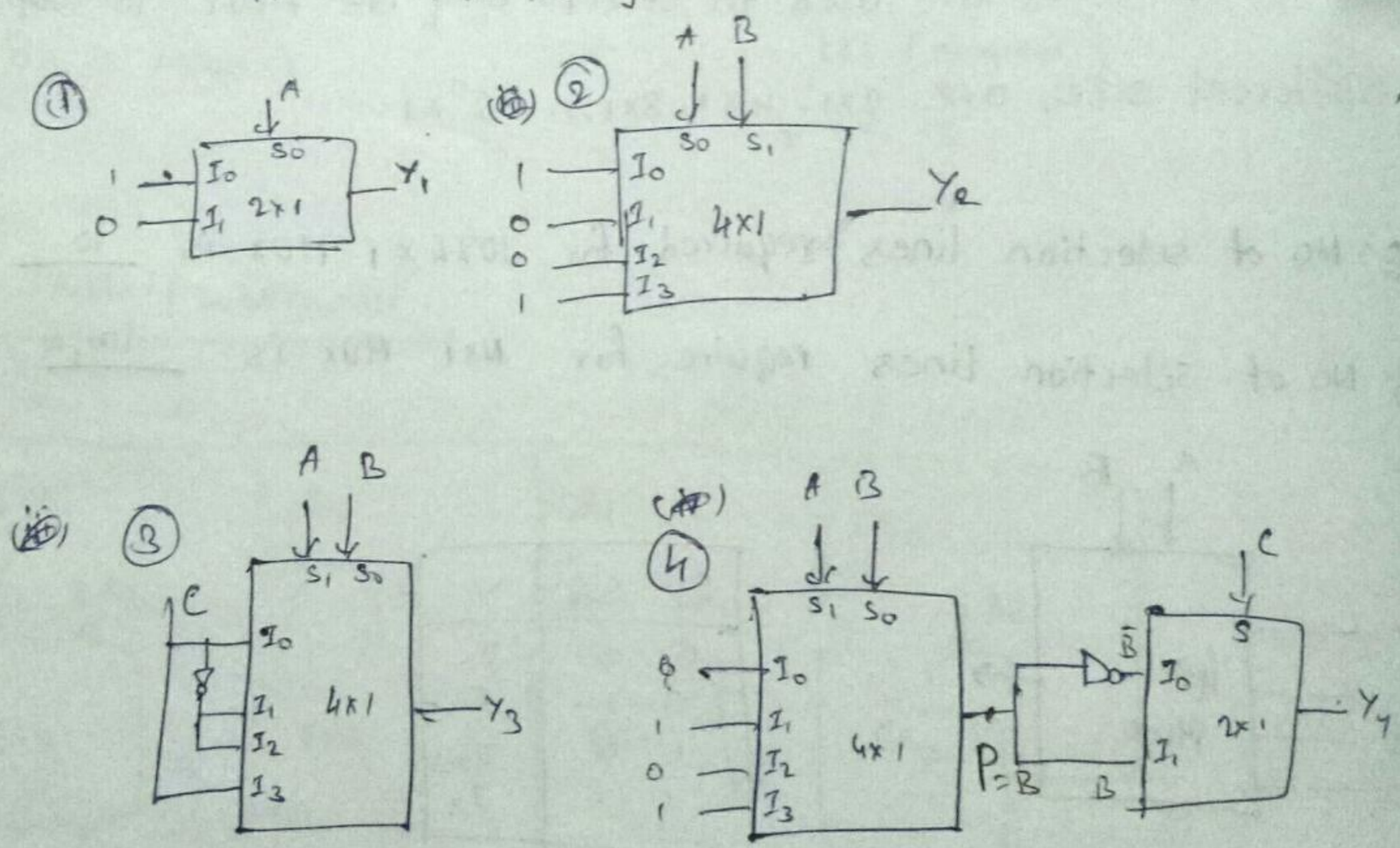
size of each AND gate = n+1

Previous

Q: No of inputs req for each AND gate to design 32x1 Mux is 6

5+1=6

Eg: What is the o/p at Y in the given below



① $Y_1 = \bar{A} I_0 + A I_1 = \bar{A}$

A	Y
0	1
1	0

② $Y_2 = \bar{A}\bar{B} I_0 + \bar{A} B I_1 + A \bar{B} I_2 + A B I_3$

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

Scanned by TapScanner

③

AB	Y
00	C
01	$\bar{C}$
10	$\bar{C}$
11	C

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

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

④

AB	P
00	0
01	1
10	0
11	1

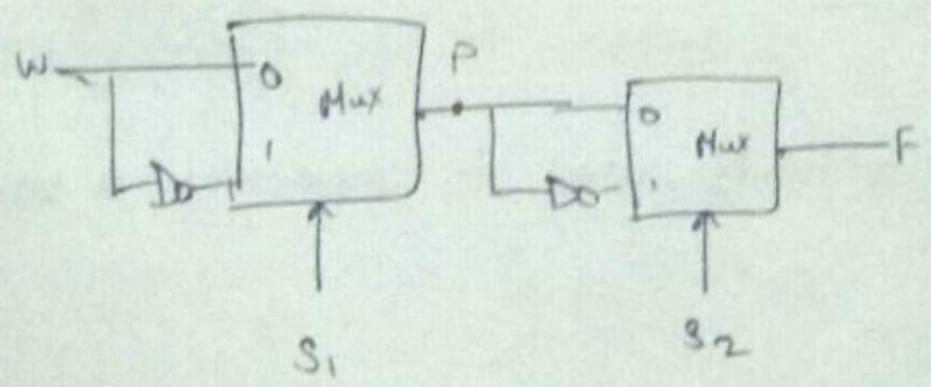
$P = \bar{A}B + AB = B$

$\Rightarrow Y_4 = \bar{C}\bar{B} + CB = B \odot C$

Problems:

(2014)

①



$P = \bar{S}_1 W + S_1 \bar{W} = S_1 \oplus W$

$F = \bar{S}_2 P + S_2 \bar{P} = S_2 \oplus P = S_2 \oplus S_1 \oplus W$

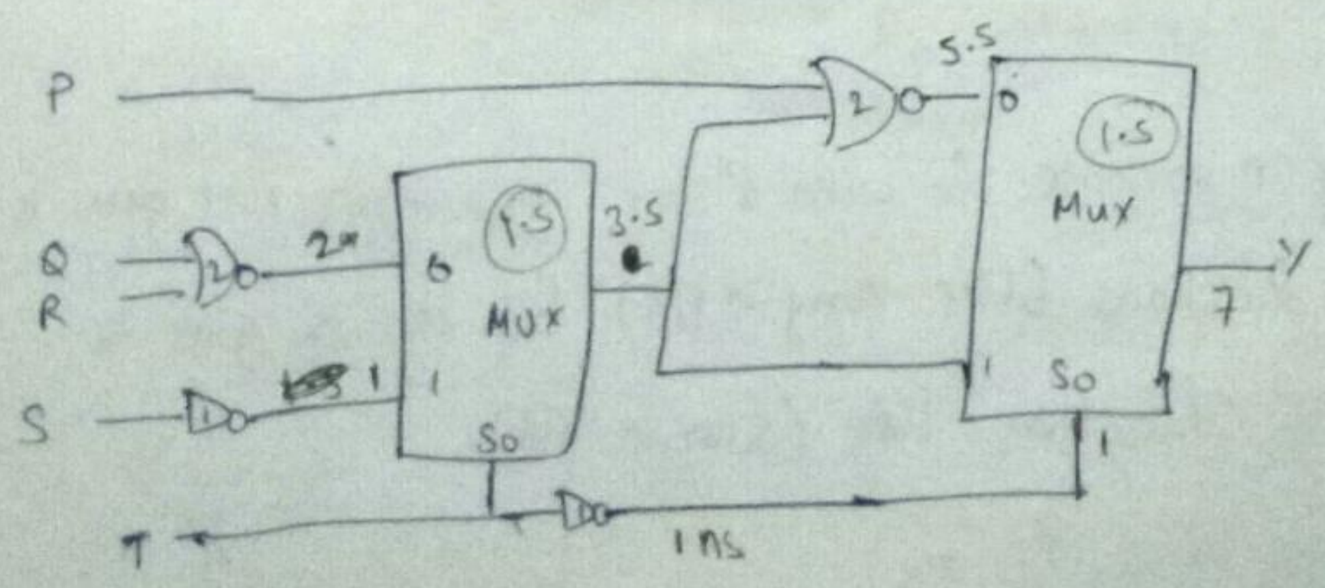
$F = W \oplus S_1 \oplus S_2$

(EEE 2016)

③

$t_{NOR} = 2 \quad t_{mux} = 1.5 \quad t_{not} = 1ns$

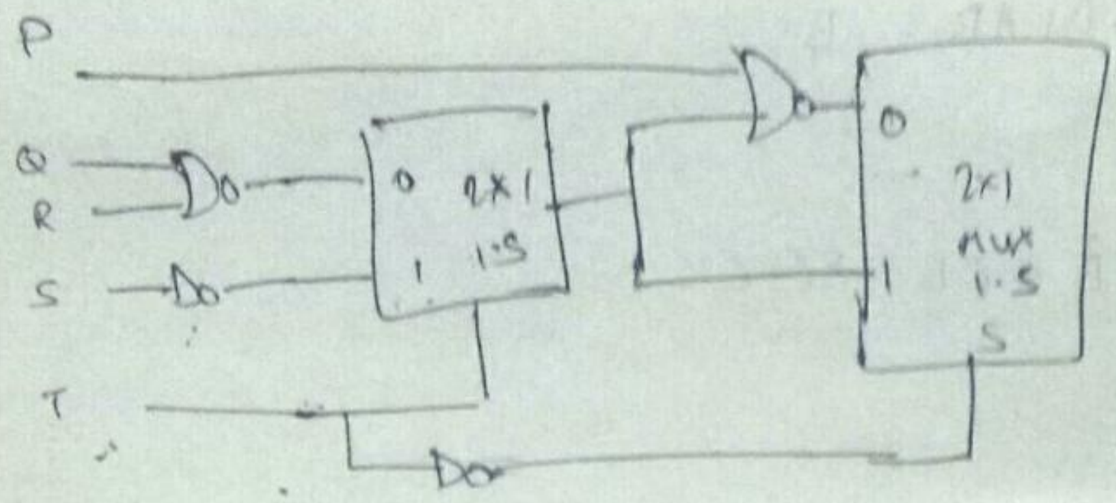
P, Q, R, S, T at same instant applied



Scanned by TapScanner

To	Data Travel path
0	$T_{NOR} + T_{MUX} + T_{MUX} = 5ns$
1	$T_{NOT} + T_{MUX} + T_{NOR} + T_{MUX} = 6ns$

Consider the circuit as below



Mux as function selector

→ Generally there are 2 methods used to implement n variable function

(i) with $2^n \times 1$ size

(ii) $2^{n-1} \times 1$ size

→ while implementing with $2^n \times 1$ size MUX all functional inputs are connected to selection lines and data inputs of Mux can be 0's & 1's

→ while implementing the f_n with $2^{n-1} \times 1$ size mux (n-1) functional inputs are connected to selection lines and data inputs for M can be 0's, 1's, free input, complement of free input.

Note:-

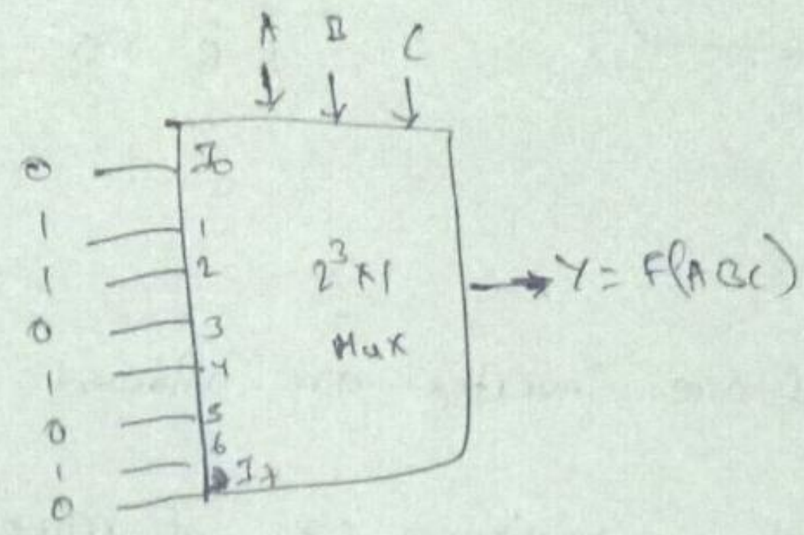
while implementing n variable fn with $2^{n-1} \times 1$ size mux, NOT gate is not needed for all functions (like carry in F/A) but NOT is gate is needed for only some functions like (Sum in FA)

Scanned by TapScanner

Eg: Implement $F(ABC) = \Sigma_m(1,2,4,6)$ with Mux

Using $2^3 \times 1$ Mux $(2^n \times 1)$:

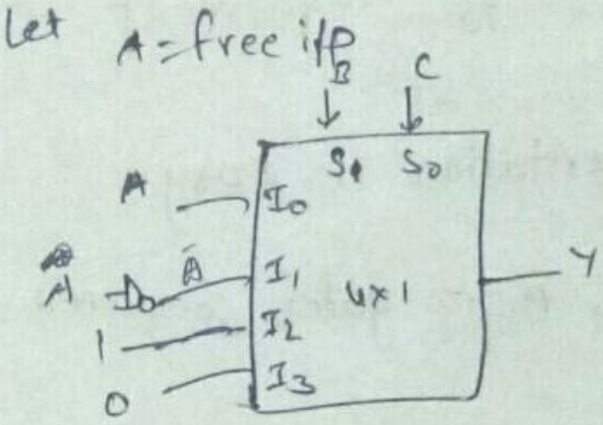
A B C	F
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	0



Using $2^{3-1} \times 1$ Mux $(2^{n-1} \times 1)$:

ABC	F
000	0
001	1
010	1
011	0
100	1
101	0
110	1
111	0

Let A = free i/p

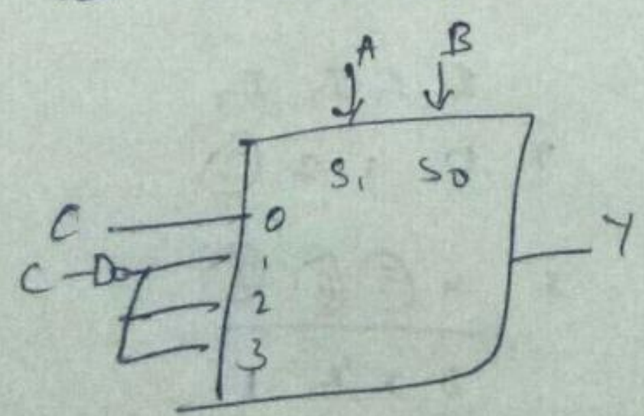


	I_0	I_1	I_2	I_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	$A\bar{A}$	1	0	

Verification

BC	Y
00	A → $A\bar{B}\bar{C}$ = 100 (4)
01	$\bar{A}$ → $\bar{A}\bar{B}C$ = 0001 (1)
10	1 → $B\bar{C}(AB\bar{C} + \bar{A}B\bar{C})$ (6,2)
11	0

let C = free input

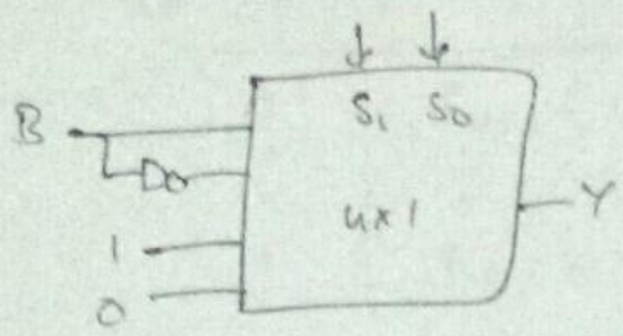


	I_0	I_1	I_2	I_3
$\bar{C}$	0	2	4	6
C	1	3	5	7
	C	$\bar{C}$	$\bar{C}$	$\bar{C}$

A B	Y
00	C : $\bar{A}\bar{B}C$ (1)
01	$\bar{C}$ } (2,4,6)
10	$\bar{C}$
11	$\bar{C}$

Scanned by TapScanner

Let B = free i/p



	I_0	I_1	I_2	I_3
$\bar{B}$:	0	1	4	5
B :	2	3	6	7
	$\bar{B}$	1	0	B

Q: One mux & one invertex are allowed to implement n variable function. what is minimum size of mux needed

(i) $2^n \times 1$ (ii) $2^{n-1} \times 1$ (iii) $2^{n+1} \times 1$ (iv) None

→ For $2^n \times 1$ Mux to implement n variable function:

→ Implementation is easy

→ Requires more gates compared to $2^{n-1} \times 1$ Mux implementation

→ For $2^{n-1} \times 1$ Mux to implement n variable function

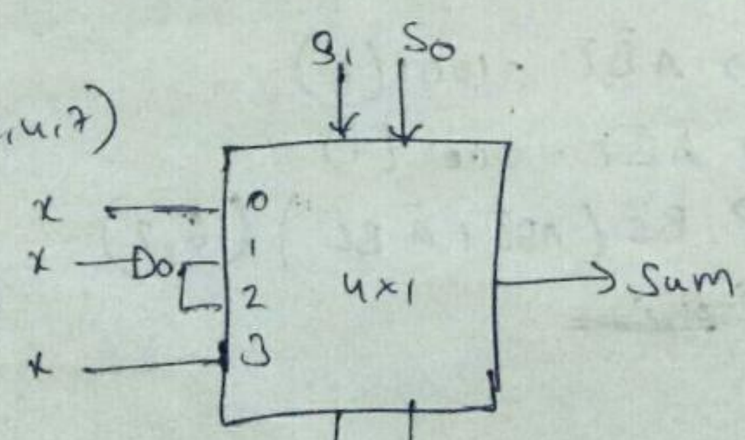
→ Implementation is complex.

→ Requires less gates.

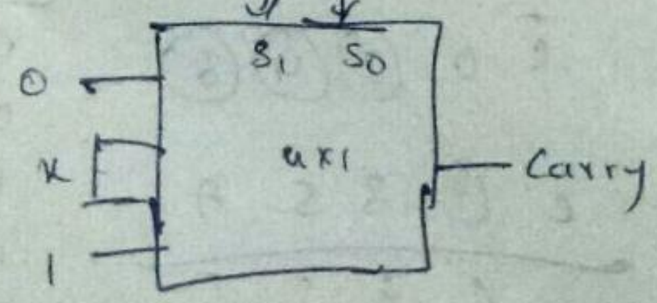
Eg: Implement FA with 4x1 matrices

$Sum(xyz) = \Sigma m(1,2,4,7)$

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



	I_0	I_1	I_2	I_3
$\bar{x}$	0	1	2	3
x	4	5	6	7
	x	$\bar{x}$	$\bar{x}$	x



	I_0	I_1	I_2	I_3
$\bar{x}$	0	1	2	3
x	4	5	6	7
	0	x	x	1

Here x is S_2 (i.e., 3rd bit in addition)

Scanned by TapScanner

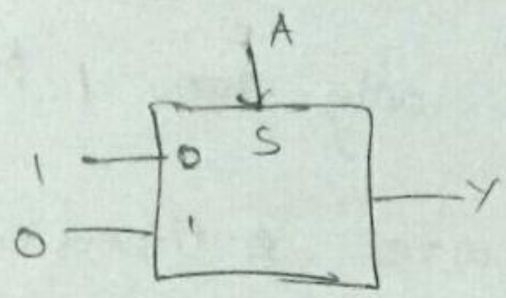
Logic gates with 2x1 Muxes.

→ It is like function implementation with 2^{n-1} muxes.

(2003)
Q: Min no of 2x1 muxes needed to implement 2 i/p XOR gate is 2 and 2 i/p AND gate is 1

NOT gate:

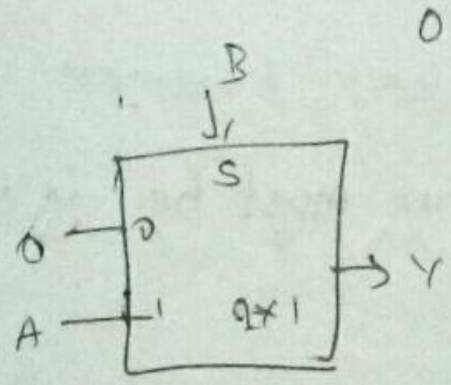
A	Y
0	1
1	0



AND gate

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

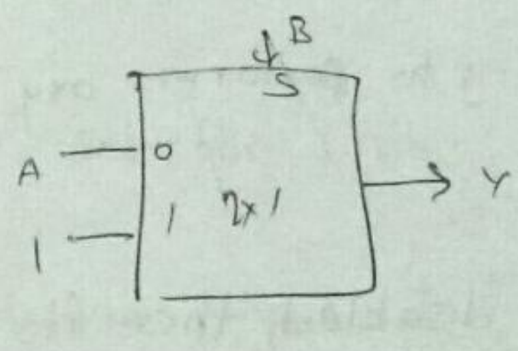
	I_0	I_1
$\bar{A}$	0	1
A	2	3
	0	A



OR gate

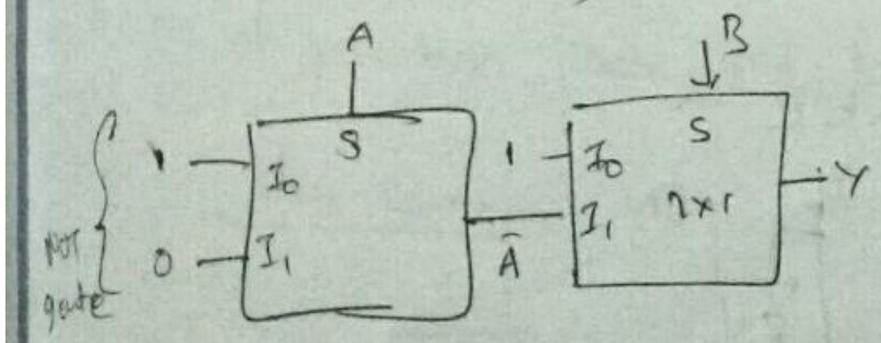
$f(AB) = \Sigma m(1,2,3)$

$\bar{A}$	0	1
A	2	3
	A	1



NAND gate

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



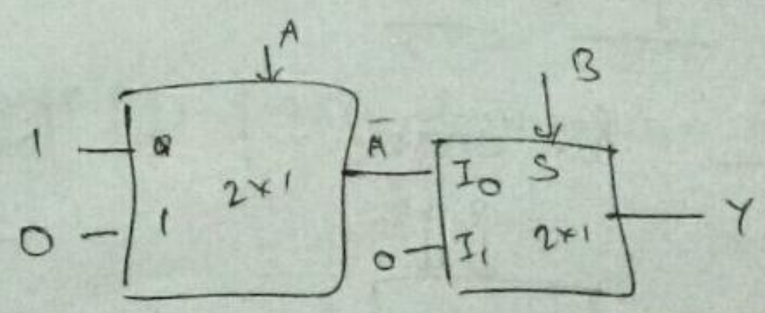
$\bar{B}+\bar{A}$

$\bar{A}$	0	1
A	2	3
	1	$\bar{A}$

↓
require not gate

NOR gate

$f(AB) = \Sigma m(0)$



$\bar{A}$	0	1
A	2	3
	$\bar{A}$	0

Scanned by TapScanner

XOR

$f(AB) = \Sigma_m(1,2)$

	I_0	I_1
$\bar{A}$	0	1
A	2	3
	A	$\bar{A}$

∴ require two Muxes

EXNOR

$f(A) = \Sigma_m(0,3)$

	I_0	I_1
$\bar{A}$	0	1
A	2	3
	$\bar{A}$	A

∴ requires two Muxes

→ All basic ~~2 i/p~~ gates require only ~~one~~ 1 Mux.

→ All universal & derived gates require 2 Muxes.

Enable input in a Mux:

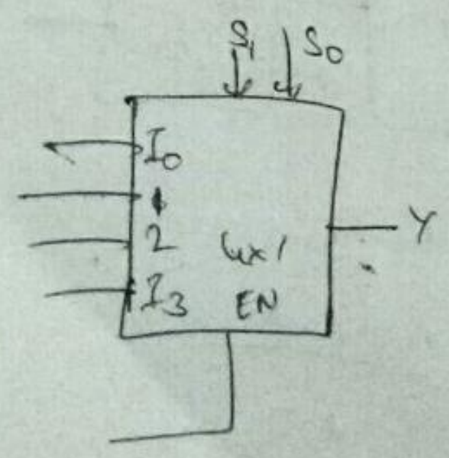
→ Some Muxes are designed with this special control input known as enable input.

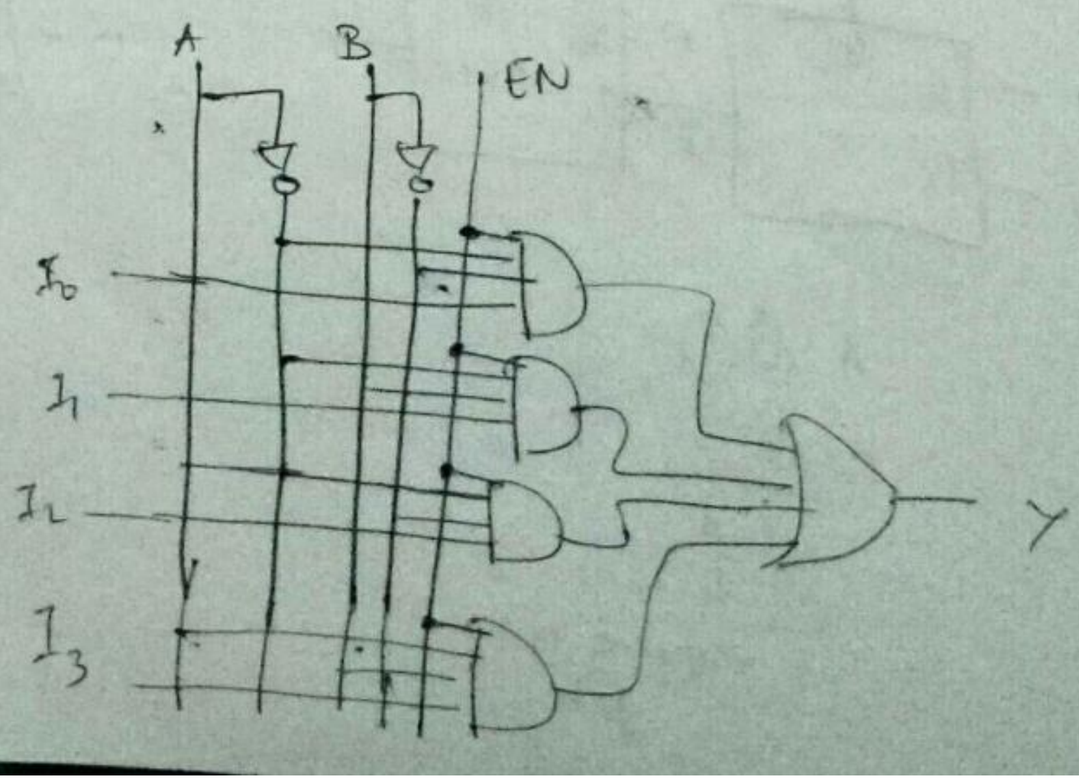
→ Before going to perform any operation the mux must be in enabled condition.

→ If mux is disabled then its O/P will be forced to '0' regardless of data inputs & selection lines.

Active high enable:

'1' for enable ; '0' for disable

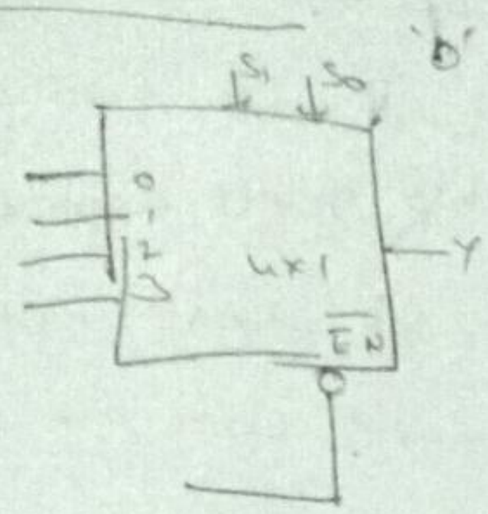


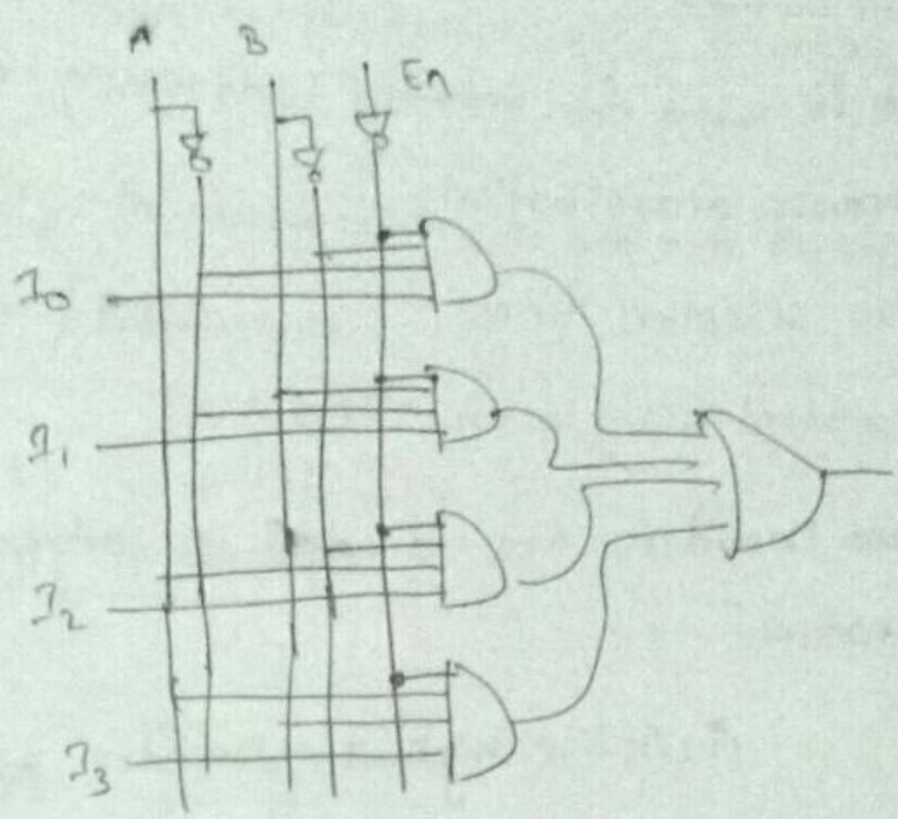


Scanned by TapScanner

Active low Enable.

'0' for enable '1' for disable





Demux :

→ It is also known as Data distributor, one to many selector, serial to parallel selector.

Standard size is 1×2^n

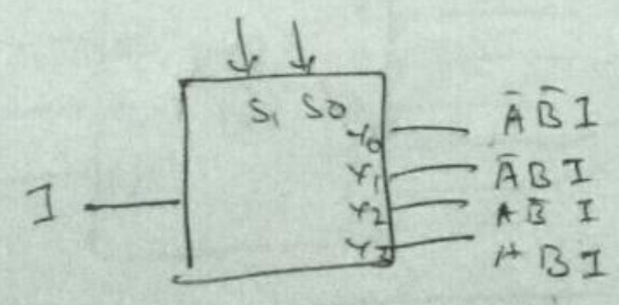
2^n = no of o/p with 'n' selection lines

→ selection lines are used to select the input 'I' to one of the inputs.

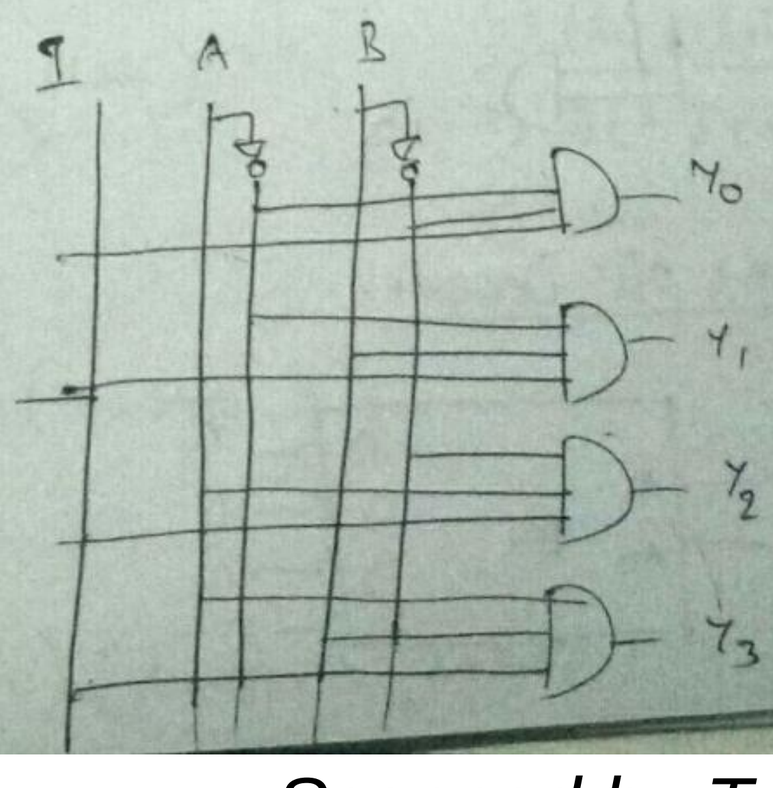
→ Different sizes are 1×2, 1×4, 1×8 .... 1×2^n

Q: No of selection lines req for designing 1x256 demux is 8

1x4 Demux



AB	Y_0	Y_1	Y_2	Y_3
00	I	0	0	0
01	0	I	0	0
10	0	0	I	0
11	0	0	0	I



Scanned by TapScanner

Decoder:

→ It is used for memory addressing and chip select application in memory organization.

→ No selection lines.

→ Standard size is $n \times 2^n$ (max)

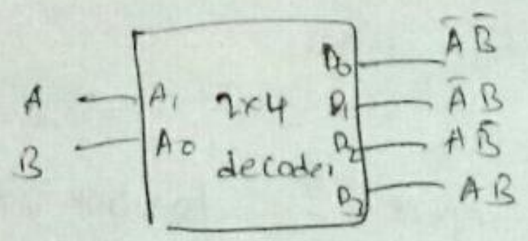
→ Decoders are divided in 2 types that depends on its output nature.

(i) Active high o/p Decoder → only one o/p bit is high

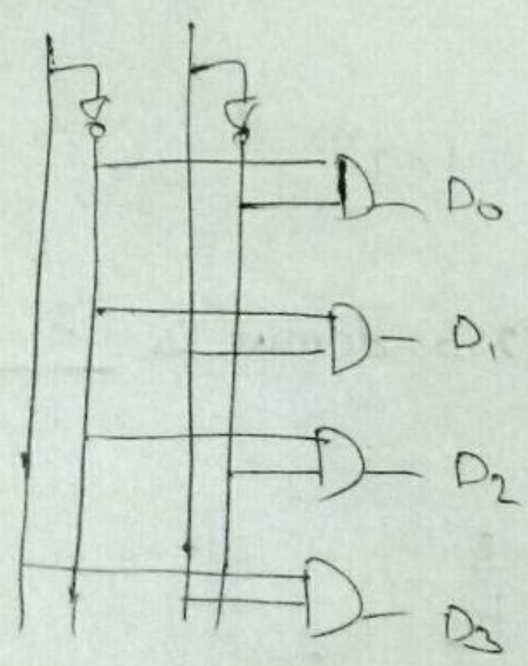
(ii) Active low o/p Decoder → only one o/p bit is low

☆ → whatever the size of the decoder and nature of it's o/p, always only one o/p bit is different to all other o/p bits.

Active high o/p decoder.

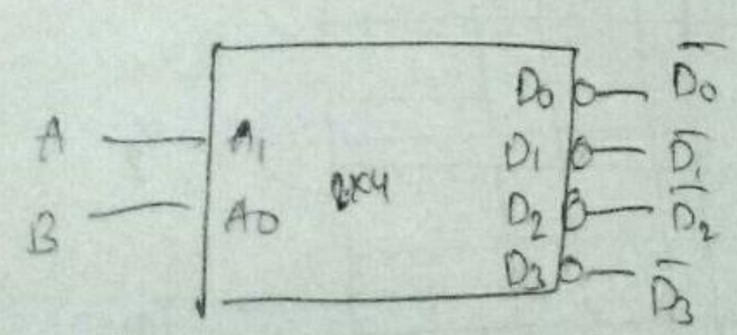


AB	D_0	D_1	D_2	D_3
0 0	1	0	0	0
0 1	0	1	0	0
1 0	0	0	1	0
1 1	0	0	0	1



ABCDE	Hig o/p given by
11011	D_{27}

Active low o/p Decoder



AB	$\bar{D}_0$	$\bar{D}_1$	$\bar{D}_2$	$\bar{D}_3$
0 0	0	1	1	1
0 1	1	0	1	1
1 0	1	1	0	1
1 1	1	1	1	0

Scanned by TapScanner

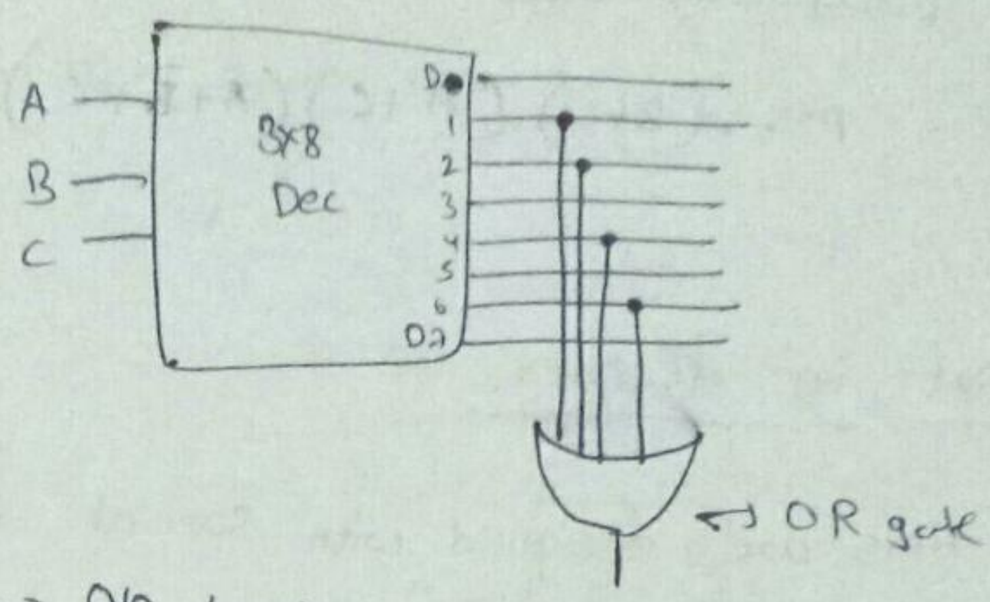
Decoder as function selector:

- Generally $n \times 2^n$ size decoder is used to implement one 'n' variable function.
- All functional inputs are connected to decoder input and minterms are conneted as input for OR gate / NAND gate

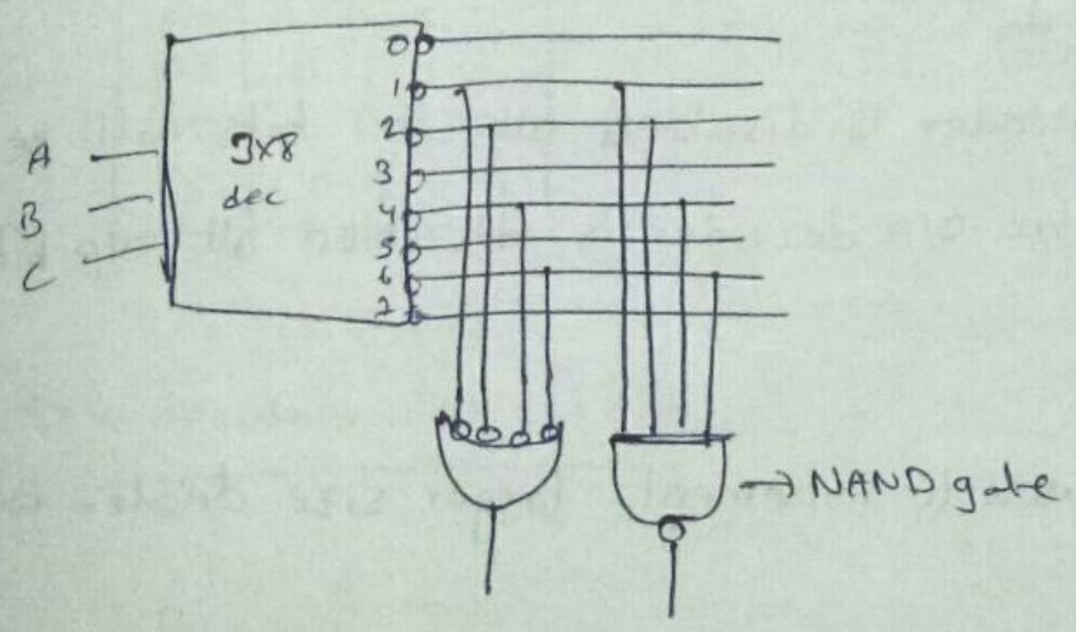
Eg: Implement $f(ABC) = \Sigma m(1,2,4,6)$ with Decoder

ABC	F
000	0
001	1
010	1
011	0
100	1
101	0
110	1
111	0

Active high O/P decoder

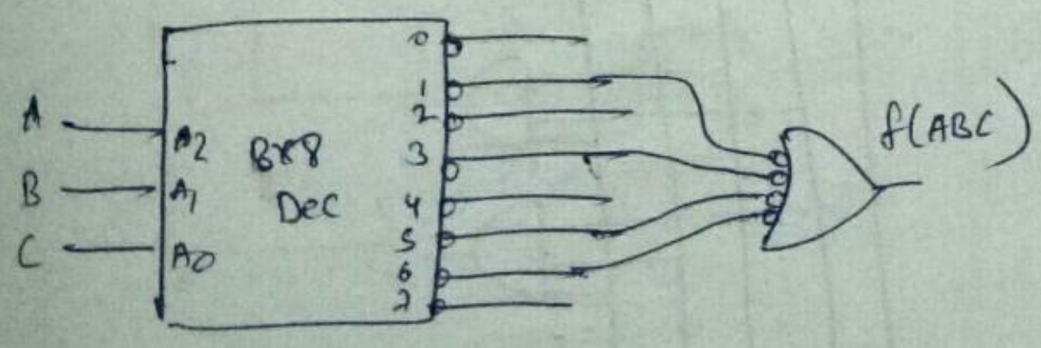


Active low O/P decoder



100P

Q The minimized POS expression for the given below diagram



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

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

c) $\bar{B}C + \bar{A}C + AB\bar{C}$.

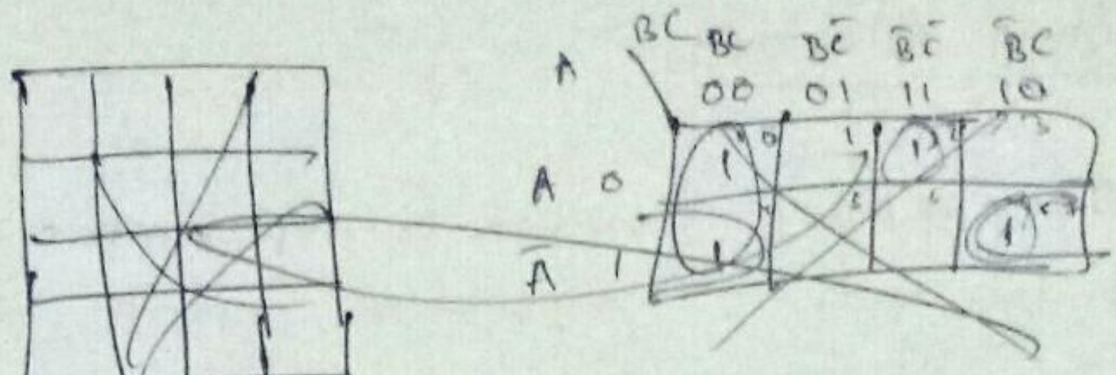
d) None

$f(ABC) = \Sigma m(1,3,5,6) = \Pi M(0,2,4,7)$

POS expression : ~~$ABC + A\bar{B}C$~~ +

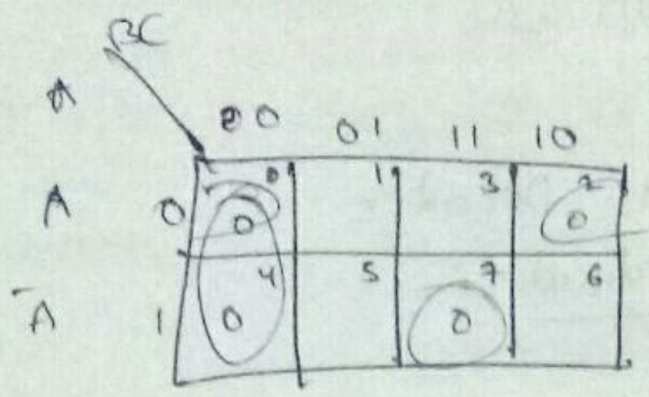
$(A+B+C)(A+\bar{B}+C)(\bar{A}+B+C)(\bar{A}+\bar{B}+\bar{C})$

Scanned by TapScanner



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

(1, 4) (1) (4, 8)



pos: $(B+C)(A+C)(\bar{A}+\bar{B}+\bar{C})$

Enable input in decoder:

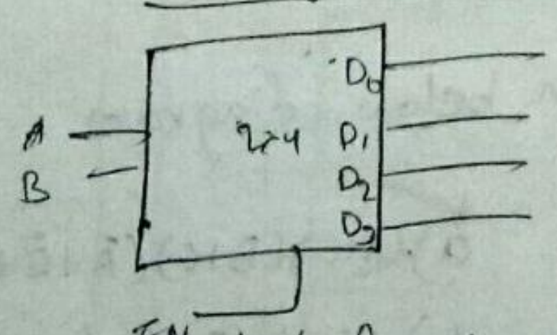
→ Some decoders are designed with special input known as enable input.

→ Before going to perform any operation the decoder should be enabled.

→ If active high o/p decoder is disabled all o/p bits will be forced to low and if active low o/p decoder is disabled all o/p bits will be forced to high.

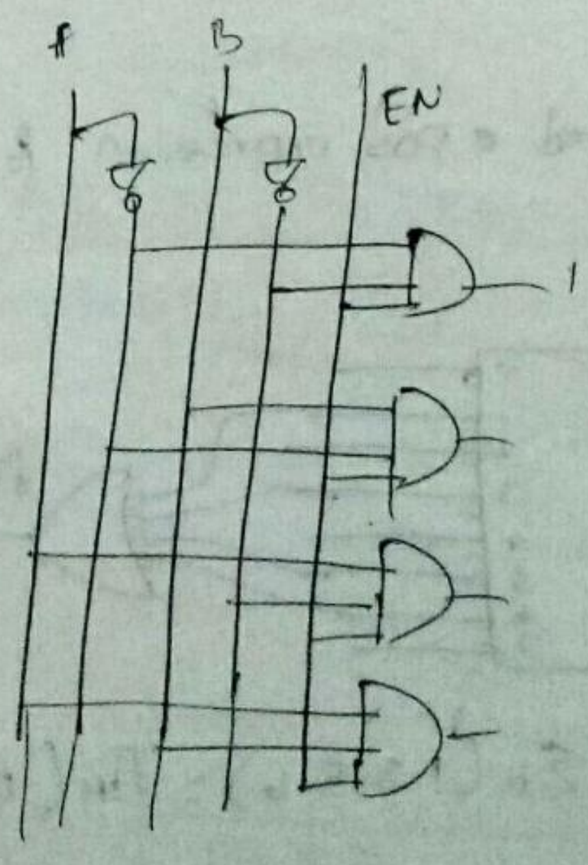
→ This enable input is used to implement larger size decoder with smaller size decoders.

Active high o/p decoder



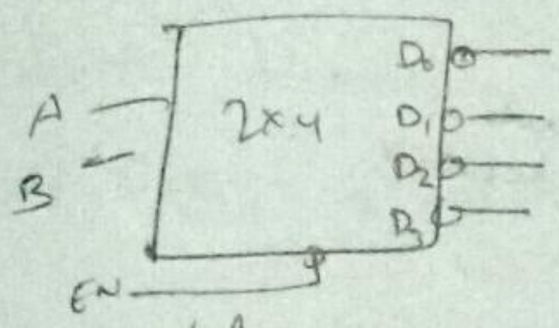
'0' for disable

'1' for enable



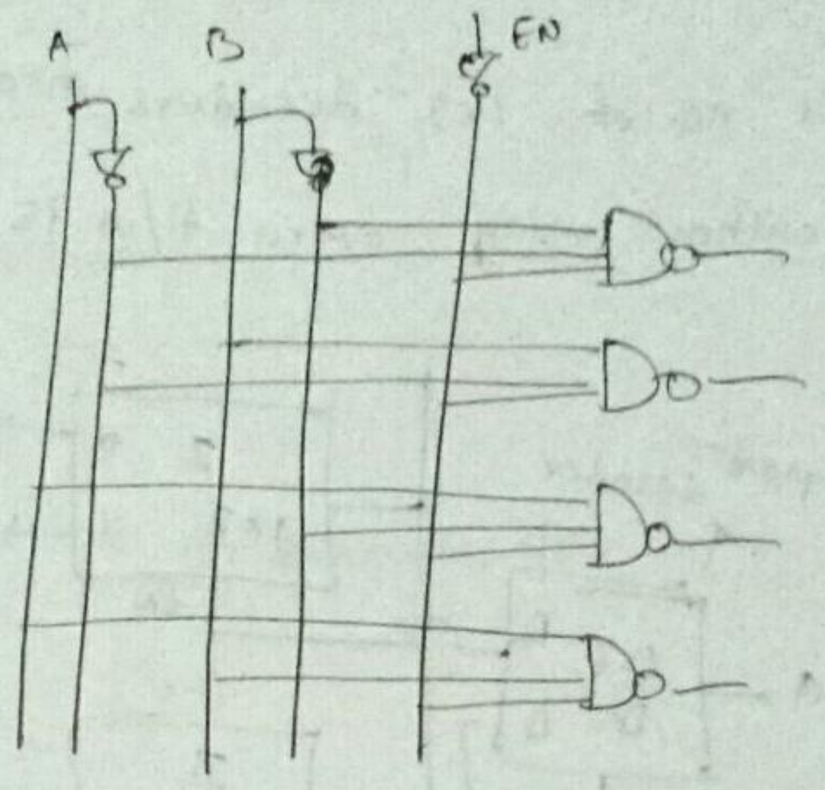
Scanned by TapScanner

Active low o/p decoder



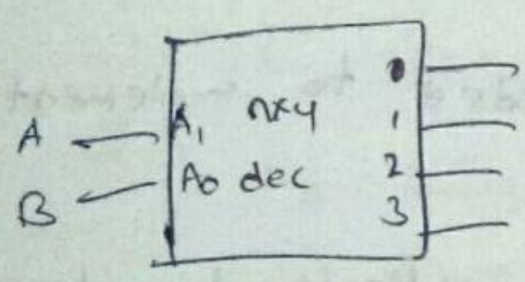
'0' for enable

'1' for disable



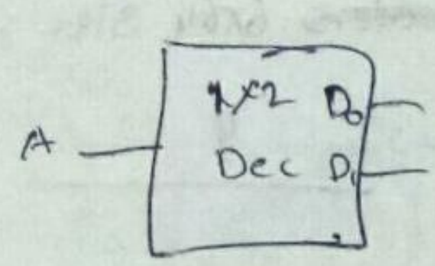
Eg: Implement 2x4 decoder with 1x2 decoders

Target size



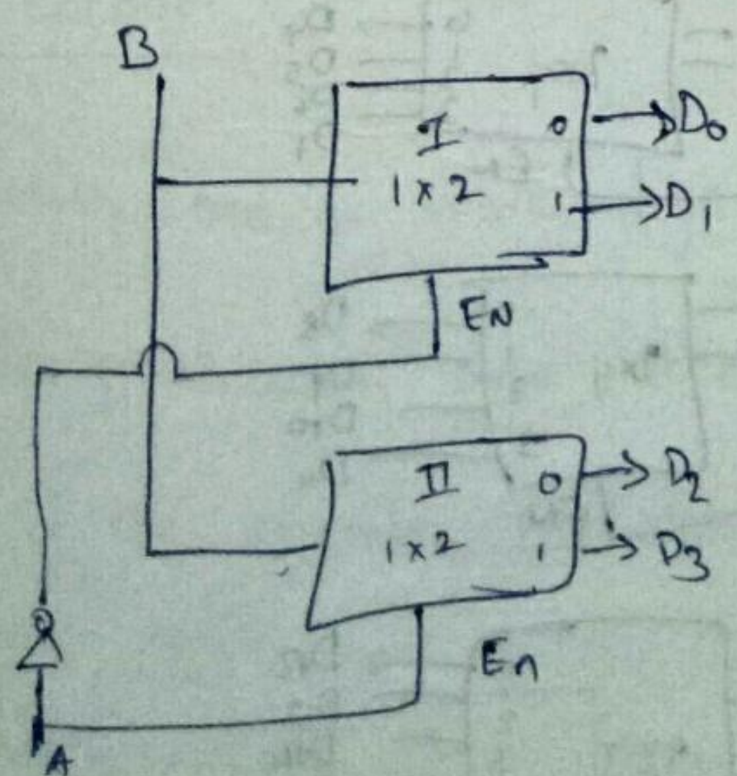
AB	D_0	D_1	D_2	D_3
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

Basic size



A	D_0	D_1
0	1	0
1	0	1

2x4 decoders with 1x2



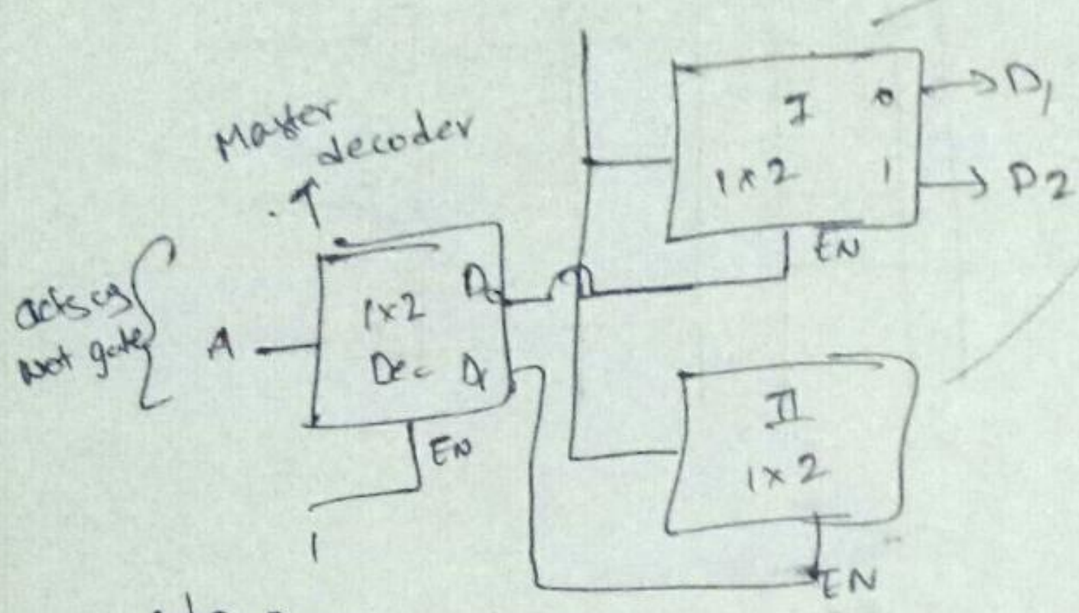
AB	D_0	D_1	D_2	D_3
	I		II	
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

disabled when A=0

disable when A=1

Scanned by TapScanner

Q: Min no of 1x2 decoders needed to implement 2x4 decoder without using extra H/w is 3



AB	D_0 D_1 D_2 D_3
00	1 0 0 0
01	0 1 0 0
10	0 0 1 0
11	0 0 0 1

} disabled when A=0

disable when A=1

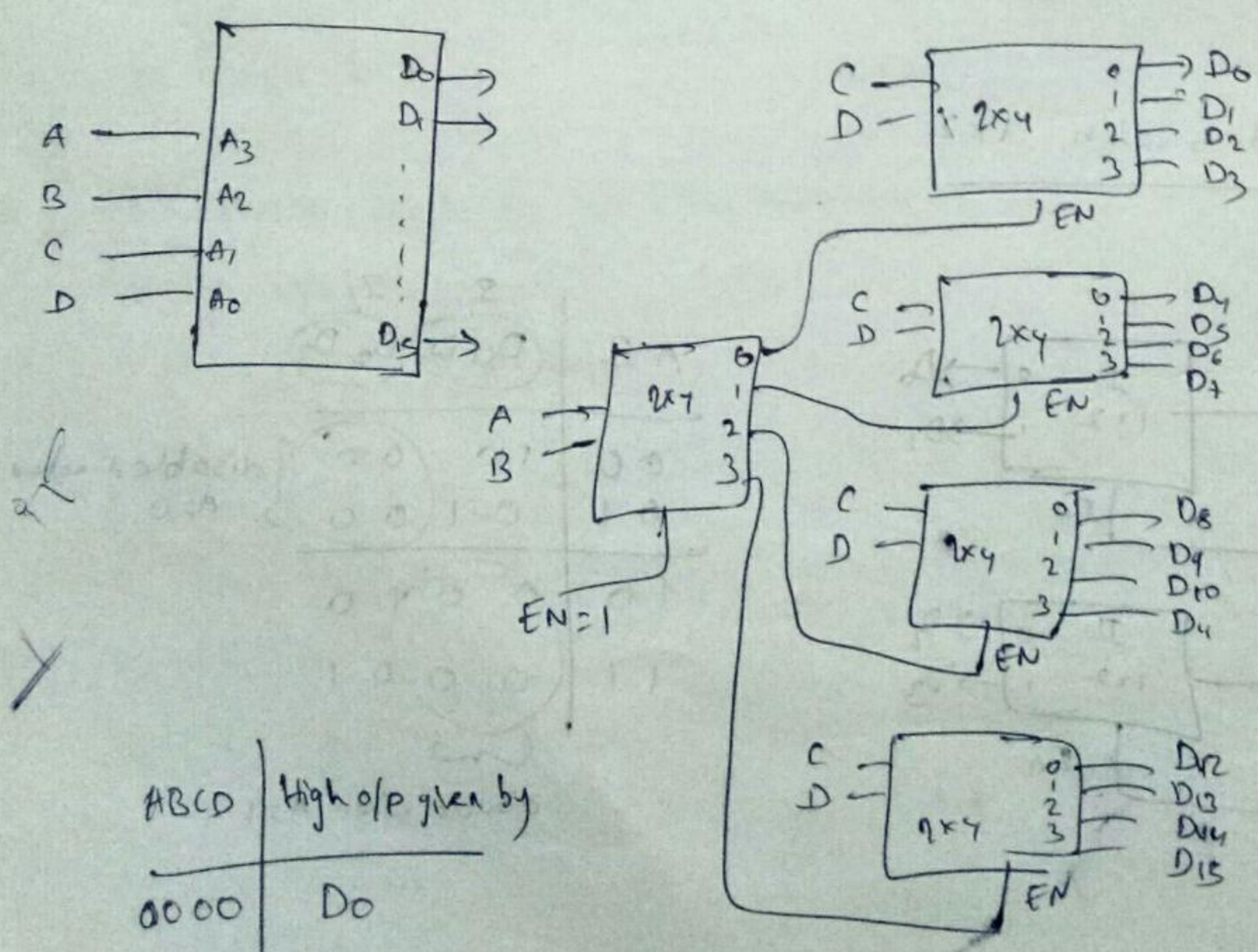
A	D_0 D_1
0	1 0
1	0 1

2007

Q Min no of ~~size decoders~~ 6x64 size decoders needed to implement 3x8 size decoder is 9

$\frac{6\times64}{3\times8} = 8$, 1 Master decoder

Eg: Implement 4x16 decoder with 2x4 decoder



ABCD	High o/p given by
0000	D_0
0111	D_7
1001	D_9
1111	D_{15}

Scanned by TapScanner

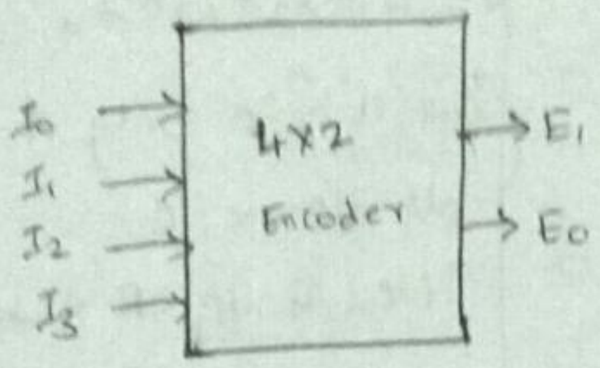
Encoder:

→ It performs decoder's inverse function.

standard size is $2^n \times n$

→ Always only one high signal is permitted at ~~output~~ i/p.

→ Different sizes are 2×1, 4×2, 8×3 .... $2^n \times n$

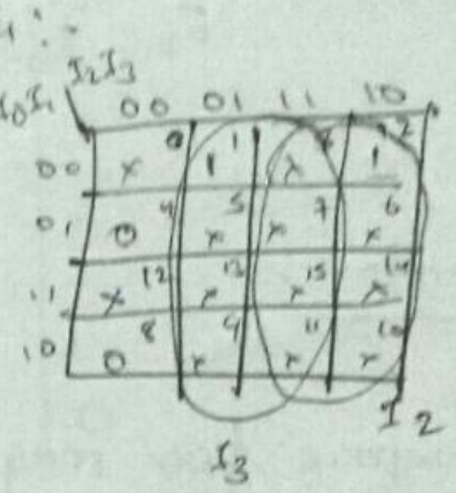


I_0	I_1	I_2	I_3	E_1	E_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

$E_1 = I_2 + I_3$

$E_0 = \cancel{I_0 + I_1}$ $I_1 + I_3$

for E_1:-



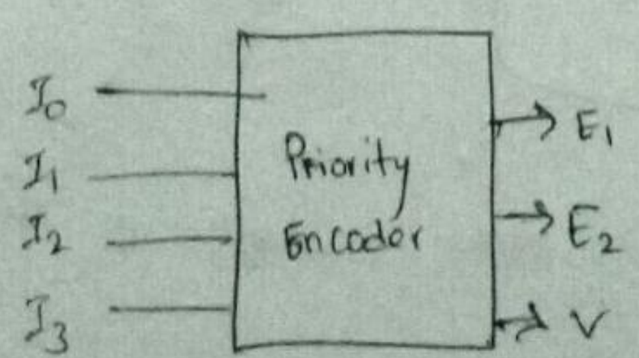
→ If i/p contains more than one high ~~bits~~ signal (Eg: 1100) then it is invalid i/p.

Priority Encoder:

→ It permits one or more high signals at i/p.

→ It is designed with one special o/p bit(V).

→ Output of the encoder becomes valid if V=1 only.



Scanned by TapScanner

	I_0	I_1	I_2	I_3	E_1	E_0	↓
0	0	0	0	0	x	x	0 (invalid)
(8 to 15)	1	x	x	x	0	0	1
(4 to 7)	0	1	x	x	0	1	1
(2 to 3)	0	0	1	x	1	0	1
(1)	0	0	0	1	1	1	1

For E_1

I_0I_1 \ I_2I_3	00	01	11	10
00	x	1	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$E_1 = \bar{I}_0 \bar{I}_1$

For E_2

I_0I_1 \ I_2I_3	00	01	11	10
00	x	1	0	0
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

$E_2 = \bar{I}_0 I_1 + \bar{I}_0 I_2$

$N = I_1 + I_2 + I_3 + I_4$

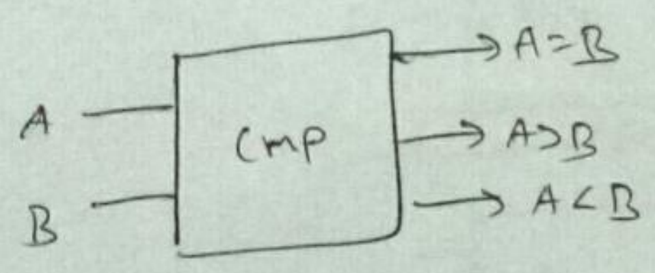
(Coz it is '0' only if all i/p are 0 i.e., 4 i/p OR gate)

Magnitude Comparator:

→ It is used to compare two magnitudes (values)

One bit
Two bit
Three bit
n bit

one bit cmp:



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

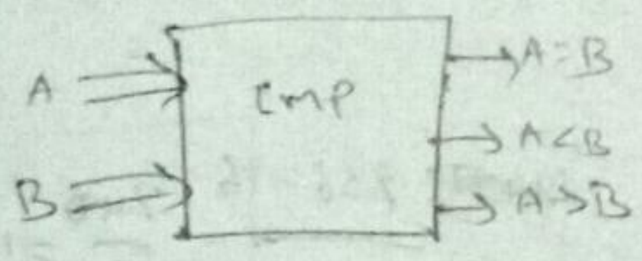
$F_{A=B} = A \odot B$

$F_{A<B} = \bar{A} B$

$F_{A>B} = A \bar{B}$

Scanned by TapScanner

2 bit comperator



A B	A=B	A>B	A<B
00 00	1	0	0
0001	0	0	1
0010	0	0	1
0011	0	0	1
0100	0	1	0
0101	1	0	0
0110	0	0	1
0111	0	0	1
1000	0	1	0
1001	0	1	0
1010	1	0	0
1011	0	0	1
1100	0	1	0
1101	0	1	0
1110	0	1	0
1111	1	0	0

Here

one data size = 2

combination size = 4

Max no of combinations = 2^4

$F_{A=B}$ provides high o/p for 2^2 combinations = 4

$F_{A<B}$ - - - - $\frac{2^4-2^2}{2} = 6$

$F_{A>B}$ - - - - $\frac{2^4-2^2}{2} = 6$

In n bit comperator

one data size = n bit

combination size = 2^n bits

Max no of combinations = 2^{2n}

No of A=B combination = 2^n

No of A<B " $= \frac{2^{2n}-2^n}{2}$

No of A>B " $= \frac{2^{2n}-2^n}{2}$

(202)

Q No of A>B combinations that provide high o/p in 2 bit Comparator is

6

i.e, $\frac{2^{2n}-2^n}{2} = \frac{2^4-2^2}{2} = \frac{16-4}{2} = 6$

Scanned by TapScanner

Q: No of A<B combinations that provide high o/p in 4 bit comparator is 120

i.e. $\frac{2^{2n}-2^n}{2} = \frac{2^8-2^4}{2} = \frac{256-16}{2} = \frac{240}{2} = 120$

5. Sequential Circuits

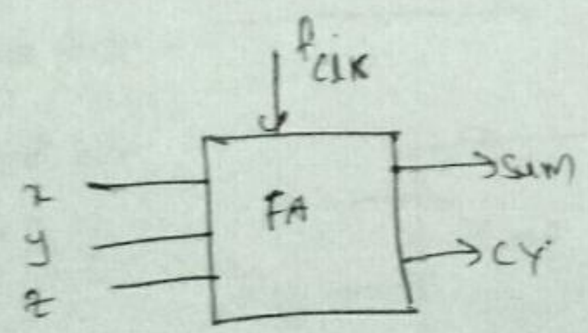
→ All sequential circuits are clocked circuits and frequency of the clock pulse computes the o/p arrival time.

→ Generally universal gates are used to design a sequential circuits.

→ In the given sequential circuit if the clock i/p signal is '0' then no changes are present at the o/p irrespective of i/p data.

→ To operate ~~the se~~ for sequential circuit reliable operation, the max clock frequency to be applied is less than $1/t_{pd}$ where t_{pd} is ckt propagation delay

Consider below FA



$T_{sum} = 1ns$

$T_{CY} = 1ns$

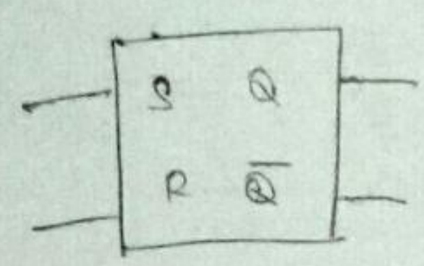
f_{clk}	T (ns)
100 MHz	10
200 MHz	5
500 MHz	2
900 MHz	1.1
1 GHz	1
2 GHz	0.5 (X) not permitted

because it is less than minimum time delay given

Scanned by TapScanner

.SR Latch

→ It is the pre-stage for a Flip flop



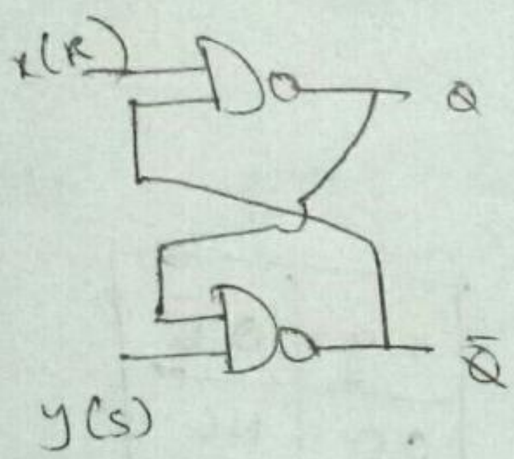
S = set = It sets the 'Q' o/p

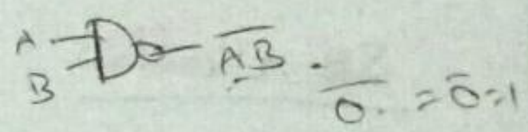
R = Reset = It resets the 'Q' o/p

whenever $Q = \bar{Q}$ then that type of i/p is considered as invalid.

→ It contains memory, but no clock.

SR NAND Latch





x y	Q $\bar{Q}$
0 0	1 1 (invalid)
0 1	1 0
1 0	0 1
1 1	N.C

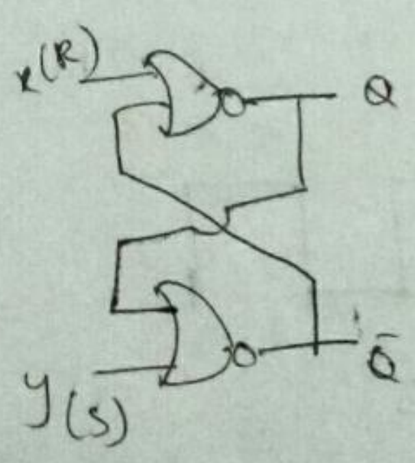
x y	Q
0 0	x
0 1	1 Set
1 0	0 Reset
1 1	NC

∴ y ⇒ x = R

⇒

S R	Q
0 0	X
0 1	0
1 0	1
1 1	NC

SR NOR Latch



x y	Q $\bar{Q}$
0 0	N.C
0 1	1 0
1 0	0 1
1 1	0 0 (invalid)

x y	Q
0 0	NC
0 1	1 Set
1 0	0 Rest
1 1	x

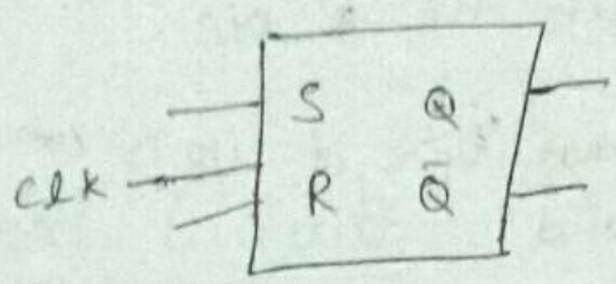
∴ y = S
x = R

S R	Q
0 0	NC
0 1	0
1 0	1
1 1	x

Scanned by TapScanner

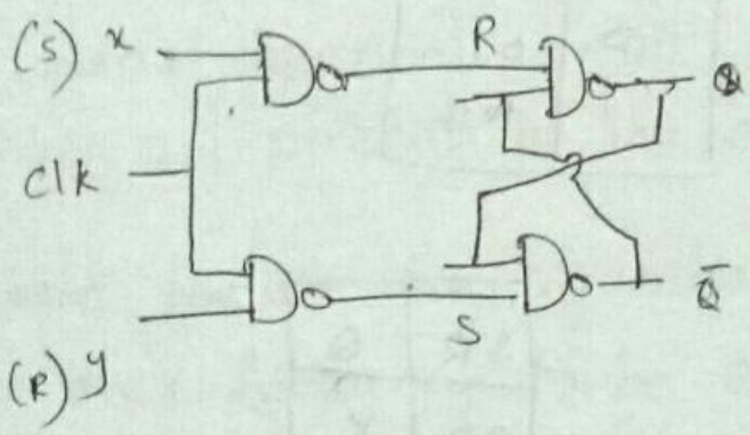
SR Flip flop

→ When clock pulse is connected to SR latch then it acts as SR Flip Flop



~~SR Flipflop~~

~~→ when clock pulse is connected~~ 1



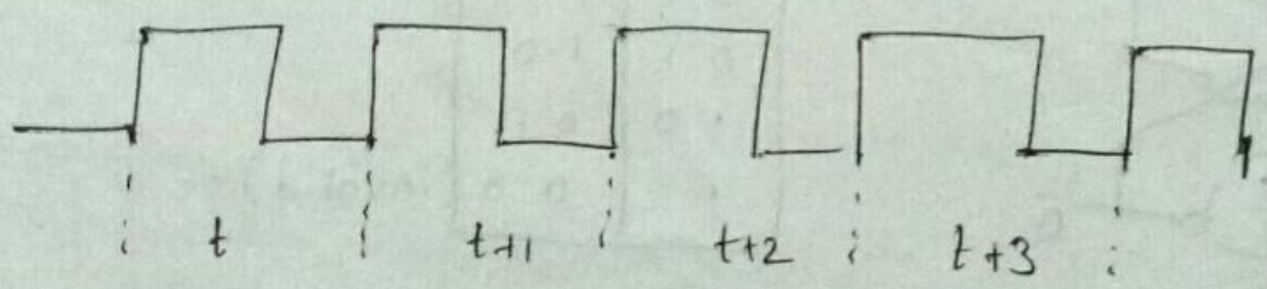
x y	Q $\bar{Q}$
0 0	NC
0 1	0 1
1 0	1 0
1 1	1 1 (x) invalid

Here Clk is not zero

x y	Q
0 0	NC
0 1	0 reset
1 0	1 set
1 1	x

SR	Q
0 0	NC
0 1	0
1 0	1
1 1	x

Clk



Output (Q) Q_t Q_{t+1} Q_{t+2} Q_{t+3}

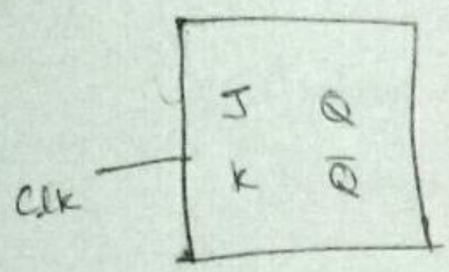
Q_t = Present State (PS)

Q_{t+1} = Next state (NS)

S R	Q_{t+1}
0 0	Q_t (NC)
0 1	0
1 0	1
1 1	x

Scanned by TapScanner

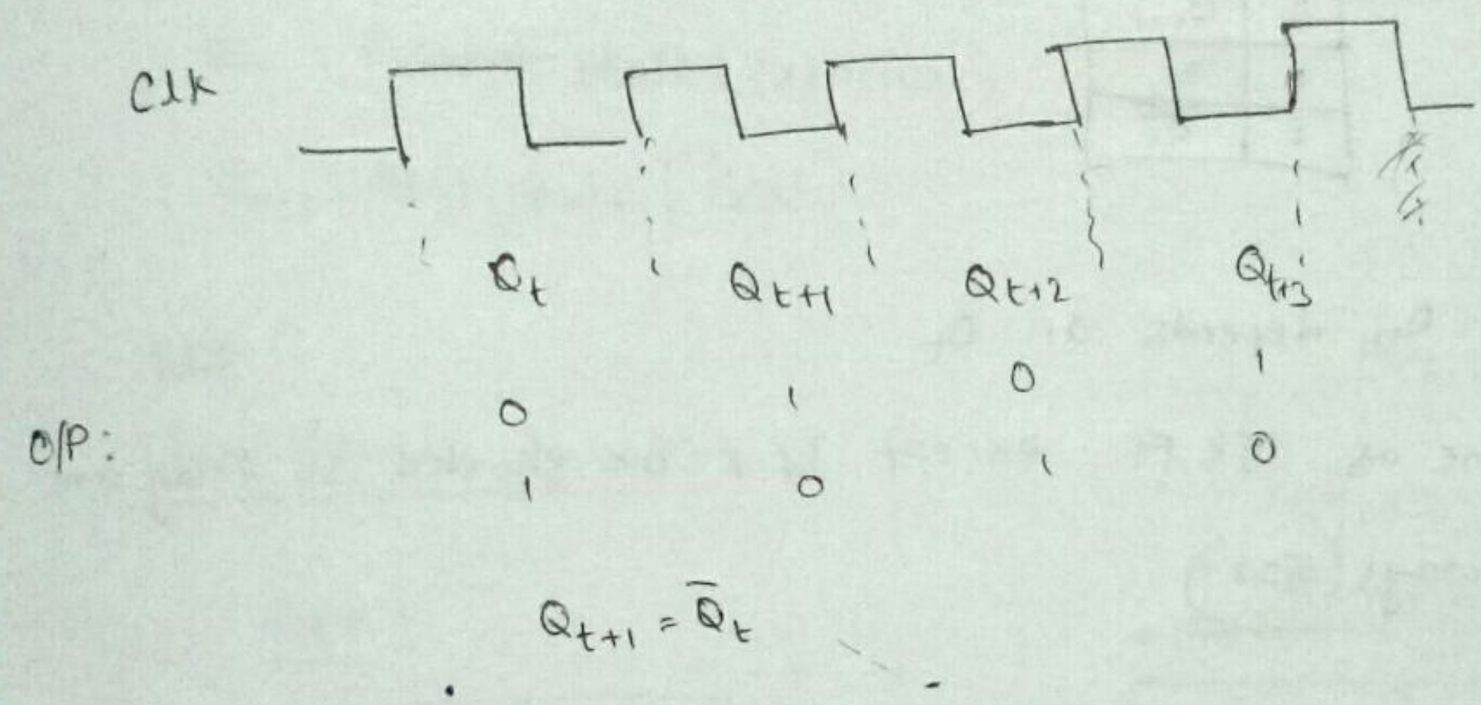
JK Flipflop



	JK	Q_{t+1}
Like SR FF	00	Q_t
	01	0
	10	1
	11	$\bar{Q}_t$ (Toggle)

Toggle: It is the process of complementing the O/P of a FF for each new clock

→ Toggle is used in counter design.



$Q_{t+1} = \bar{Q}_t$

3 bit up counter

Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Single input flip flops

→ ~~Exciton~~ Excitation table is easy.

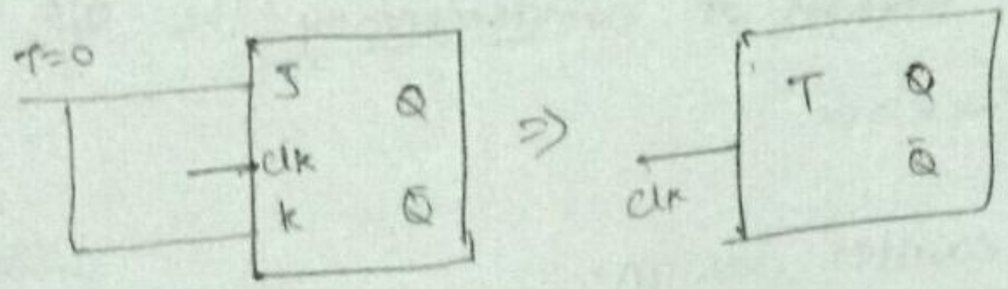
JKFF

- TFF, J=K
- DFF (data FF, direct FF, delay FF, dumb FF), J≠K

Scanned by TapScanner

	JK	Q_{t+1}
TFF	00	Q_t
	11	$\overline{Q_t}$
DFF	01	0
	10	1

TFF using JKFF



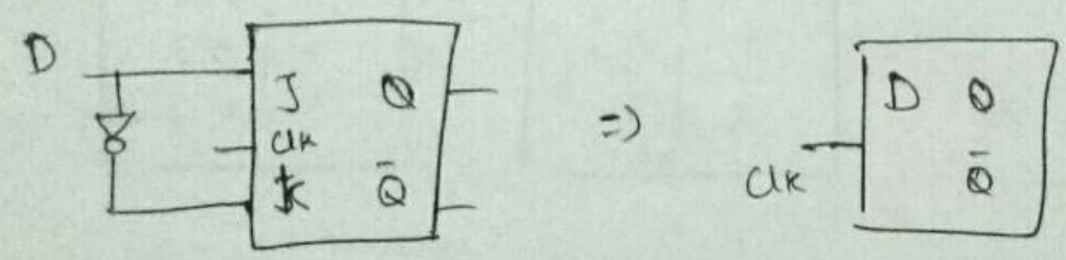
T	JK	Q_{t+1}
0	00	Q_t
1	11	$\overline{Q_t}$

T	Q_{t+1}
0	Q_t
1	$\overline{Q_t}$

→ In TFF Q_{t+1} depends on Q_t

→ It is same as JK FF except J & K are shorted so they are equal always (J=K)

DFF using JKFF



D	JK	Q_{t+1}
0	01	0
1	10	1

D	Q_{t+1}
0	0
1	1

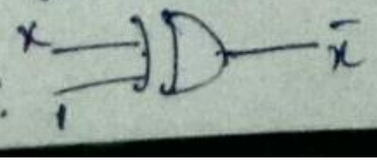
→ Here J ≠ K.

Q: To implement DFF using JKFF we require

a) NOT gate b) ~~NAD~~ NAND c) NOR

d) EXOR e) 2x1 Mux f) Any of the above

∴ Not gate can be implement with any of given 5 gates

NOT with XOR: 

Scanned by TapScanner

Flip Flop Tables

→ Each FF table can be divided in 3 ways:

(i) General table
(ii) Characteristic table
(iii) Excitation table

→ General table tells FF's nature. (Tables that are drawn so far are general tables)

→ Characteristic table tells the relations b/w Q_t & Q_{t+1}

→ Excitation table is used to design any synchronous sequential circuit.

Q_t : Present state (expected)

Q_{t+1} : Next state (final)

~~DFF~~

Characteristic tables:

DFF :

D	Q_t	Q_{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

$Q_{t+1} = D\bar{Q}_t + DQ_t$

$Q_{t+1} = D$ (Characteristic equation)

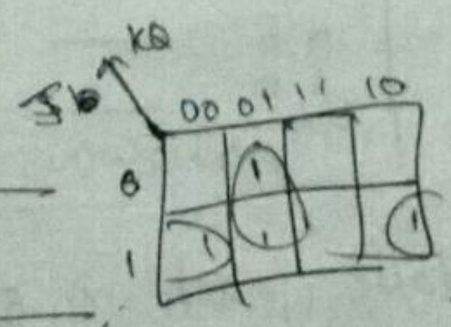
TFF

	T	Q_t	Q_{t+1}
NC	0	0	0
	0	1	1
Toggle	1	0	1
	1	1	0

$Q_{t+1} = T \oplus Q_t$ (Characteristic eq of TFF)

JKFF

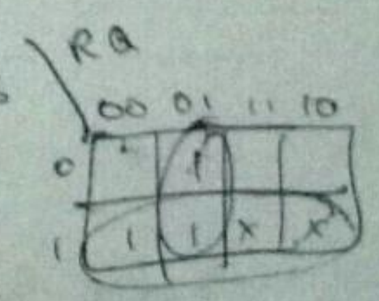
	J K	Q_t	Q_{t+1}
NC	0 0	0	0
	0 0	1	1
0	0 1	0	0
	0 1	1	0
1	1 0	0	1
	1 0	1	1
Toggle	1 1	0	1
	1 1	1	0



$Q_{t+1} = J\bar{Q} + \bar{K}Q$

SRFF

S	R	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x



$S + \bar{R}Q$

Scanned by TapScanner

Excitation tables:

→ First write our target states (Q_t & Q_{t+1})

→ Then define the i/p for the available Flip Flop

TFF

general table

T	Q_{t+1}
0	Q_t (NC)
1	$\bar{Q}_t$

Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation table

DFF

general table

D	Q_{t+1}
0	0
1	1

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

JKFF

	Jk	Q_{t+1}
T	00	NC
	11	Toggle
D	01	0
	10	1

Q_t	Q_{t+1}	Jk	
0	0	00	No change
		01	Direct '0' o/p
		0X	
0	1	11	Toggle
		10	Direct '1' o/p
		1X	
1	0	11	Toggle
		01	Direct '0' o/p
		X1	
1	1	00	No change
		10	Direct '0' o/p
		X0	

→ Here in building exciting table each $Q_t Q_{t+1}$ is checked for possibility in DFF & TFF

Eg: for 00 we can see it as No change (or) Direct '0' o/p

Scanned by TapScanner

SRFF

SR	Q_{t+1}
00	NC
01	0
10	1
11	x

Q_t	Q_{t+1}	S R	
0	0	00	— No change.
		01	— '0': ~~set~~ reset by R
		0x	
0	1	10	— '1': set by S
1	0	01	— '0': reset by R
1	1	00	— No change
		10	— '1': set by S
		x0	

Counter & It's design

→ Counter is used to count the no of states.

→ Each state is available for one clock cycle time.

→ n bit counter produces max 2^n states.

→ If n bit counter is used to count all 2^n states then it is binary counter otherwise it is mod counter.

→ Mod counter is used to count the target no of states only.

→ Mod k counter is used to count k no of states.

→ In real time applications mod counter is useful than binary counter, but design cost of ~~bin~~ mod counter is more expensive than design cost of binary counter.

→ There are two methods to design a counter:

(i) Synchronous counter

(ii) Asynchronous counter (Ripple counter)

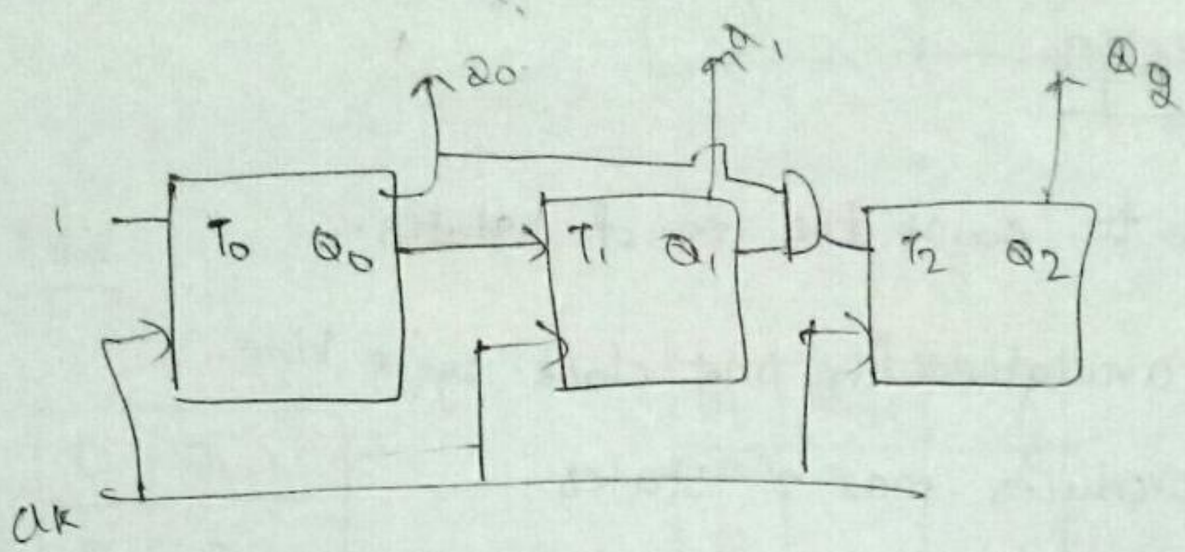
Design of Synchronous Counter:

→ All flip flops' clock i/ps are connected to main clock generator and flip flop i/ps are defined with excitation table.

Scanned by TapScanner

3 bit upcounter

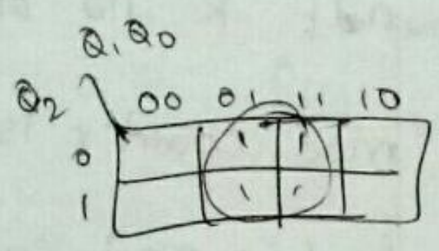
	$Q_2 Q_1 Q_0$	T_2	T_1	T_0
Q_t	0 0 0	0	0	1
Q_{t+1}	0 0 1	0	1	1
Q_{t+2}	0 1 0	0	0	1
Q_{t+3}	0 1 1	1	1	1
⋮	1 0 0	0	0	1
⋮	1 0 1	0	1	1
⋮	1 1 0	0	0	1
Q_{t+7}	1 1 1	1	1	1



O/P : $Q_2 Q_1 Q_0$

$$T_2(Q_2 Q_1 Q_0) = \overline{Q_2} Q_1 Q_0 + Q_2 Q_1 Q_0$$
$$= Q_1 Q_0$$

for T_1 :



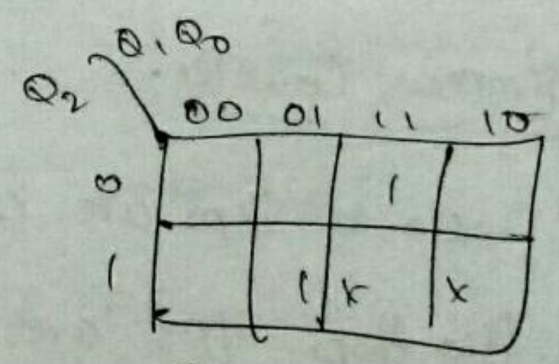
$$T_1(Q_2 Q_1 Q_0) = Q_0$$

for (T_0)

$$T_0(Q_2 Q_1 Q_0) = 1$$

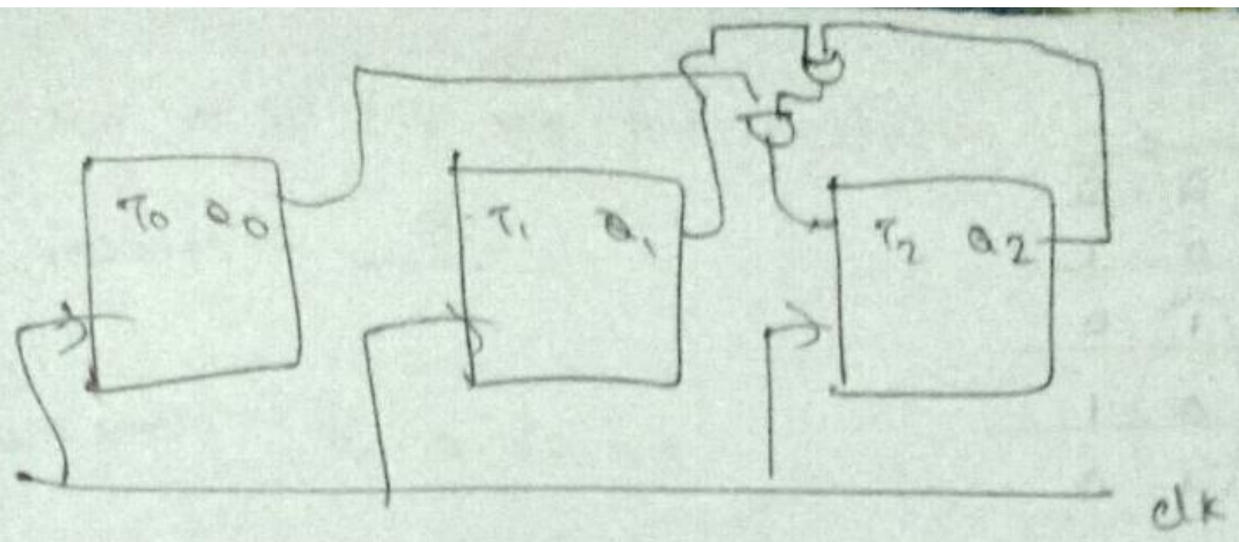
Mod 6 up counter:

$Q_2 Q_1 Q_0$	T_2	T_1	T_0
0 0 0	0	0	1
0 0 1	0	1	1
0 1 0	0	0	1
0 1 1	1	1	1
1 0 0	0	0	1
1 0 1	1	0	1
1 1 0	x	x	x
1 1 1	x	x	x



$$T_2 = Q_1 Q_0 + Q_2 Q_0$$
$$= Q_0(Q_1 + Q_2)$$

Scanned by TapScanner



→ Here T_2 ~~also~~ & alone req 2 gate which makes mod counter design expensive

Eg: Design a counter for the states of 0, 3, 6, 2, 5, ~~and rep~~

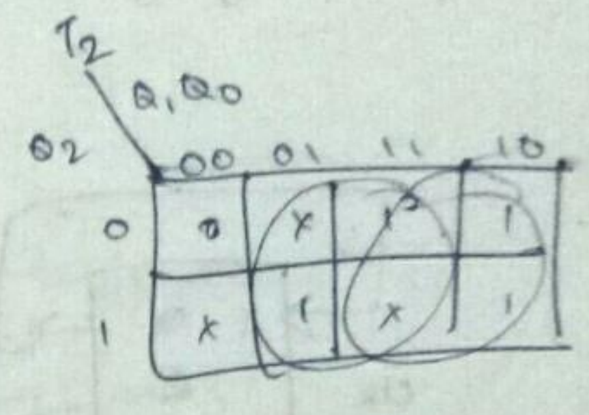
string: - 0, 3, 6, 2, 5

No state is repeated

Highest digit = 6 (110)

∴ No of FF's req = 3

	Q_2	Q_1	Q_0	T_2	T_1	T_0
0	0	0	0	0	1	1
3	0	1	1	1	0	1
6	1	1	0	1	0	0
2	0	1	0	1	1	1
5	1	0	1	1	0	1
1	0	0	1	x	x	x
4	1	0	0	x	x	x
7	1	1	1	x	x	x



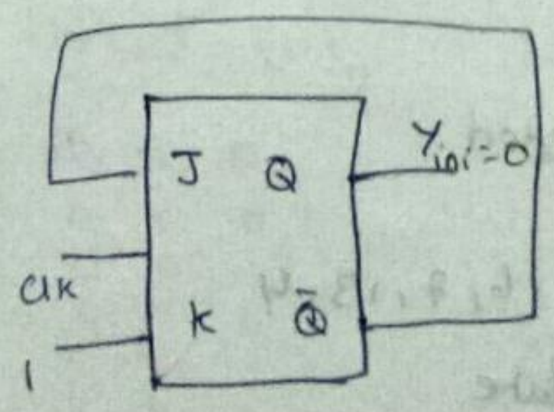
$T_2 = Q_1 + Q_0$

Slly derive expression for T_1 & T_2

08/09/19

Q) In the below diagrams initial o/p at Y is '0'. what is output signal at 'Y' for next 3 clocks?

1)



a) 1, 1, 1

b) 0, 0, 0

c) 1, 0, 1

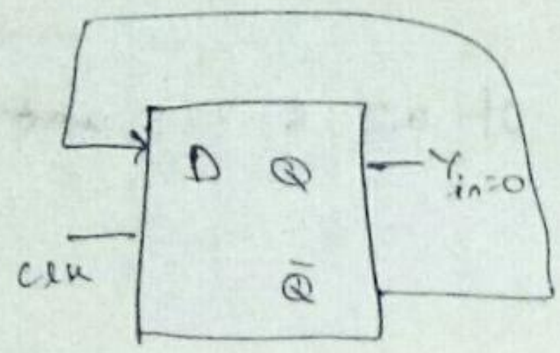
d) 0, 1, 0

Scanned by TapScanner

Clk	J	K	Q	$\bar{Q}$
	0	1	0	1
↑	1	1	1	0
↑	0	1	0	1
↑	1	1	1	0

∴ 1, 0, 1

2)

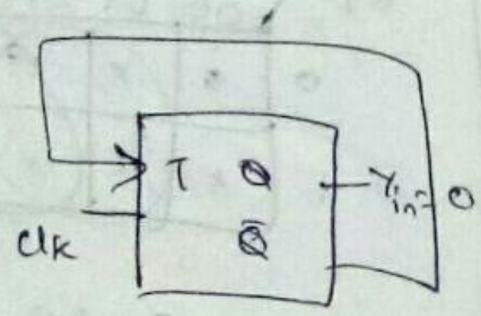


a) 0, 0, 0
b) 1, 1, 1
c) 1, 0, 1
d) 0, 1, 0

Clk	D	Q	$\bar{Q}$
		0	1
↑	1	1	0
↑	0	0	1
↑	1	1	0

∴ 1, 0, 1

3)



a) 0, 0, 0
b) 1, 1, 1
c) 1, 0, 1
d) 0, 1, 0

Clk	T	Q	$\bar{Q}$
		0	1
↑	1	1	0
↑	0	1	0
↑	0	1	0

∴ 1, 1, 1

Q: Min no of FFs req for designing a counter for generating the states 2, 5, 0, 9, 3, 1, 12, 6, 7, 13, 4 and repeats is 4

~~Max value = 14 (1110)~~

~~i.e, 4 FFs are needed~~

Given string 2, 5, 0, 9, 3, 1, 12, 6, 7, 13, 4

There is no repeated state

Highest state $= 13_{10} = (1101)_2$

∴ 4 FFs are needed

Scanned by TapScanner

Q: Minimum no of FFs req for producing ~~0,2,1,~~ 0,1,2,1,3 and repeats is <u>3</u>

Sol:

Given string is 0,1,2,1,3

state '1' is repeated, hence this design requires additional FFs known as dummy FFs

Note:

Before writing the excitation table all states should be distinct.

Q_1	Q_0	T_1	T_0
0	0	0	1
0	1	1	1
1	0	1	1
0	1	1	0
1	1	1	1

meaningless

∴ To distinguish same states we use a <u>dummy filpflop</u>

So Consider the excitation table as

Active FF → Q_1 Q_0 Q_D → Dummy FF

Q_1	Q_0	Q_D
0	0	0
0	1	0
1	0	0
0	1	1
1	1	0

different No

These bits can be be either all '0' or all '1'.

∴ We need 3 FFs

<u>Problems</u>

(Q15)

6) Given 0,0,1,1,2,2,3,3

each state repeated for 2 times

Max digit = 3 = $(11)_2$

Q_0	Q_1	Q_D
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

∴ we need 3 FFs

2 active FFs

1 dummy FF

Scanned by TapScanner

(2016)

1) Given string 0, 1, 0, 2, 0, 3

Max value = $3 = (11)_2$

$\therefore$ 2 active FFs needed

0 is repeated for 3 time

So we need two Dummy FFs

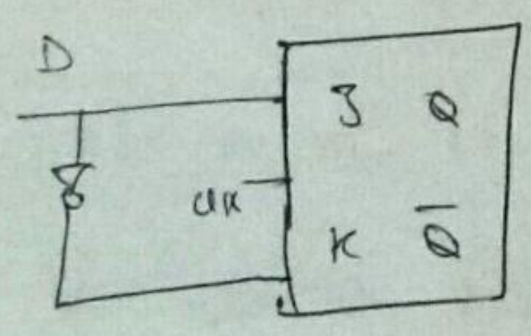
Q_1	Q_0	Q_{D_1}	Q_{D_2}	
0	0	0	0	— 1st time
0	1	0	0	
0	0	0	1	— 2nd time
1	0	0	0	
0	0	0	0	— 3rd time
1	1	0	0	

$\therefore$ we need 4 FFs

Flip Flop Conversions:

→ Any target FF can be implemented with available FF. In this Process first write characteristic table for the target FF then excitation table for available FF.

DFF using ~~JKK~~ JKFF:

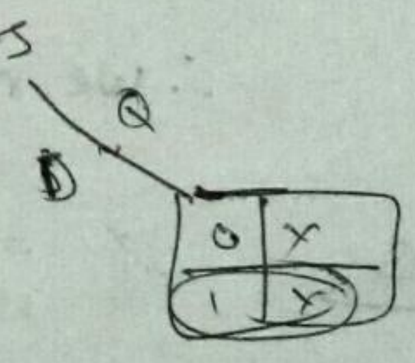


$J = D$

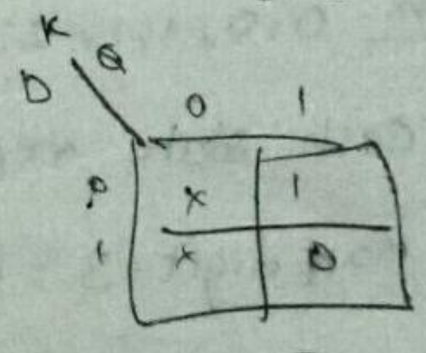
$K = \bar{D}$

$J = \bar{K}$

D	Q	Q_{t+1}	J	K
0	0	0	0	x
0	1	0	x	1
1	0	1	1	x
1	1	1	x	0



$J = D$



$K = \bar{D}$

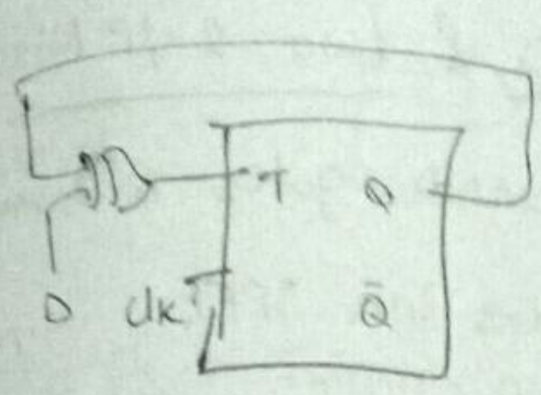
Scanned by TapScanner

DFF using TFF

D	Q_t	Q_{t+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

$T = \bar{D}Q + D\bar{Q}$

$T = D \oplus Q$

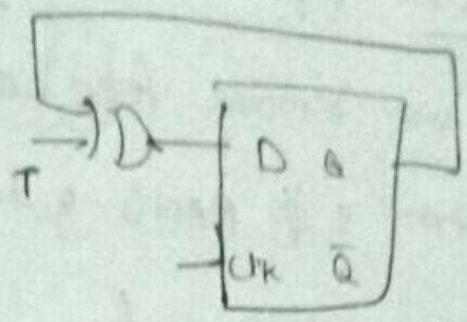


TFF using DFF

T	Q_t	Q_{t+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

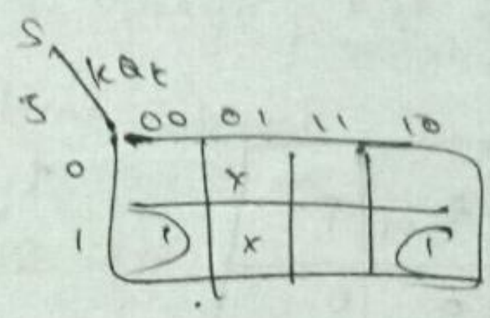
$D = \bar{T}Q_t + T\bar{Q}_t$

$= T \oplus Q_t$

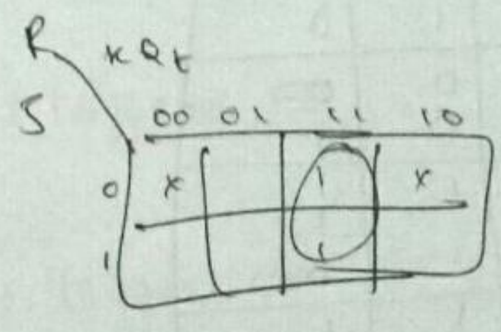


JKFF using SRFF

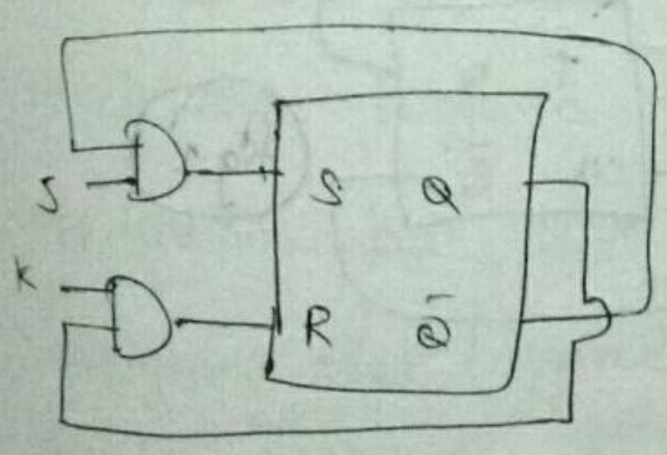
	J K	Q_t	Q_{t+1}	S R
NC	0 0	0	0	0 x
	0 0	1	1	x 0
0	0 1	0	0	0 x
	0 1	1	0	0 1
1	1 0	0	1	1 0
	1 0	1	1	x 0
toggle	1 1	0	1	1 0
	1 1	1	0	0 1

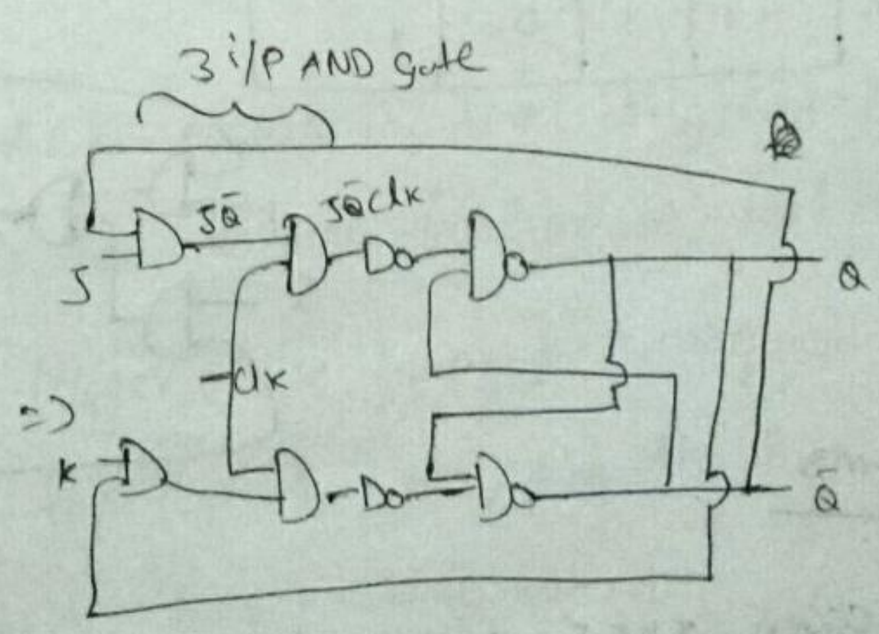


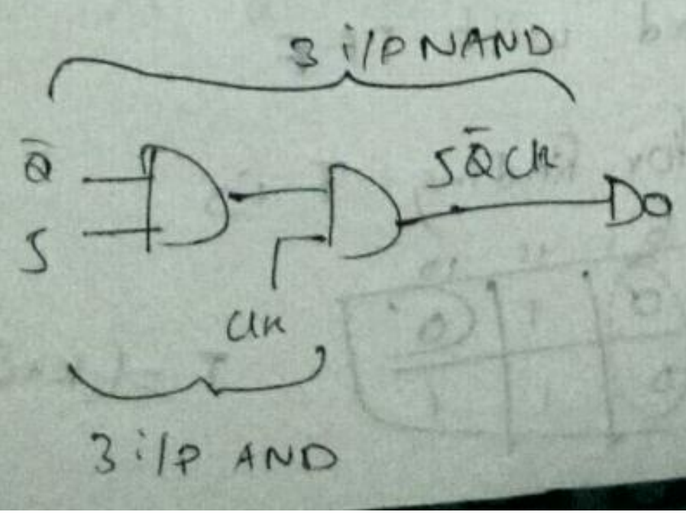
$S = J\bar{Q}$

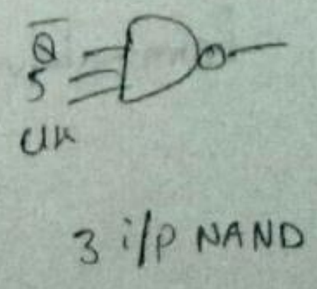


$R = KQ$



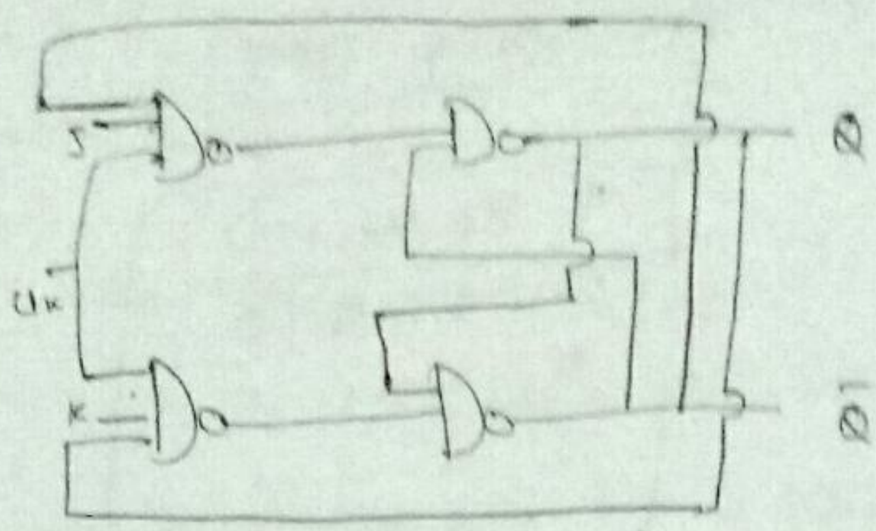






Scanned by TapScanner

⇒



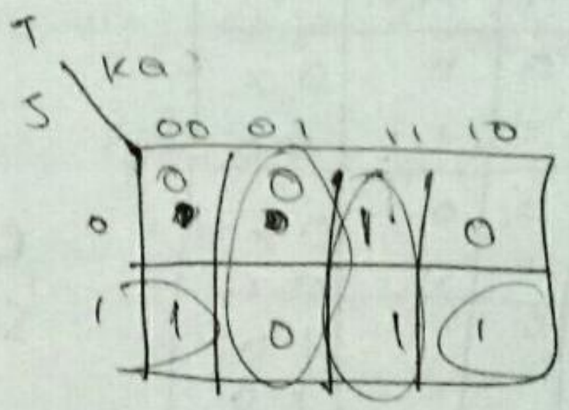
From

⇒ ~~Ex~~ ~~are~~ above diagram we can conclude

* JKFF requires two 3 i/p NAND & two 2 i/p NAND gates
* If we remove feedbacks for 3 i/p NAND gates and ~~make~~ replace with ~~then~~ 2 i/p NAND gates then it turns into SRFF.
* SRFF requires four 2 i/p NAND gates.

JKFF using TFF

	J K	Q_t	Q_{t+1}	T
NC	0 0	0	0	0
	0 0	1	1	0
0	0 1	0	0	0
	0 1	1	0	1
1	1 0	0	1	1
	1 0	1	1	0
Toggle	1 1	0	1	1
	1 1	1	0	1



$KQ + J\bar{Q}$

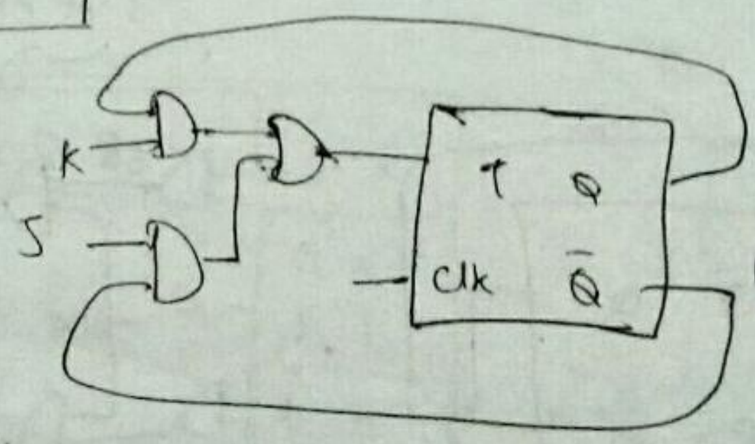
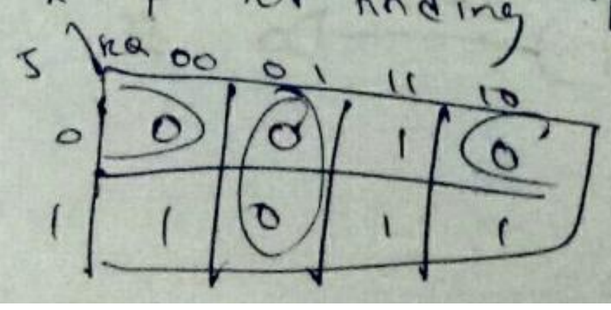


fig 1

Problems

3) Given JKFF implemented with TFF

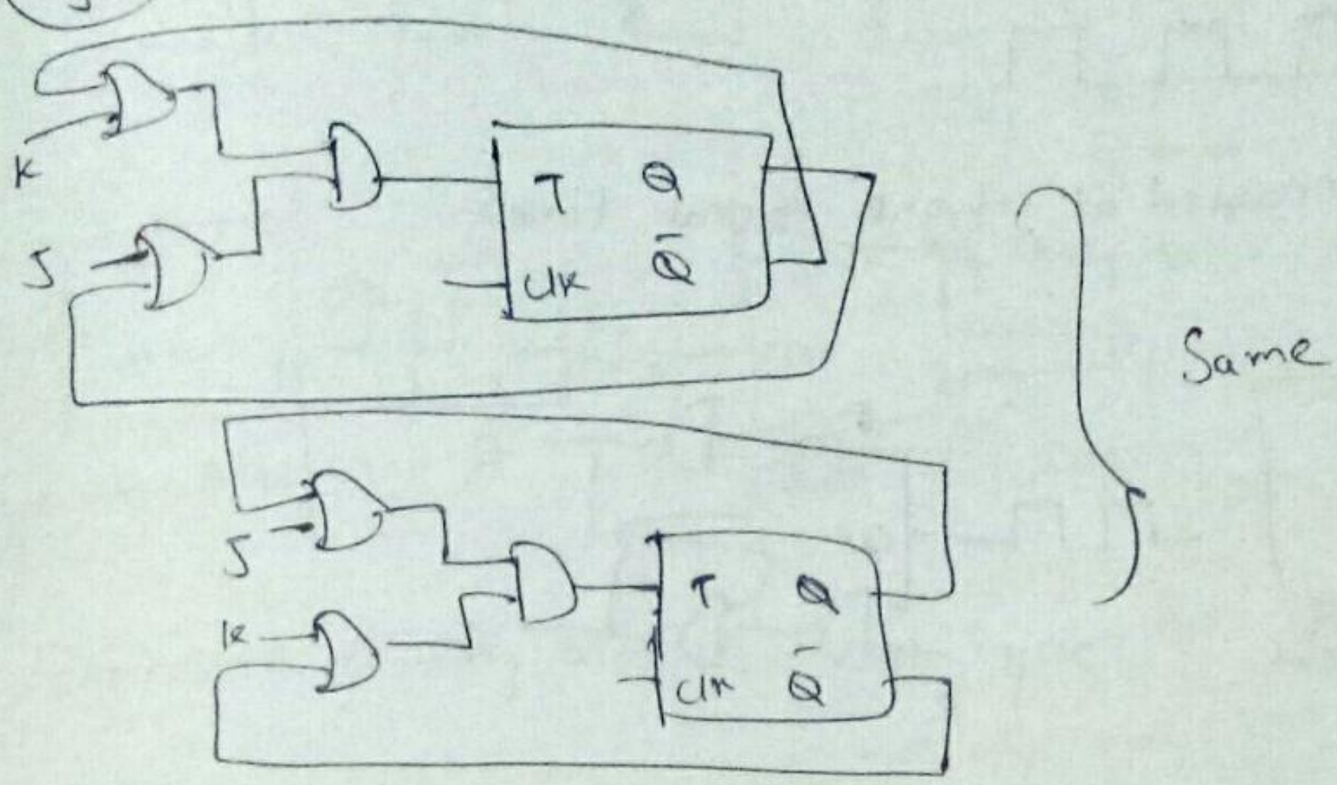
from above k-map for finding T is



$T = (K+\bar{Q})(J+Q)$

Scanned by TapScanner

Here by finding SOP expression none of options matches to (fig 1), So we find P.O.S



Flip Flop Triggering:

→ Triggering is also known as clocking.

→ Triggering is the process of applying high signal on clock terminal at our desired time interval.

→ Different trigger techniques are

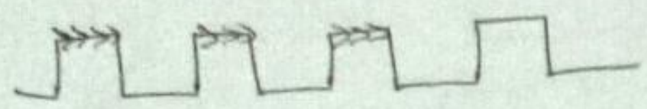
(i) +ve level } level triggered FF
(ii) -ve level

(iii) +ve edge } edge triggerd FF
(iv) -ve edge

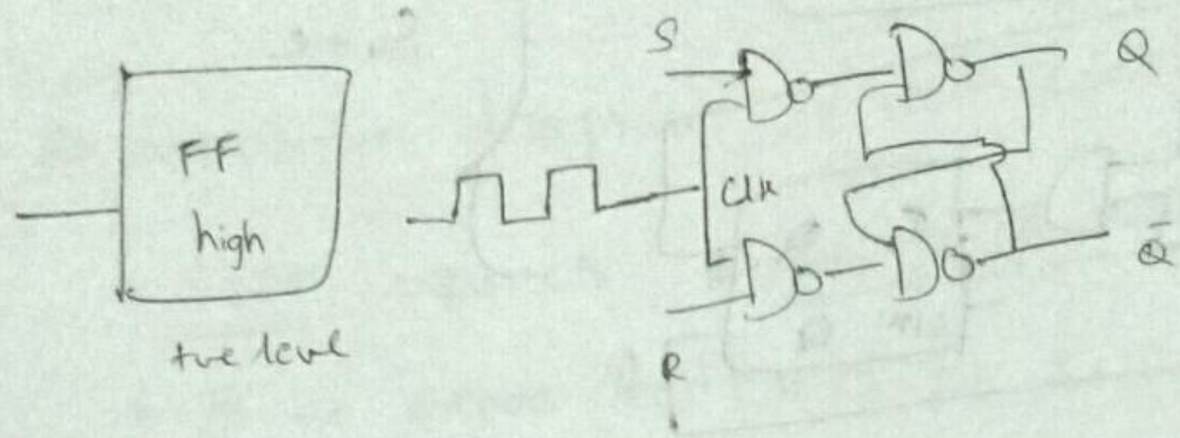
→ Race Around Condition (RAC) occurs in level triggered flip flops. Hence edge triggered flip flop become familiar in real time applications, but schmitt trigger (Pulse trainer) is required for designing edge triggered flip flop for providing the high signal to clock terminal at user's target time interval.

Scanned by TapScanner

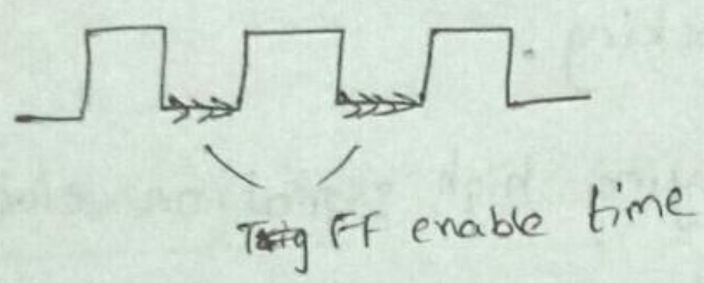
+ve level trigger

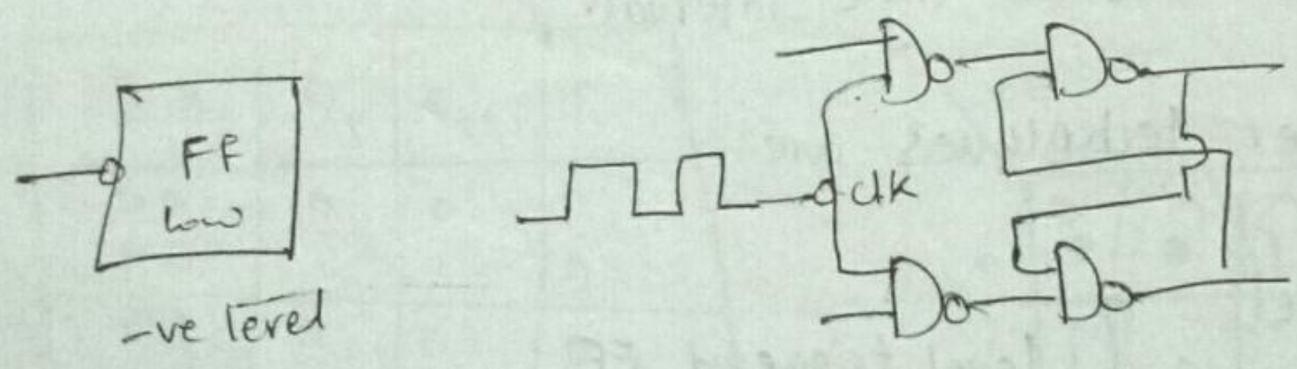


→ The FF is enabled at high signal times

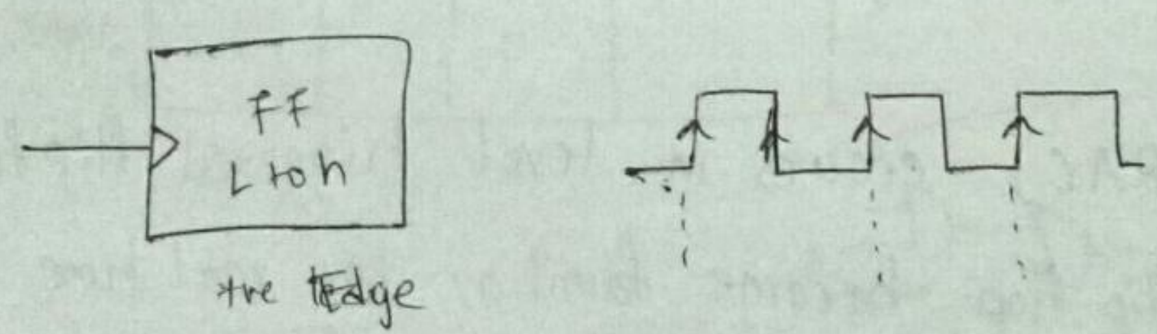


-ve level trigger

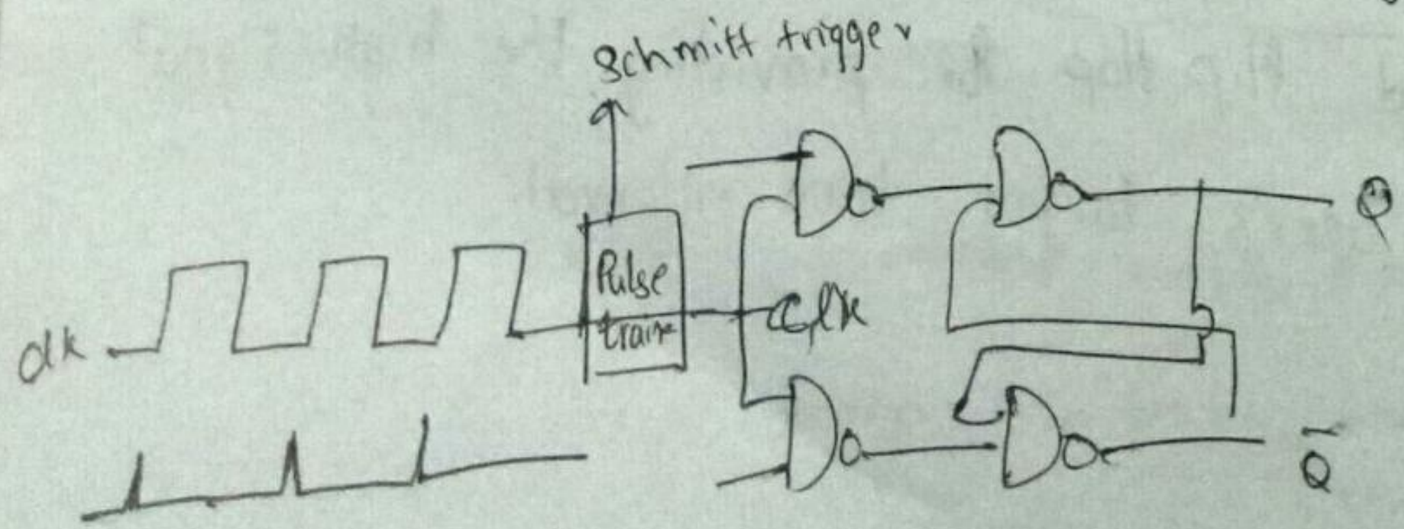




+ve edge trigger

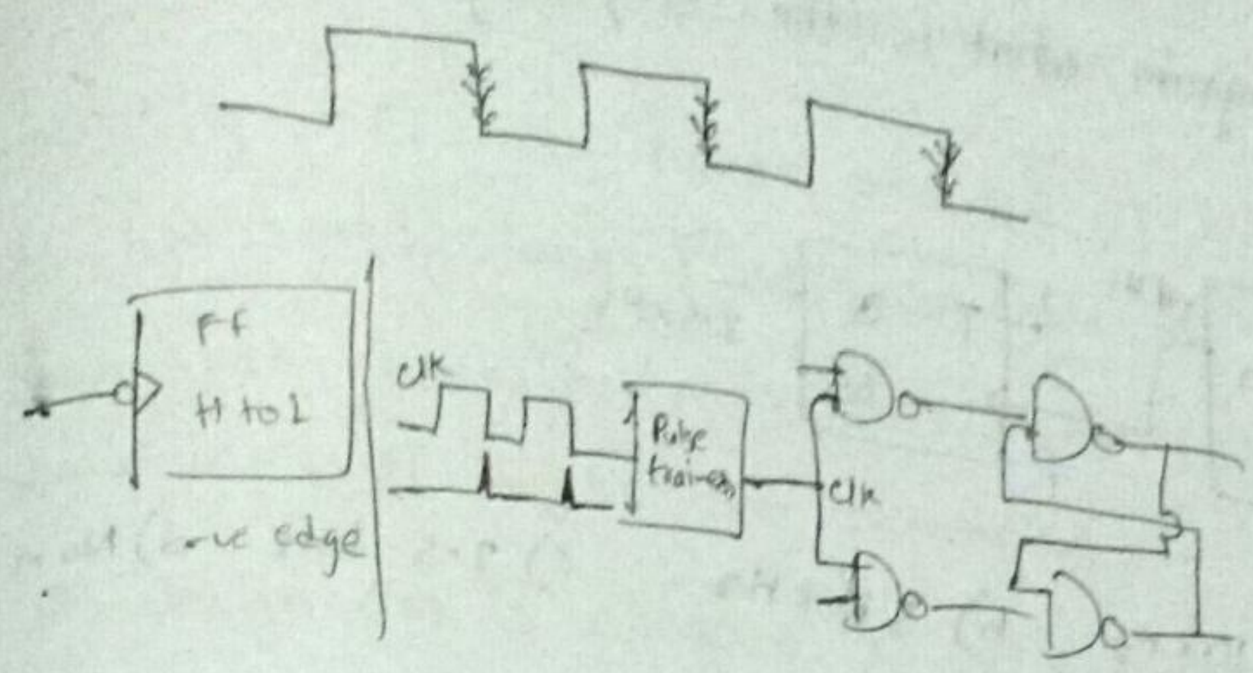


→ FF is enabled exactly once for each cycle for short time.



Scanned by TapScanner

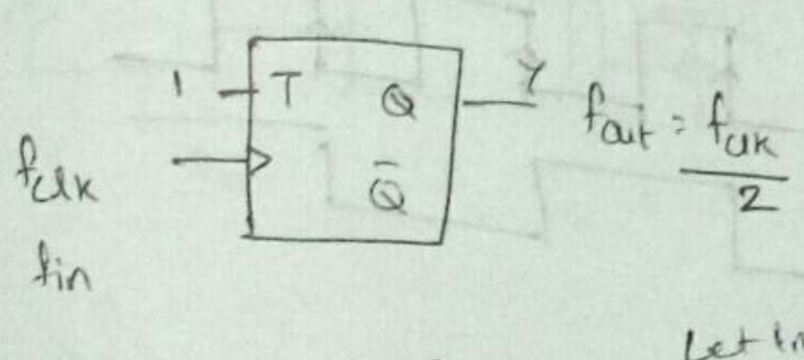
-ve edge trigger



→ FF is enabled exactly once for each cycle for short time

TFF:

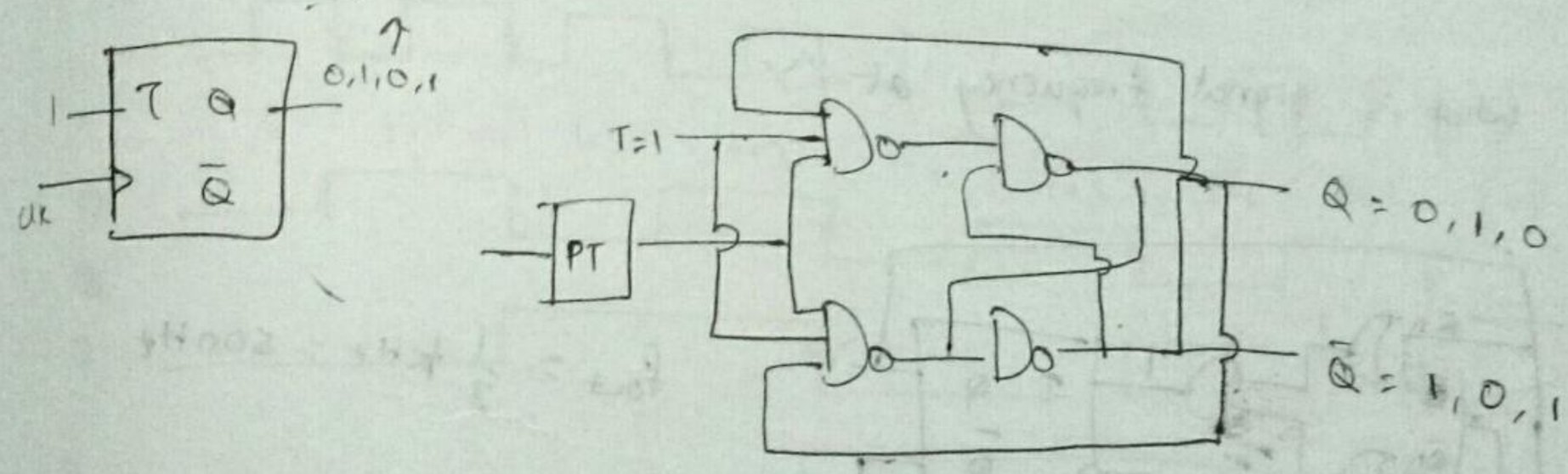
It is also known as frequency divider and mod 2 counter when T=1

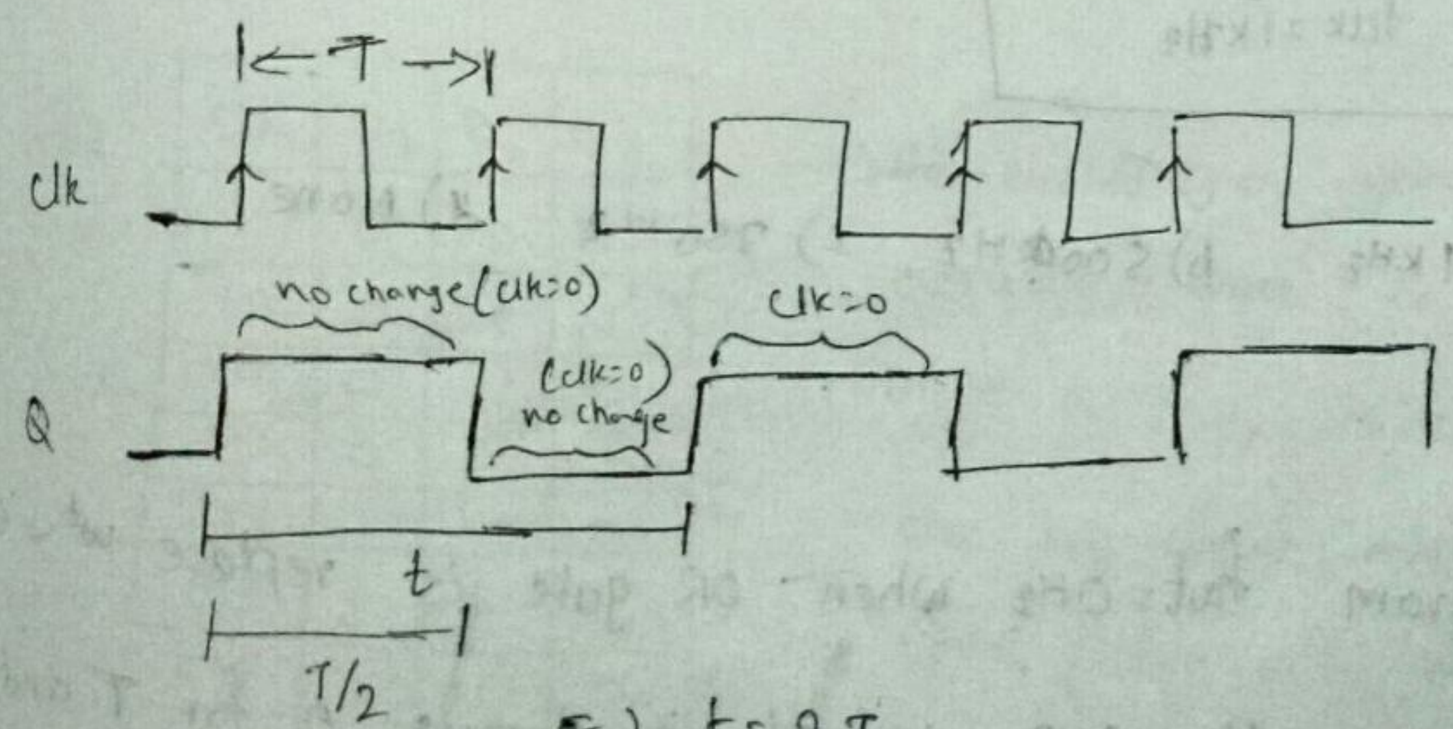


$f_{out} = \frac{f_{clk}}{2}$

Let initial value of Q be 0

∴ acts as counter



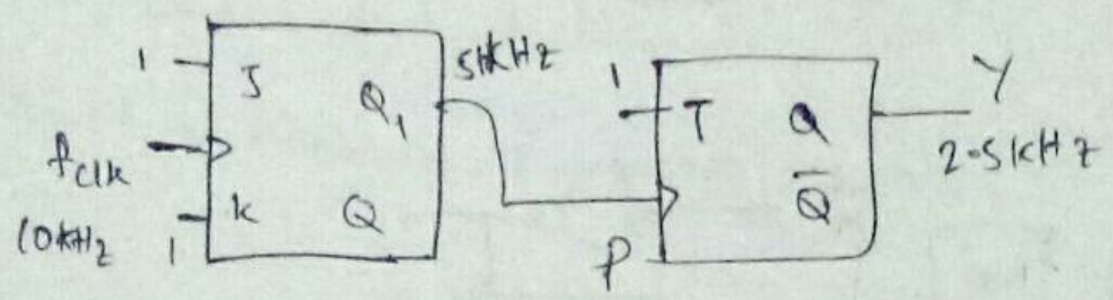


⇒ $t = 2T$

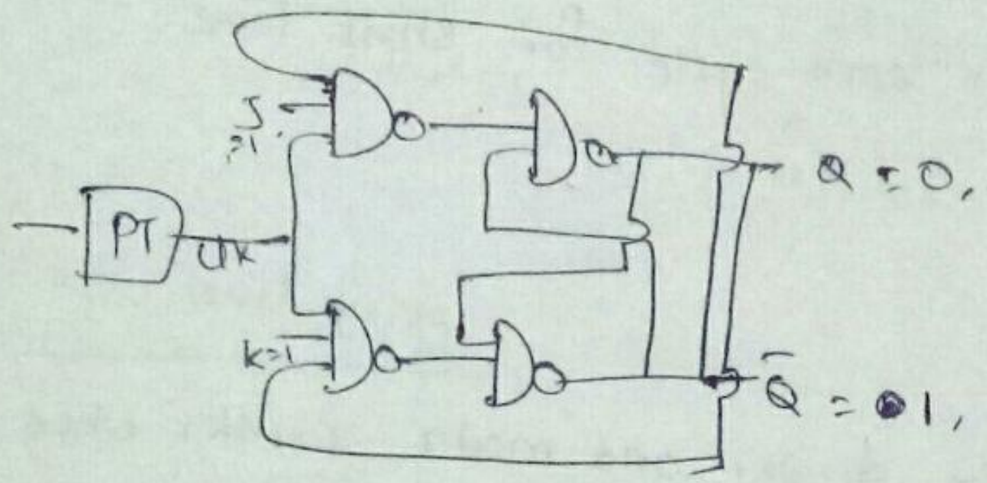
⇒ $f_{out} = f_{in}/2$

Scanned by TapScanner

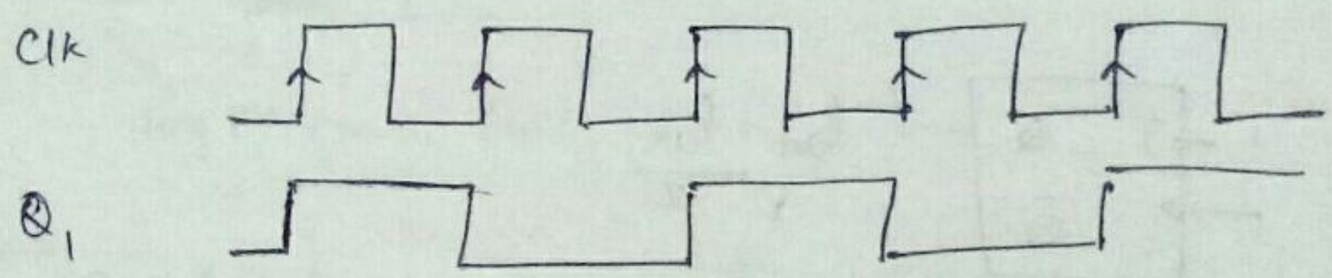
Q: In below diagram what is the frequency at Y?



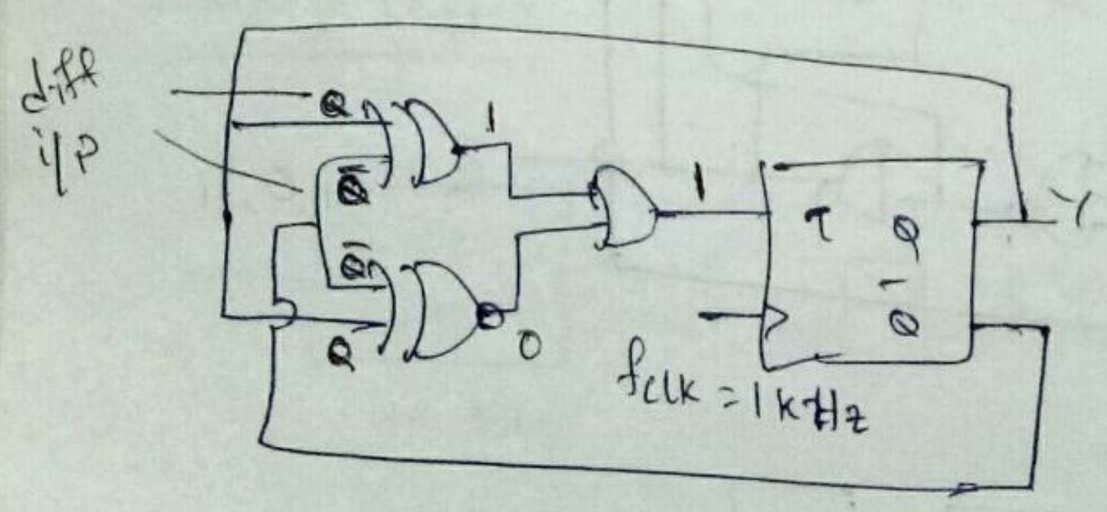
a) 10kHz b) 5kHz c) 2.5 kHz ✓ d) None



J	k	Q_t	$\bar{Q}_t$	Q_{t+1}	$\bar{Q}_{t+1}$
—	—	0	1		
1	1	0	1	1	0
1	1	1	0		



Q: What is signal frequency at Y



$f_{out} = \frac{1}{2}$ kHz = 500Hz

a) 1kHz b) 500Hz ✓ c) 250Hz d) None

Note:

In above diagram f_{out} = 0Hz when OR gate is replace with AND gate. Because the AND gate o/p is always 0 for T and TFF produces no change for '0' i/p. Thus the o/p is a straight line (high or low).

Scanned by TapScanner

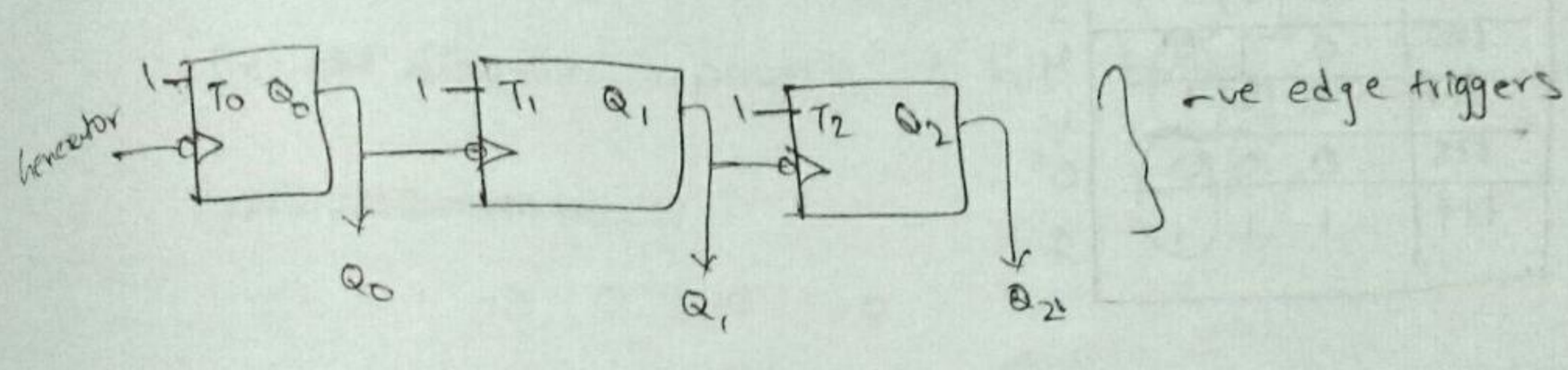
Design of Ripple Counters (Asynchronous Counter):

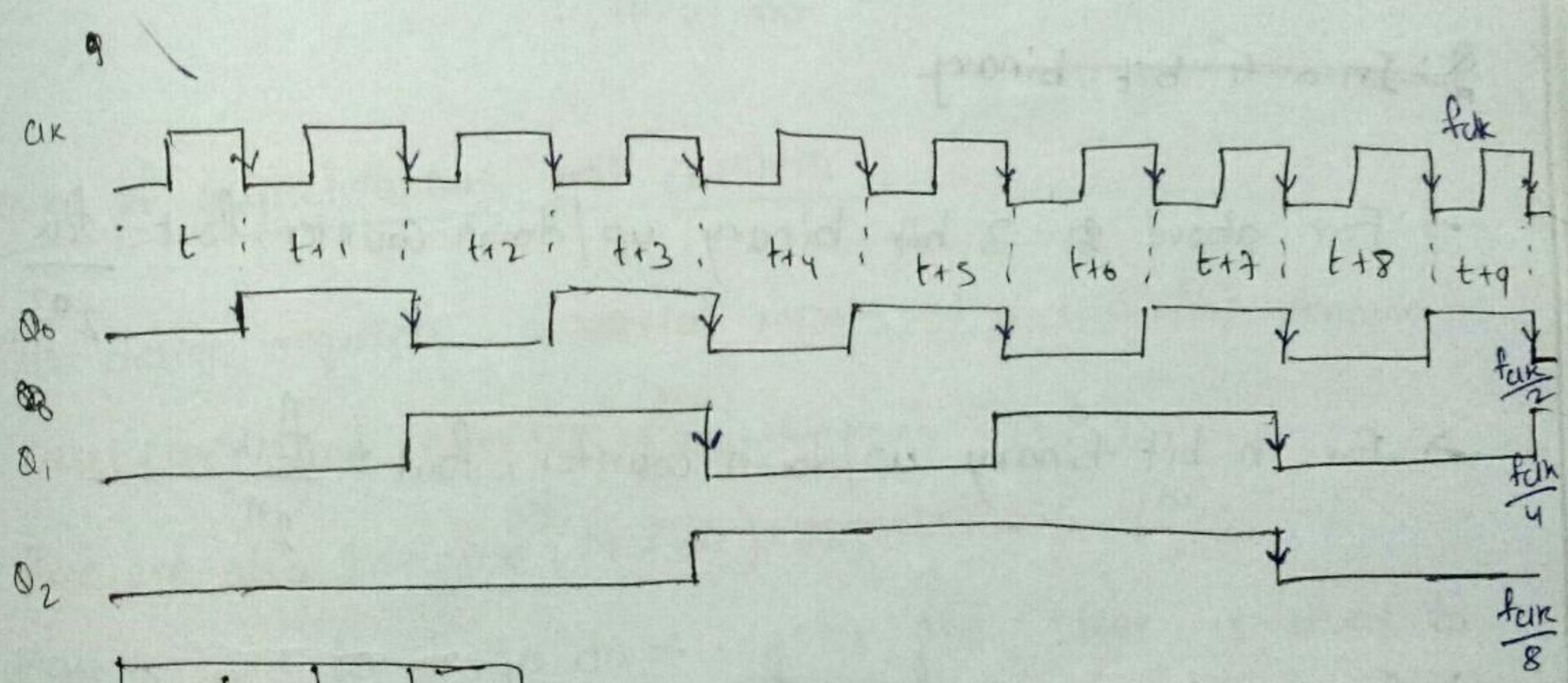
→ Generally TFF are taken and all T i/ps are connected to high signal always, excitation table is not required.

→ Main clock generator is connected to LSB FF only and remainining all clocks are connected in ripple format.

→ Its design cost is cheaper. It is especially used for frequency division application.

3 bit up Counter





Clk	Q_2	Q_1	Q_0	
t	0	0	0	0
t+1	0	0	1	1
t+2	0	1	0	2
t+3	0	1	1	3
t+4	1	0	0	4
t+5	1	0	1	5
t+6	1	1	0	6
t+7	1	1	1	7
t+8	0	0	0	0
t+9	0	0	1	1

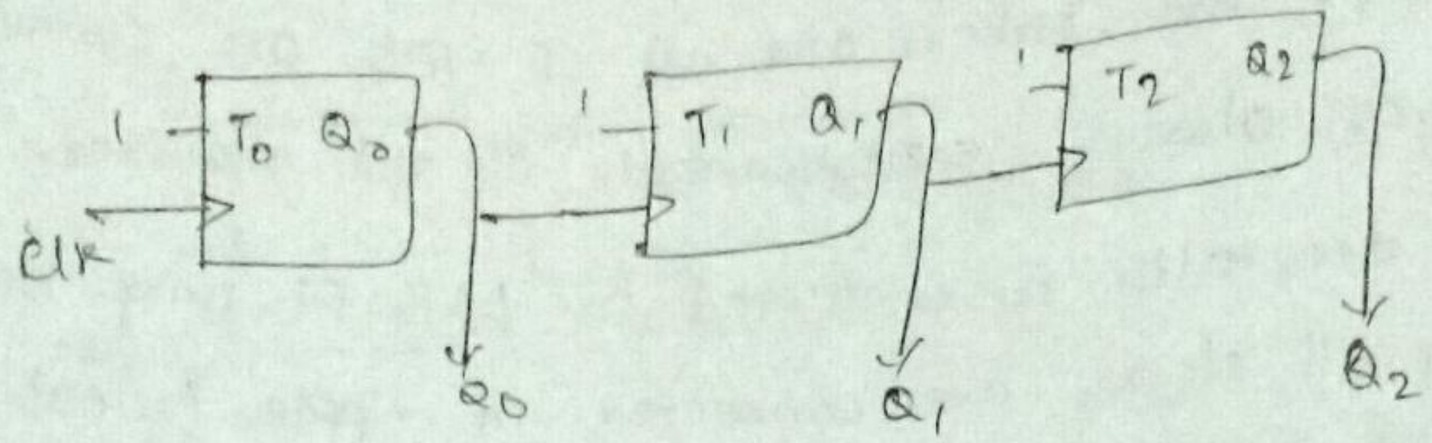
→ The circled ones represent where toggle occured (i.e, high to low).

→ Once a toggle is occured we change the bit value (inverse) and continue same value until next toggle occurs

The above 2 points help in designing the table without drawing all waves. A shortcut method.

Scanned by TapScanner

3 bit down counter



clk	Q_2	Q_1	Q_0	
t	0	0	0	0
t+1	1	1	1	7
t+2	1	1	0	6
t+3	1	0	1	5
t+4	1	0	0	4
t+5	0	1	1	3
t+6	0	1	0	2
t+7	0	0	1	1
t+8	0	0	0	0
t+9	1	1	1	7

Note

+ve edge TFF - down counter

-ve edge TFF - up counter

~~Q: In a 4 bit binary~~

→ For above 3 bit binary up/down counter, $f_{out} = \frac{f_{clk}}{2^3}$

→ For n bit binary up/down counter, $f_{out} = \frac{f_{clk}}{2^n}$

→ For mod k counter $f_{out} = \frac{f_{clk}}{k}$

Q: In a 4 bit binary counter the present state is 0110, what is o/p after 18 clocks

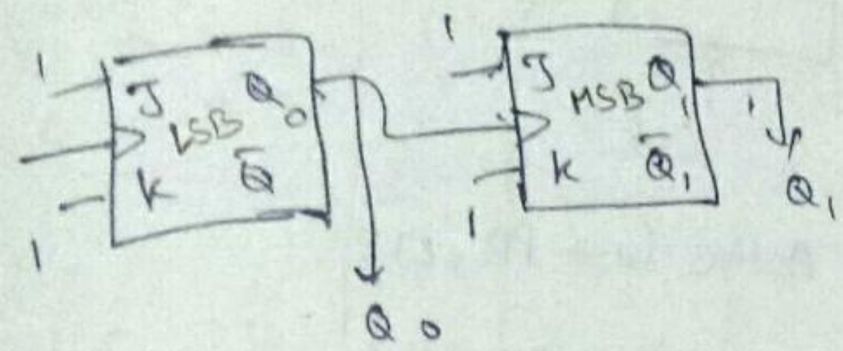
a) 0110 b) 0111 c) 1111 d) 1000 ✓

→ For every 16 clocks (2^4) value will be repeated

$\therefore 18 \% 16 = 2 \Rightarrow 0110 + (10)_2 = 1000$

Scanned by TapScanner

(2011) Q: In the below diagram, initial o/p at Q_1, Q_0 is '00' what is o/p for next 4 clocks?



Q_1	Q_0	
0	0	→ initial (given)
1	1	→ +ve edge
1	0	⇒ toggle is done when low to high
0	1	
0	0	

Ans

a) 00, 01, 10, 11

b) 11, 10, 01, 00 ✓

c) 10, 11, 01, 00

d) None

In the above diagram $J = K = 1$ i.e., TFF with $T = 1$

∴ The ~~the~~ diagram represents 2 bit binary down counter

~~i.e., 00, 01, 10, 11~~

⇒ 0, 3, 2, 1, 0

11, 10, 01, 00

Design of asynchronous mod counters :

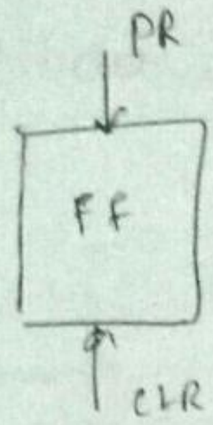
→ This design requires 2 special inputs of a flip flop known as preset (pr) and clear (clr) (asynchronous inputs).

→ These are also known as (pr & clr) asynchronous i/ps.

→ Preset is used to design down counter and clear is used to design up counter.

→ During normal operations these i/ps should be in disabled condition because if preset is enabled o/p filpflop will be forced to high and if clear is enabled the o/p the flipflop will be forced to low immediately regardless of flipflop data i/ps & clock

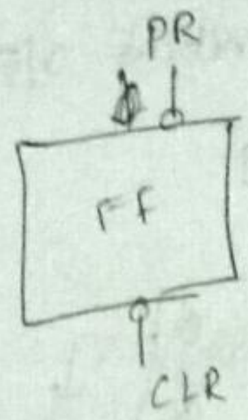
Scanned by TapScanner



Active high

'0'-disabled

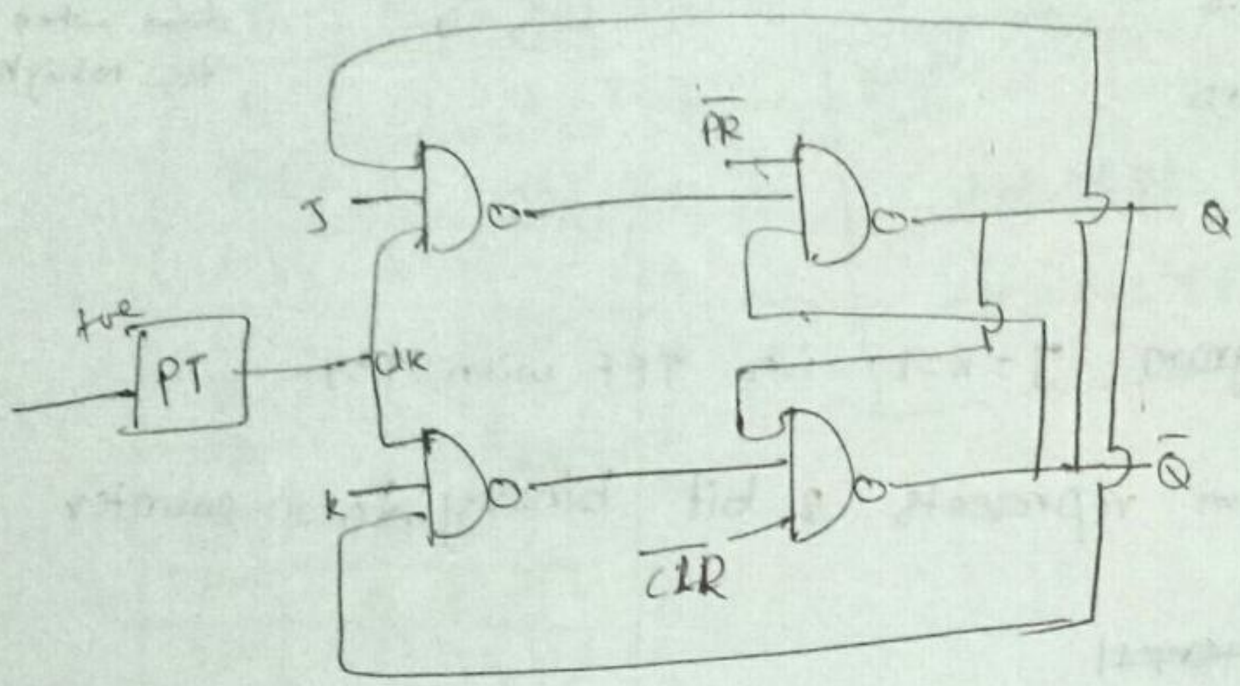
'1'-enabled



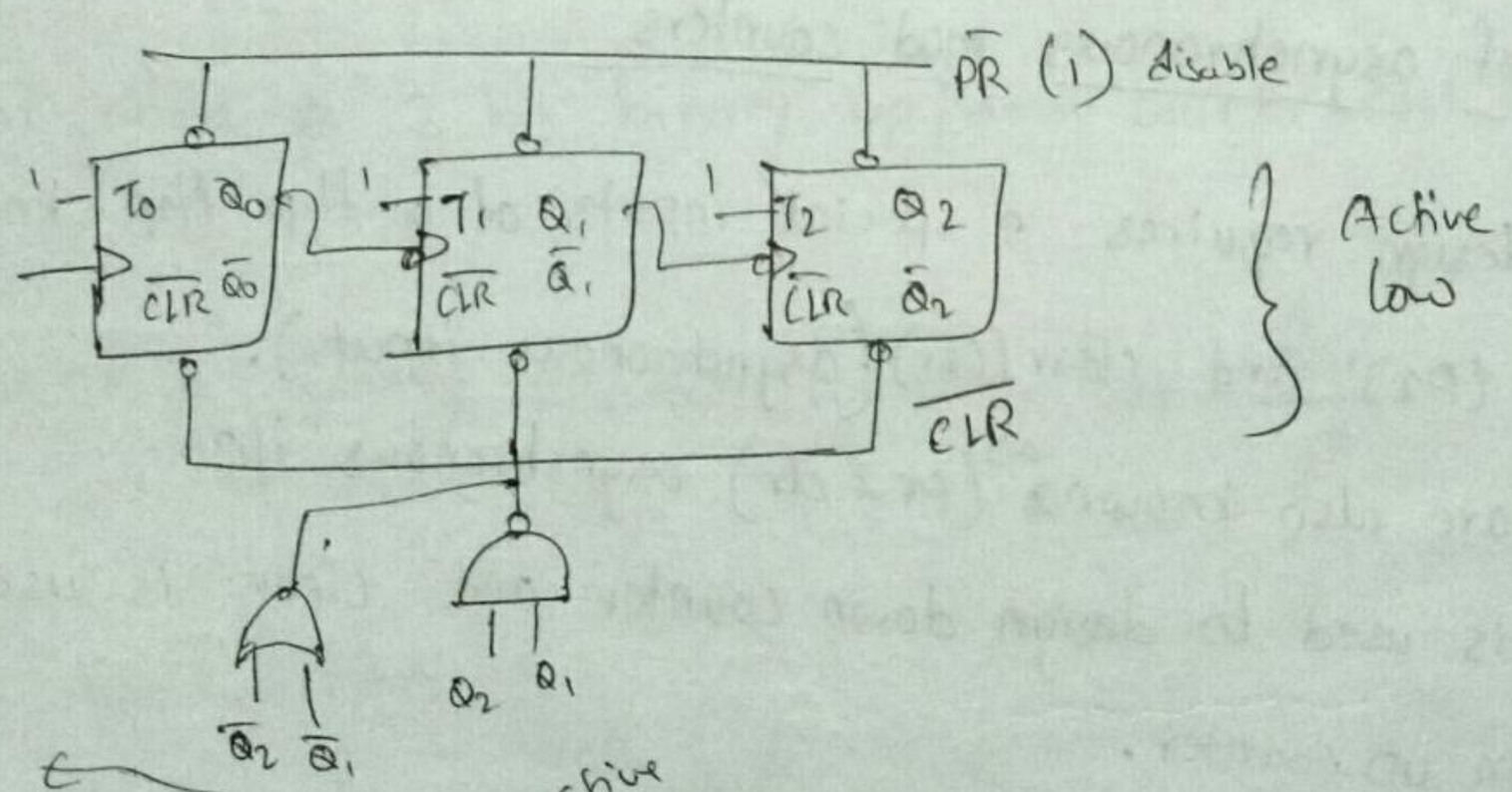
Active low PR, CLR

'0'-enabled

'1'-disabled



Mod 6 upcounter



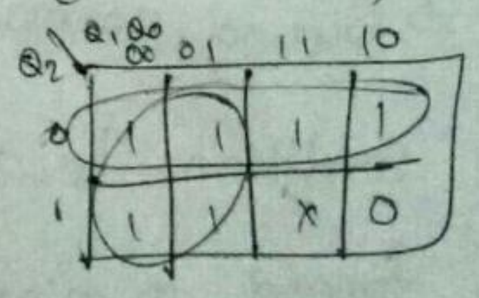
active high

active high

$Q_2 Q_1 Q_0$	$\overline{CLR}$	CLR
000	1	0
001	1	0
010	1	0
011	1	0
100	1	0
101	1	0
110	0	1
111	X	X

$\overline{CLR}(Q_2Q_1Q_0) = \Sigma m(0,1,2,3,4,5) + d(7)$

$= \overline{Q_2 Q_1}$



Silly

$CLR(Q_2Q_1Q_0) = Q_2Q_1$

Scanned by TapScanner

Mod 13 up counter

Q_3	Q_2	Q_1	Q_0	$\overline{CLR}$ (active low)	CLR (active high)
0	0	0	0	1	0
0	0	0	1	1	0
⋮				⋮	⋮
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	x	x
1	1	1	1	x	x

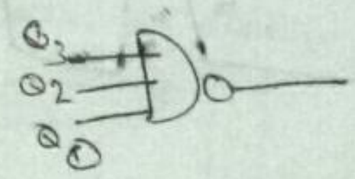
$\overline{CLR}$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	0	x	x
10	1	1	1	1

} POS

$$\overline{CLR} = \overline{Q_3} + \overline{Q_2} + \overline{Q_0}$$

$$= \overline{Q_3 Q_2 Q_0}$$



Sily $CLR = Q_3 Q_2 Q_0$

Mod 83 upcounter

→ 0 to 82 accept

$83 = (1010011)_2$
$Q_6\, Q_5\, Q_4\, Q_3\, Q_2\, Q_1\, Q_0$

Here we have to stop counting at 83

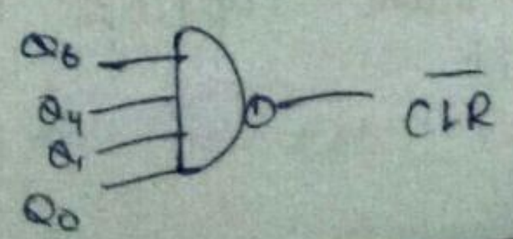
∴ 83's binary value have 4 1's. So we need four i/p NAND gate for which i/ps are 1's digit place in 83's binary representation

→ No of FFs required = 7

→ Name of gate is NAND (OR can also be used by taking inputs for $\overline{Q_i}$)

→ No of i/ps of NAND (or) OR gate = 4

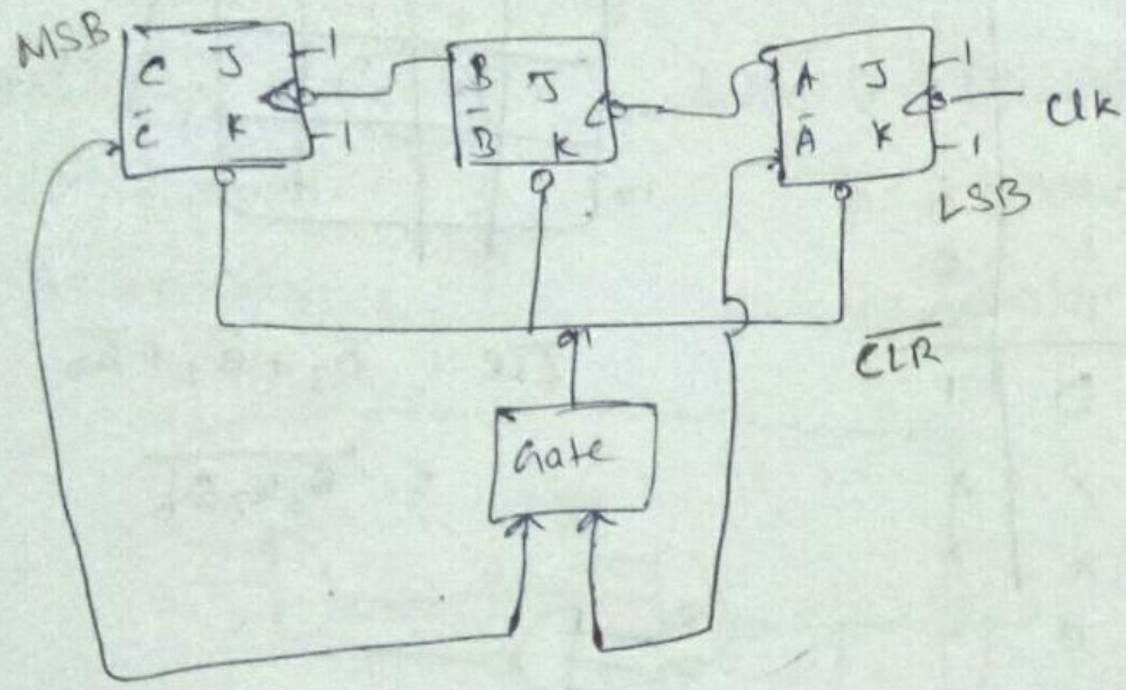
→ Gate is



Scanned by TapScanner

Q: In the given below mod 5 counter, the name of the gate is ____

a) AND b) OR ✓ c) NAND d) NOR



Since the inputs of gates are complement its is OR gate.

Method II

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

→ for gate to enable clear it should produce o/p 0.

⇒ 0 to 0 i.e. OR gate

Race Around Condition (RAC):

→ When flip flop o/p is toggled more than one time in one clock cycle completion the Race Around Condition occurs.

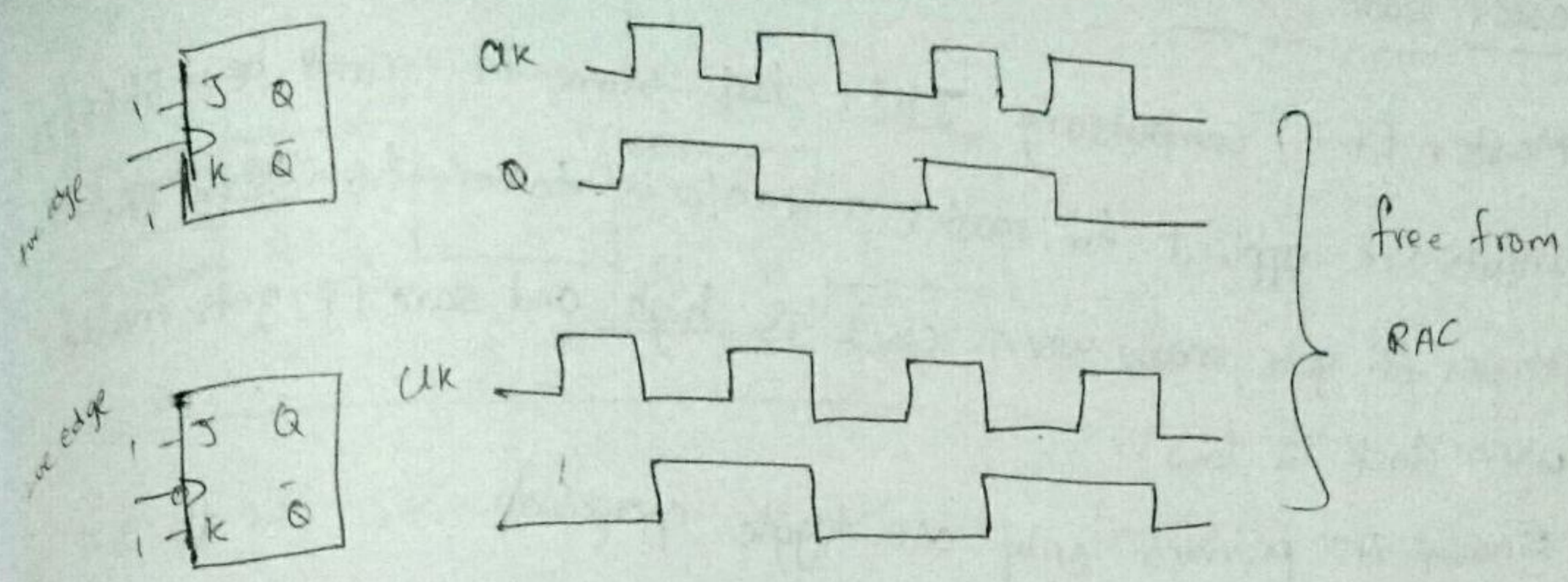
→ SR flip flop and DFF are free from RAC.

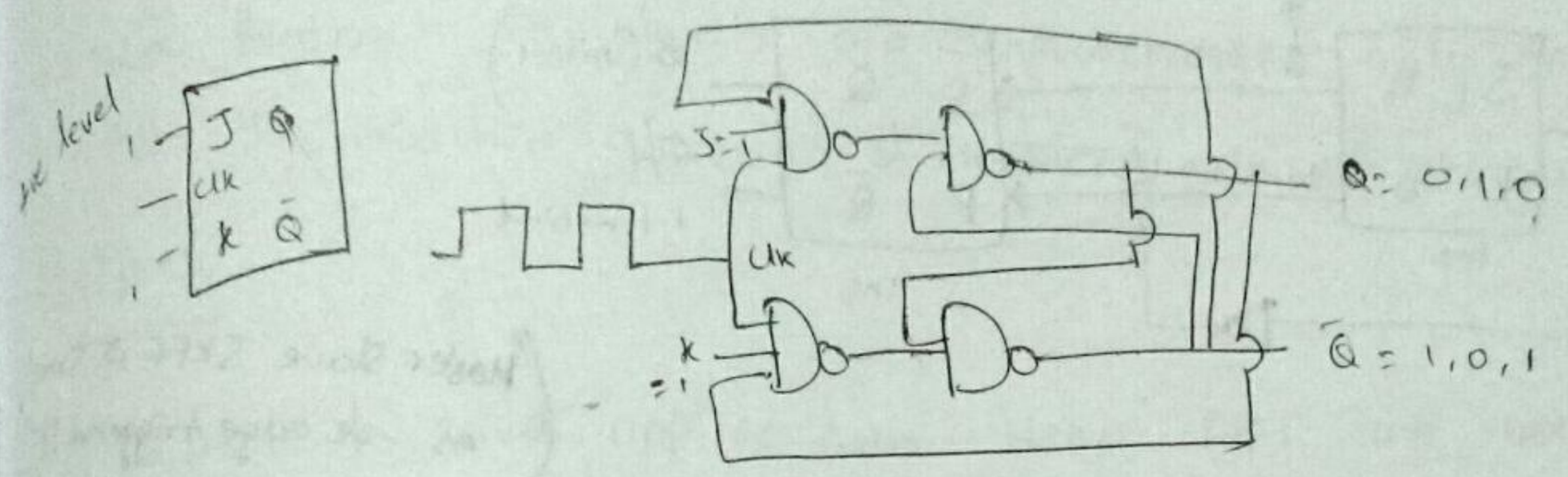
→ Edge triggered JKFF and edge triggered TFF are free from RAC.

★ → RAC occurs only in level triggered JKFF and level triggered TFF.

→ To prevent RAC the master slave flipflop is used.

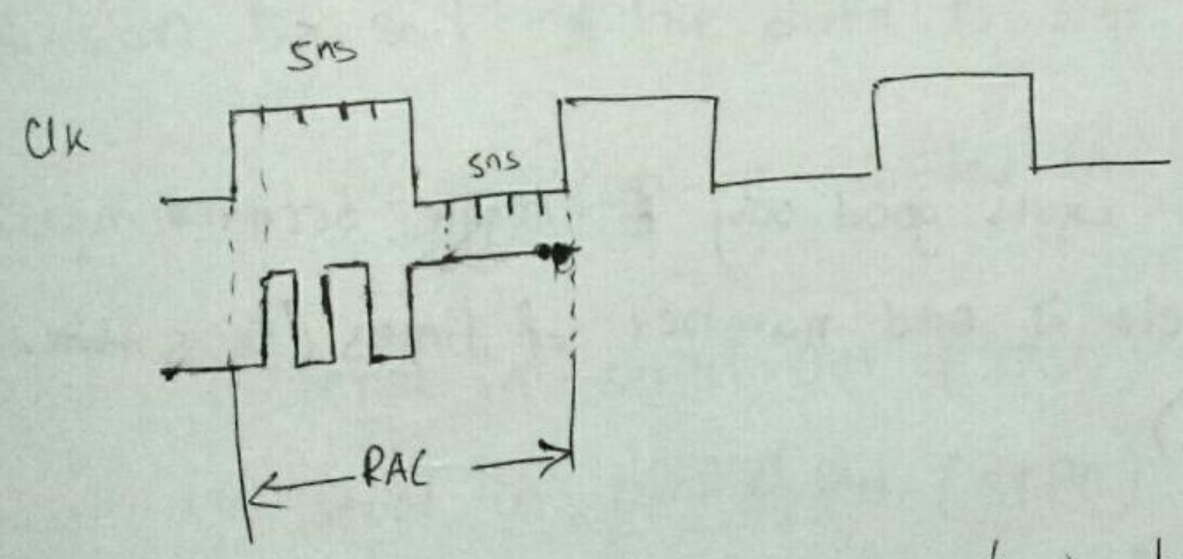
Scanned by TapScanner





$\Delta t = t_{PD} = 1ns$ (i.e, it takes 1ns to produce Q & $\bar{Q}$ for given i/p)

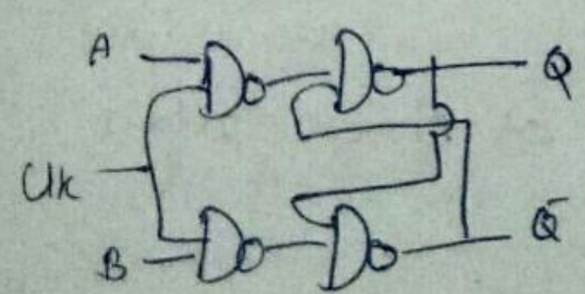
Let frequency of clock be 100 MHz ⇒ T = 10ns



In the above figure the o/p is toggled more than once for given on clock cycle.

2012

Q: In the below diagram, RAC



✓ a) does not occur

b) occurs when A=1, B=1, clk=0

c) occurs when A=0, B=0, clk=0

d) occurs when A=1, B=1, clk=1

This SRFF which is free from RAC

Scanned by TapScanner

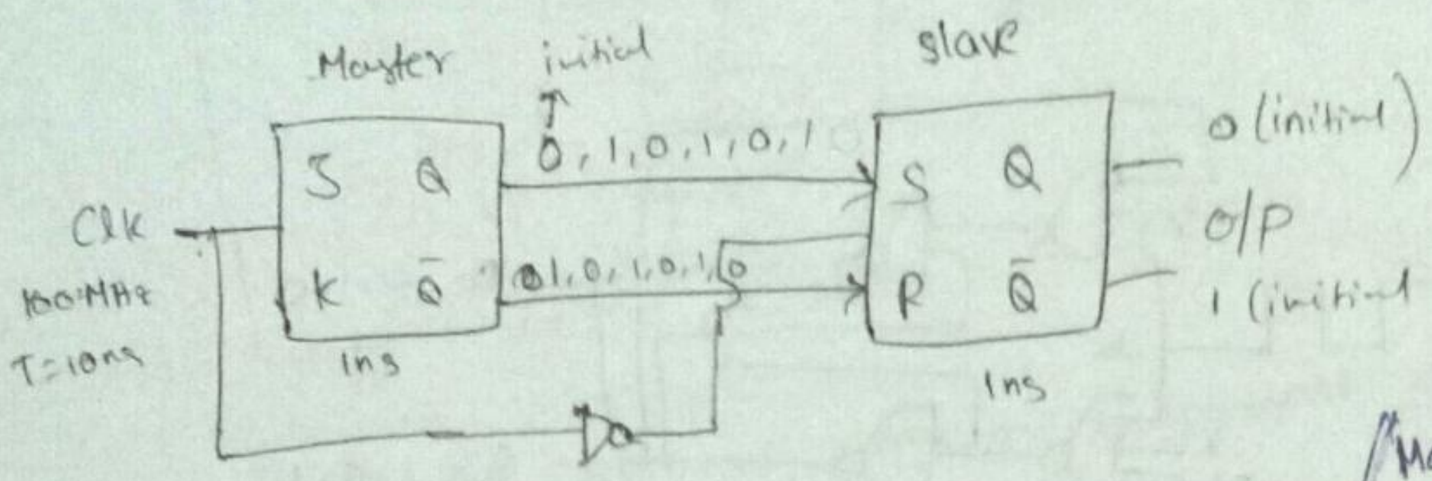
Master slave JKFF

→ Master FF is compulsary JKFF but slave FF can be SRFF/JKFF

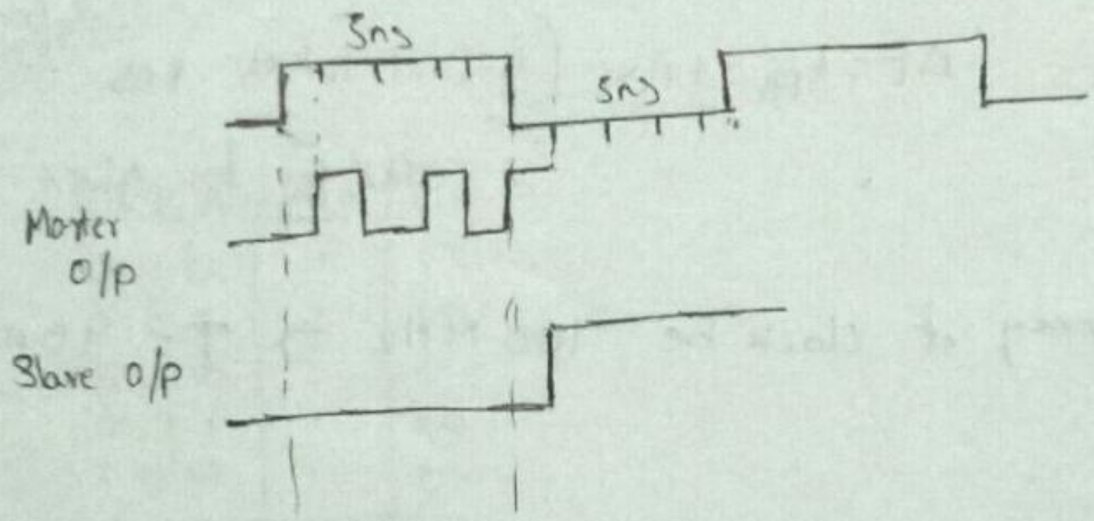
→ Inputs are applied for master and o/p is taken from slave FF.

→ Master FF gets enable when clock is high and slave FF gets enable when clock is low.

→ Finally it provides only one toggle o/p.



(Master slave JKFF is same as -ve edge triggered JKFF)

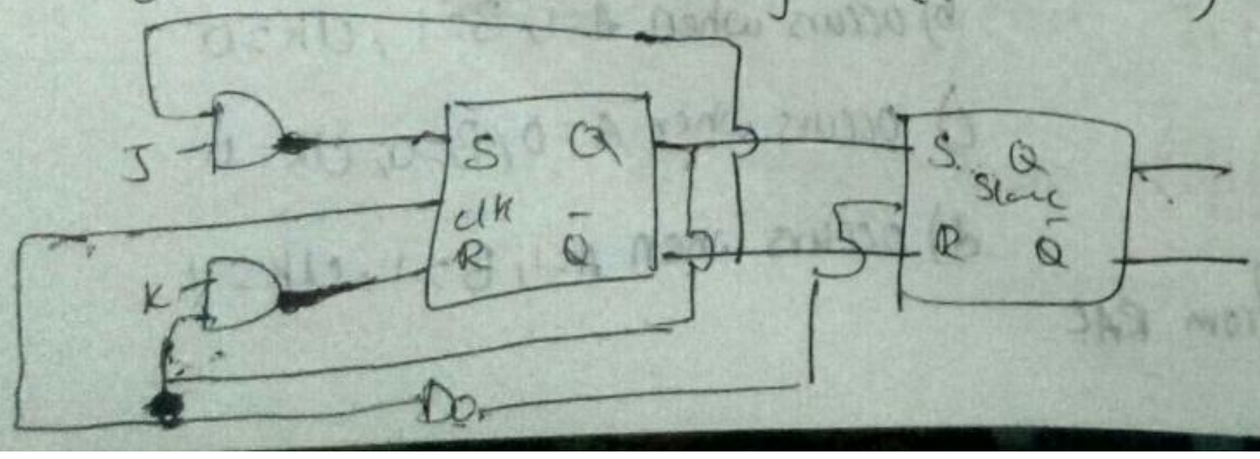


→ The above circuit works good only if toggle occured in master in one cycle is odd number of times (Eg: 5 times in above diagram)

→ If it occurs even number of time the slave o/p is a straight line.

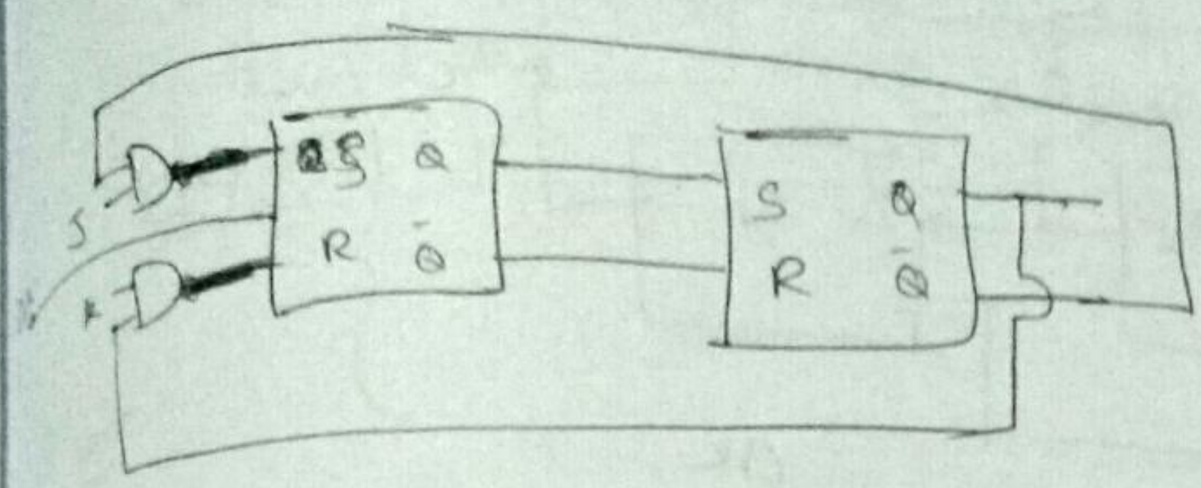
Modified diagram to toggle the master FF o/p only one time during clock high:

Consider previous diagram (the above) as shown below



Scanned by TapScanner

It is modified as



→ The i/p at JK changes only if the feedback to which it is connected change.

→ Here feedback is slave's o/p which changes only once and thus the modified circuit's master's o/p also changes once.

Registers

→ It is used to save applied data. Hence DFF are used for its design

→ n-bit register design requires n number of DFFs

→ Register is also used to perform binary multiplication and binary division by shifting the data to left and right respectively

→ Data in a register can be shifted in four ways.

(i) Serial in serial out (SISO)
(ii) Serial in parallel out (SIPO)
(iii) Parallel in serial out (PISO)
(iv) Parallel in parallel out (PIPO)

→ In most of the real time application serial In parallel Out technique is used.

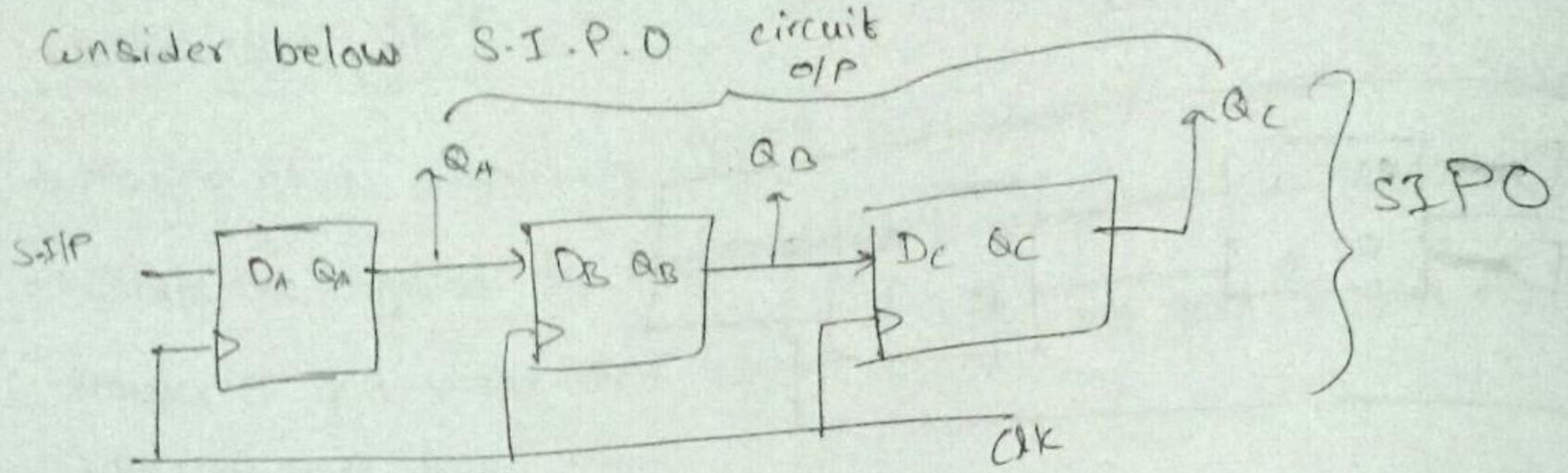
$A_3\ A_2\ A_1\ A_0$

Left shift $A_2\ A_1\ A_0\ 0$ (This mulplies data by 2)

Right shift $0\ A_3\ A_2\ A_1$ (This divides data by 2 i.e, Integer division $(5/2=2)$)

Scanned by TapScanner

Consider below S.I.P.O circuit



$D_A = S.I/P$

$D_B = Q_A$

$D_C = Q_B$

S.I/P	D_A D_B D_C	clk	Q_A Q_B Q_C	
			0 0 0	
1	1 0 0	↑	1 0 0	to shift bit '1'
1	1 1 0	↑	1 1 0	
1	1 1 1	↑	1 1 1	
0	0 1 1	↑	0 1 1	to shift bit '0'
0	0 0 1	↑	0 0 1	
0	0 0 0	↑	0 0 0	

Ring Counter (RC)

→ The SIPO register acts as R.C when last FF O/P is connected to first FF input.

→ During initialization, it requires preset technique (only for one of the flipflops)

→ In n bit R.C the no of useful states is $\underline{n}$ and no of unused states is $\underline{2^n - n}$

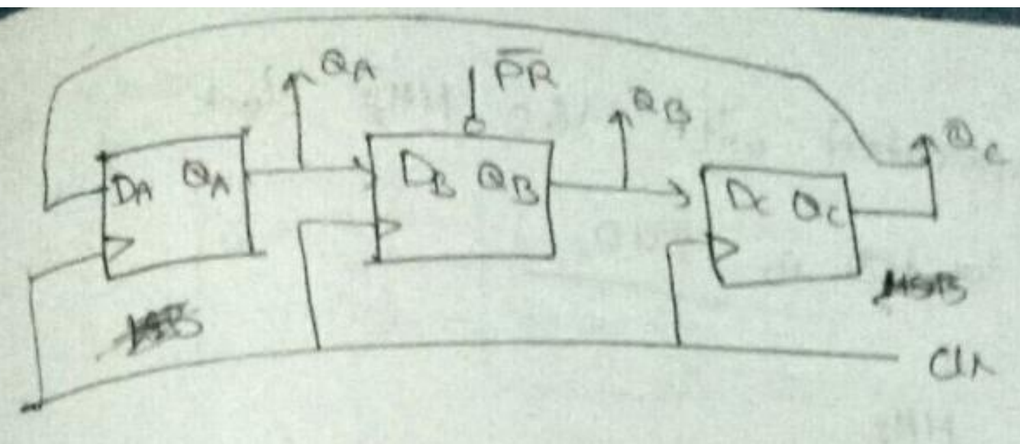
→ n bit R.C is also known as Mod $\underline{n}$ counter and $f_{out} = \frac{f_{in}}{n}$

Eg: A 5 bit RC is operated with 120 MHz Clock

① No of unused states is $\underline{27}$.

② f_{out} is $\underline{24}$ MHz.

Scanned by TapScanner



$D_A = Q_C$
$D_B = Q_A$
$D_C = Q_B$

Note:

whatever the size of ring counter, always only one o/p bit is high ~~active~~ like active high o/p decoder.

D_A D_B D_C	Clk	Q_A Q_B Q_C
initial		0 1 0
0 0 1	↑	0 0 1
1 0 0	↑	1 0 0
0 1 0	↑	0 1 0

→ Preset (because initial state requires ~~atleast~~ exactly one '1' bit in Q_A (or) Q_B (or) Q_C)

3 useful states out of 2^3 states

$2^3 - 3 = 5$ states are unused

Johnson Counter (JC)

→ It is also known as
Twisted R.C
Ringtail Counter and
Moebius counter

→ It does not require preset technique during initialization.

→ n bit JC is also known as mod $\underline{2n}$ counter and fout is $\frac{f_{in}}{2n}$.

→ In n bit JC no of useful states is $\underline{2n}$

and no of unused states is $\underline{2^n - 2n}$

D_A D_B D_C	Clk	Q_A Q_B Q_C
initial		0 0 0
1 0 0	↑	1 0 0
1 1 0	↑	1 1 0
1 1 1	↑	1 1 1
0 1 1	↑	0 1 1
0 0 1	↑	0 0 1
0 0 0	↑	0 0 0

~~$D_A = \overline{D_C}$~~

$D_A = \overline{Q_C}$
$D_B = D_A$
$D_C = D_B$

Scanned by TapScanner

Eg: A 8 bit J.C is operated with 160 MHz clock

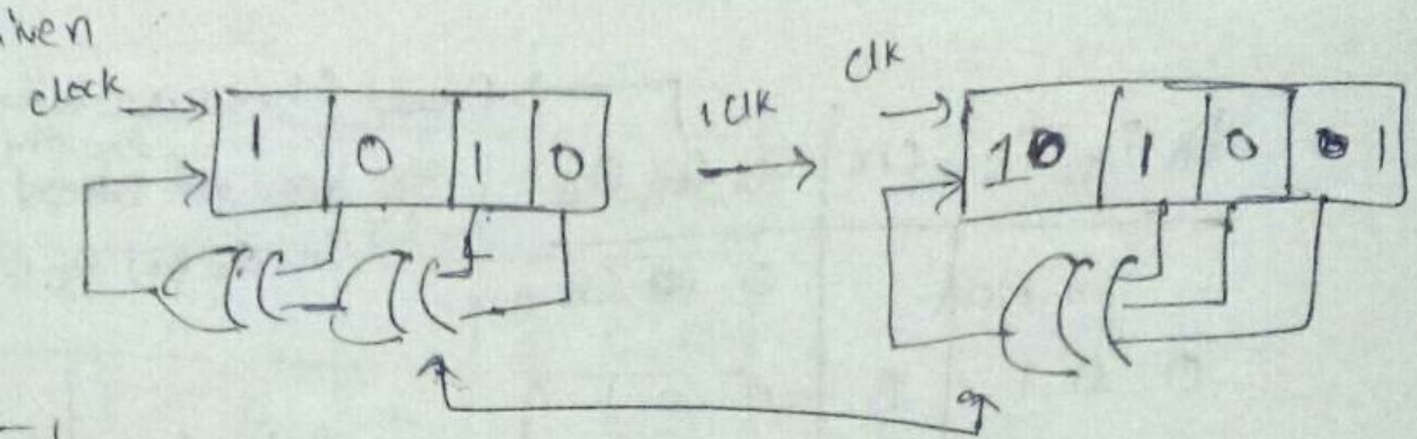
1) No of unused states is 240

2) fout is 10 MHz

Problems

(2)

Given



3 i/p XOR gate

1010
1101 – 1
0110 – 2
0011 – 3
0001 – 4
1000 – 5
0100 – 6
1010 – 7

0101
~~1010~~
1010

∴ 3 ~~clocks~~ are required

∴ 7 clocks

2014 2M

Q: Let $k = 2^n$. A circuit is built by giving the o/p of n-bit binary counter as input to $n \times 2^n$ size decoder. This circuit is equivalent to

a) k bit binary upcounter

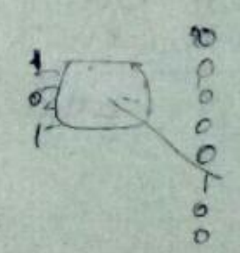
b) k bit binary down counter

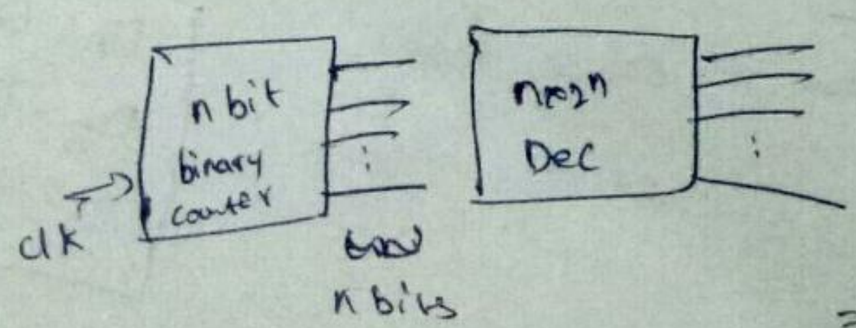
c) k bit ring counter

d) k bit Johnson Counter

2^3

3×8





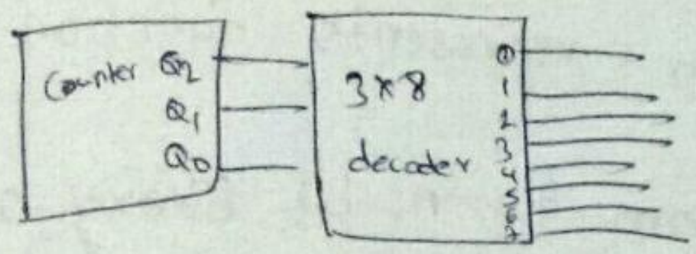
2^n lines

of which only one line is high

$\Rightarrow 2^n$ bit R.C $\Rightarrow$ k bit R.C ($\because 2^n = k$)

Scanned by TapScanner

Let $n=3 \Rightarrow k=8$



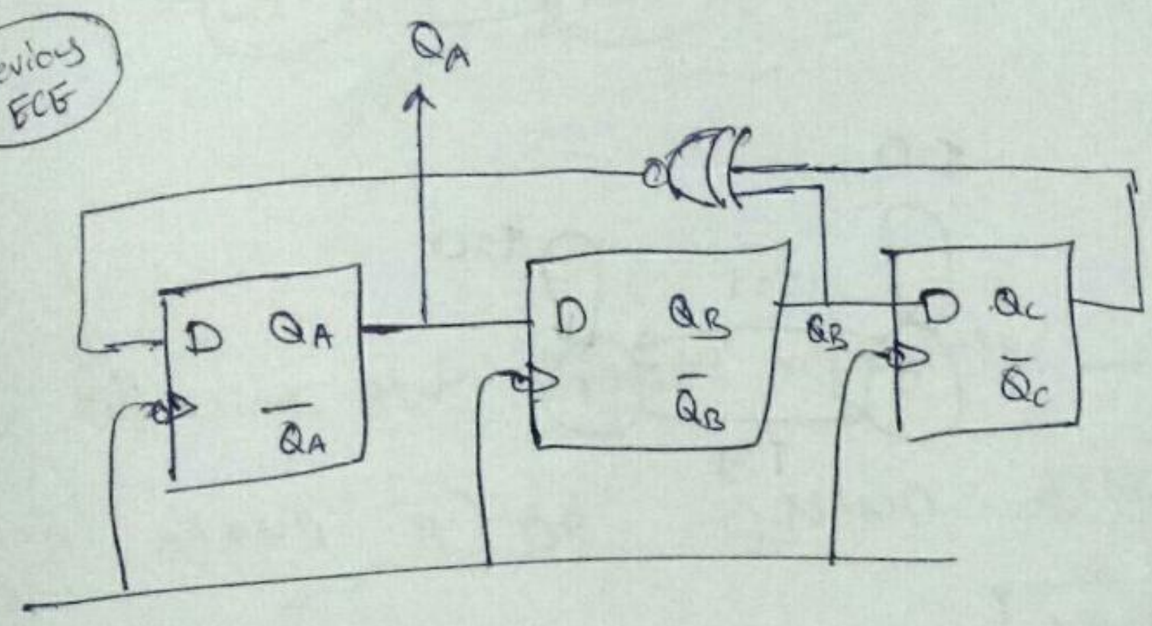
$Q_2\ Q_1\ Q_0$	$D_0\ D_1\ D_2\ D_3\ D_4\ D_5\ D_6\ D_7$
0 0 0	1 0 0 0 0 0 0 0
0 0 1	0 1 0 0 0 0 0 0
⋮	⋮
1 1 1	0 0 0 0 0 0 0 1

∴ This represents 8 bit ~~Johnson~~ Ring Counter

i.e, k bit Ring Counter

Problems

(8) Previous ECE



$D_A D_B D_C$	CLK	Q_A	Q_B	Q_C
Initial		0	0	0
1 0 0	↑	1	0	0
1 1 0	↑	1	1	0
0 1 1	↑	0	1	1
1 0 1	↑	1	0	1
0 1 0	↑	0	1	0

$D_A = \cancel{Q_B Q_C + 0}\ Q_B \odot Q_C$

$D_B = Q_A$

$D_C = Q_B$

Q_A is 0 1 1 0 1 0 ....

State table & State diagrams:

→ State table is also known as state transition table and it is ~~cal~~ closer to excitation table.

→ State diagram is the collection of circles. One circle represen one state, circle internal content is used to represent a state.

$2^n \geq k$)

Scanned by TapScanner

→ At the external side of the circle the numerator represents data i/p and denominator represents function o/p.

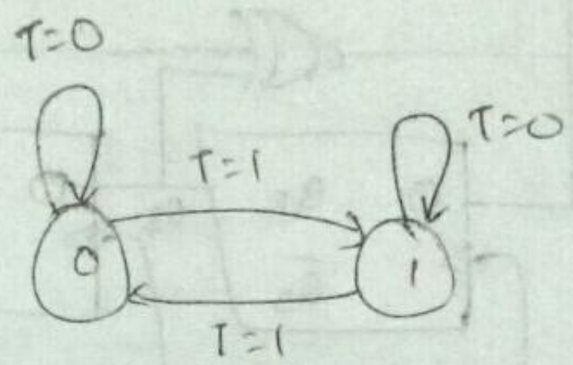
→ To draw the state diagram for n bit binary counter we require 2^n circles.

~~3 bit binary up counter state diagram~~:

State diagrams for Flip Flops:

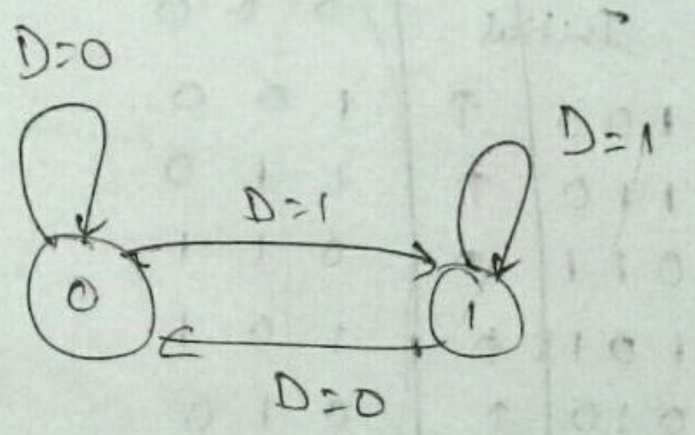
TFF

Q	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0



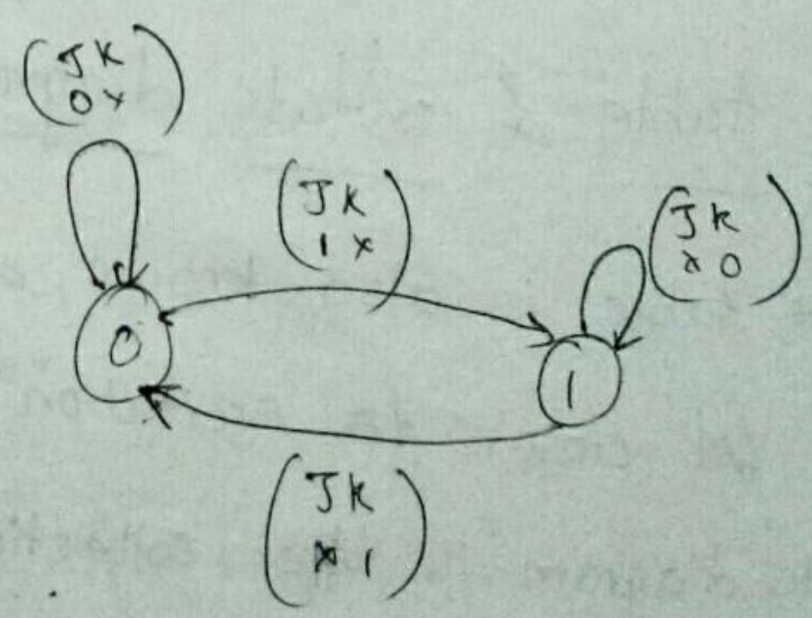
DFF

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1



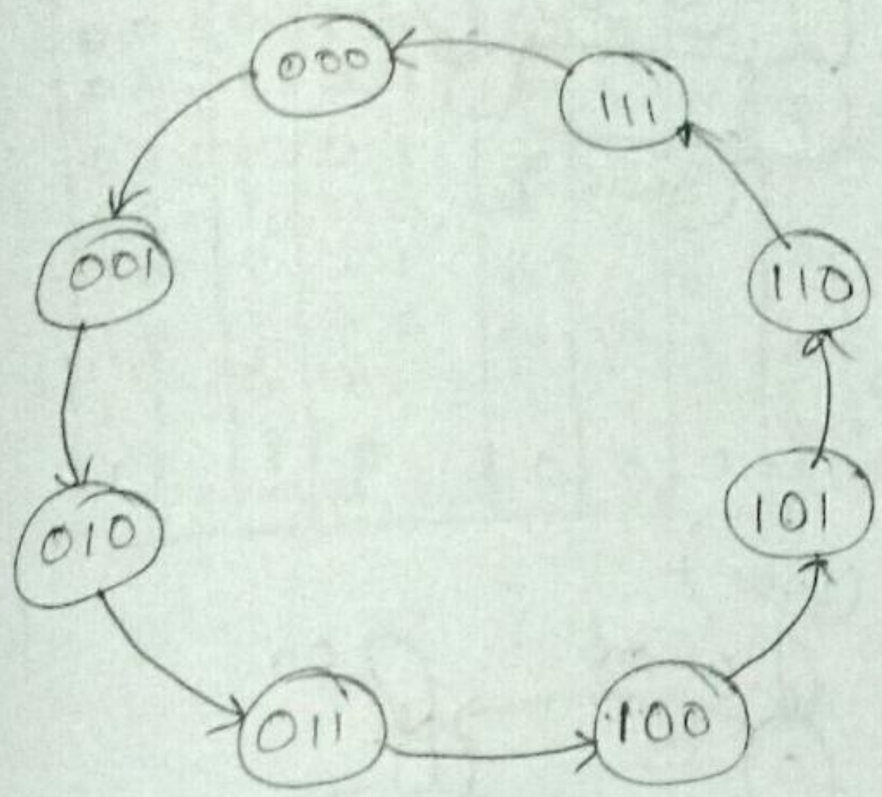
JKFF

Q_t	Q_{t+1}	JK
0	0	0x
0	1	1x
1	0	x1
1	1	x0



Scanned by TapScanner

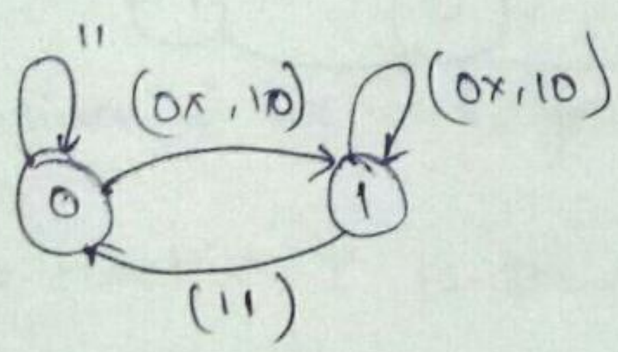
3 bit binary upCounter State diagram:



2015

Q: The following state diagram represents ____ gate

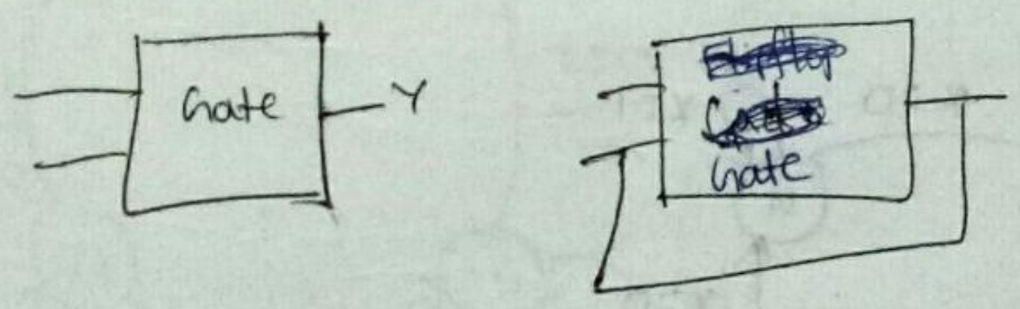
a) AND B) OR c) NAND d) NOR



AB	Y
00	1
01	1
10	1
11	0

i.e, NAND gate

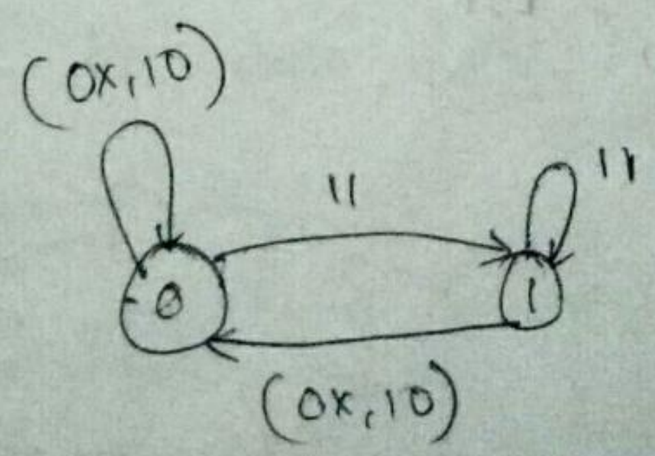
State diagrams for logic gates:



Here Present I/P depends on previous o

AND gate

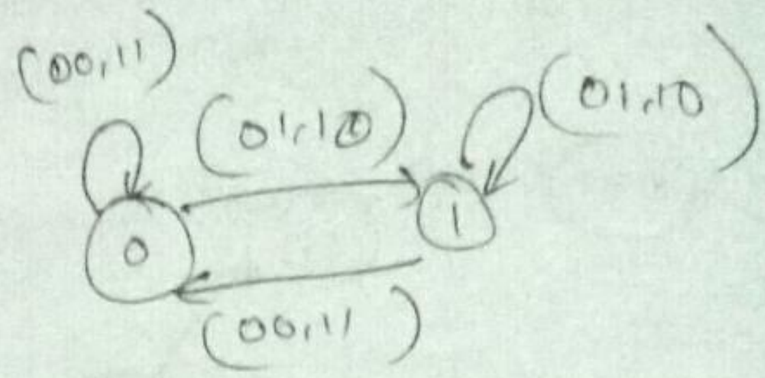
Q_t	Q_{t+1}	AB
0	0	0x, 10
0	1	11
1	0	0x, 10
1	1	11



Scanned by TapScanner

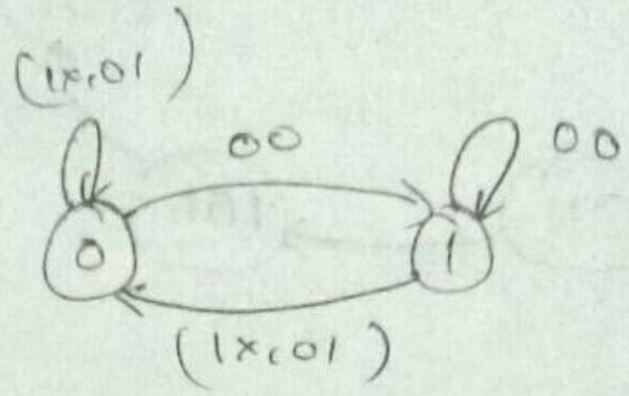
XOR gate

Q_t	Q_{t+1}	AB
0	0	00, 11
0	1	01, 10
1	0	00, 11
1	1	01, 10



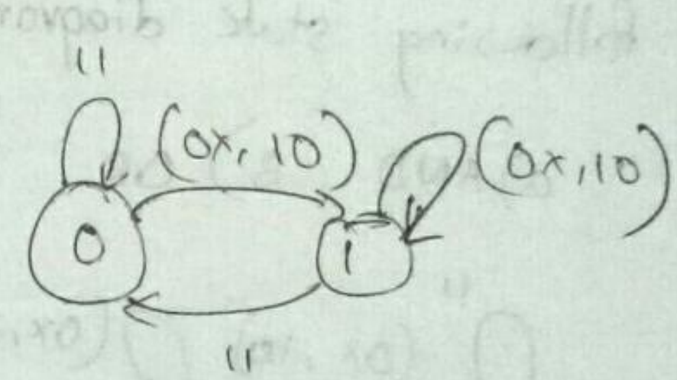
NOR gate

Q_t	Q_{t+1}	AB
0	0	1x, 01
0	1	00
1	0	1x, 01
1	1	00



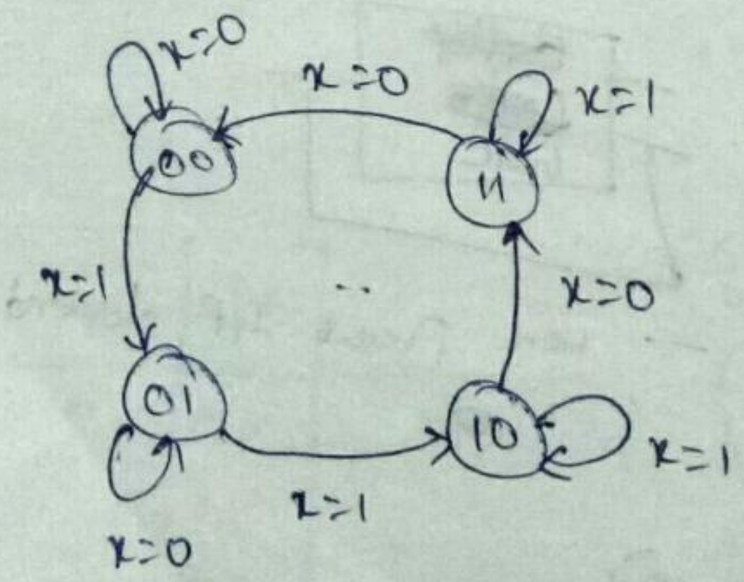
NAND gate

Q_t	Q_{t+1}	AB
0	0	11
0	1	(0x, 10)
1	0	11
1	1	(0x, 10)



Design the logic diagram for the below state diagram with TFF:

x = data i/p



Scanned by TapScanner

Q_t P.S Q_1Q_0	x	Q_{t+1} N.S Q_1Q_0	T_1	T_0	Y	
0 0	0	0 0	0	0	0	
0 0	1	0 1	0	1	0	
0 1	0	0 1	0	0	0	
0 1	1	1 0	1	1	1	
1 0	0	1 1	0	1	0	3
1 0	1	1 0	0	0	1	5
1 1	0	0 0	1	1	1	6
1 1	1	1 1	0	0	1	7

carry in FA

T_1

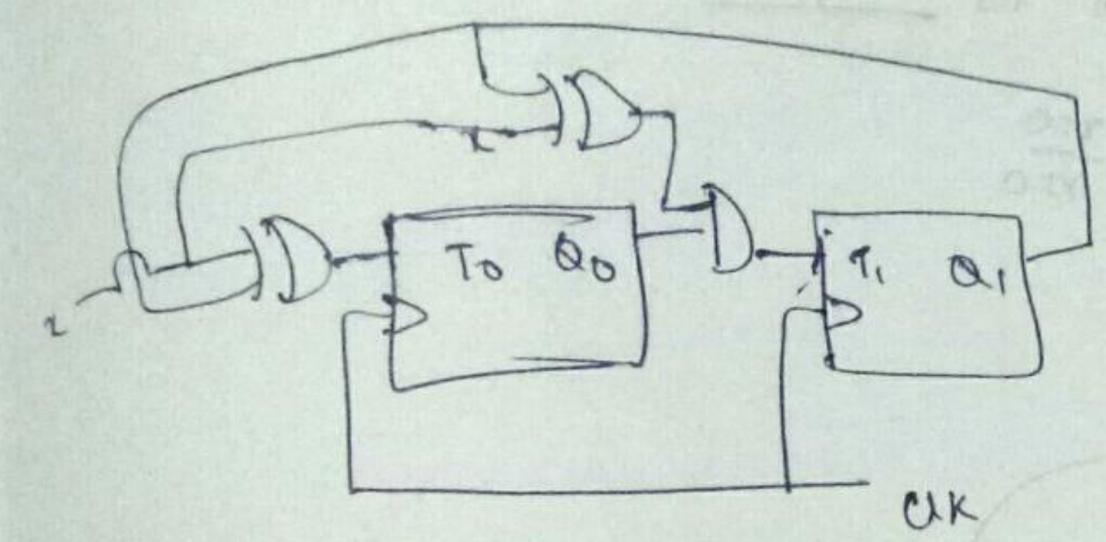
Q_1 \ Q_0x	00	01	11	10
0		1	1	
1	1			1

$T_1 = \cancel{\bar{Q}_1 x + 0}\ \bar{Q}_1 Q_0 x + Q_1 Q_0 \bar{x}$

$= \cancel{Q_0(Q_1 \oplus 0)}$

$= Q_0(Q_1 \oplus x)$

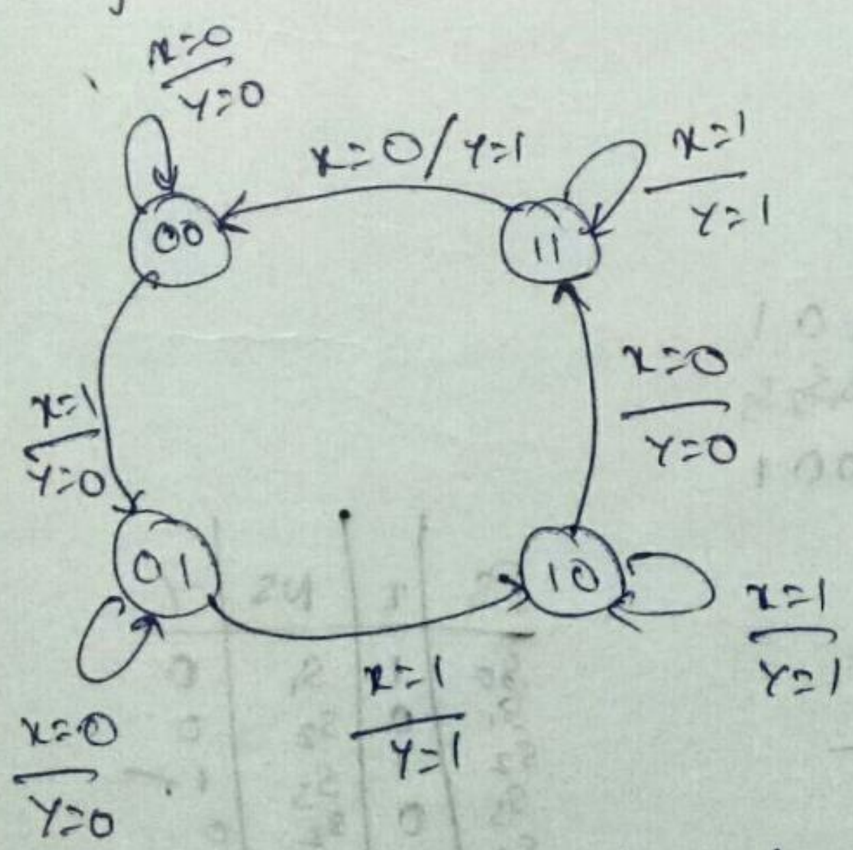
Slly. $T_0 = Q_1 \oplus x$



ckt 1

Now consider if we want generate o/p function (Y) for carry in FA

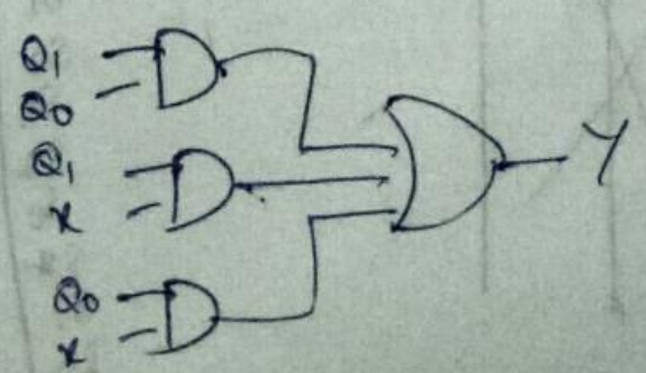
The O/P function Y is shown in above state table. The state diagram is



for carry in FA

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

Also we add below ckt to above ckt 1

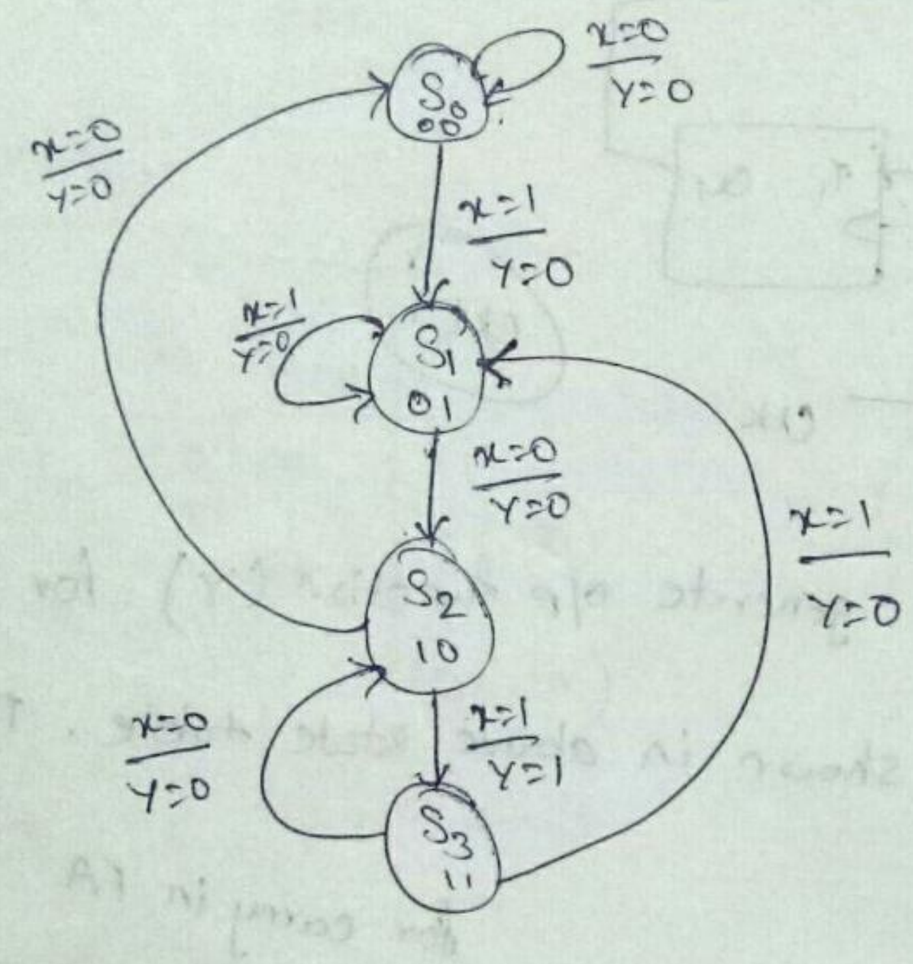


Scanned by TapScanner

~~The sequential~~

→ The sequential synchronous circuit is also known as Finite state Machine (FSM).

Q: The state diagram for the FSM is shown below. The initial state of the FSM is S_0. If the input data sequence is 1010110100110 1, starting with MSB the number of time output(Y) becomes high is 4



Initial state S_0

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

S_1 S_2 S_3 S_2 S_3 S_1 S_2 S_3 S_2 S_0 S_1 S_1 S_2 S_3

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

Method 2:

PS	x	NS	Y
S_0	1	S_1	0
S_1	0	S_2	0
S_2	1	S_3	1
S_3	0	S_2	0
S_2	1	S_3	1
S_3	1	S_1	0
S_1	0	S_2	0
S_2	1	S_3	1
S_3	0	S_2	0
S_2	0	S_0	0
S_0	1	S_2	0
S_2	1	S_1	0
S_1	0	S_2	0
S_2	1	S_3	1

∴ 4 times

Scanned by TapScanner

Doubts :

→ Analog Systems vs Digital Systems

→ which of the following expression remove hazard from : $xy + z\bar{x}$?

(a) $xy + z\bar{x}$

(b) $xy + z\bar{x} + wyz$

(c) $xy + z\bar{x} + yz$

(c) $xy + z\bar{x} + wz$

Scanned by TapScanner