Quine McCluskey method ⟷ Minterms

1 1 0 1

K-map: 2 vars, 3 vars, 4 vars, 5 vars

Simplify the following Boolean fn by using Quine McCluskey method

$f(A, B, C, D) = \Sigma m(0, 2, 3, 6, 7, 8, 10, 12, 13) = SOP$

Sol: (1) List all the minterms in the binary form

Minterms	Binary representation
m_0	0 0 0 0
m_2	0 0 1 0
m_3	0 0 1 1
m_6	0 1 1 0
m_7	0 1 1 1
m_8	1 0 0 0
m_{10}	1 0 1 0
m_{12}	1 1 0 0
m_{13}	1 1 0 1

2) Arrange the minterms according no of 1's

Minterms	Binary Representation (8 4 2 1)	
m_0	0 0 0 0	→ zero 1's
m_2	0 0 1 0	→ One 1's
m_8	1 0 0 0	
m_3	0 0 1 1	→ two 1's
m_6	0 1 1 0	
m_{10}	1 0 1 0	
m_{12}	1 1 0 0	
m_7	0 1 1 1	→ three 1's
m_{13}	1 1 0 1	

1st
2nd
3rd
4th

(3) Compare each binary no with ~~every~~ term in the next adjacent category and if they differ only by 1 position put a check mark & copy the term in the next column with (-) hyphen in the position that they differed.

Minterms	binary ~~representation~~
(0, 2)	0 0 - 0
(0, 8)	- 0 0 0

③

minterms	A	B	C	D	
1st					
(0, 2) ✓	0	0	0	0	✓
(0, 8) ✓	–	0	0	0	✓
2nd					
(2, 3) ✓	0	0	1	–	✓
(2, 6) ✓	0	–	1	0	✓
(2, 10) ✓	–	0	1	0	✓
(8, 10) ✓	1	0	–	0	✓
(8, 12)	1	–	0	0	→ $A\bar{C}\bar{D}$
3rd					
3, 7 ✓	0	–	1	1	✓
6, 7 ✓	0	1	1	–	✓
12, 13	1	1	0	–	→ $AB\bar{C}$

(C)

④ Differ only by 1 position

	minterms	A	B	C	D	
I	(0, 2, 8, 10)	–	0	–	0	$\bar{B}\bar{D}$
	(0, 8, 2, 10)	–	0	–	0	
II	(2, 3, 6, 7)	0	–	1	–	$\bar{A}C$
	(2, 6, 3, 7)	0	–	1	–	

⑤ List the prime implicants → minterms

(8, 12) = $A\bar{C}\bar{D}$

(12, 13) = $AB\bar{C}$ (d)

(0, 2, 8, 10) = $\bar{B}\bar{D}$

(2, 3, 6, 7) = $\bar{A}C$

(5) List the prime implicants

$(8, 12) = A\bar{C}\bar{D}$ ✓ $\quad (0, 2, 8, 10) = \bar{B}\bar{D}$ ✓

$(12, 13) = AB\bar{C}$ ✓ $\quad (2, 3, 6, 7) = \bar{A}C$ ✓

(6) Select the min no of prime implicants which must cover ALL the minterms.

Prime implicant selection chart

Prime implicants	m_0	m_2	m_3	m_6	m_7	m_8	m_{10}	m_{12}	m_{13}
	↓	↓	↓						
$(8, 12) = A\bar{C}\bar{D}$								⊙	⊙ ✓
$(12, 13) = AB\bar{C}$ ✓									
$(0, 2, 8, 10) = \bar{B}\bar{D}$ ✓	⊙ ✓	⊙				⊙	⊙ ✓		
$(2, 3, 6, 7) = \bar{A}C$ ✓		⊙	⊙ ✓	⊙ ✓	⊙ ✓				

1

A B C D

3 2 1

Simplified B.E using QM $= \bar{B}\bar{D} + \bar{A}C + ABC$

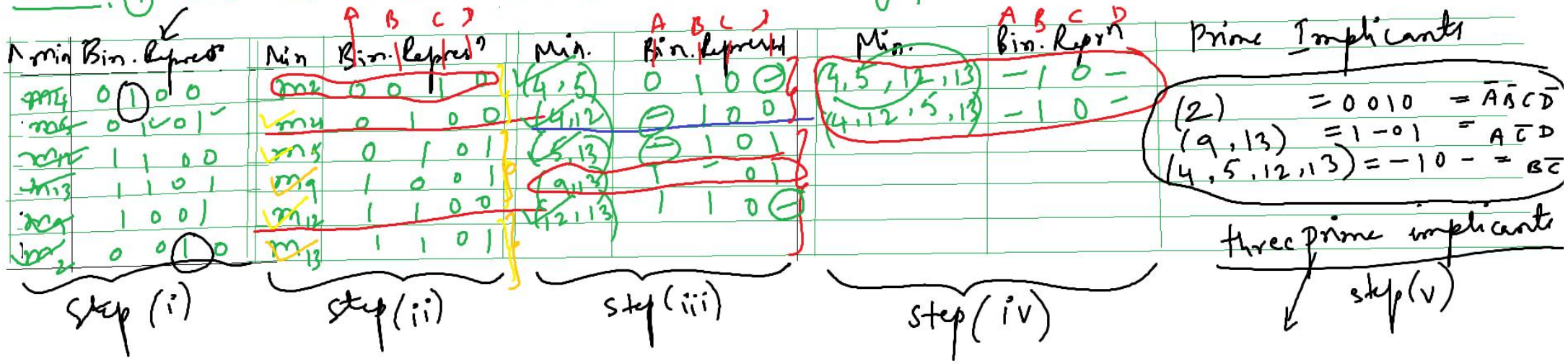
$= (-0-0) + (0-1-) + (1\ 1\ 0\ -)$

Minimize the expression using Quine McCluskey method

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

$= \Sigma m(2, 4, 5, 9, 12, 13)$

Sol: (i) List all the minterms in the binary form



Step (vi): Min no of prime implicants → ALL the minterms

Prime Implicants		m_2 (c-1)	m_4 (c-2)	m_5 (c-3)	m_9 (c-4)	m_{12} (c-5)	m_{13} (c-6)
✓ $\bar{A}B\bar{C}\bar{D}$	(2)	⊙					
✓ $A\bar{C}D$	(9, 13)				⊙		⊙
✓ $B\bar{C}$	(4, 5, 12, 13)		⊙	⊙		⊙	⊙

Final Simplified Boolean Expression = $\bar{A}\bar{B}C\bar{D} + B\bar{C} + A\bar{C}D$.

Obtain all the prime implicants of the following fn.

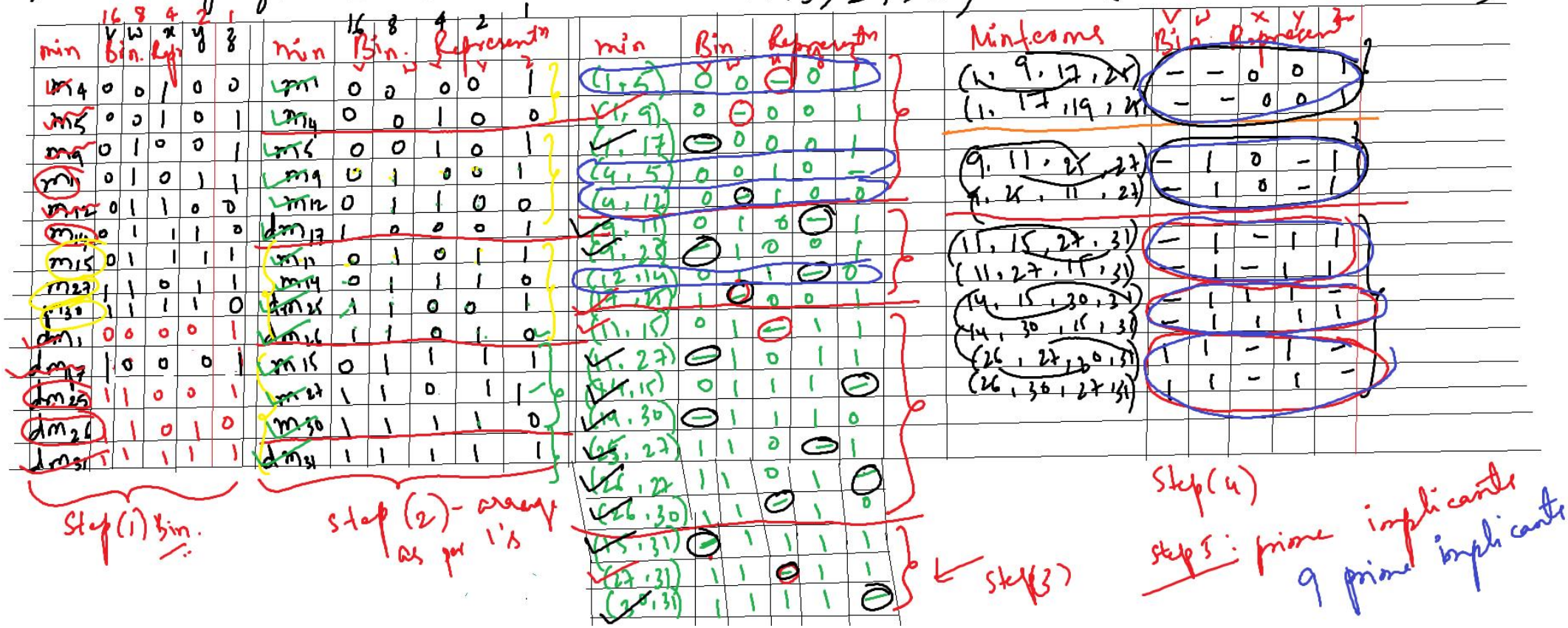
14 minterms

$f(V, W, x, y, z) = \sum m(1, 4, 5, 9, 11, 12, 14, 15, 17, 25, 26, 27, 30, 31)$

Quine McCluskey using Don't care terms (cross - X)

$x = 1$

$$f(v, w, x, y, z) = \sum m(4, 5, 9, 11, 12, 14, 15, 27, 30) + dc(1, 17, 25, 26, 31)$$



List all the prime implicants

Prime Implicants		v	w	x	y	z
		Bin. Representation				
(1, 5)	$= \bar{v}\bar{w}\bar{y}z$	0	0	–	0	1
(4, 5)	$= \bar{v}\bar{w}x\bar{y}$	0	0	1	0	–
(4, 12)	$= \bar{v}x\bar{y}\bar{z}$	0	–	1	0	0
(12, 14)	$= \bar{v}wx\bar{z}$	0	1	1	–	0
(1, 9, 17, 25)	$= \bar{x}\bar{y}z$ ✓	–	–	0	0	1
(9, 11, 25, 27)	$= w\bar{x}z$	–	1	0	–	1
(11, 15, 27, 31)	$= wyz$ ✓	–	1	–	1	1
(14, 15, 30, 31)	$= wxy$ ✓	–	1	1	1	–
(26, 27, 30, 31)	$= vwy$ ✓	1	1	–	1	–

Selection of min pr. imp = ALL = P.I. S. chart

Find Simplified Boolean Expr^n using Quine-McCluskey

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

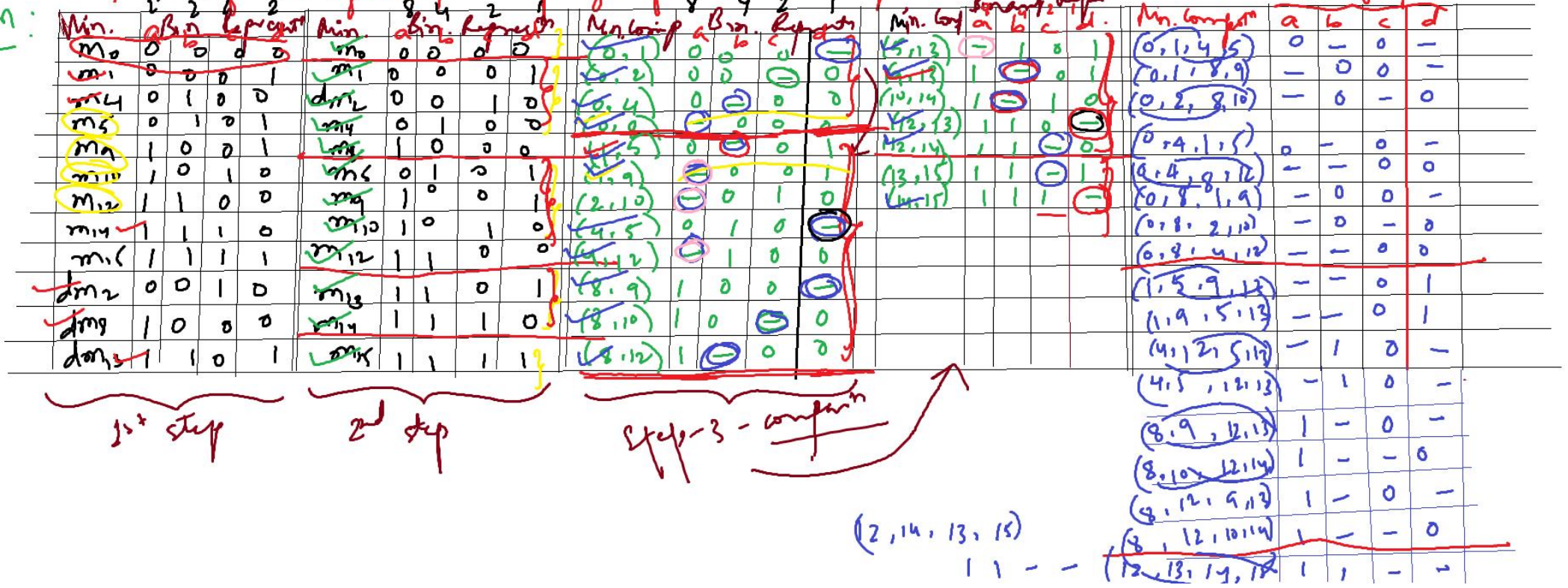
1, 9, 17, 25

26, 27, 30, 31

Prime Implicants	m_1	m_4	m_5	m_9	m_{11}	m_{13}	m_{14}	m_{15}	m_{17}	m_{25}	m_{26}	m_{27}	m_{30}	m_{31}
(1, 5)	⊙		⊙											
(4, 5)		⊙	⊙											
(4, 12)		⊙				⊙								
(12, 14)						⊙	⊙							
✓ (1, 9, 17, 25)	⊙			⊙					⊙	⊙				
(9, 11, 25, 27)				⊙	⊙					⊙		⊙		
(11, 15, 27, 31)					⊙		⊙					⊙	⊙	
(14, 15, 30, 31)							⊙	⊙					⊙	⊙
✓ (26, 27, 30, 31)											⊙	⊙	⊙	⊙

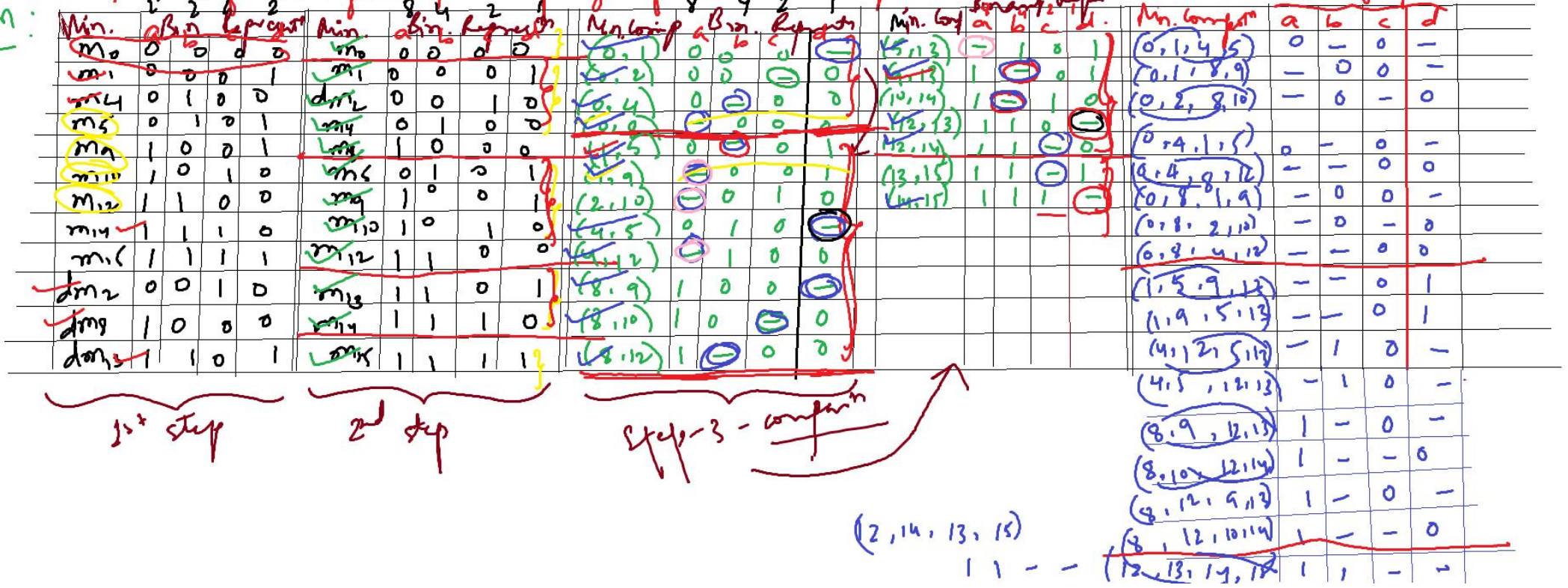
Using Quine McCluskey tabulation method, obtain the set of prime implicants for the fn $f(a,b,c,d) = \Sigma(0,1,4,5,9,10,12,14,15) + d(2,8,13)$ and hence obtain the minimal form of the given fn employing decimal representn.

Soln:



Using Quine McCluskey tabulation method, obtain the set of prime implicants for the fn $f(a,b,c,d) = \Sigma(0,1,4,5,9,10,12,14,15) + d(2,8,13)$ and hence obtain the minimal form of the given fn employing decimal representn.

Soln:



Step 6: Draw prime implicants selection chart

Final f. B.E = $\bar{b}\bar{d} + ab + \bar{c}$

$(-0-0) + (11--) + (--0-)$

Min. Comp	a	b	c	d
(0, 1, 4, 5)	0	−	0	−
(0, 1, 8, 9)	−	0	0	−
(0, 2, 8, 10)	−	0	−	0
(4, 5, 9, 13)	−	−	0	1
(4, 12, 5, 13)	−	1	0	−
(8, 10, 12, 14)	1	−	−	0
(12, 13, 14, 15)	1	1	−	−

→ $\bar{b}\bar{d}$, $a\bar{d}$, ab

Min. Comp	a	b	c	d
(0, 1, 4, 5, 8, 9, 12, 13)	−	−	0	−
(0, 1, 8, 9, 4, 12, 5, 13)	−	−	0	−
(0, 4, 8, 12, 1, 5, 9, 13)	−	−	0	−

Binar. Rep.

→ $\bar{c}$

Step 5: 4 prime implicants

$\bar{b}\bar{d} = (0, 2, 8, 10) = -0-0$

$a\bar{d} = (8, 10, 12, 14) = 1--0$

$ab = (12, 13, 14, 15) = 11--$

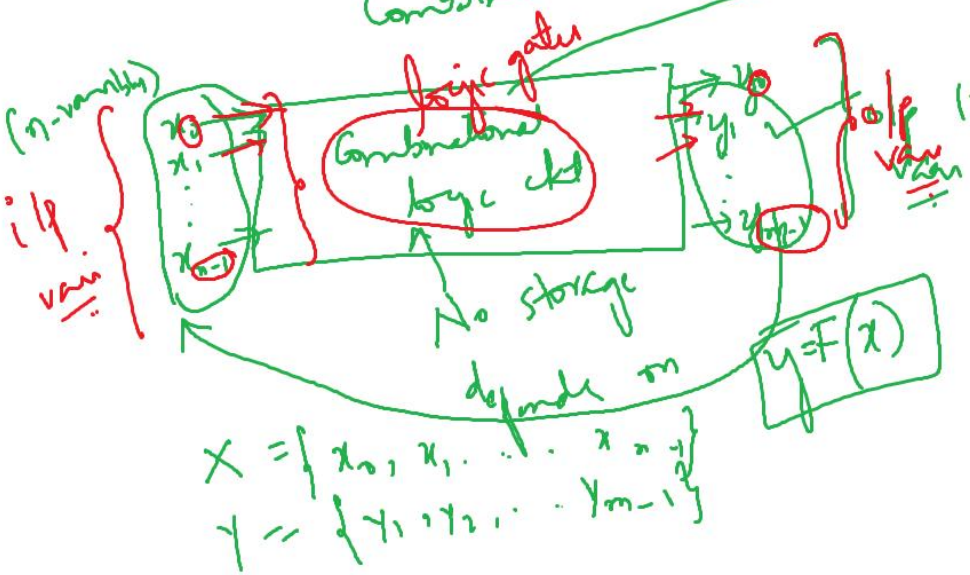
$\bar{c} = (0, 1, 4, 5, 8, 9, 12, 13) = --0-$

Prime Implicants	m_0	m_1	m_2	m_4	m_5	m_8	m_9	m_{10}	m_{12}	m_{13}	m_{14}	m_{15}
✓ $\bar{b}\bar{d} = (0, 2, 8, 10)$	⊙		⊙			⊙		⊙				
$a\bar{d} = (8, 10, 12, 14)$ ×						⊙		⊙	⊙		⊙	
✓ $ab = (12, 13, 14, 15)$									⊙	⊙	⊙	⊙ ✓
✓ $\bar{c} = (0, 1, 4, 5, 8, 9, 12, 13)$	⊙	⊙ ✓		⊙	⊙	⊙	⊙		⊙	⊙		

Unit – 2

Data processing circuits → collection of logic gates

- Combinational
- Sequential

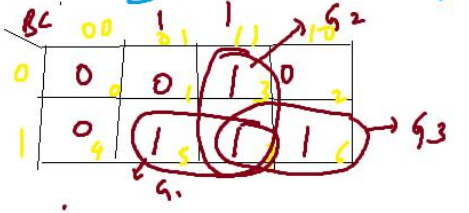


Ex: A combinational logic ckt with 3 i/p vars that will produce a logic-1 when > one i/p vars are = 1. Derive the TT. & implement the ckt.

Sol: 3 i/p variables ⇒ 2^3 = 8 combin^{n} / i/p set

Designing of L.C.

	A	B	C	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1 → $\bar{A}BC$
4	1	0	0	0
5	1	0	1	1 → $A\bar{B}C$
6	1	1	0	1 → $AB\bar{C}$
7	1	1	1	1 → ABC

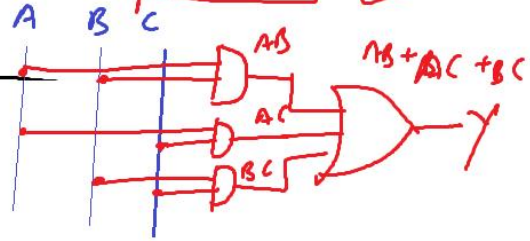


$Y = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

$Y = AC + BC + AB$

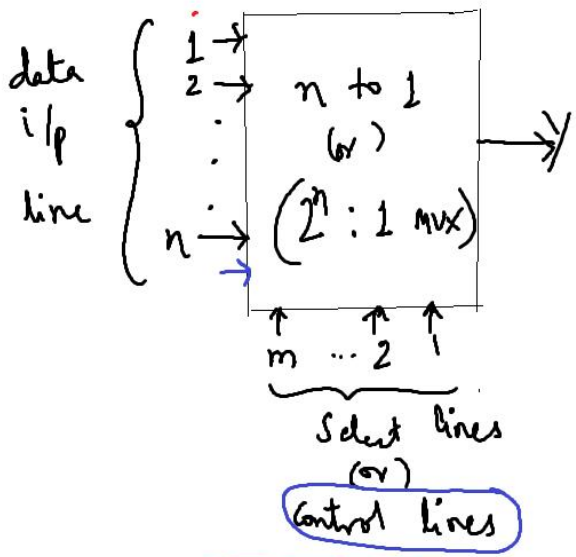
$\bar{Y} = \overline{AB + AC + BC}$

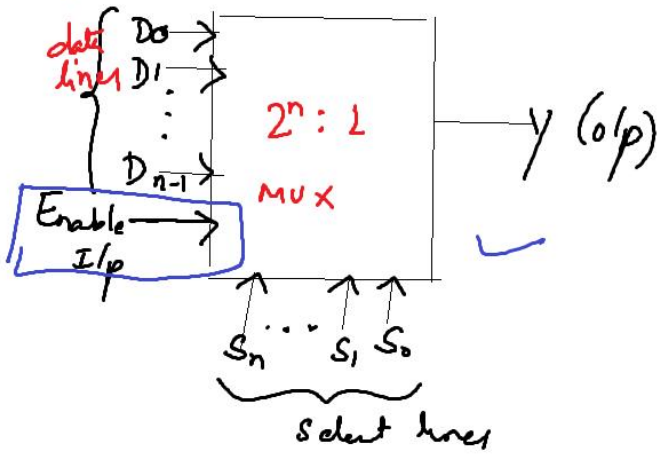
Implementation:



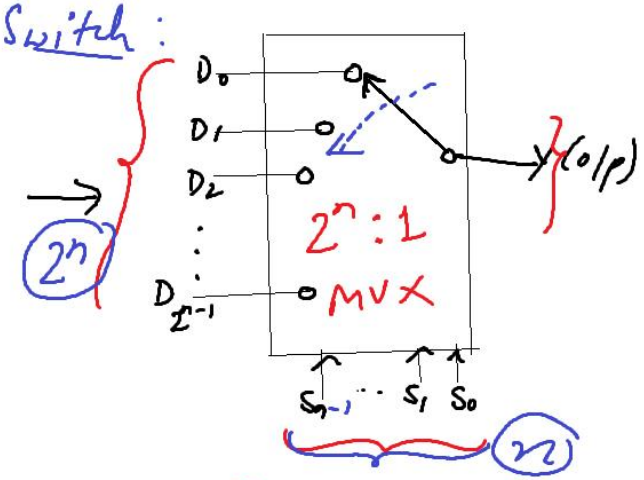
Multiplexers: (MUX) → Select single data line from several data i/p lines

↳ data selector - many to 1 -





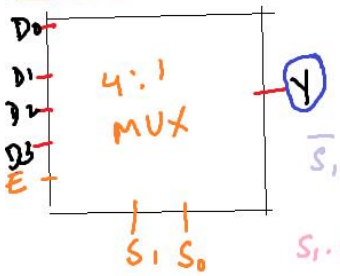
Equivalent ckt →



Equivalent ckt.

Block diagram of 2^n : 1 MUX

4:1 MUX:



Function table / Truth table

S_1	S_0	Y	E
0	0	D_0	1
0	1	D_1	1
1	0	D_2	1
1	1	D_3	1

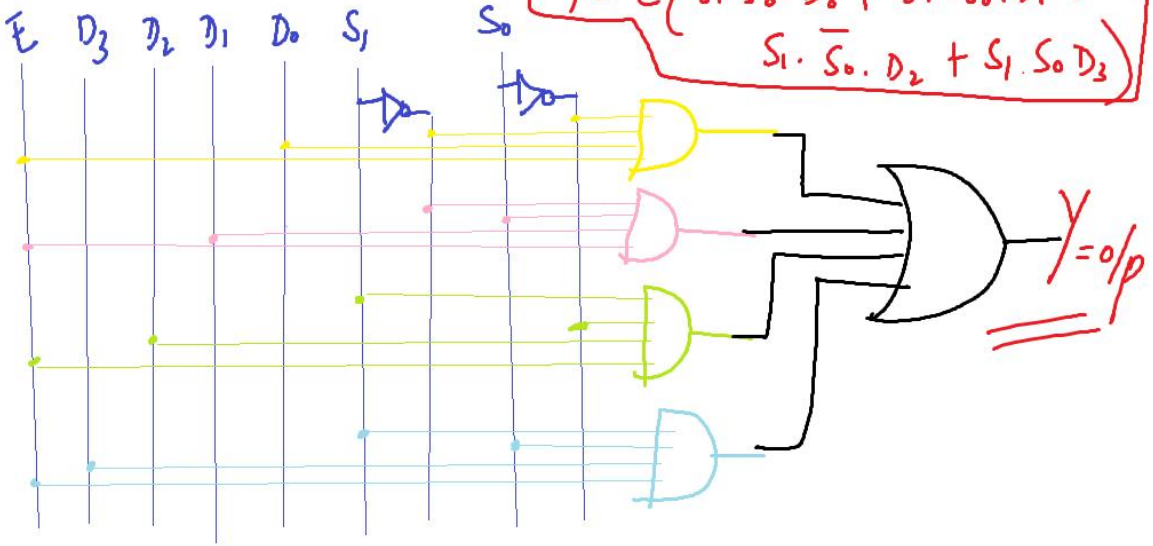
$\bar{S}_1 \bar{S}_0 . D_0 E$

$\bar{S}_1 . S_0 . D_1 E$

$S_1 . \bar{S}_0 . D_2 E$

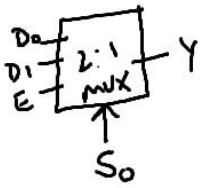
$S_1 . S_0 . D_3 E$

$$Y = E(\bar{S}_1 . \bar{S}_0 . D_0 + \bar{S}_1 . S_0 . D_1 + S_1 . \bar{S}_0 . D_2 + S_1 . S_0 D_3)$$



2:1 MUX:

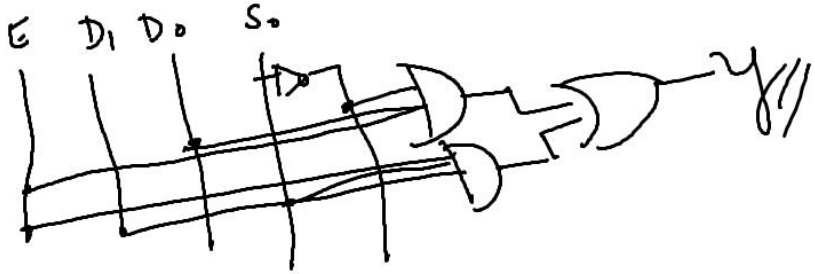
Select line = 1
data lines = 2



f^n table / Truth table

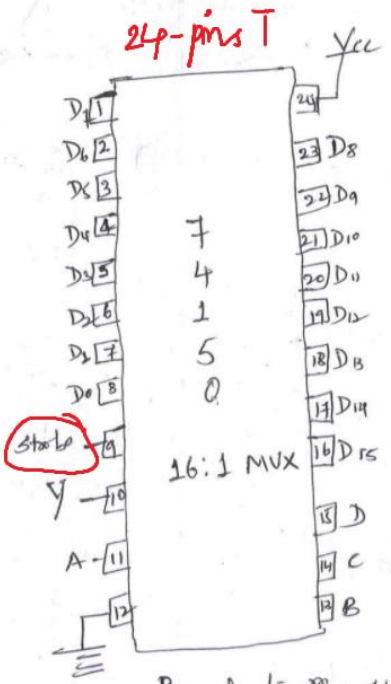
E	S_0	Y
1	0	D_0
1	1	D_1

$$Y = E\bar{S}_0 D_0 + E S_0 . D_1 = E(\bar{S}_0 . D_0 + S_0 . D_1)$$



Strobe/ Enable	A	B	C	D	Y
L	L	L	L	L	$\overline{D_0}$
L	L	L	L	H	$\overline{D_1}$
L	L	L	H	L	$\overline{D_2}$
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
L	H	H	H	H	$\overline{D_{15}}$
H	X	X	X	X	H

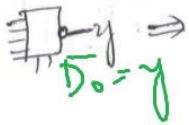
74150 Truth Table



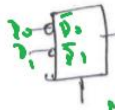
Pinout diagram of 74150

can't be determined

Bubbles on Signal line:



$D_0 = y$ ⇒ is the o/p that is the complement of the selected data-bit.



y ⇒ bubble on i/p pins, means the signal is active low.

Low-state

1 = 1

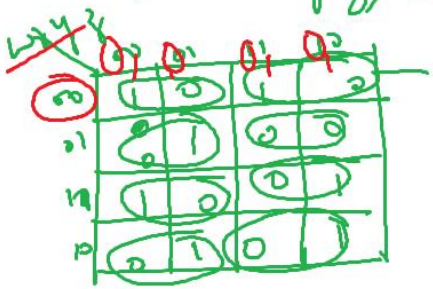
0 74150

E	S_1	S_0	Y
0	0	0	$\overline{D_0}$
0	0	1	$\overline{D_1}$
0	1	0	$\overline{D_2}$
0	1	1	$\overline{D_3}$
1	X	X	X

Realization of 4 var f^n using 8:1 MUX

Soln :: $f(w,x,y,z) = 2^4 = 16:1$

↳ 8:1



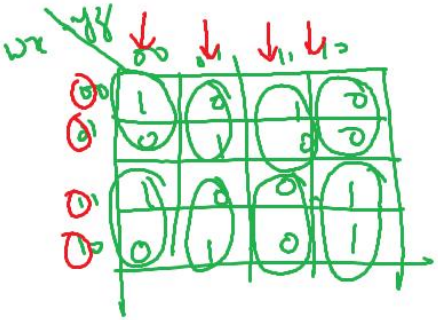
horizontal grouping

Select line

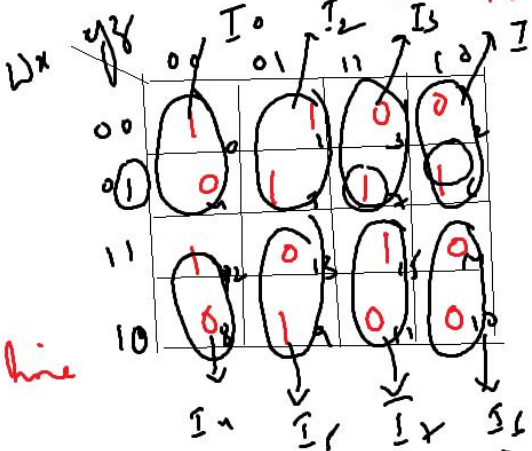
wxy = horizontal combination → (2) → data line

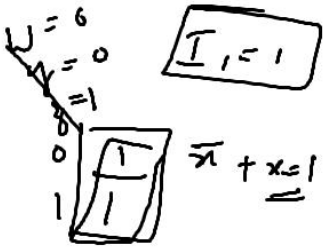
→ wyz → vertical grouping

(x) → data line i/p.



$\Sigma \bar{0}, \bar{1}, 5, 6, 7, 9, 12, 15$



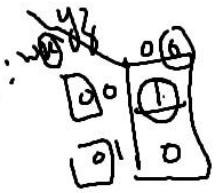


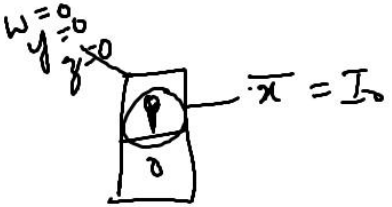
$I_1 = 1$

$\bar{x} + x = 1$

$I_2 = x$

Submap :

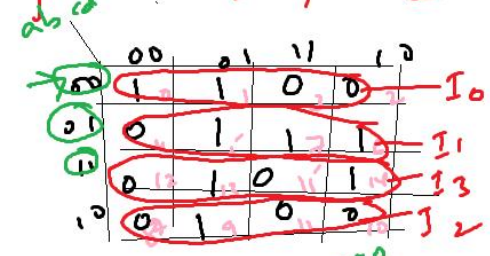




$\bar{x} = I_0$

Realize 4 var B.f using 4:1 MUX

$f(a,b,c,d) = \Sigma m(0,1,5,6,7,9,13,14)$



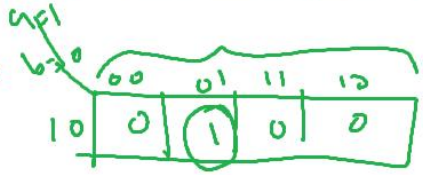
Sub maps : I_0

$\overline{I_0} = \overline{c}$

$I_1 =$

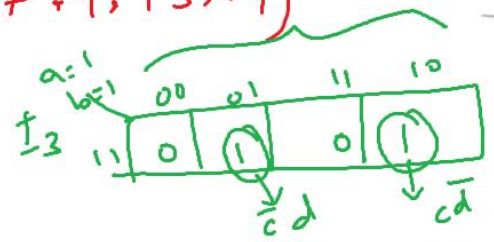
$d + c \Rightarrow I_1 = c + d$

$I_2 =$



$I_2 = \overline{c}.d$

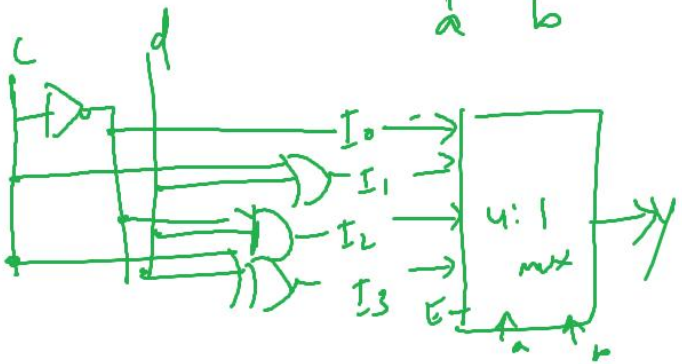
(2) Rea

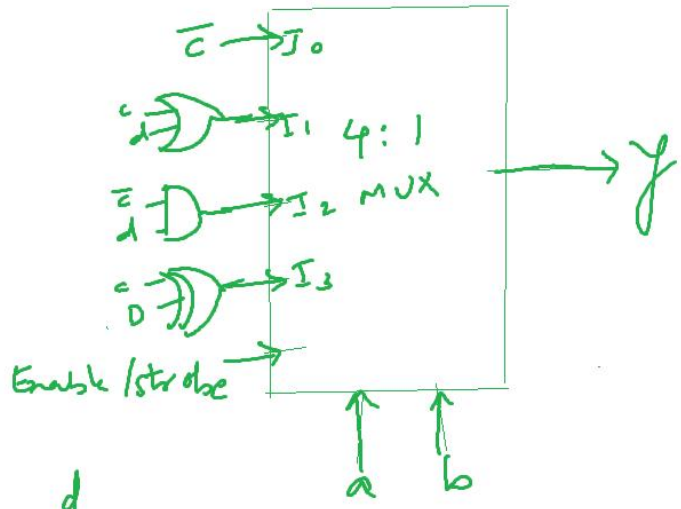


$\therefore I_3 = \overline{c}d + c\overline{d}$

$I_3 = c \oplus d$

I_0

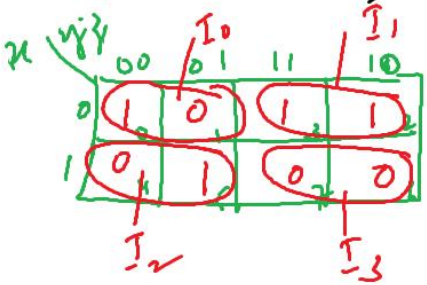




Realizn of 3-var fn using 4:1 mux:

$f(x, y, z) = \sum m(0, 2, 3, 5)$

Soln :-

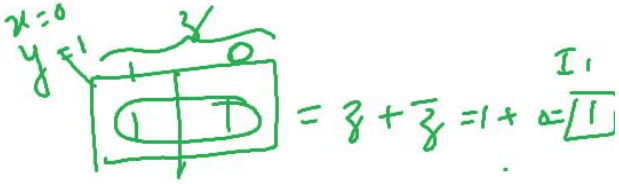


bit = 0 / 1

byte = 0 1 0 1 0 1 0 1 — 8-bits

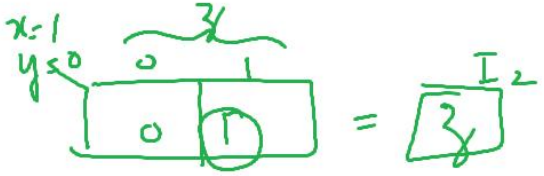
∴ (A) (B) (C)

Submap for I_1:



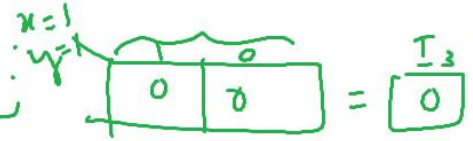
$= z + \bar{z} = 1 + 0 = 1$ → I_1

Submap for I_2:



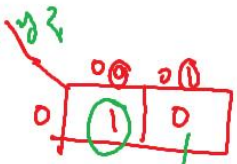
$= \bar{z}$ → I_2

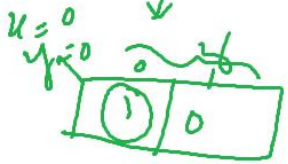
Submap for I_3:



$= 0$ → I_3

Submap:





$= I_0 = \bar{z}$

x y → constant values → select lines

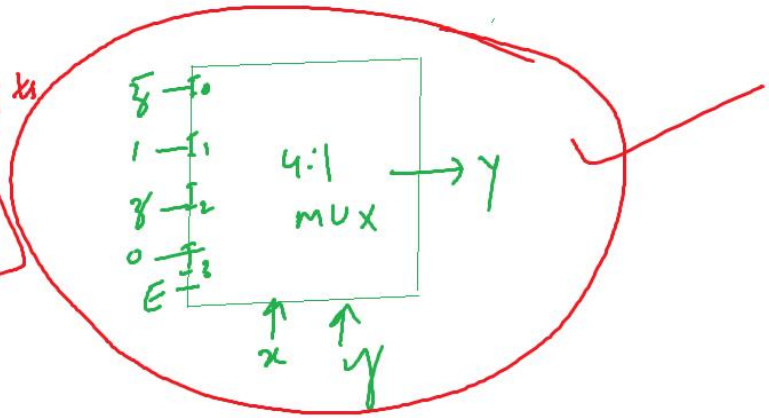
z → varying → data line i/p.

nibble = byte/2 = 8 bits/2

nibble = 4 bits

byte = n bits

n-1 | n-2



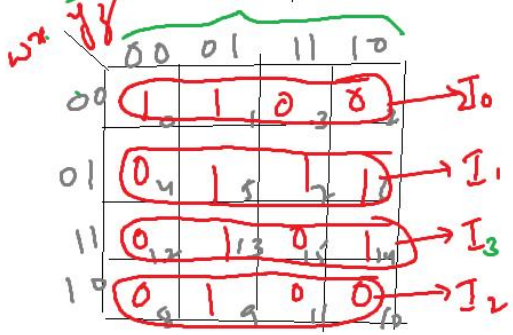
Realization of 4 variable f using 4:1 MUX

$f = \Sigma m(0, 1, 5, 6, 7, 9, 13, 14)$

Soln:

4-variables → 2^4 → 16:1 MUX

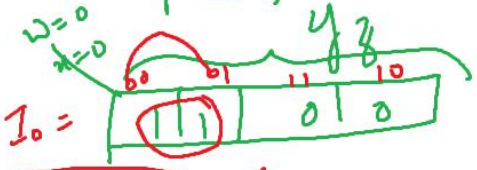
2 variable → 2^2 → 4:1 MUX



Common - constant literals → select lines

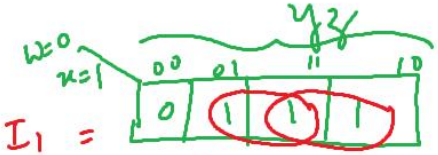
varying → data lines (yz)

Submaps: for I_0, I_1, I_2 and I_3



$I_0 =$

$I_0 = \bar{y}$ ✓

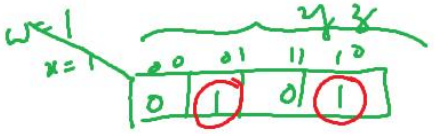


$I_1 =$

$= z + y$

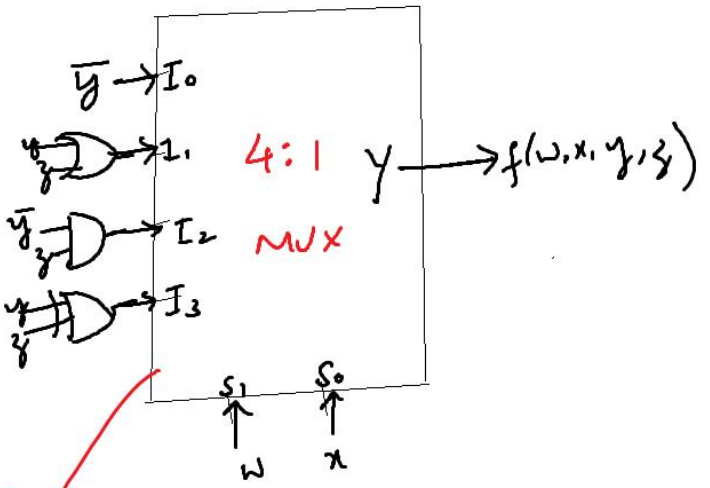
$I_1 = y + z$ ✓

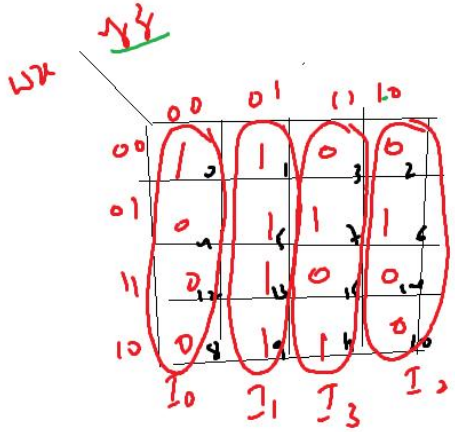
$I_2 = \bar{y}z$ ✓

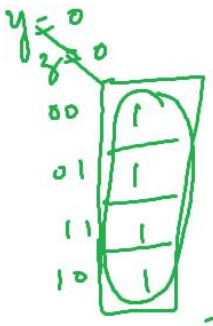


$I_3 = \bar{y}z + y\bar{z} = y \oplus z$

$I_3 = y \oplus z$







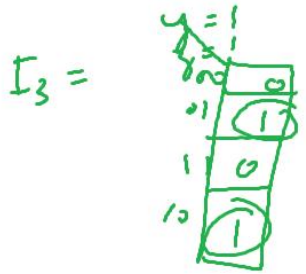
$= w\bar{x} + \bar{w}x + wx + w\bar{x}$

$(w \oplus x) + (\bar{w} \oplus x)$

$1 + 0 = 1$

$I_1 = 1$ ✓

$I_2 = \bar{w}x$ ✓

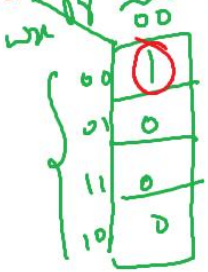


$\bar{w}x + w\bar{x}$

$I_3 = w \oplus x$ ✓

Submaps for I_0, I_1, I_2, I_3

$I_0 = y'z'$ Select lines

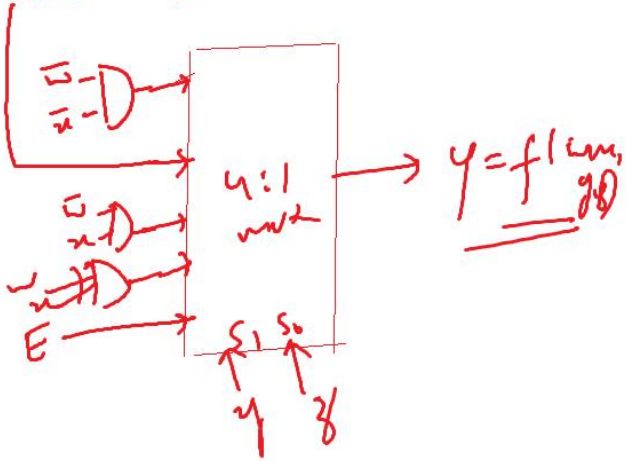


→ wx → data lines

yz → select lines

$I_0 = \bar{w} \cdot \bar{x}$ ✓

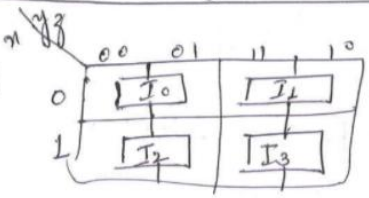
Vcc = logic 1



Realization of 3 vari boolean fn using 4-to-1 MUX

2 ways:

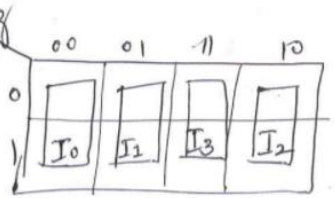
①



⇒ xy – select lines

z – data line

②

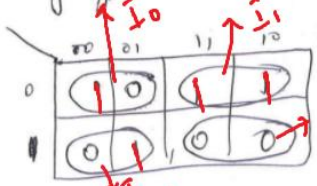


⇒ yz – select lines

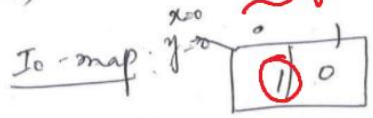
x – data line

Eg: First method:

$f(x,y,z) = \sum m(0,2,3,5)$ Sum of products

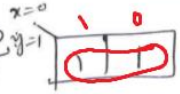


I_0 – map:



$= \bar{z}$ ✓

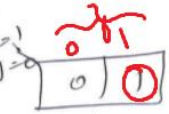
I_1 – map:



$= \bar{y} + \bar{z} = 1 + 0$

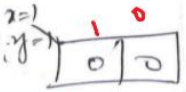
$= 1$ ✓

I_2 – map:



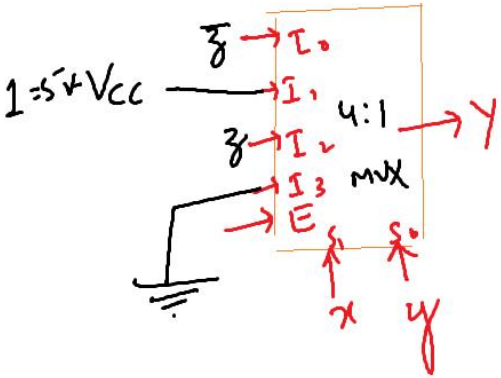
$= z$ ✓

I_3 – map:



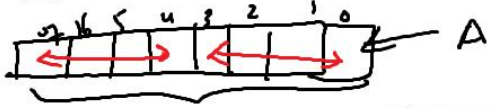
$= 0$ ✓

$I_0 = \bar{z}$ $I_1 = 1$ $I_2 = z$ $I_3 = 0$



1 bit = 0 or 1

1 byte = 8-bits



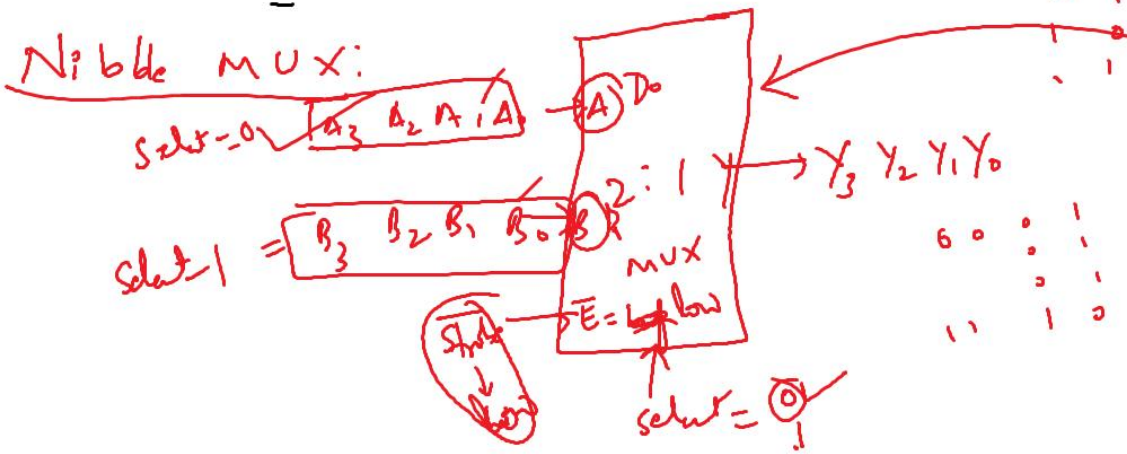
1 word = 2-bytes =

1 nibble = $\frac{1\text{ byte}}{2} = \frac{8\text{-bits}}{2}$ = 4 bits

Nibble MUX:

Select = 0 → $A_3\ A_2\ A_1\ A_0$

Select = 1 → $B_3\ B_2\ B_1\ B_0$



A B

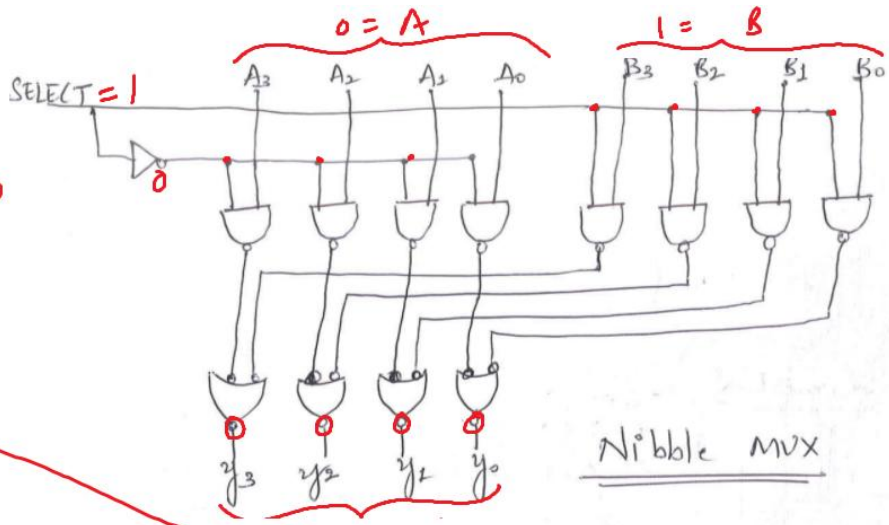
0 0

0 1

1 0

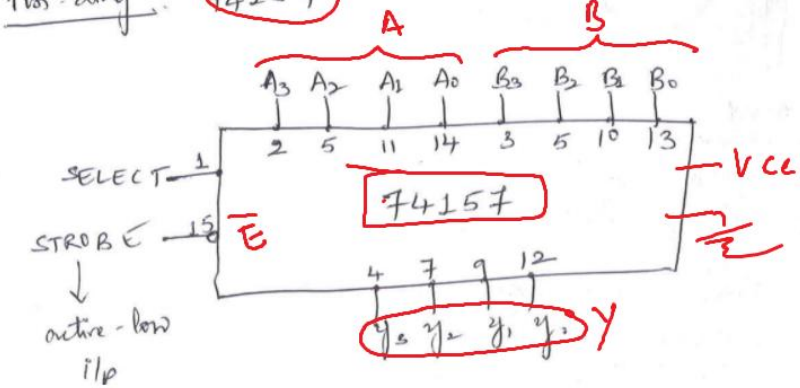
1 1

A < 0 / 1



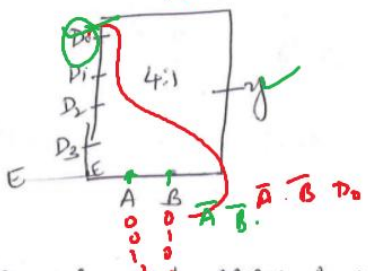
Nibble MUX

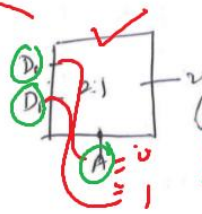
Pin-diagram: 74157



1) Show how 4:1 MUX can be obtained using only 2:1 MUX

Soln :





$\bar{A}\cdot D_0 + A\cdot D_1$

$\bar{A}\,\bar{B}\,D_0$

Logic eqn for 2:1 MUX : $y = \bar{A}\,D_0 + A\,D_1$ — (1)

Logic eqn for 4:1 MUX : $y = \bar{A}\,\bar{B}\cdot D_0 + \bar{A}\cdot B\cdot D_1 + A\,\bar{B}\cdot D_2 + A\cdot B\cdot D_3$

$y = \bar{A}(\bar{B}\cdot D_0 + B\cdot D_1) + A(\bar{B}\cdot D_2 + B D_3)$

⇒ 4 s
cho
⇒ 2-to
the

Logic eqn of two 2:1 MUX

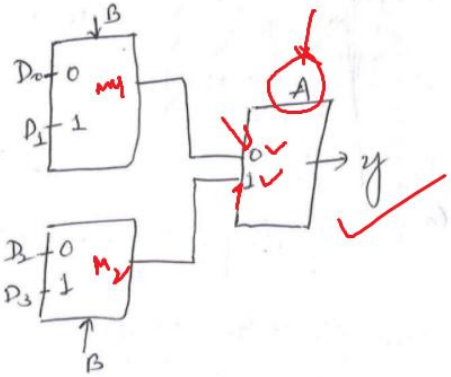
The logic eqn of 4:1 MUX can be re-written as

$y = \bar{A}(\bar{B}D_0 + BD_1) + A(\bar{B}D_2 + BD_3)$ — (2)

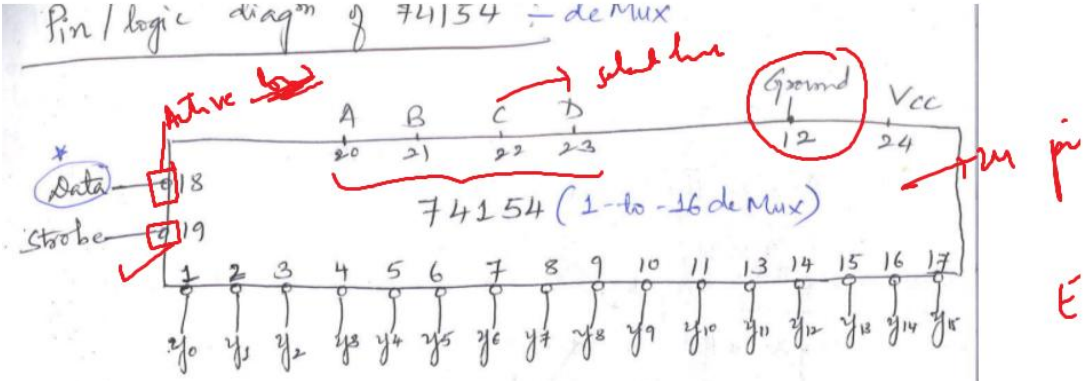
B_1 B_2

Compare eqn (1) & (2).

We need two 2:1 MUX to realize the two bracketed terms where B serves as select i/p. The o/p these 2 MUXs can be sent to a 3rd MUX as data i/ps where A serves as select i/p & we get 4:1 MUX.



Pin / logic diagm of 74154 – deMux



D	strobe E	A	B	C	D	Y_0	Y_1	Y_{15}
0	0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	0	1
1		X	X	X	X	1	1	1
	1	X	X	X	X	1	1	1

Worked Example: Show how two 1-to-16 MUX can be connected to get 1-to-32 deMUX.

Soln: 1-to-32 deMUX has 5 select variables A B C D E.

Four of them B C D E — fed to two 1-to-16 deMUX.

Fifth A — used to select one of these two deMux through strobe i/p.

If A = 0, the top 74154 is choosen.

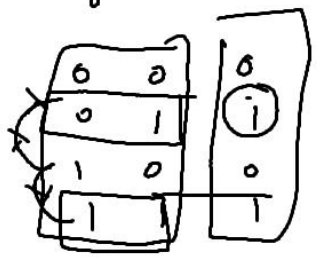
& B C D E directs data to one of the 15 o/p's of that IC.

If A = 1, the top 74154 is choosen;

Adders & Subtractors : Combinational ckts

Binary

present i/p combin^n



Binary Adders:

- Half Adder
  - 2 bin. i/ps
    - Augend & addend
  - 2 bin. o/ps
- Full Adder
  - 3 bin. i/ps (Aug, Addn, C_i)
  - 2 bin. o/ps

$0+0=0$

$0+1=1$

$1+0=1$

$1+1=10$ (1 → MSB, 0 → LSB)

1
1
10
Sum
Carry

Binary Half Adder

x_i – Augend

y_i → addend

C → carry

S → Sum

x_i	y_i	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

II:

0 → Augend

+ 0 → addend

0 0

sum

0 1 1
0 1 0
0 1 1
carry sum

I/ps x_i, y_i → Half Adder → S, C → O/ps

Realisation / Implⁿ

$C = x_i \cdot y_i$

K-map : Carry :

x_i \ y_i	0	1
0	0	0
1	0	1

Sum = x_i y_i

x_i \ y_i	0	1
0	0	1
1	1	0

$Sum = \bar{x}_i \cdot y_i + x_i \cdot \bar{y}_i$

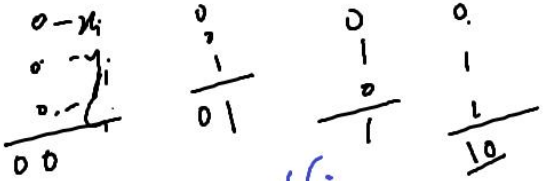
$S = x_i \oplus y_i$

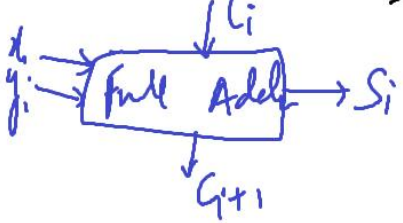
Binary Full Adder ckt : → 3 i/p, Carry-out = C_{i+1}

→ 2 o/p

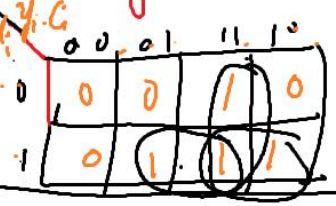
Truth table:

x_i	y_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1 ✓
0	1	0	0	1 ✓
0	1	1	1	0
1	0	0	0	1 ✓
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1 ✓



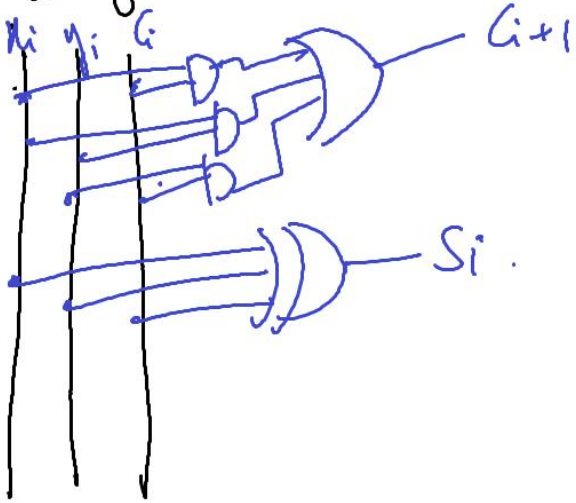


K-map:

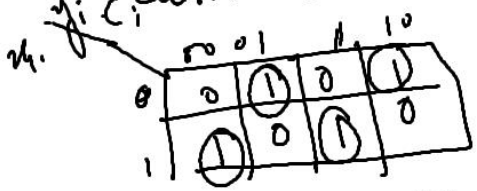


$C_{i+1} = x_i \cdot C_i + x_i \cdot y_i + y_i \cdot C_i$

Realization:



Sum = S_i



$S_i = \bar{x_i} \cdot \bar{y_i} \cdot C_i + \bar{x_i} \cdot y_i \cdot \bar{C_i} + x_i \cdot \bar{y_i} \cdot \bar{C_i} + x_i \cdot y_i \cdot C_i$

$= C_i (\bar{x_i} \bar{y_i} + x_i y_i) + \bar{C_i} (\bar{x_i} \cdot y_i + x_i \cdot \bar{y_i})$

$= C_i \overline{(x_i \oplus y_i)} + \bar{C_i} (x_i \oplus y_i)$

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

$S_i = C_i \oplus (x_i \oplus y_i)$

Binary Subtractor :

Half → 2 i/p

Full → 3 i/p

2 o/p

b_i ← borrow

x_i	y_i	d_i	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Bin. Half Subtractor :

0 − 0 = 0

0 − 1 = 1

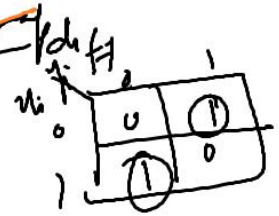
1 − 0 = 1

1 − 1 = 0

2 bin i/p: Minuend − x_i
Subtrahend − y_i

2 bin o/p: difference − d_i
borrow bit − b_i

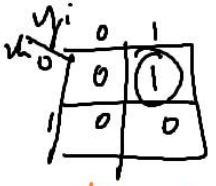
K-map



$$d_i = \bar{x_i} \cdot y_i + x_i \cdot \bar{y_i} = x_i \oplus y_i$$

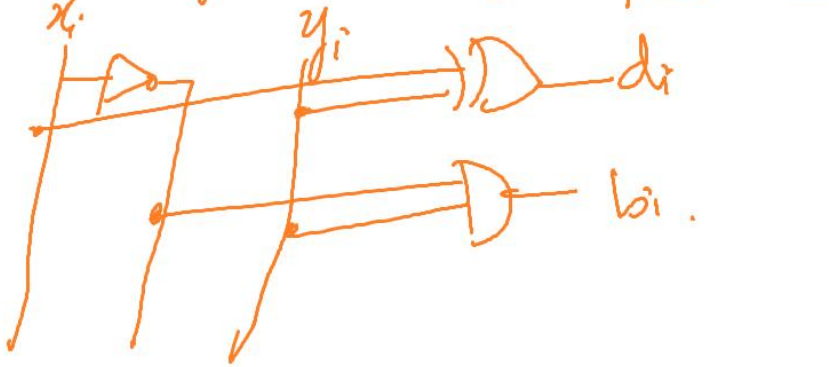
$$d_i = x_i \oplus y_i$$

borrow :



$$\bar{x_i} \cdot y_i = b_i$$

Logic diagram / ckt / Realization / Implementation



Binary full subtractor

3 i/p – x_i, y_i, b_i

o/p – b_{i+1}, d_i

x_i	y_i	b_i	d_i	b_{i+1}
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

x_i y_i = → subtrahend

Minuend $y_i b_i$

$d_i = x_i$

k-map

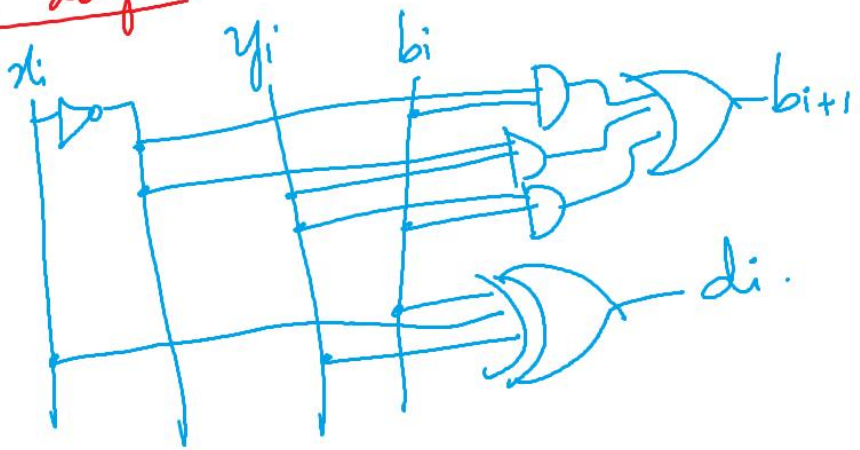
= same to S.O.E of Full Adder

but b_i replaced with b_i bar.

do write the simplifn steps.

$$d_i = b_i \oplus (x_i \oplus y_i)$$

Logic diagm.



k-map:

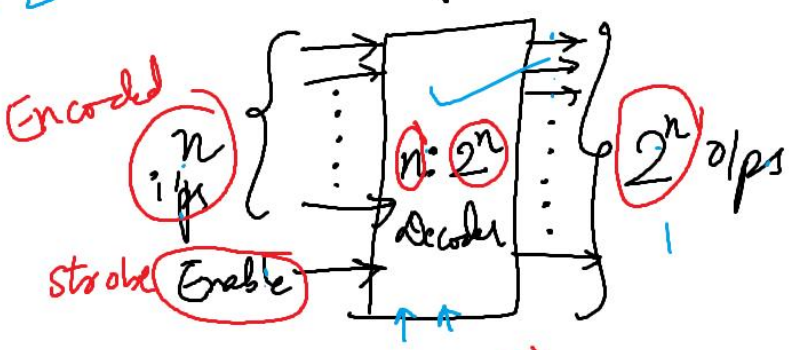
b_{i+1} =

$$\bar{x}_i \cdot b_i + \bar{x}_i \cdot y_i + y_i \cdot b_i$$

= b_{i+1} = borrow-out

Decoder: Combinational ckt with multiple i/ps & multiple o/ps.
↳ converts CODED i/ps into CODED o/ps, where the i/p & o/p codes are different

$n : 2^n$



n-i/ps — (0 to (n-1))

2^n-o/ps - (0 to 2^n-1)

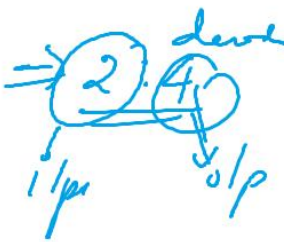
Binary decoder: 2 i/p lines

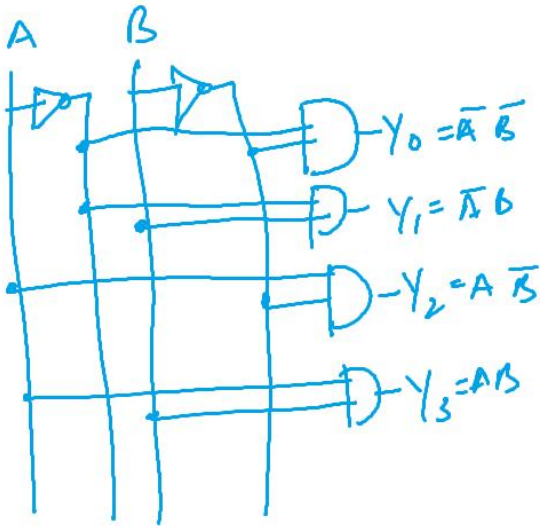
E	A	B	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0
0	X	X	0	0	0	0

2:4 decoder

$n : 2^n$

$2 : 2^2$

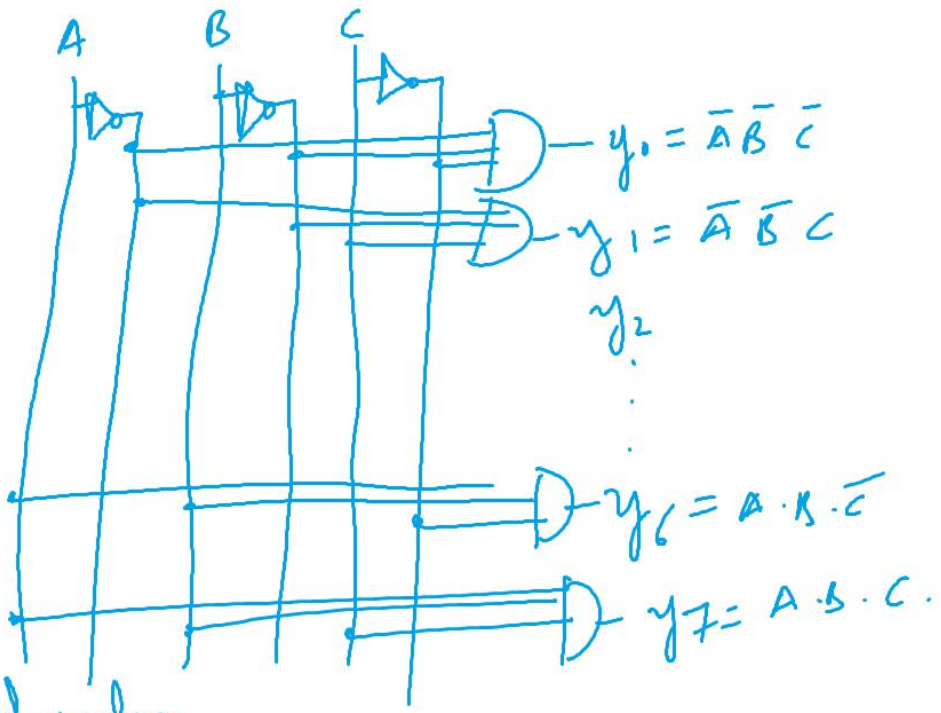




Draw the ckt for 3 to 8 decoder

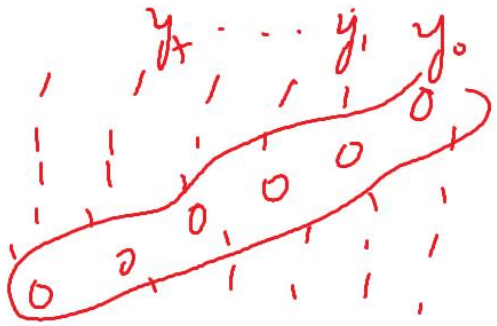
Soln: Truth table → ckt diagram.

E	A	B	C	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0



3 : 8 line decoder

74138 : 3 to 8 decoder
- commercially available
- 3 bin. i/ps
- 8 individual active-LOW o/ps
- $\overline{E}$ = low strobe = active low enable i/p



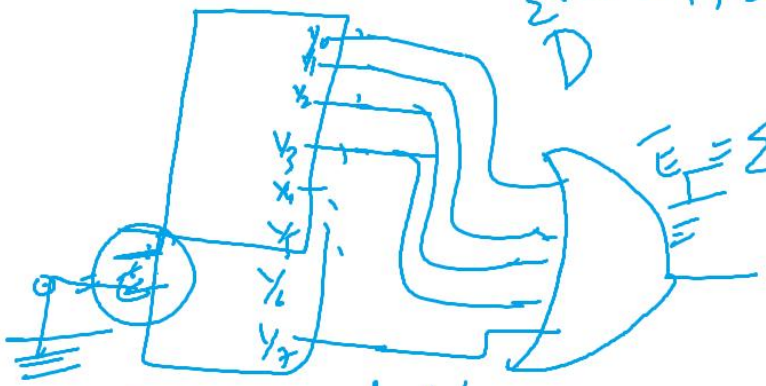
Realization of multiple o/p for using Bin. Decoder

1) For active HIGH o/p $\Rightarrow$ o/p = 1
   SOP (minterm) implement
   POS (maxterm) ↓ " —
2) For active LOW o/p $\Rightarrow$ o/p = 0
   SOP → ?
   POS → ?

HIGH o/p : 1

SOP → DNF

→ OR ing



$\Sigma m = 1, 2, 4, 7$

D

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

$D = \overline{\overline{D}}$

$m_0 = \bar{x} \cdot \bar{y} \cdot \bar{z}$ → minterm - product term

$\overline{m_0} = \overline{\bar{x} \cdot \bar{y} \cdot \bar{z}}$

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

$M_0 = (x + y + z)$ → maxterm - sum term

$m = \overline{M}$

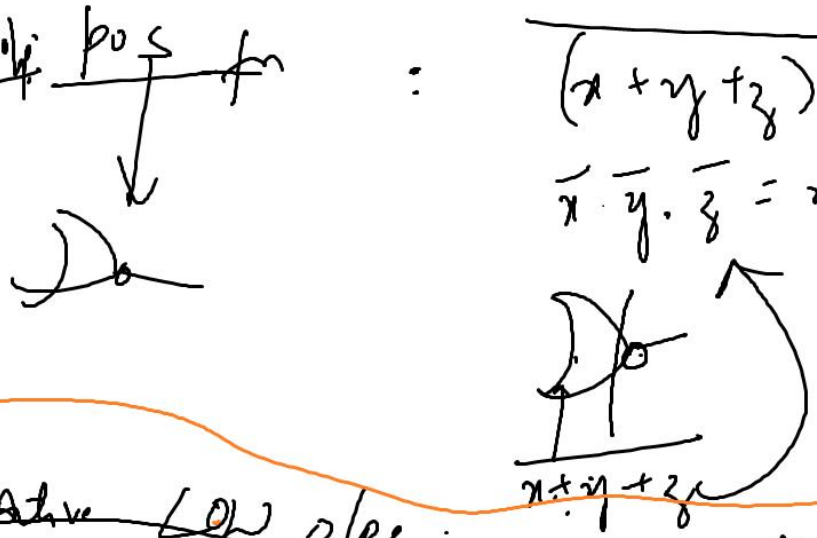
(or)

$\overline{m} = M$

$D = \overline{\overline{D}}$

High if POS fn : $\overline{(x+y+z)}$ = max

$\bar{x}\cdot\bar{y}\cdot\bar{z}$ = min



For Active LOW o/ps:

POS → AND gate

SOP → NAND gate

For Active HIGH o/ps

SOP → OR gate

POS → NOR gate

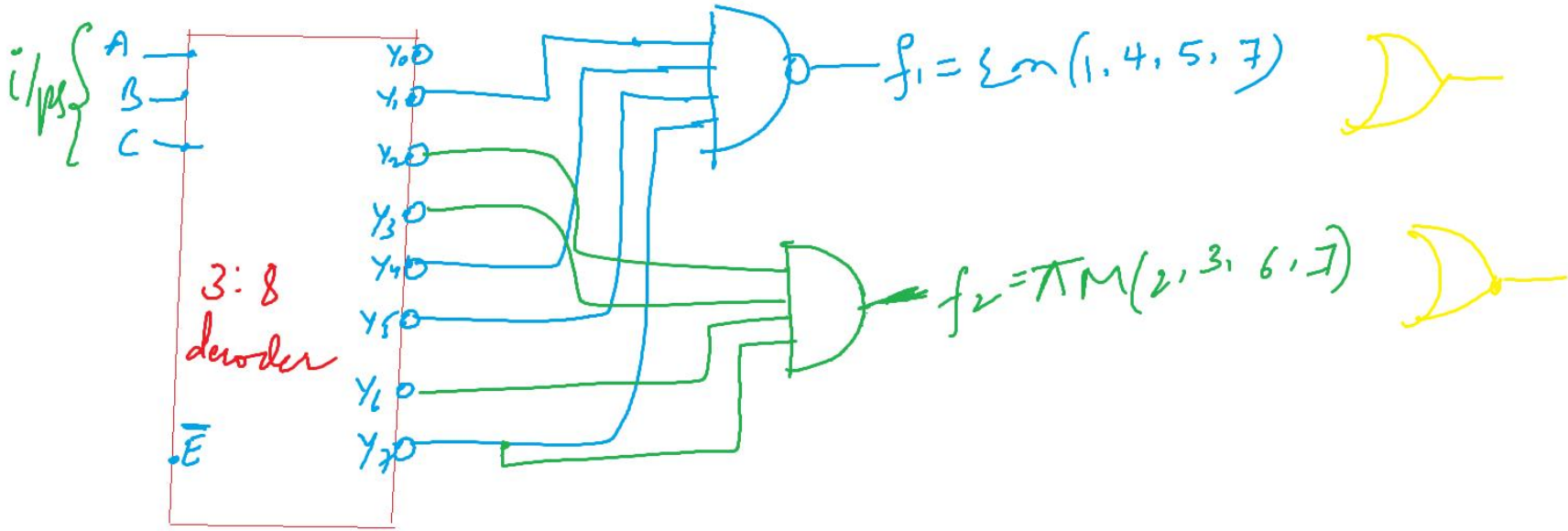
Implement following multiple o/p fn using 74138 (Active LOW o/ps)

$f_1 = \Sigma m(1, 4, 5, 7)$ — NAND, OR

$f_2 = \Pi M(2, 3, 6, 7)$ — AND, NOR

Active HIGH o/ps

Soln:

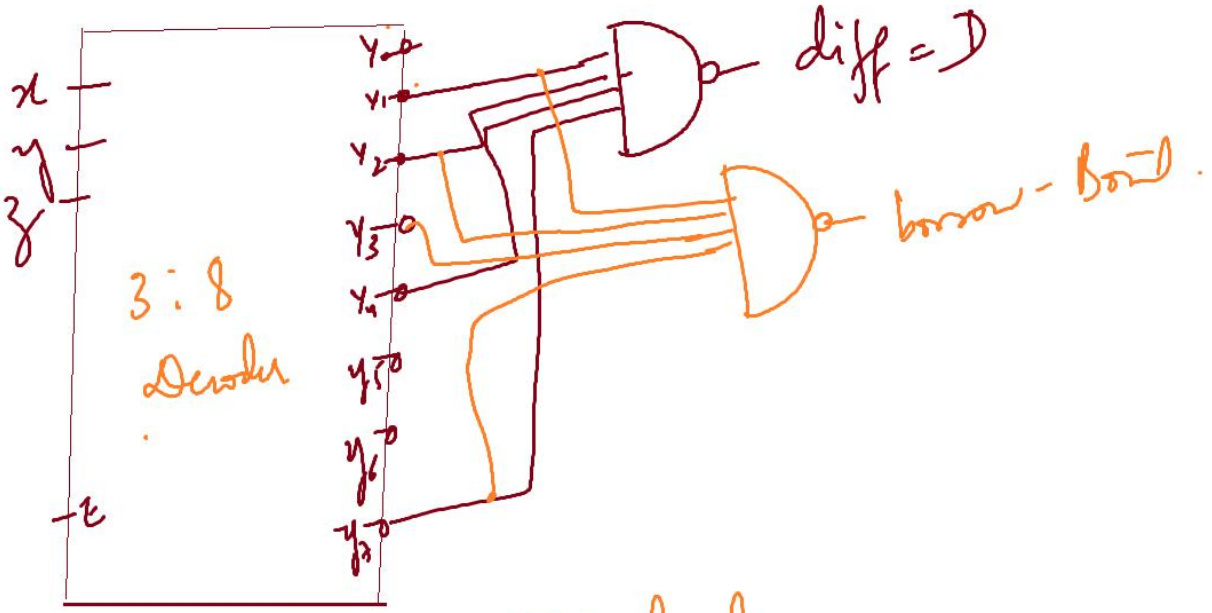


Implement full subtractor using a decoder and write TT.

	x	y	z	D	Bout
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

T.T for F.S

1 high



Implements of FS using 3:8 decoder

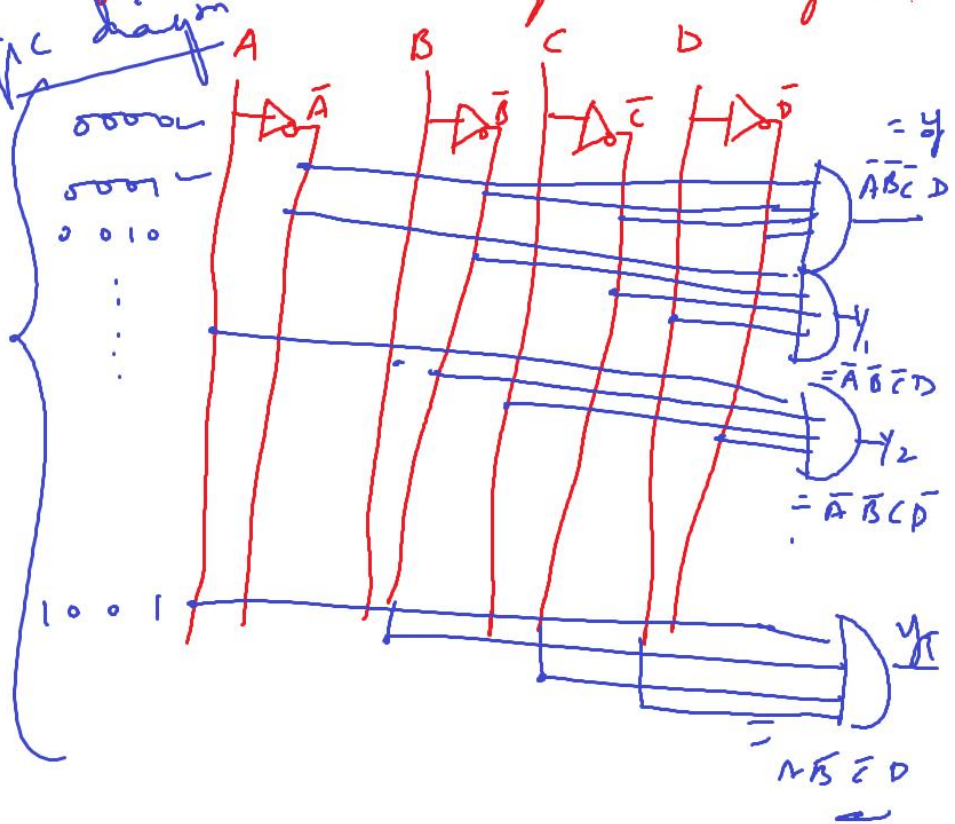
BCD to decimal decoder : 1 of 10-decoder. (only 1 of the 10 o/ps is high)

Binary Coded Decimal — Decimal decoder

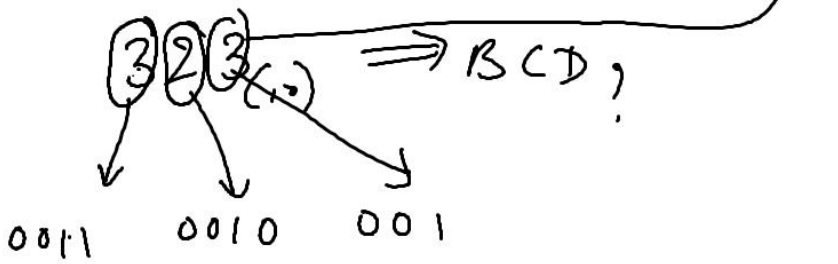
(16) $\frac{(0-9)}{(10-15)}$ X (7445) — LOW o/ps

	A	B	C	D	y_9	y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
→	0	0	0	0	1	1	1	1	1	1	1	1	1	0
	0	0	0	1	1	1	1	1	1	1	1	1	0	1
	0	0	1	0	1	1	1	1				0	1	1
	0	0	1	1							0	1	1	1
	0	1	0	0						0	1	1	1	1
	0	1	0	1					0	1	1	1	1	1
	0	1	1	0				0	1	1	1	1	1	1
	0	1	1	1			0	1	1	1	1	1	1	1
	1	0	0	0		0	1	1	1	1	1	1	1	1
	1	0	0	1	0	1	1	1	1	1	1	1	1	1

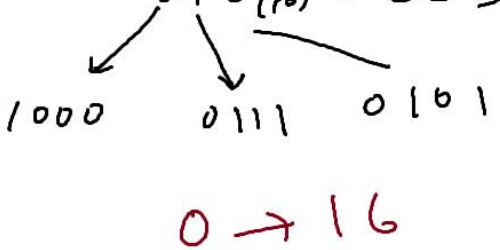
Logic diagram



BCD : nibble equivalent (4-bit binary no).



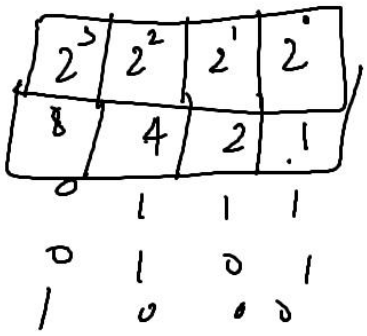
$875_{(10)}$ = BCD - ?



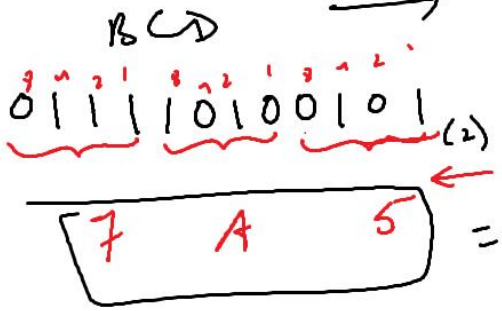
0 → 16

2 | 3
1 - 1

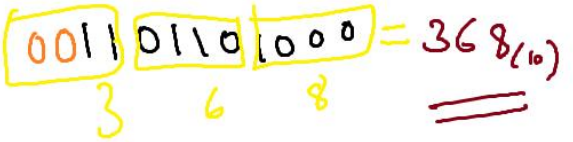
2 | 7
2 | 3 - 1
1 - 1



BCD → Decimal eqn)

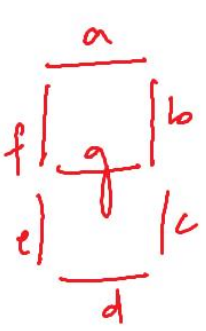


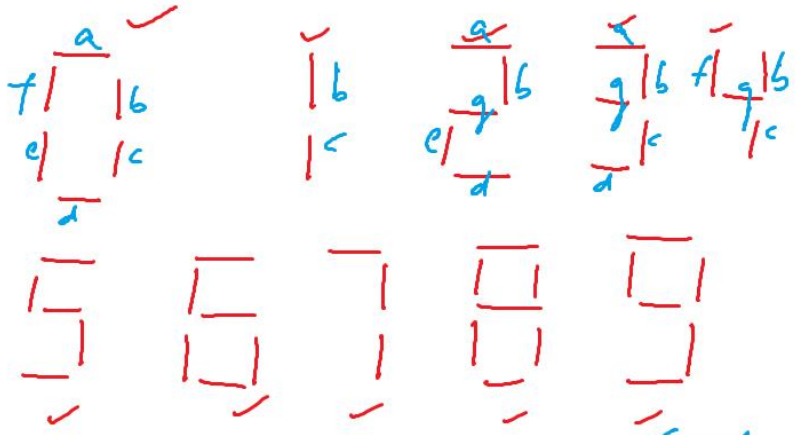
10 - A
11 - B
12 - C
13 - D



① BCD - Decimal

② 7 segment decoder – 7446



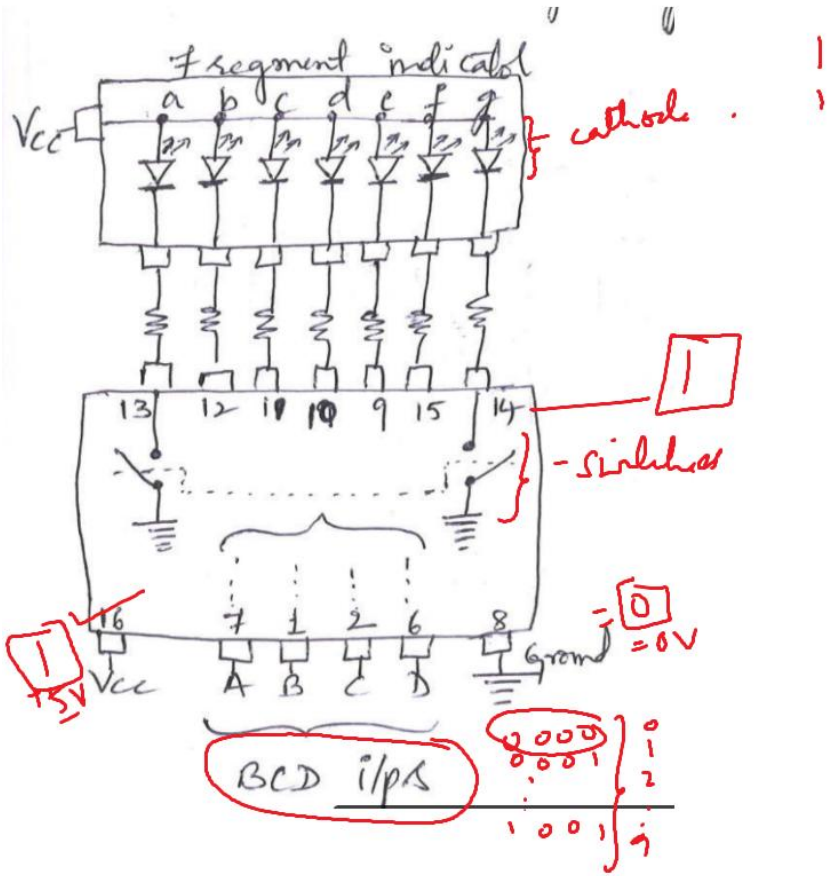


Digit	Segments activated	Display
0	a, b, c, d, e, f	
1	b, c	
2	a, b, d, e, g	
⋮	⋮	⋮
8	a, b, c, d, e, f, g	
9	a, b, c, d, f, g	

0

(10 digits)

7 Segment indicator


7 segment indicator
a b c d e f g
Vcc
cathode
13 12 11 10 9 15 14
1
switches
16 7 1 2 6 8
0
=0V
1
5V
Vcc
Ground
A B C D
BCD i/ps
0000
0001
1001
0
1
2
9

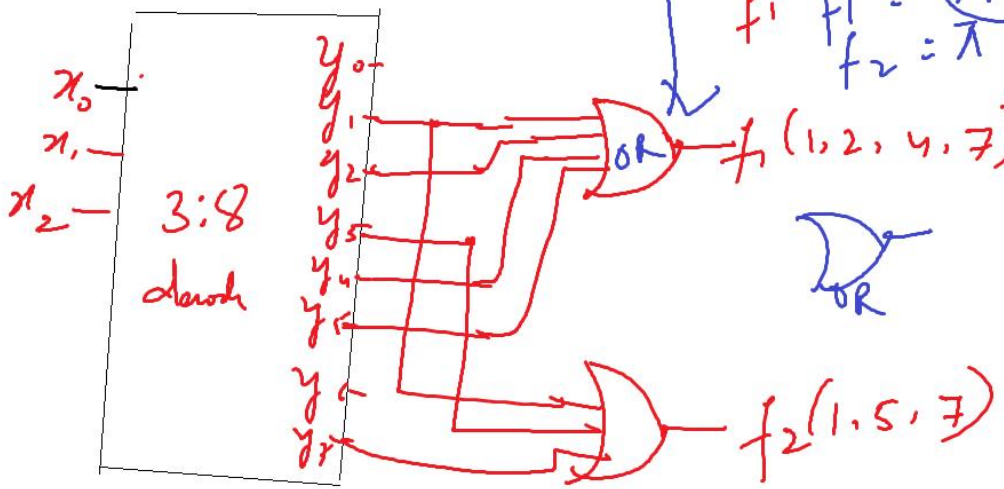
Logic design using decoders:

1) Realize / Implement 3:8 decoder for given fns

$f_1(x_2, x_1, x_0) = \sum m(1,2,4,5)$ — byc 1

$f_2(x_2, x_1, x_0) = \sum m(1,5,7)$

(MSB x_2, LSB x_0)



NOR.

$f_1 = \prod M(1,2,4,7)$

$f_2 = \prod M(1,5,7)$

OR

comp NOR

SOP $= m_1 + m_2 + m_4 + m_7$ — OR

=

$m = \overline{M}$

$M = \overline{m}$

POS $= M_1 \cdot M_2 \cdot M_4 \cdot M_7$

$= \overline{m_1 \cdot m_2 \cdot m_4 \cdot m_7}$

$= \overline{m_1} + \overline{m_2} + \overline{m_4} + \overline{m_7}$

$= \overline{m} = \overline{\sum m}$

$= \prod M$

LOW – o/p :

(7445 - IC)

Sop (Σm) = NAND

Pos (πM) = AND

7445: LOW o/ps →

ACTIVE HIGH o/p

Sop (Σm) = OR

pos (πM) = NOR

Default

Encoders:

74157 — low i/p

74157 — active-low output

Decimal value	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	A	B	C	D
	Inputs (10 i/ps)										Outputs (4 o/ps)			
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	0
2	1	1	0	1	1	1	1	1	1	1	1	1	0	1
3	1	1	1	0	1	1	1	1	1	1	1	1	0	0
4	1	1	1	1	0	1	1	1	1	1	1	0	1	1
5	1	1	1	1	1	0	1	1	1	1	1	0	1	0
6	1	1	1	1	1	1	0	1	1	1	1	0	0	1
7	1	1	1	1	1	1	1	0	1	1	1	0	0	0
8	1	1	1	1	1	1	1	1	0	1	0	1	1	1
9	1	1	1	1	1	1	1	1	1	0	0	1	1	0

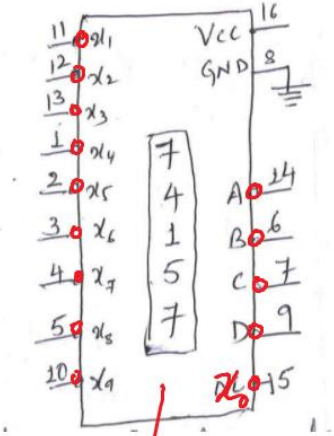
3 of LOW i/p - LOW o/p

HIGH i/o/ps

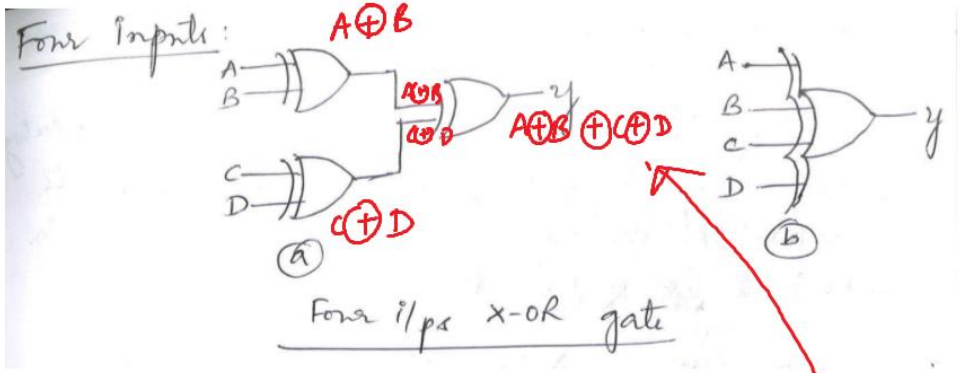
... x_9	A	B	C	D
	0	0	0	0
	0	0	0	1
	0	0	1	0
	0	0	1	1
	0	1	0	0
	0	1	0	1
	0	1	1	0
	0	1	1	1
0	1	0	0	0
1	1	0	0	1

74157

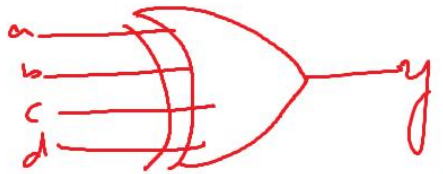
10 - i/p lines
4 - o/p lines
1 - Vcc
1 - GND
16 - pin IC



2^n : n Encoder



two - 2 i/ps Ex-OR gate
one - 4 i/ps EX-OR gate



$y = a \oplus b \oplus c \oplus d$

(0, 2, 4, 8 ...)

Even no of 1's = odd parity

Odd no. of 1's = Even parity

(1, 3, 5, 9 ...)

} generate & check the parity bit in binary i/p ___ EX-OR gates are used.

Truth Table : When ABCD i/ps has an odd no of 1s, o/p = 1

Comment	A	B	C	D	y
Even	0	0	0	0	0
Odd	0	0	0	1	1
Odd	0	0	1	0	1
Even	0	0	1	1	0
Odd	0	1	0	0	1
Even	0	1	0	1	0
Even	0	1	1	0	0
Odd	0	1	1	1	1
Odd	1	0	0	0	1
Even	1	0	0	1	0
Even	1	0	1	0	0
Odd	1	0	1	1	1
Even	1	1	0	0	0
Odd	1	1	0	1	1
Odd	1	1	1	0	1
Even	1	1	1	1	0

First ABCD entry to product an o/p 1 is 0001 ; it has odd no of 1s. The next ABCD entry to product an o/p 1 is 0010 ; again an odd no of 1s. An o/p-1 also occurs for these ABCD i/ps 0100, 1000, 1011, 1101, and 1110, each having an odd no of 1s.

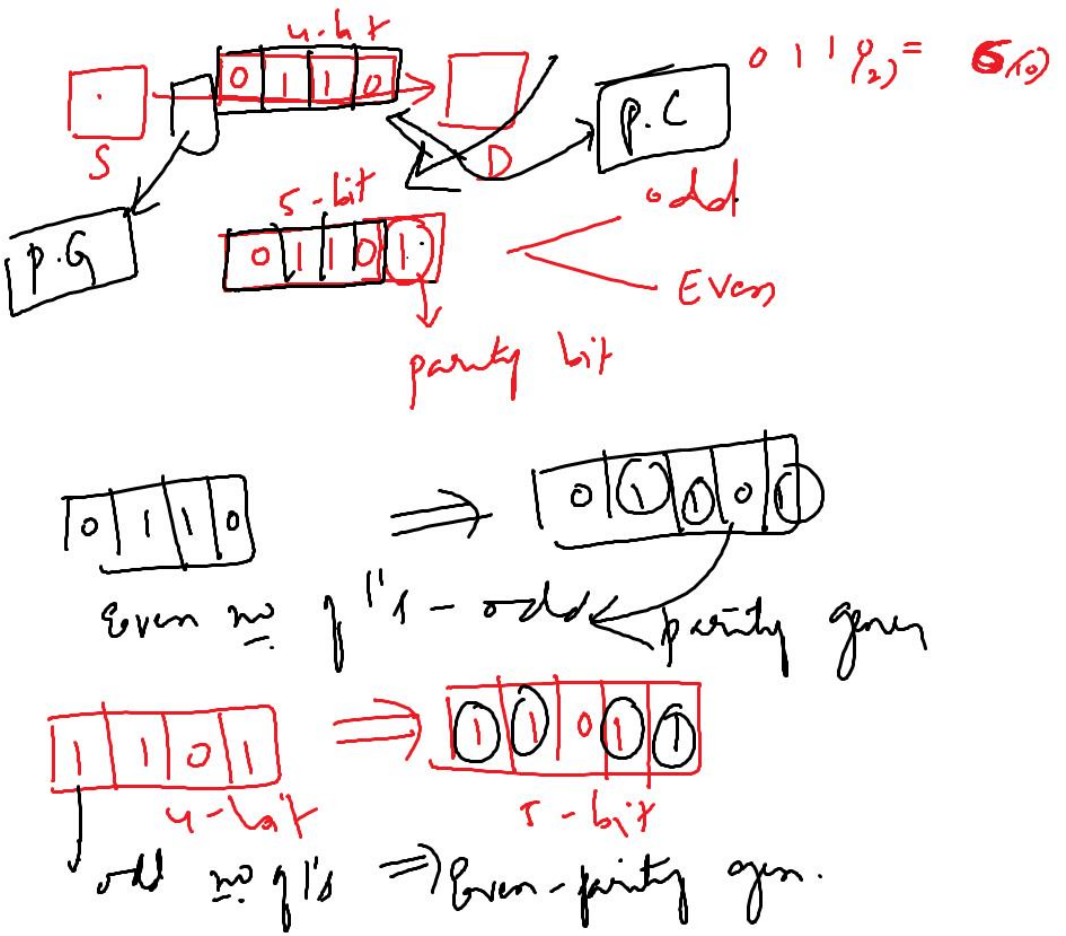
Parity Generators and checkers:

→ A parity-bit is used for the purpose of detecting errors during transmn of binary inform". It is an extra-bit included with a binary message to make the no. of 1s either odd or even.

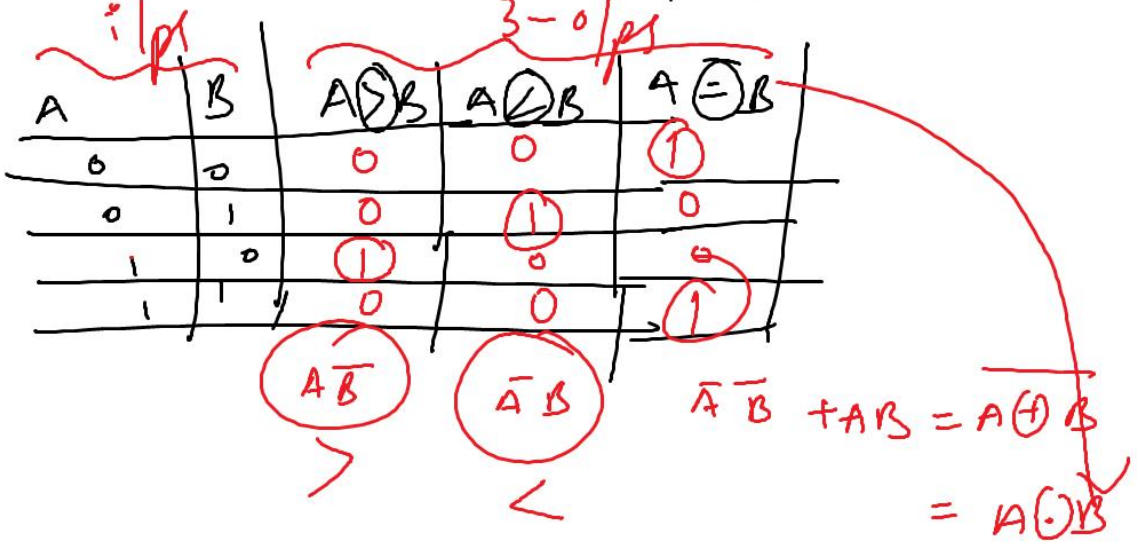
→ The message, including the parity bit is transmitted & then checked at the receiving end for errors. An error is detected if the checked parity does NOT correspond with the one transmitted.

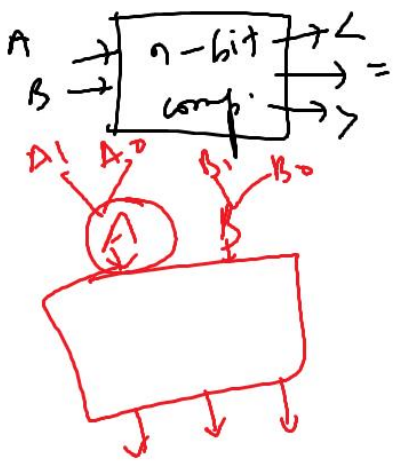
→ Parity generator: Circuit that generates the parity bit in the transmitter.

→ Parity checker: Circuit that checks the parity bit in the receiver.

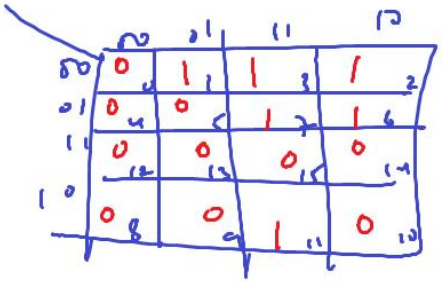


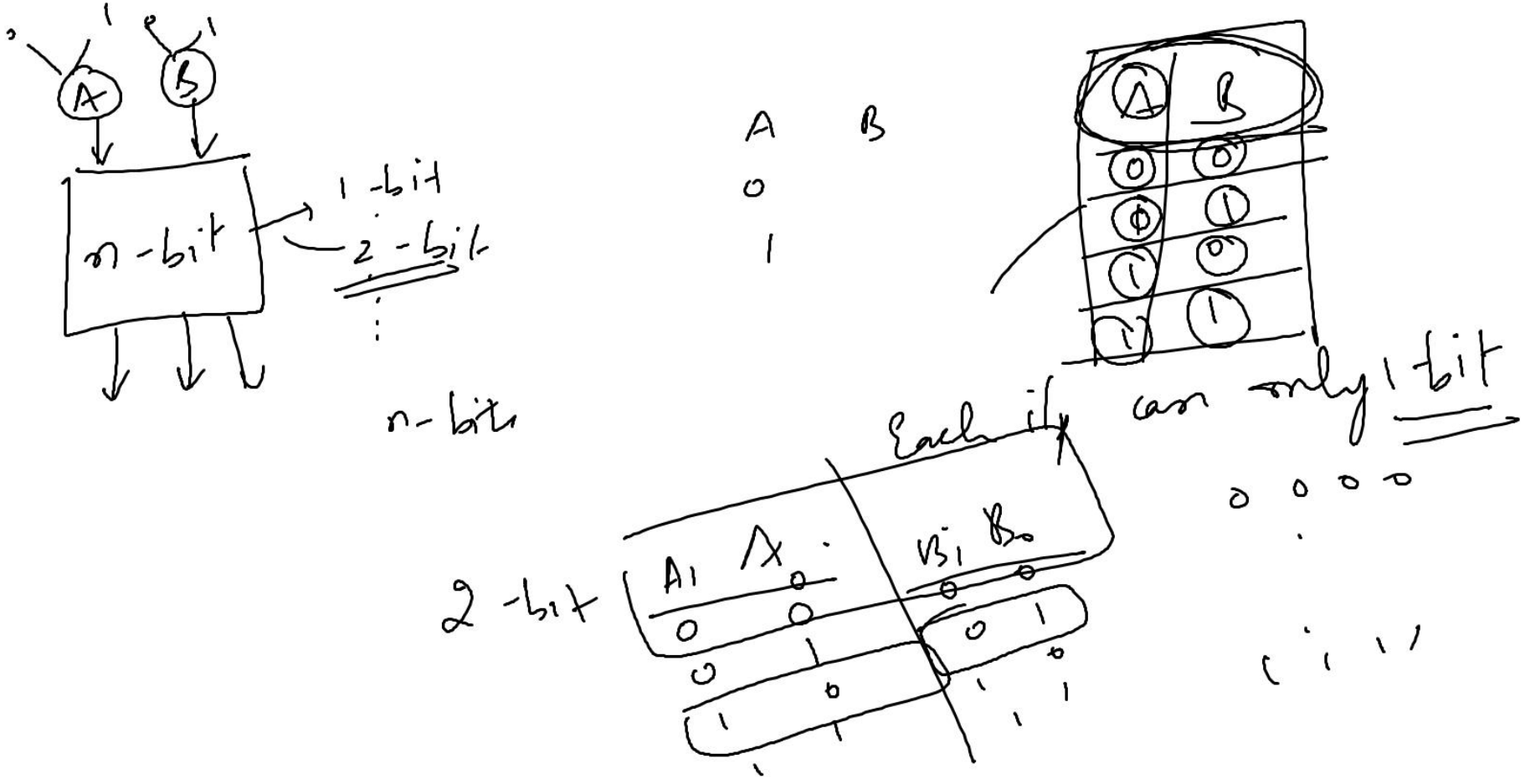
Comparator: Combinational ckt → magnitude of any 2 nos.
Related





	A		B		Less	Great	Equal
	A_1	A_0	B_1	B_0	A<B	A>B	A=B
0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	0	0
4	0	1	0	0	0	1	0
5	0	1	0	1	0	0	1
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	0
8	1	0	0	0	0	1	0
9	1	0	0	1	0	1	0
10	1	0	1	0	0	0	1
11	1	0	1	1	1	0	0
12	1	1	0	0	0	1	0
13	1	1	0	1	0	1	0
14	1	1	1	0	0	1	0
15	1	1	1	1	0	0	1




A
B
n-bit
1-bit
2-bit
n-bit
A B
Each if can only 1 bit
2-bit
A1 A0
B1 B0

Programmable Logic Arrays / Programmable Array Logic

(PLAs) (PALs)

Programmable logic devices : n x p x m

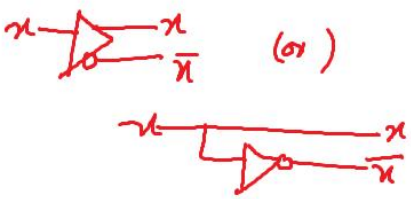
(PLDs)

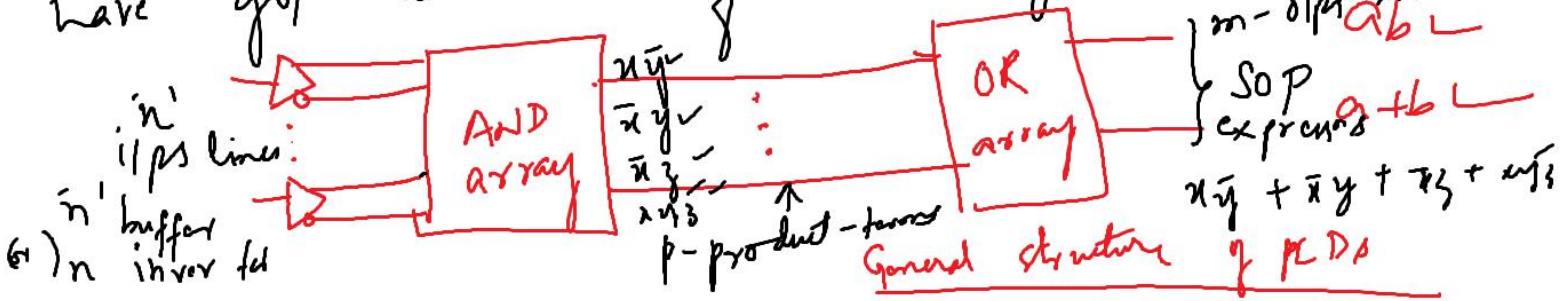
- H/w programming
- Combinational ckts
- Sequential ckt

current i/p – o/ps – Combin

current i/p + prevm – o/ps – Seq

→ ICs which have got the collection of AND arrays and OR arrays



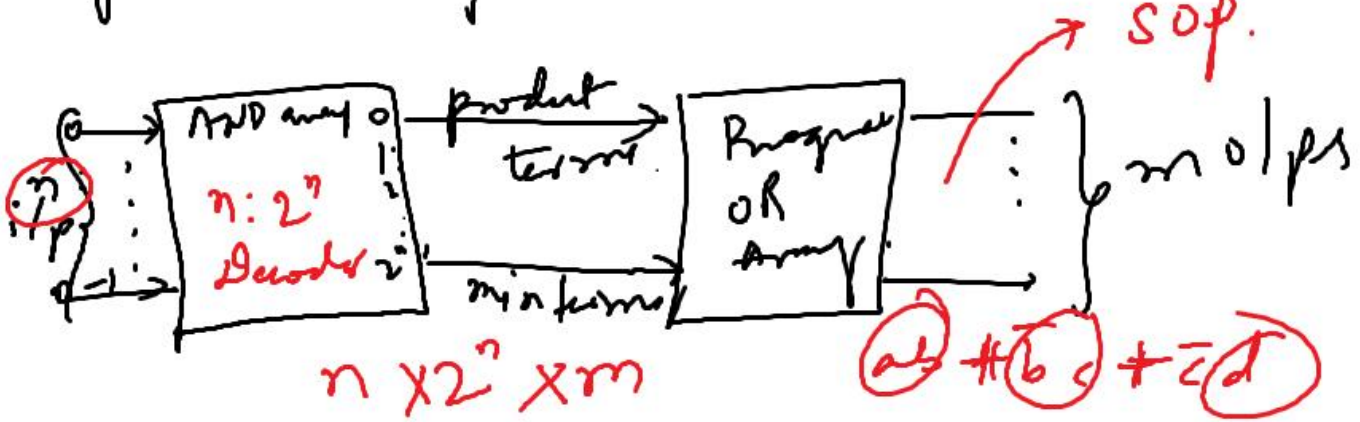


General structure of PLDs

PLD : 3 types → AND arrays & OR arrays

1) PROM — Fixed — Programmable
2) PLA — Programmable — Programmable
3) PAL — Programmable — Fixed

1) PROM : Programmable ROM



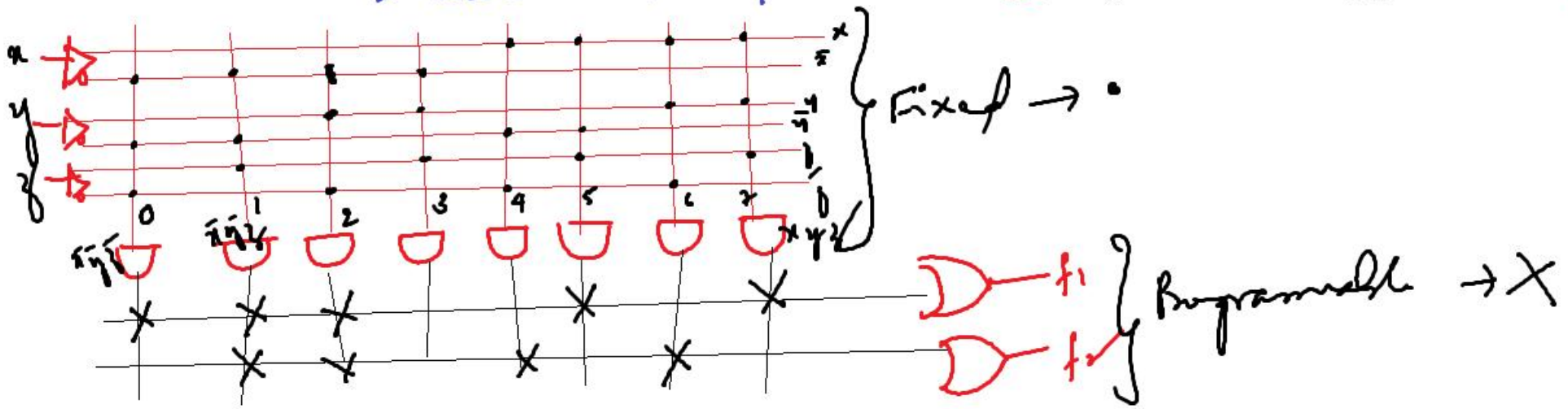
Eg: PROM: Realize a PROM for given 2 fns

AND array - fixed
OR array - Programmable

1) $f_1(x, y, z) = \Sigma m(0, 1, 2, 5, 7)$

$f_2(x, y, z) = \Sigma m(1, 2, 4, 6)$

No: 3-vars: $\Rightarrow 2^n$ combinations $\Rightarrow 2^n$ product term $= 2^3 = 8$ minterms. $\Rightarrow$ AND gate for each



PROM : $3 \times 2^3 \times 2 \Rightarrow 3 \times 8 \times 2$

IC

PLA : Programmable Logic array : $3 \times 4 \times 2$ ($n \times p \times m$)

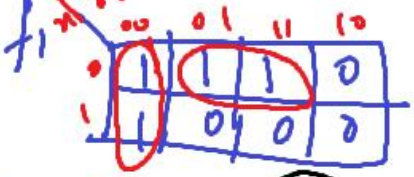
Realize PLA for given 2 fn's.

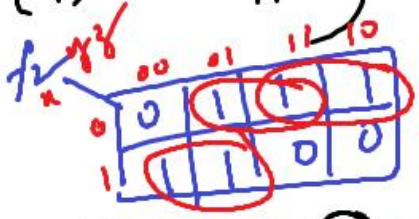
1) $f_1(x, y, z) = \sum m(0, 1, 3, 4)$

$f_2(x, y, z) = \sum m(1, 2, 3, 4, 5)$

$3 \times 5 \times 2$ → 4

Soln :

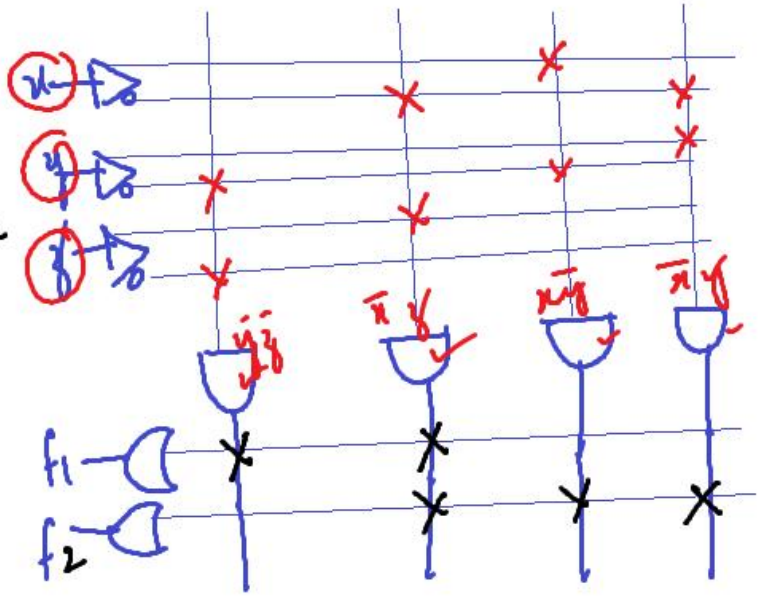




$f_1 = \bar{y}\bar{z} + \bar{x}z$

$f_2 = x\bar{y} + \bar{x}\bar{z} + \bar{x}y$

7404 : 4 OR gates



3 × 8 × 2 — PROM

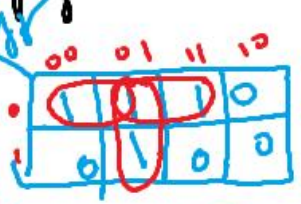
3 × (4) × 2 PLA

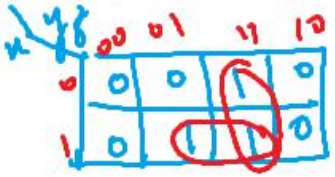
Eg(2): Realize PLA for the following fns

$f_1(x,y,z) = \Sigma m(0,1,3,5)$

$f_2(x,y,z) = \Sigma m(3,5,7)$

Soln:





$f_1 = \bar{x}\bar{y} + \bar{x}z + \bar{y}z$

5 product terms

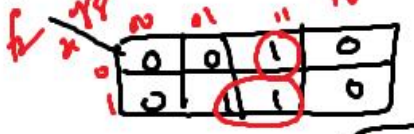
$f_2 = xz + yz$

⇓ 3 × 5 × 2

Totally 5 terms but 3 × (4) × 2 PLA is available

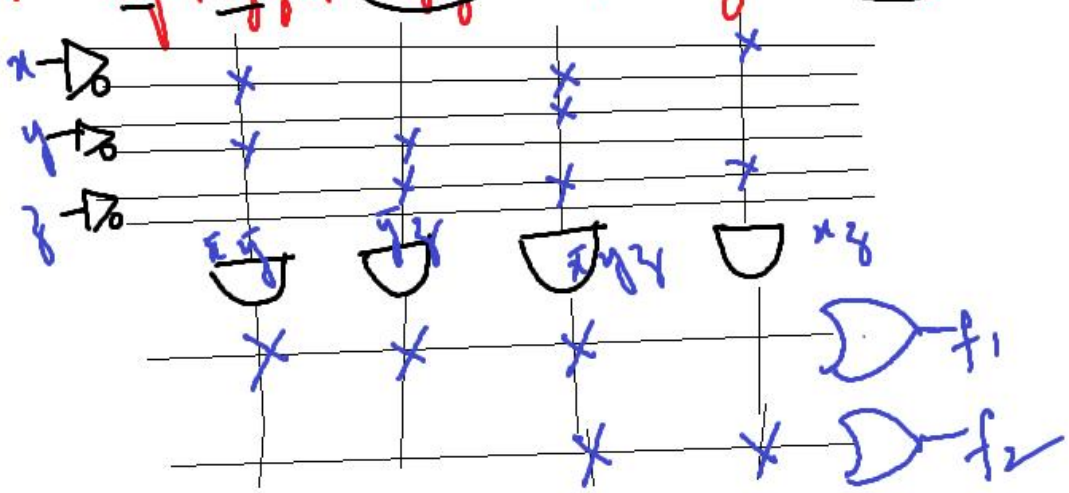
∴ only 4 product terms

∴



$f_1 = \bar{x}\bar{y} + \bar{y}z + \bar{x}yz$

$f_2 = xz + \bar{x}yz$



PAL : Programmable Array Logic : AND - programable — X
OR - fixed

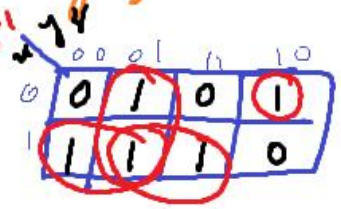
→ Max. of 3 product terms are realizable in PAL (for each fn)

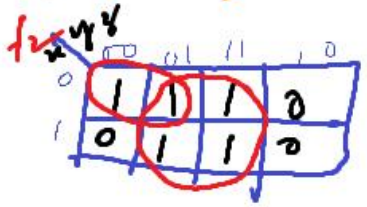
① Realize PAL for the given fns

$f_1(x, y, z) = \Sigma m(1, 2, 4, 5, 7)$

$f_2(x, y, z) = \Sigma m(0, 1, 3, 5, 7)$

Soln :





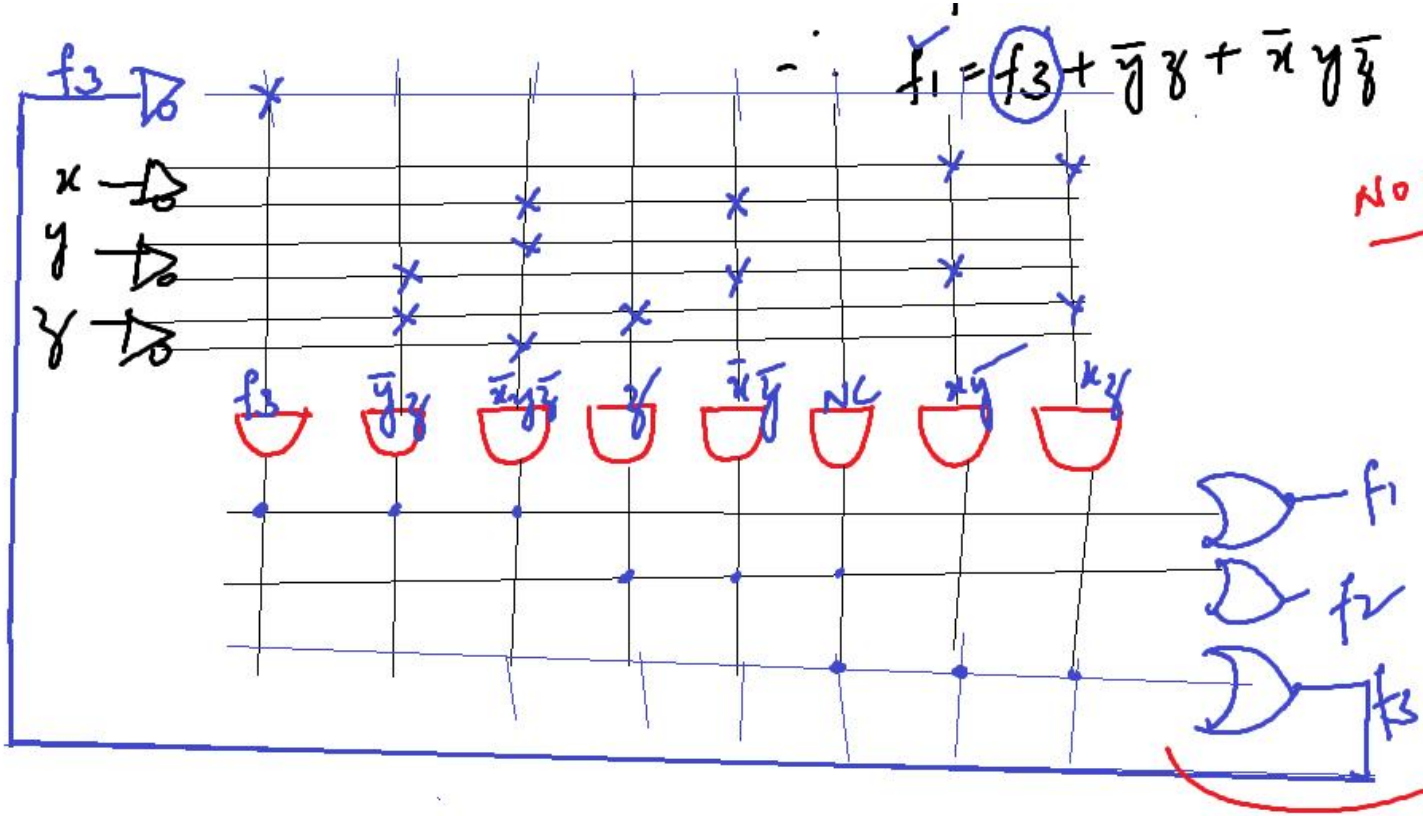
$f_1 = \underbrace{x\bar{y} + xz}_{f_3} + \bar{y}z + \bar{x}y\bar{z}$

$f_2 = z + \bar{x}\bar{y} + NC$

$\therefore f_1 = f_3 + \bar{y}z + \bar{x}y\bar{z}$

$f_2 = z + \bar{x}\bar{y} + nc$

$f_3 = x\bar{y} + xz + nc$



$f_1 = f_3 + \bar{y}z + \bar{x}y\bar{z}$

$f_2 = z + \bar{x}\bar{y} + xc$

$f_3 = x\bar{y} + xz + xc$

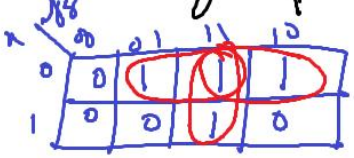
NOTE: For each f_n, there must be max of 3 product term.

$3 \times 8 \times 2$

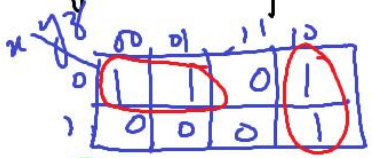
PLA: 3×4×2 Realize PLA for the given f^n's: (i) $f_1(x,y,z) = \Sigma m(1,2,3,7)$

$f_2(x,y,z) = \Sigma m(0,1,2,6)$

Soln:



$f_1 = \bar{x}z + \bar{x}y + yz$



$f_2 = \bar{x}\bar{y} + y\bar{z}$

No common term(s) in b/w f^n's, take complement of (grouping 0's)

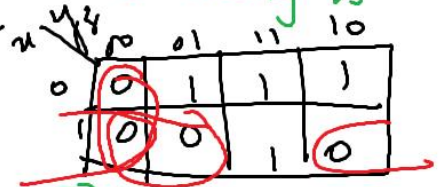
$f_1 = \bar{x}z + \bar{x}y + yz$

$f_2 = \bar{x}y + y\bar{z}$

$\bar{f}_1 = \bar{y}\bar{z} + x\bar{z} + xy$

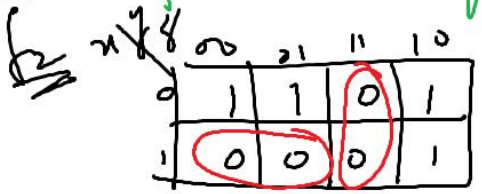
$\bar{f}_2 = x\bar{y} + yz$

$\bar{f}_1$ both the f^n's



$\bar{f}_1 = \bar{y}\bar{z} + x\bar{z} + x\bar{y}$

$\bar{f}_2$



$\bar{f}_2 = x\bar{y} + yz$

Compare to get common minterms: f_1 with f_2; f_1 with $\bar{f}_1$ (or) f_2; f_2 with $\bar{f}_2$; $\therefore$ f_1 with $\bar{f}_2$ (or) f_2 with $\bar{f}_1$; $\therefore$ $\bar{f}_1$ with $\bar{f}_2$

∴ Case 1: $f_1(x,y,z) = \bar{x}z + \bar{x}y + yz$

$\bar{f}_2(x,y,z) = x\bar{y} + yz$ } 4 terms

∴ Case 2: $\bar{f}_1(x,y,z) = \bar{y}\bar{z} + x\bar{z} + x\bar{y}$

$f_2(x,y,z) = x\bar{y} + yz$ } 4 terms

Compared f_1 with f_2, ($\bar{f}_1$) $\bar{f}_2$: to get common terms

f_2 with f_1, $\bar{f}_1$, $\bar{f}_2$ → Implement → Ex-OR

$f_1 = \bar{x}z + \bar{x}y + yz$

$f_2 = \bar{x}y + y\bar{z}$

$\bar{f}_1 = \bar{y}\bar{z} + x\bar{z} + x\bar{y}$

$\bar{f}_2 = x\bar{y} + yz$

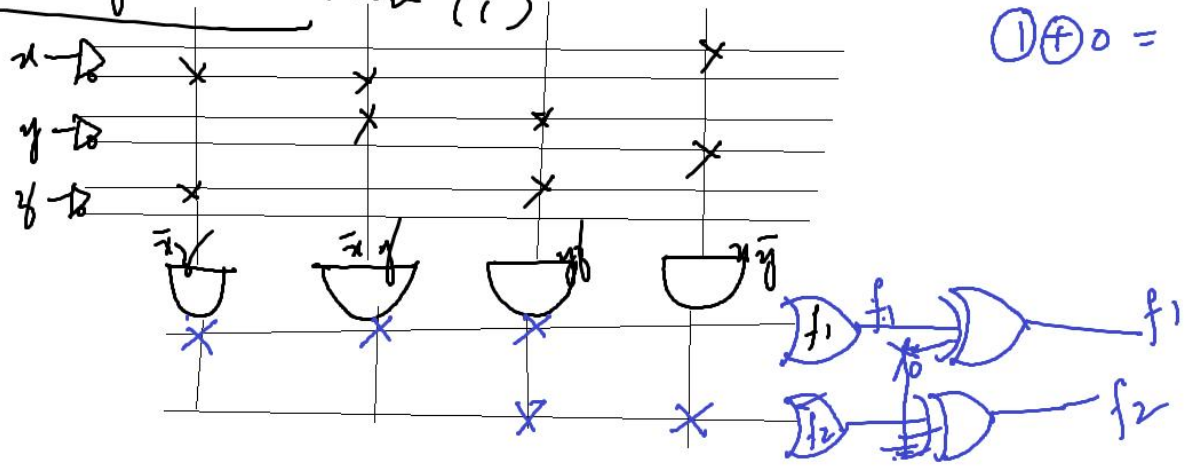
∴ f_1 with $\bar{f}_2$

i.e. $x\bar{y}$ T.

$1 \oplus 0 =$

	0	0	0
0			1
1	0		1
			0

PLA diagram not : Case (i)



Case 2: $\bar{f}_1(x,y,z) = \bar{y}\bar{z} + x\bar{z} + x\bar{y}$

$\bar{f}_2(x,y,z) = x\bar{y} + yz$

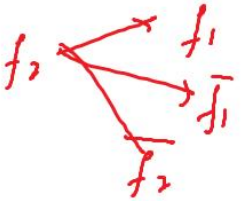
} 4 terms

PLA diagram note: Case (ii)

Note: When NO common product term(s) is/are found, then take complement of both the f's (given).

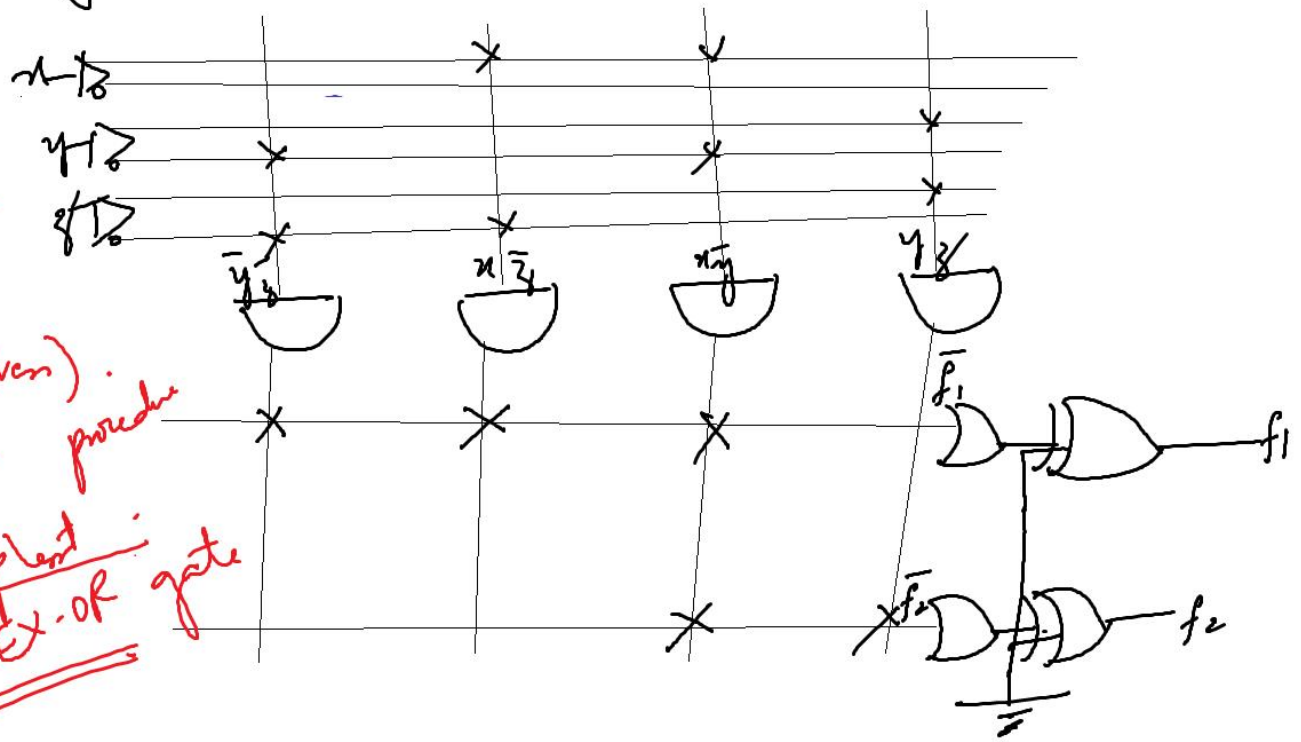
$f_1 \rightarrow \bar{f}_1$

$f_2 \rightarrow \bar{f}_2$



→ PLA procedure

→ Implement using EX-OR gate



Common way of specifying the connections in PLA

PLA table: case (i)

Product terms	Inputs x	y	z	Outputs f_1	f_2
$\bar{x}z$	0	–	1	1	–
$\bar{x}y$	0	1	–	1	–
yz	–	1	1	1	1
$x\bar{y}$	1	0	–	–	1

Inputs: 0 → complemented form
1 → Uncomplemented form/true var
– → No connection exists

case (ii)

Product terms	Inputs x	y	z	Outputs f_1	f_2
$\bar{y}\bar{z}$	–	0	0	1	–
$x\bar{z}$	1	–	0	1	–
$x\bar{y}$	1	0	–	1	1
yz	–	1	1	–	1

Outputs: 1 → connection exists b/w AND gate & OR gate
– → No connection exists

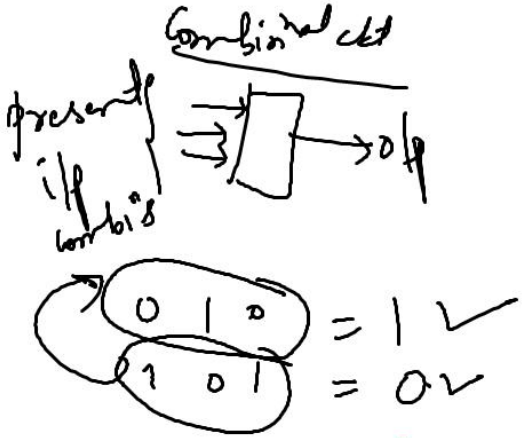
Unit-3: Flip-flops → Sequential ckts

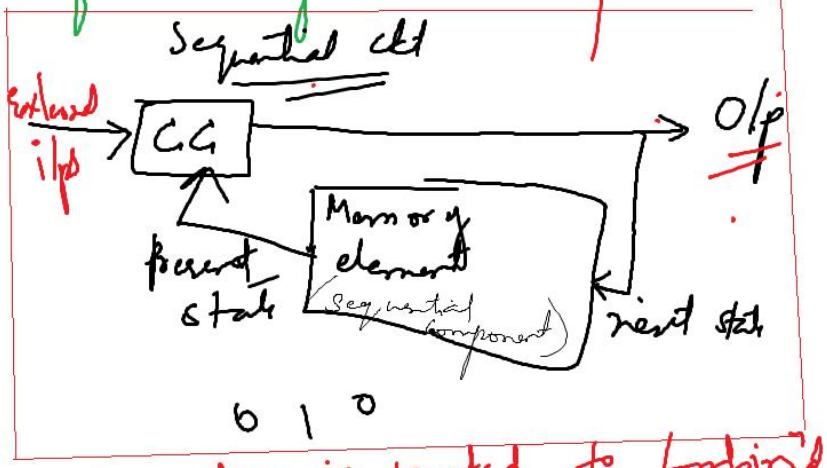
There are many applns in which the digital o/ps are reqd to be generated in accordance with the seq of i/p signals are received. This reqt can NOT be satisfied using combinational ckts

Finite State M/c (FSM)

Sequential ckt: Time sequence of external i/ps, internal states (present & next state) & o/ps

Eg: Counters & Registers

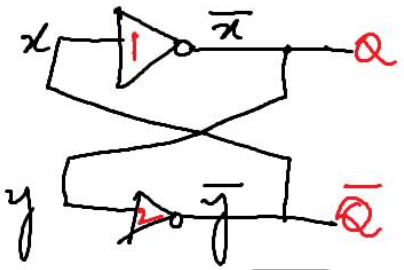




Mem elems is connected to combinational ckt

Memory Element used in the sequential ckts is a FLIP-FLOP which is capable of storing 1-bit binary information

The basic bistable element: NOT (Inverter)

① 

2 i/ps 'x' and 'y'
2 o/ps 'Q' and '$\bar{Q}$'
2 cross-coupled NOT gates (feedback)

When $x=0$: $Q = \bar{x} = 1$

$\because y = \bar{x} = 1 \Rightarrow \bar{Q} = \bar{y} = 0$

$Q = \bar{x} = y = 1$ & $\bar{Q} = \bar{y} = x = 0$

$\therefore Q = 1$ and $\bar{Q} = 0$ — first stable cond^{n}s.

When $x=1$:

$Q = \bar{x} = 0$

$\therefore y = \bar{x} = 0 \Rightarrow \bar{Q} = \bar{y} = 1$

$Q = \bar{x} = 0$ & $\bar{Q} = \bar{y} = x = 1$

$\therefore$ $Q = 0$, $\bar{Q} = 1$ — second stable state

Positive logic:

o/p $Q = 1 \rightarrow$ state 1 $\Downarrow$ 1-state (SET)

o/p $Q = 0 \rightarrow$ state 0 $\Downarrow$ 0-state (RESET)

When $Q=0$, then $\bar{Q}=1$ and $Q=1$, $\bar{Q}=0$

$\therefore Q$ & $\bar{Q}$ are complementary

Binary symbol stored in the basic bistable element — content / state of the element

$Q \Rightarrow$ normal o/p

$\bar{Q} \Rightarrow$ Complementary o/p.

Equilibrium condn: 2 o/ps signals are about half-way b/w logic-0 and logic-1
o/p is NOT valid — metastable state.

Amt of time a device stays in a metastable state is unpredictable due to clk NOISE. ∴ Must be avoided.

Latches: storage devices (o/p immediately responds to changes in the i/p lines)

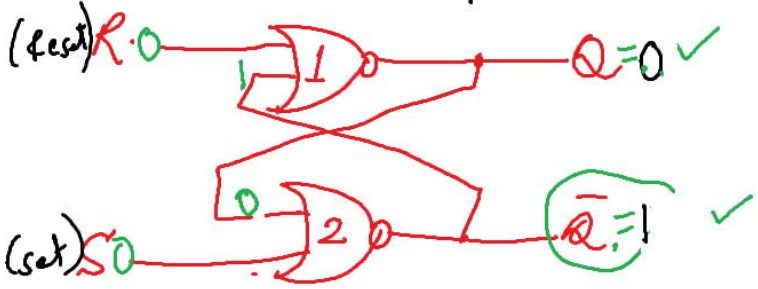
1) The Set & Reset Latch (SR-latch) → 2 i/ps (1) S → Set, R → Reset

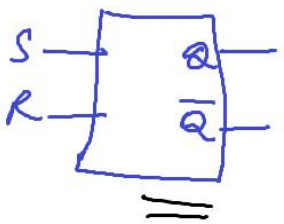
2 cross-coupled NOR gates

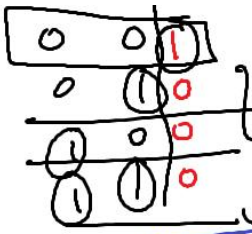
2 o/ps: Q, $\bar{Q}$

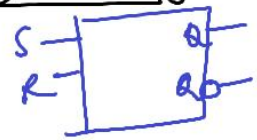
Latch → 1-bit memory cell

SR: 2 cross-coupled NOR gates









S	R	Q	$\bar{Q}$
0	0	9	-
0	1	9	-
1	0	9	
1	1	9	

(i) case 1: When S = 0 and $\bar{R}$ = 0

Initially assume: Q = 1 and $\bar{Q}$ = 0 ⇒ Q = 1 and $\bar{Q}$ = 0

NOR-1 ⇒ 0 0

NOR-2 ⇒ 1 0

With $\bar{Q}$ = 0, both i/ps to NOR-1 are @ logic-0.

So the o/p Q = 1.

With Q = 1, one i/p to NOR-2 is @ logic-1, hence its o/p $\bar{Q}$ = 0.

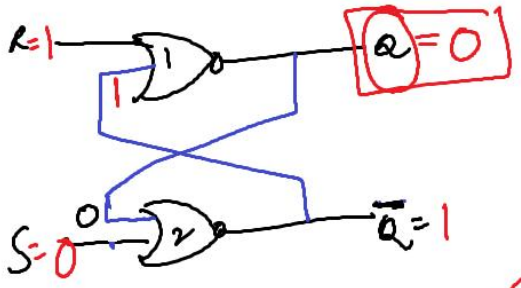
Thus shows that when S = R = 0 (low), the o/p state does NOT change.

Q = 0 ✓ and $\bar{Q}$ = 1 ✓ ⇒ Q = 0 and $\bar{Q}$ = 1 NOR-1: 0 1 and NOR-2: 0 0

Case 2 : S = 0 and R = 1

i.e., Q = 0 and $\bar{Q}$ = 1

RESET state



S	R	Q	$\bar{Q}$
0	0	0	1
		1	0
0	1		Reset
1	0		Set
1	1		

No change in o/p

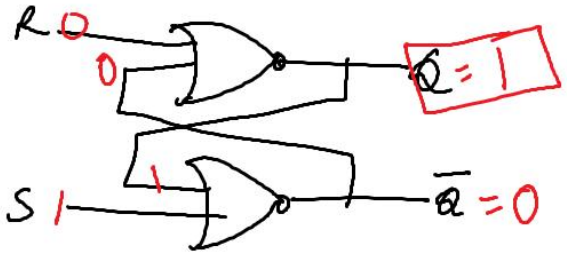
Reset = 0

Set = 1

NOR		
0	0	1
0	1	0
1	0	0
1	1	0

NOT i/p	o/p Q
0	1
1	0

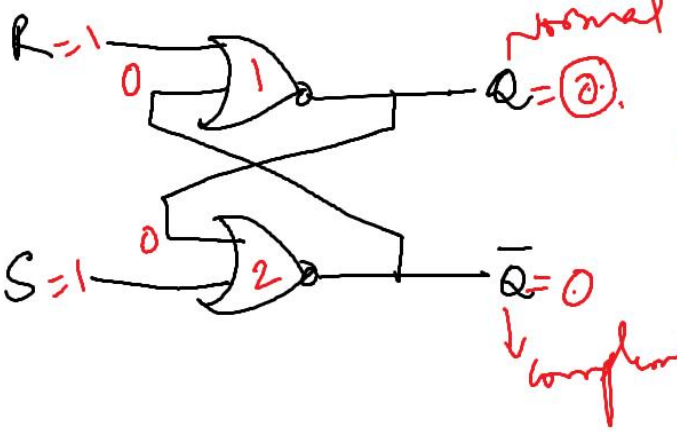
Case 3 : S = 1 and R = 0

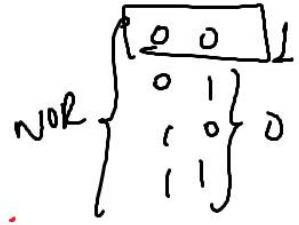


i.e., Q = 1 and $\bar{Q}$ = 0

SET state

Case 4: S=1 and R=1





S	R	Q
0	0	No change in o/p
0	1	Reset
1	0	set
1	1	Indeterminate

Indeterminate

$Q = \bar{Q} = 0$

Q = 0
$\bar{Q}$ = 0

$\Rightarrow \bar{Q} = 1$

prohibited state

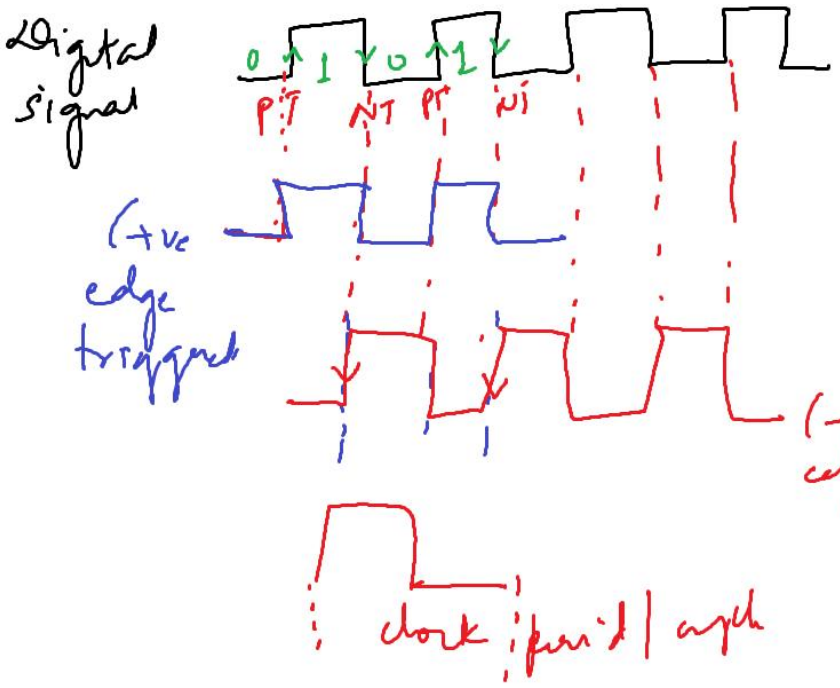
Truth table:

S	R	Q	$\bar{Q}$
0	0	No change	
0	1	RESET	
1	0	SET	
1	1	*	*

present state
next state

Q Q+

Clock : A clock signal is a particular type of signal that oscillates b/w HIGH state and LOW state and it is utilized to coordinate the actions of the circuits. It is produced by the clock generator

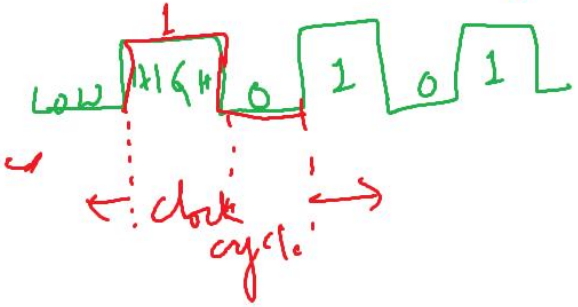


Digital signal : square wave

Analog signal : sine-wave.

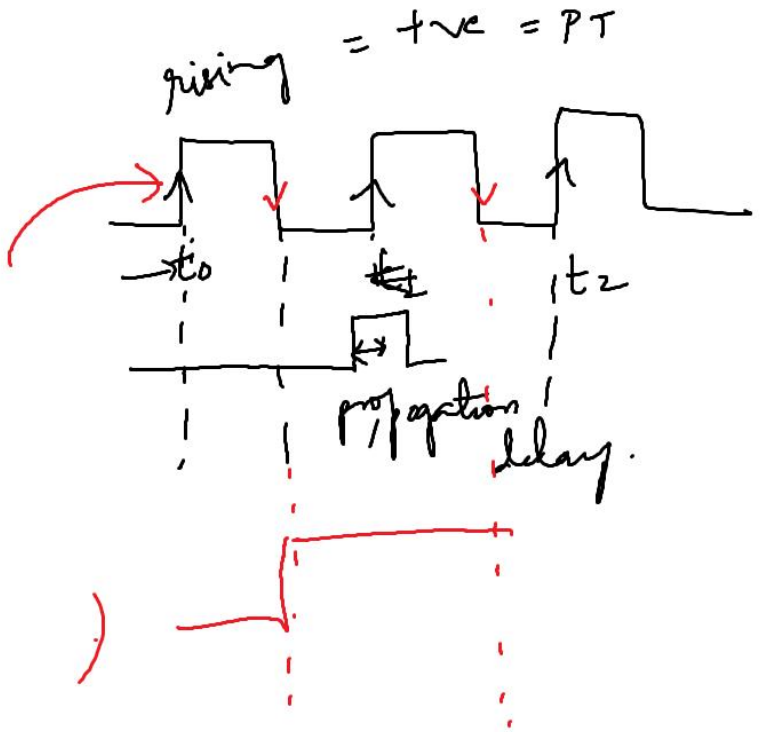
rising edge — rising edge (+ve edge triggered

falling edge (-ve edge trigg



Clock is set : clocked

+ve edge triggered



Flip flop: 0 / 1 → 1-bit of info^n 0 / 1

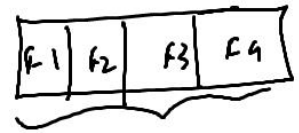
Registers: collection of flip flop

Arithmetic oper^n

Logic oper^n

4-bit regist = 4 flip flops = NIBBLE

8-bit reg = 8 FF = 1-byte



Shift registers :

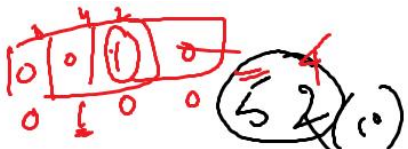
Left shift

= for a = a << 1 (applying arithmetic rules)

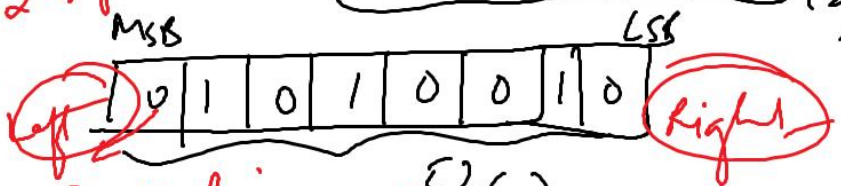
= a << 1

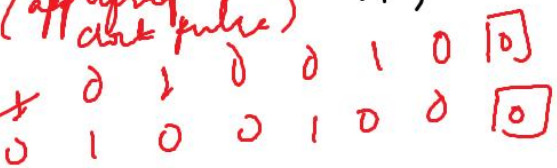
a = a << 2

a = a >> 2 → bitwise oper^n



0101 0010 (2)





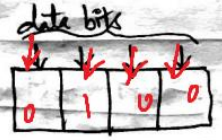
$2^{n-1} \ldots 2^1 \; 2^0$

Symbolic representn of diff types of data movement in shift re

(a) Serial shift right, then out

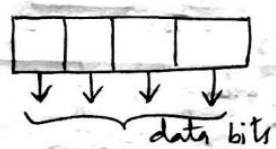
(b) Serial shift left, then out

(c) parallel shift in



// clock pls

(d) parallel shift out

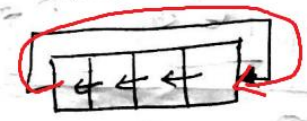


// clock pls

a << 2

(e) Rotate right

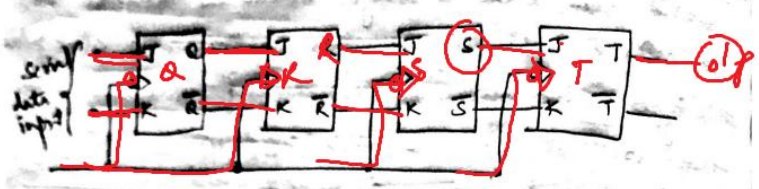
(f) Rotate left



Eg:

MSB LSB

The FFs used to construct registers are usually edge triggered JK, SR (or) D type.



(OR)

use D-FF (but not mandatory)

a) Shift (left) mode: fig shown below is serial-in serial out shift-left reg. (SISOSL)

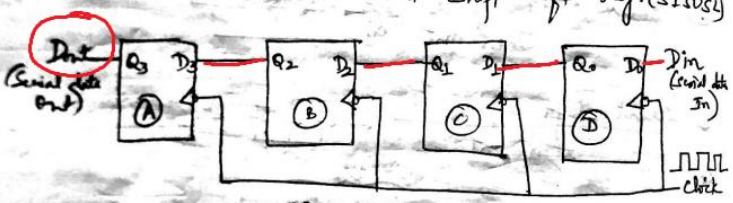


fig: Shift-left reg.

← D-flip flop

SISO - Serial-in Serial-out Shift

+ve clock

NT clock

Eg: Illustration of entry of 4-bit binary no 1111 into the reg, beginning with the left-most bit. Initially, reg is CLEARED. So $Q_3Q_2Q_1Q_0 = 1111$

collect 4-D ff ⇒ the o/p of all D-ff = 0

$Q_3Q_2Q_1Q_0 = 0000$

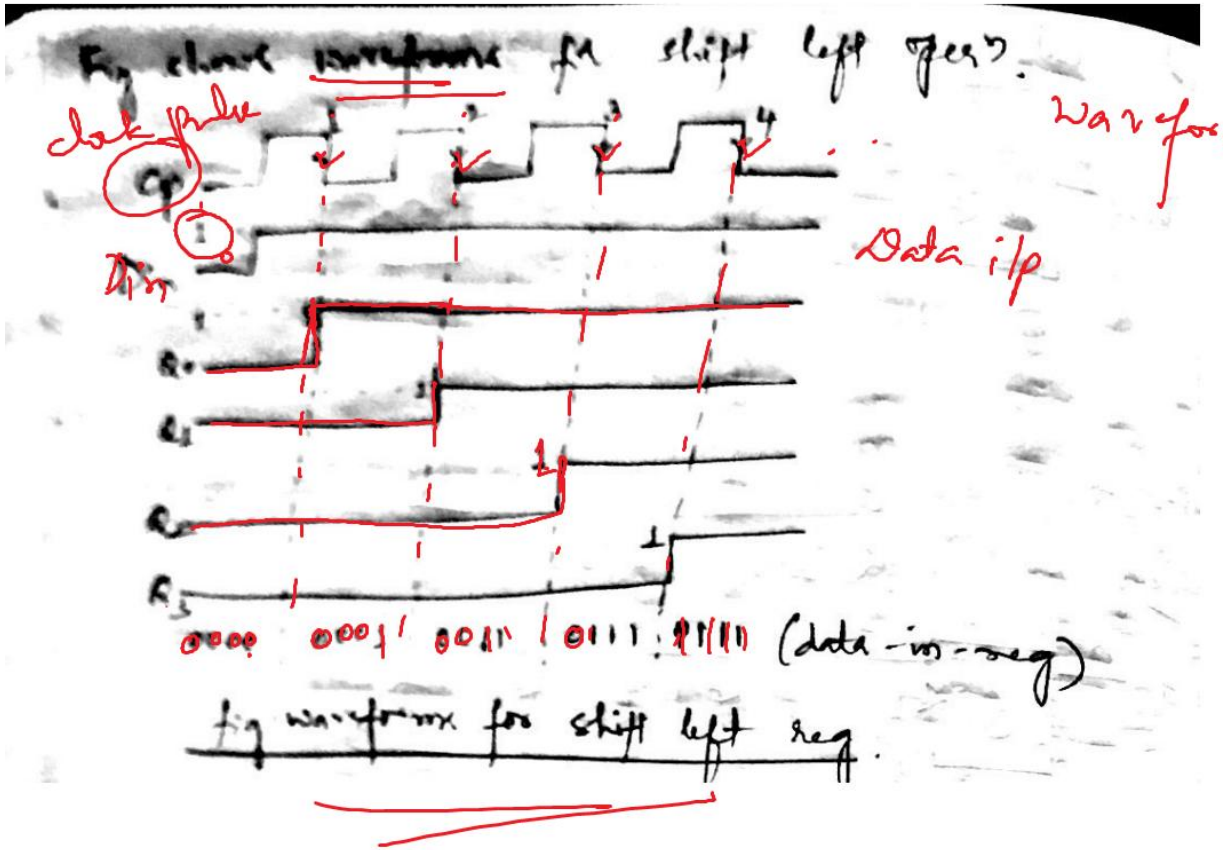
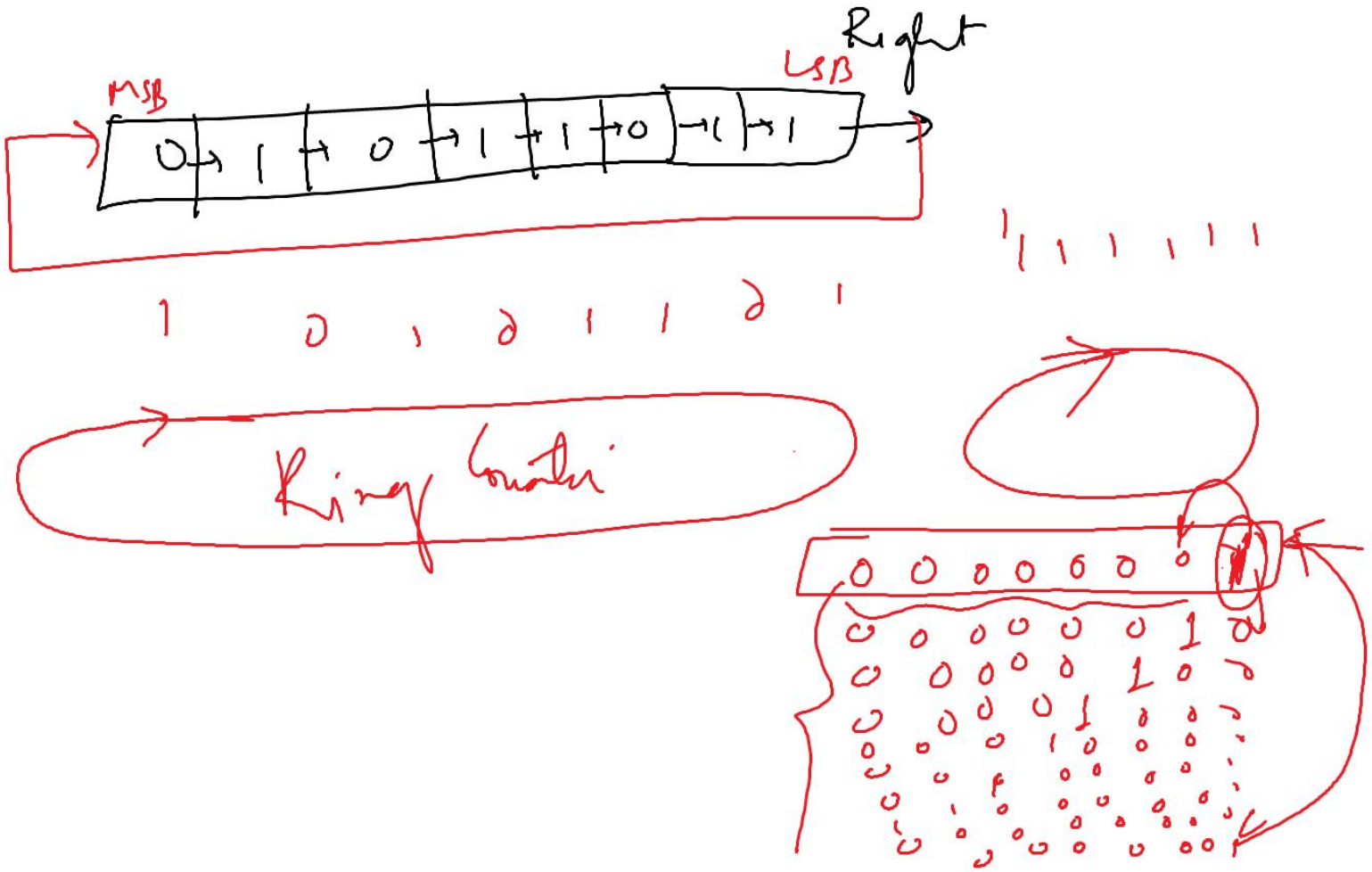


fig waveforms for shift left reg.

Waveform or a timing diagram

Dia

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


Right
MSB
LSB
0 1 0 1 1 0 1 1
Ring Counter

clock	Flip-flops				
	T	Q	R	S	T
0		0	0	0	0
1		1	0	0	0
2		1	1	0	0
3		1	1	1	0
4		1	1	1	1
5		0	1	1	1
6		0	0	1	1
7		0	0	0	1
8		0	0	0	0
9		1	0	0	0

fig (b) State Table of Johnson counter

4 FF

8 clock pulses

N FF → 2N clock pulses

repeated now on

↓ Ring counter

MSB Q R S LSB T

T

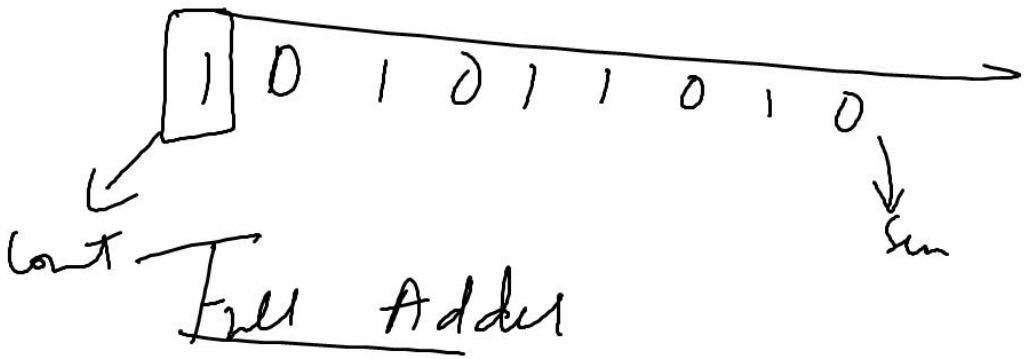
complement

Twisted Johnson

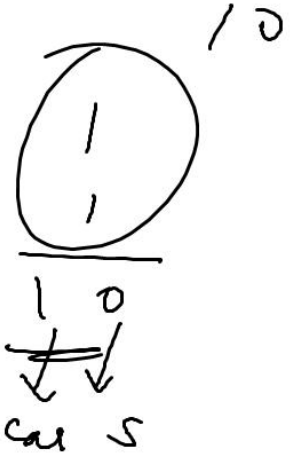
8-bit

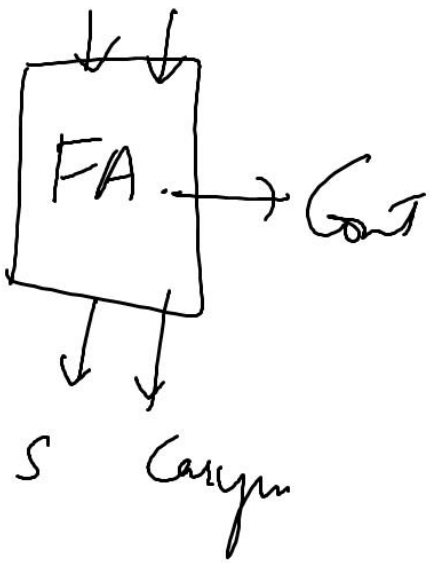
1	0	0	1	1	0	1	1
0	0	1	1	1	1	1	1

(Zero 8-bits no's to be added)



Full Adder





Counters:

Register which is capable of counting the no. of clock pulses arriving @ its i/p.

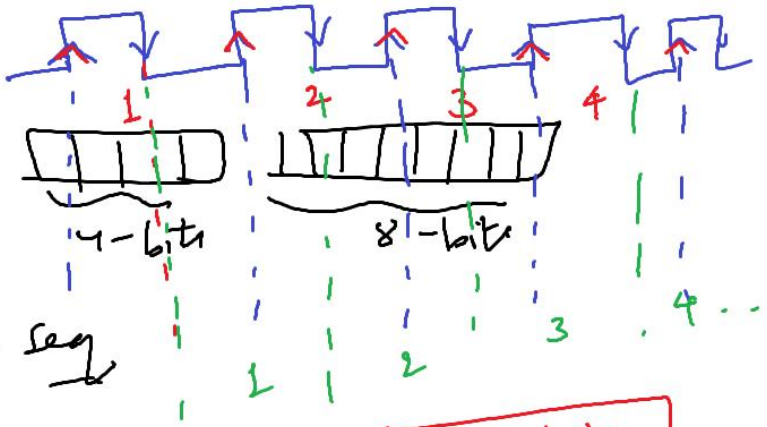
Count: the no. of clock pulses arrived

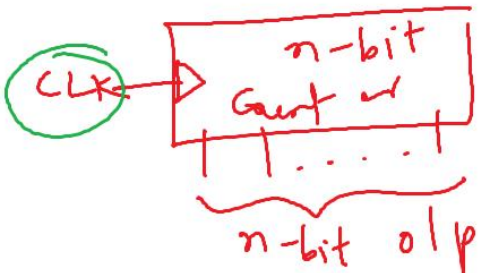
On arrival of each clock pulse, the counter is incremented by 1 (or) decremented by 1.

Flip-flop → 1-bit $\begin{matrix}Q\\ \bar{Q}\end{matrix}$

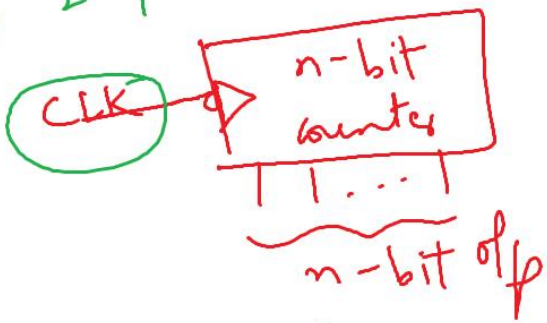
Register — Collection of FFs

Counter — which is used for generating counting seq.





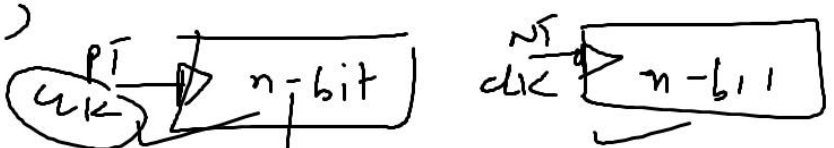
(1) PT (positive edge triggered n-bit counter



(2) NT (negative edge triggered n-bit counter

Logical Symbol of counter (binary counter)

External clock is applied to the clock i/p of the counter.



n-Flip flops. $\Rightarrow 2^n$ distinct $\neq$ ip sth

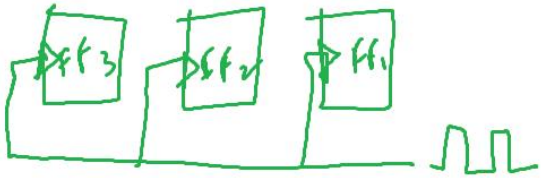
$\therefore$ 3-FF $\Rightarrow 2^3 = 8$ distinct o/p sizes.

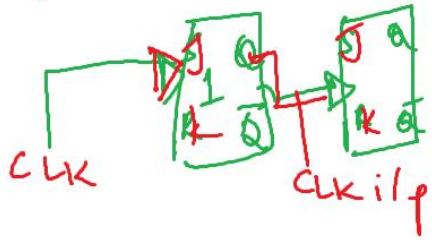
2 types: Synchronous, Asynchronous

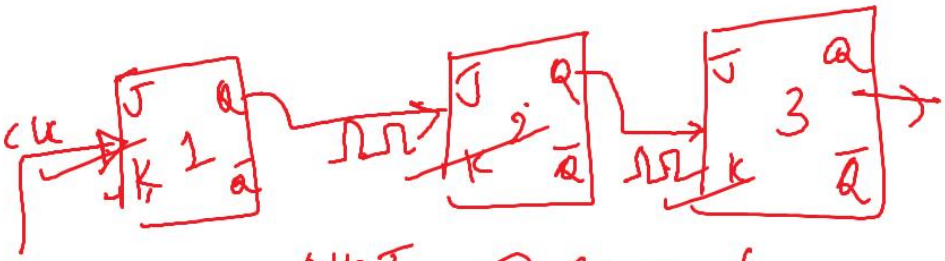
Each FF in the counter is triggered @ the SAME time.

Ripple

o/p of each F-F is connected to the clock i/p the next higher FF.







NOT @ same time.

Asynchronous counter

1) O/p of 1st FF drives the clock for the next FF
2) All the FFs are NOT clocked simultaneously
3) Logic ckt is very simple even for more no of states
4) Speed : low speed (propogation delay)
5) COST : is less.

Synchronous counter

1) There is NO connection b/w o/p of 1st FF & the clock i/p of the next FF
2) All the FFs are clocked simultaneously.
3) Design involves complex logic ckt as no of states increases.
4) Speed : high speed.
5) cost is more

Modulus of a counter: Total no of states a counter can indicate.

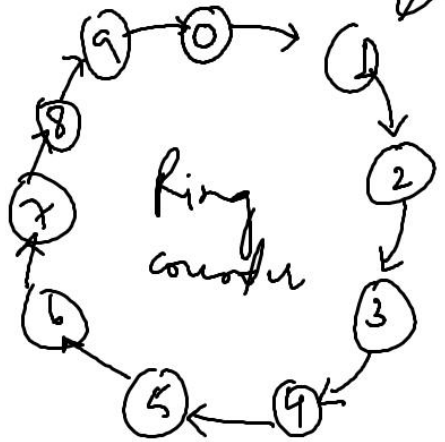
Eg: mod-6 counter ⇒ goes through 0, 1, 2, 3, 4, 5

mod-3 " ⇒ . . . 0, 1, 2

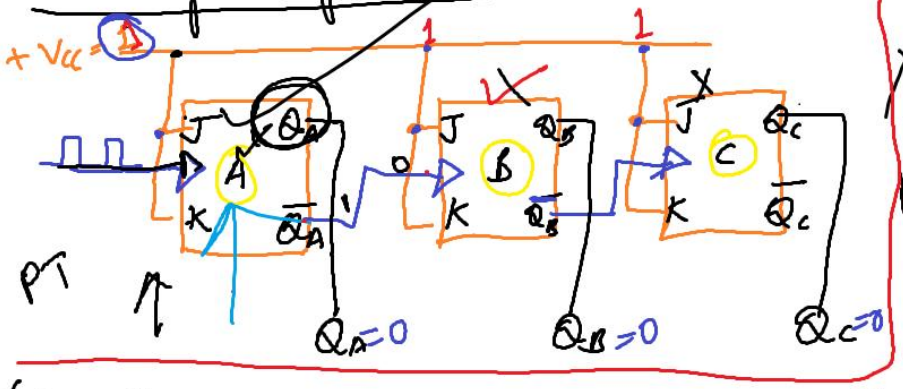
n counts ⇒ . . . 0, 1, 2 . . . (n-1)

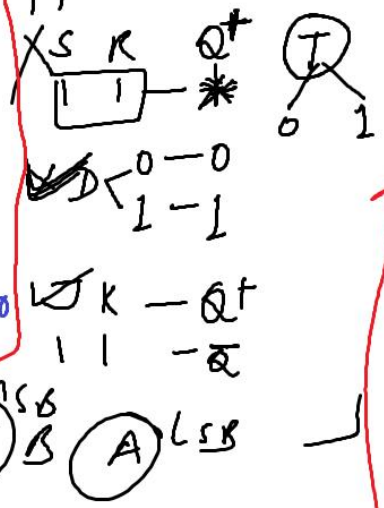
mod-10 counter: 0 → 1 → 2 → . . . → 9

Applns of Shift reg



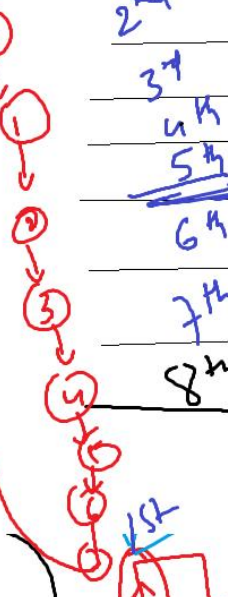
Binary Asynchronous (3-bit A.C) Counter / Ripple counter :

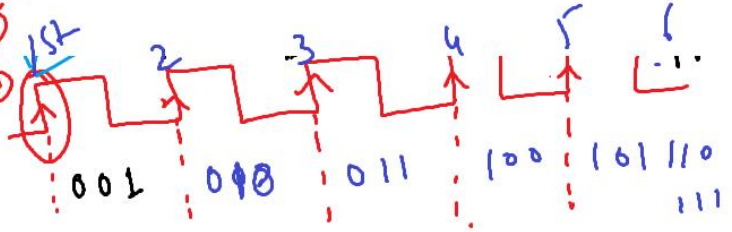




(SIPO) If JK i/ps are tied together, then it is T-flip flop.

Transition	$\overline{Q_C}$	$\overline{Q_B}$	$\overline{Q_A}$	Q_C	Q_B	Q_A
clock pulse						
⇒ Initially	1	1	1	0	0	0
1st clock $\overline{Q_A}$	1	1	0	0	0	1
2nd	1	0	1	0	1	0
3rd	1	0	0	0	1	1
4th	0	1	1	1	0	0
5th	0	1	0	1	0	1
6th	0	0	1	1	1	0
7th	0	0	0	1	1	1
8th	1	1	1	0	0	0





+Vcc = +5V ⇒ logic 1

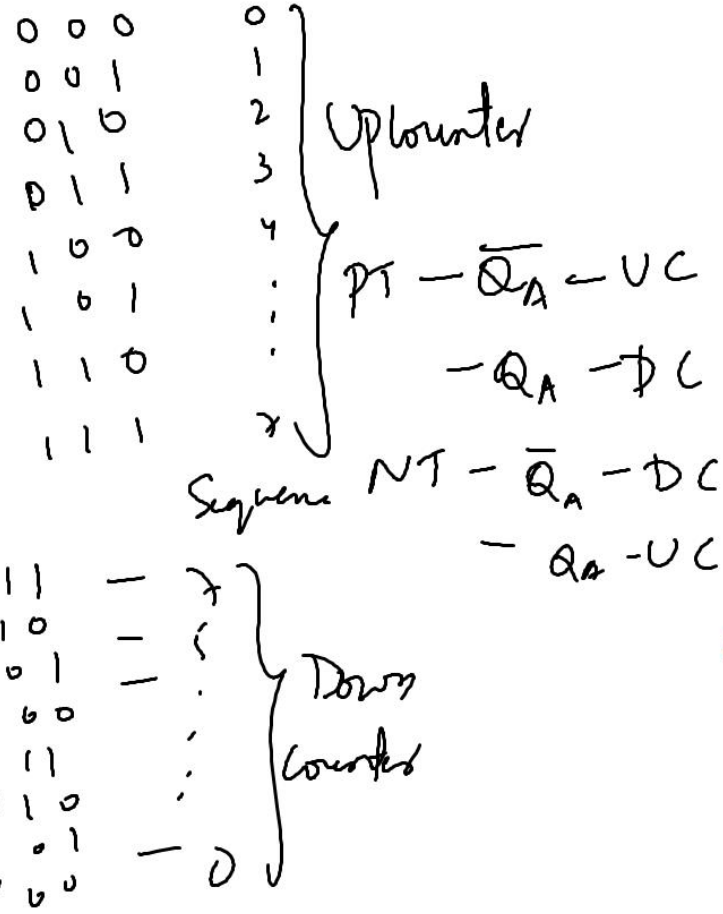
GND = 0V ⇒ logic 0

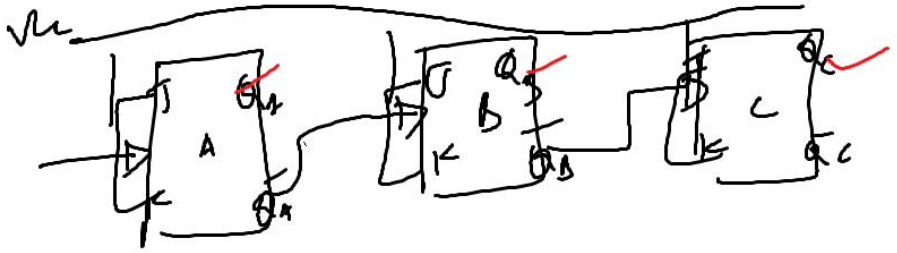
= Triggered

0 - 1 ⇒ PT

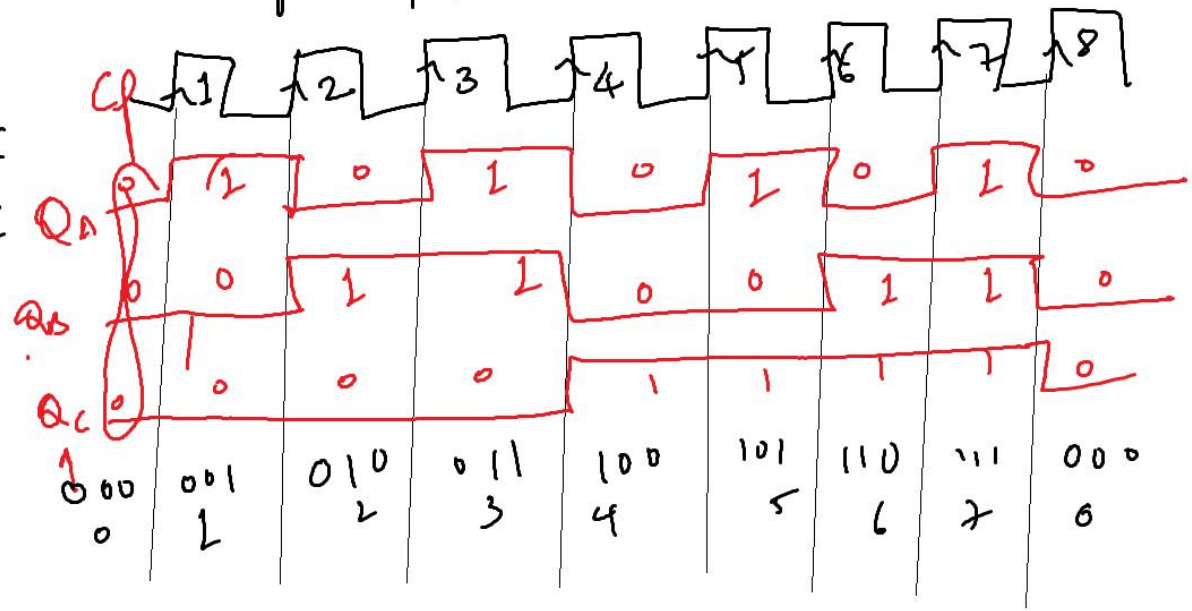
1 - 0 ⇒ NT ✗

J	K	Q_{n+1}
0	0	Q_n
1	1	$\overline{Q_n}$ = toggling of prev state

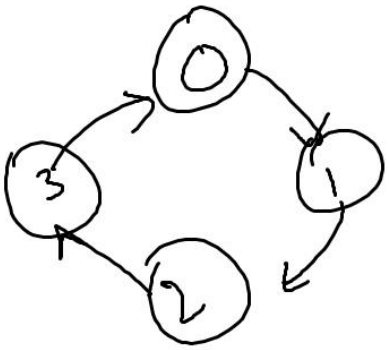




Timing diagram / Wave form - 3 bit Ripple Counter



Implement 2-bit Ripple counter for generating Upcounter sequence (0, 1, 2, 3, 0) & draw timing diagram.



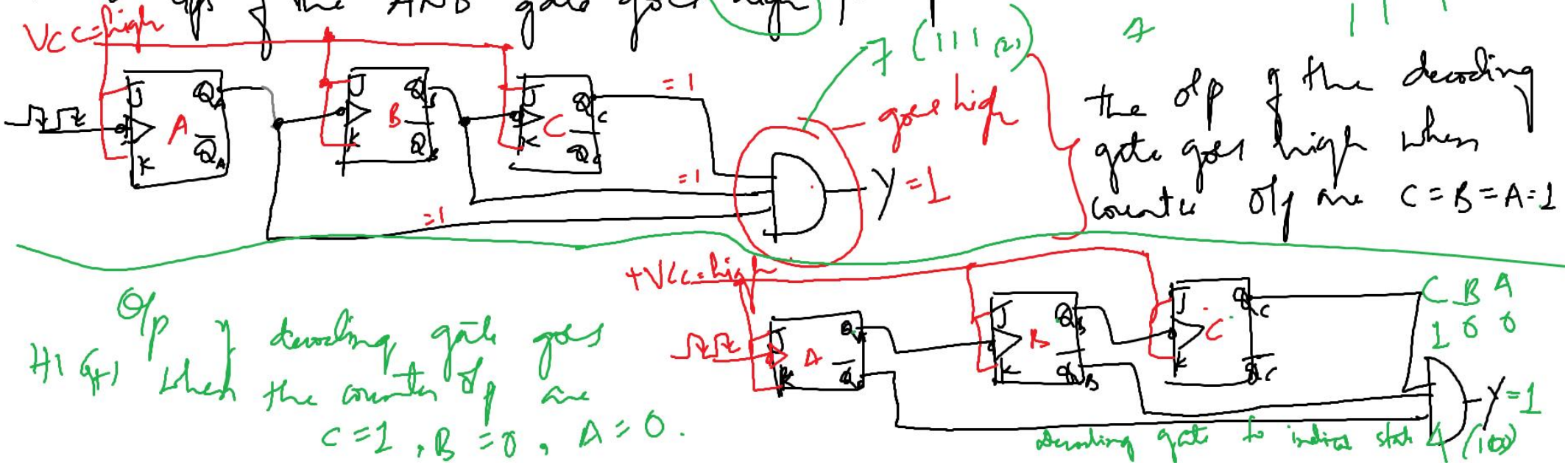
3-bit $\Rightarrow 2^3 \Rightarrow 8$ ...0...7

2-bit $\Rightarrow 2^2 = 4$ (0, 1, 2, 3, 0)

Asynchronous counter :

Decoding gates : used to indicate the counter has reached to a particular state.

O/ps of the counter are connected to the AND - gate as i/ps and the o/p of the AND gate goes high for particular state.

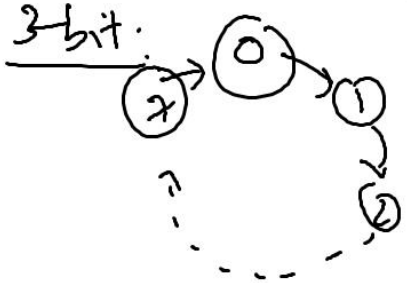


O/p of decoding gate goes HIGH when the counter o/p are C=1, B=0, A=0.

Note: Connect corresponding o/ps of Decoding gate i/p to indicate the desired state:

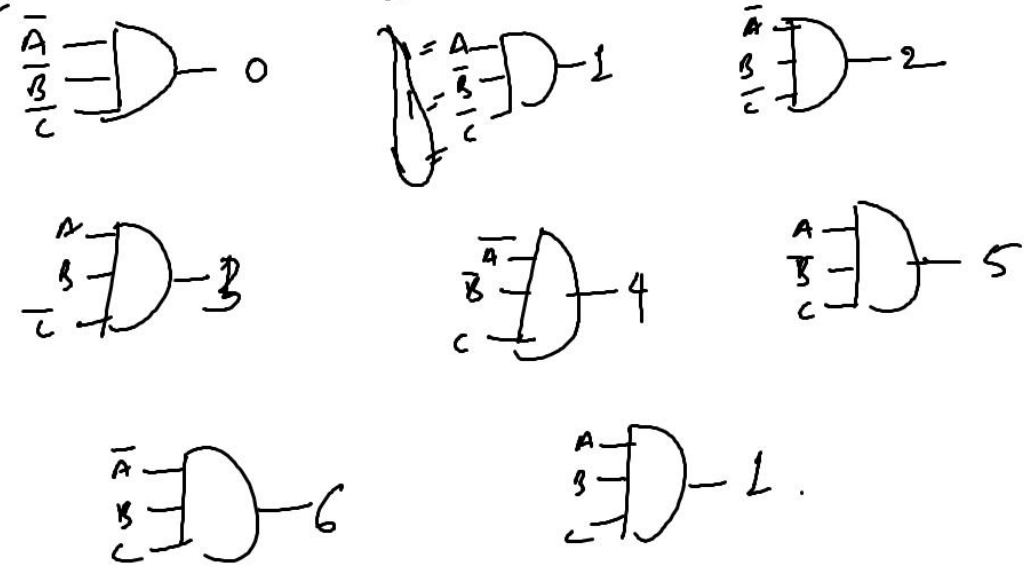
Connection for all possible state detection for 3-bit Counter: (3-FF - JK FF)

3-bit:

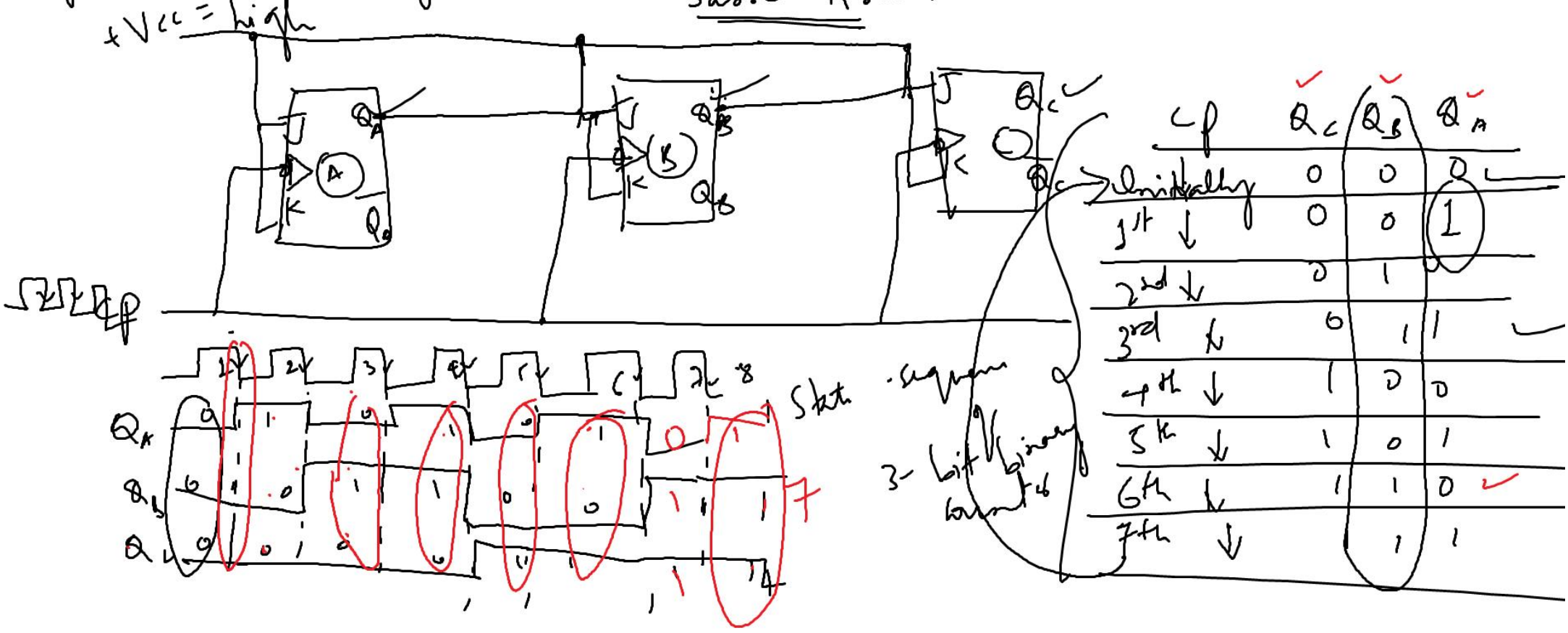


C	B	A	
0	0	0	0 ✓
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Decoding-gate connections



Synchronous Binary Counters: Each FF in the counter is triggered @ the same time.



cp	Q_C	Q_B	Q_A
Initially	0	0	0
1st ↓	0	0	1
2nd ↓	0	1	0
3rd ↓	0	1	1
4th ↓	1	0	0
5th ↓	1	0	1
6th ↓	1	1	0
7th ↓		1	1

Design a Synchronous Counter:

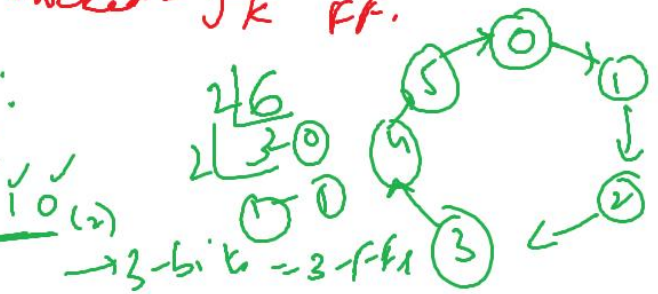
1) Determine the no of FF needed.
   If $n \rightarrow$ no of FF then
   $2^n \geq$ no of states in the counter
   $n \rightarrow$ no of FF
   $N \rightarrow$ no of states
2) Choose the type of FF to be used
3) Using Excitation table for selected FF, determine the excitation-table for Counter
4) Use k-map (or) any other simplification method to derive FF i/p f^{n}s
5) Draw a logic diagram

Eg: Design a synchronous mod-6 counter using clocked-JK FF.

Soln:

$6_{(10)} = 110_{(2)}$

→ 3-bits = 3 FFs



Step 1: Find the no of FFs.

$2^n \geq N$

$n \rightarrow$ no of bits (FFs)

$N \rightarrow$ no of states.

Here $N = 6$ — Mod-6

$2^n \geq 6$

$n = 3$

$4 \geq 6$ ✗

$n = 2$

$8 \geq 6$ ✓

$n = 3$ → 3-bits - 3-FFs are reqd

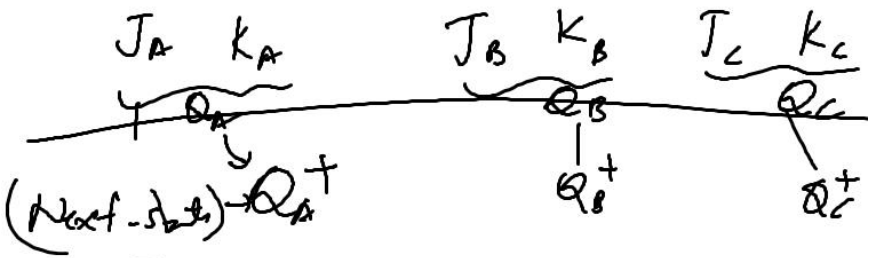
Step -2: Bit-excitable of JK-ff.

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

— through JK-state transition diagram

Step-3: Determine the transition Table.

3-JK-FF are reqd



Present state			Next state			J_A	K_A	J_B	K_B	J_C	K_C
Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+						

Design of a Synchronous counter:

Design a syn mod-6 counter using JK ff.

Sol: (1) No of ff.

2 | 6
2 | 3 - 0
1 - 1

$6_{(10)} = 110_{(2)}$

→ 3-bits ⇒ 3-FF.

(or)

n → no of FF
N - no of states = 6

$2^n \geq N$

$2^2 \geq 6$

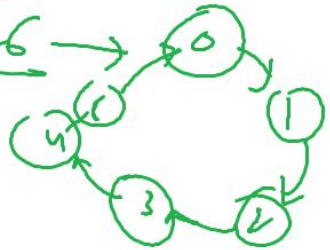
$4 \not\geq 6$

$2^3 \geq 6$

$8 \geq 6$

∴ n = 3

mod-6 →



(2) Type of FF : JK

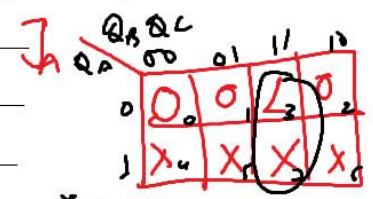
(3) Excit^n table of choose FF

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

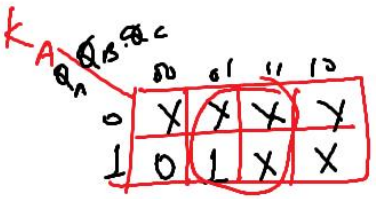
(4) Excitn table for a counter.

④ Excitation table for mod-6 counter / Transition table of mod-6 counter.

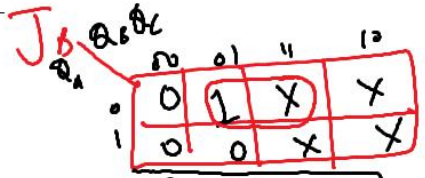
	Present state			Next state			Flip-flop inputs					
	Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	0	0	X	1	X	X	1
2	0	1	0	0	1	1	0	X	X	0	1	X
3	0	1	1	1	0	0	1	X	X	1	X	1
4	1	0	0	1	0	1	X	0	0	X	1	X
5	1	0	1	0	0	0	X	1	0	X	X	1
6	1	1	0	X	X	X	X	X	X	X	X	X
7	1	1	1	X	X	X	X	X	X	X	X	X



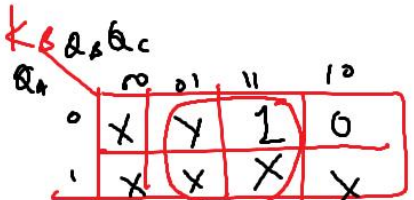
$\therefore J_A = Q_B \cdot Q_C$



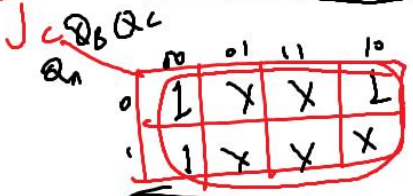
$\therefore K_A = Q_C$



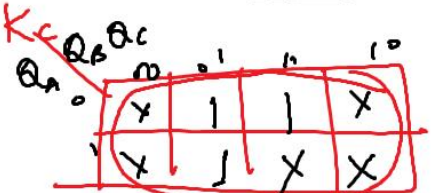
$\therefore J_B = \bar{Q}_A \cdot Q_C$



$\therefore K_B = Q_C$



$\therefore J_C = 1$



$\therefore K_C = 1$

⑤ Use k-map simplification method — FF f^{n}s.
(for each FF i/p — write k-map)
(3 i/ps ⇒ Q_A, Q_B, Q_C)

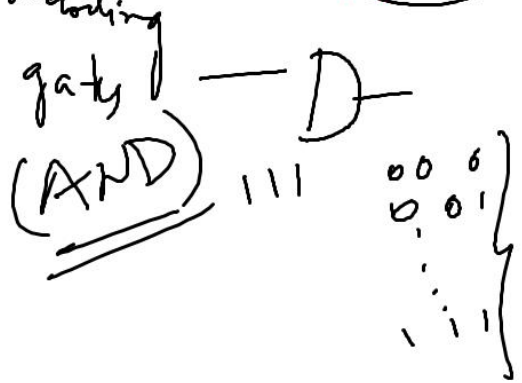
$J_A = Q_B \cdot Q_C$ ✓ $K_A = Q_C$ ✓

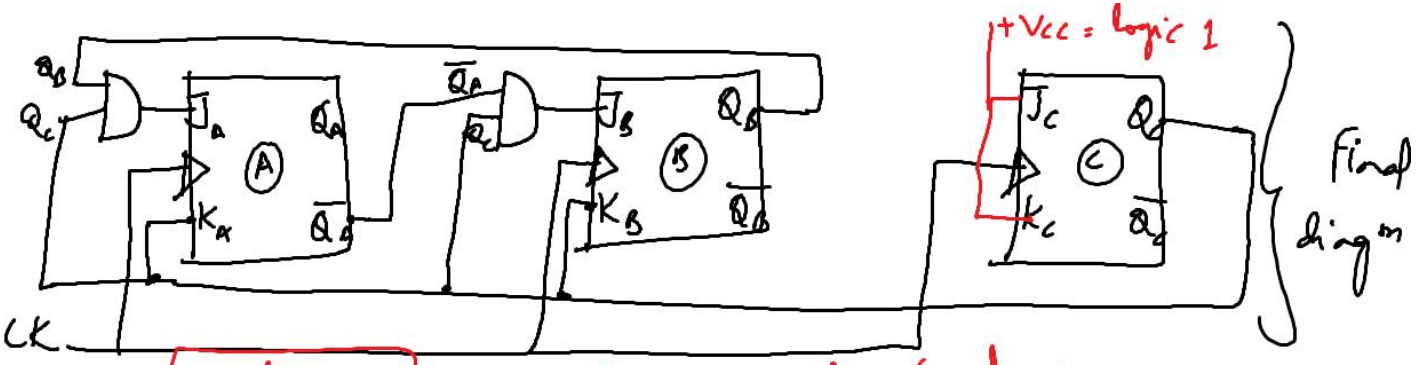
$J_B = \bar{Q}_A \cdot Q_C$ ✓ $K_B = Q_C$ ✓

$J_C = 1$ ✓ $K_C = 1$ ✓

(i) Draw the logic diagram (Implementation of Counter) Synch.

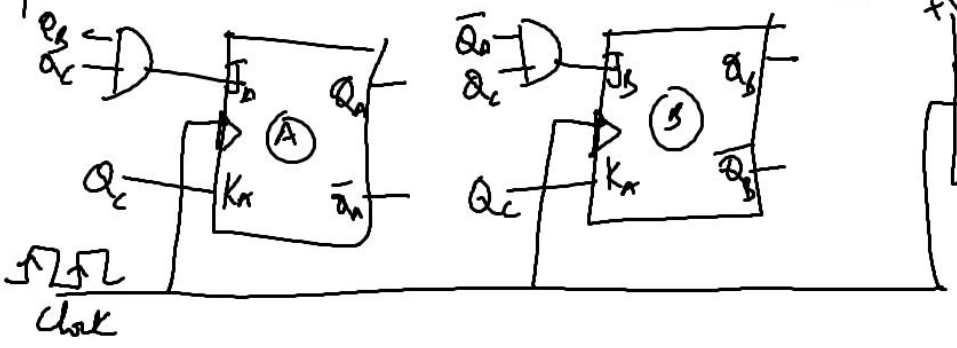
Decoding gates (AND)





Implementation of mod-6 Synch Counter

NOTE: To avoid crossing of lines & to have a better clarity the ckt diagram can also be represented by using signal-names.



Simplified representation of above final diagram

Design a Sync mod 6 Counter using D-flip flop

1) SR
2) JK
3) D
4) T

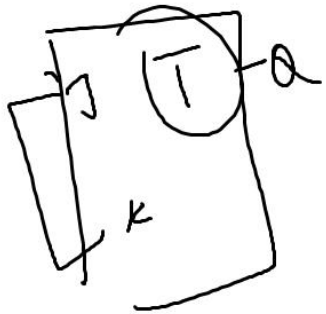
Next state

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	X

D	Q^+
0	0
1	1

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

T	Q^+
0	Q
1	$\bar{Q}$

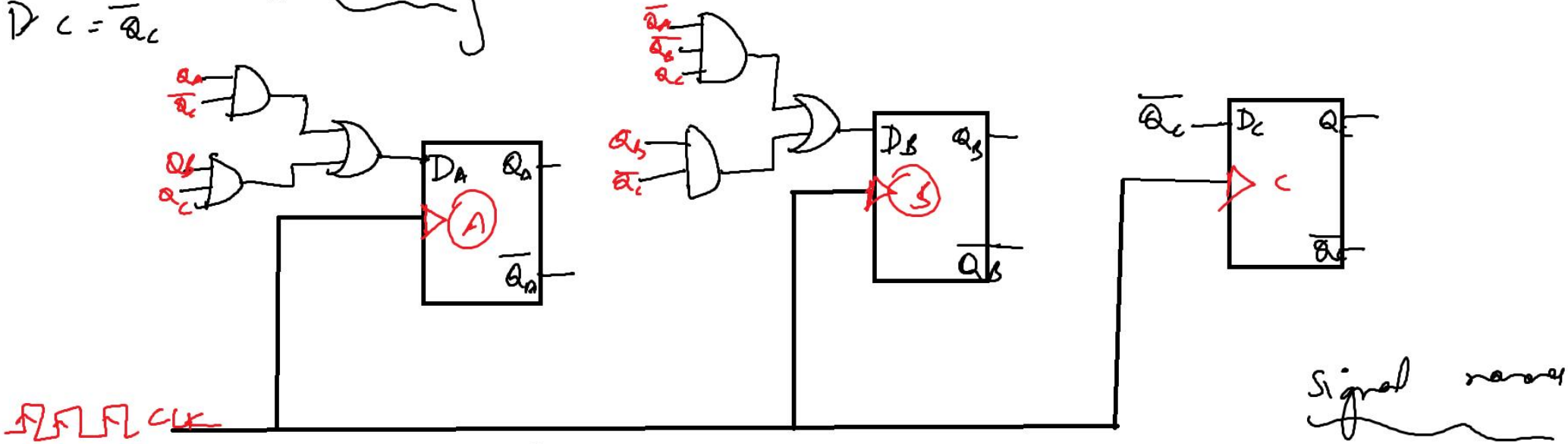


$D_A = Q_A\bar{Q}_C + Q_BQ_C$

$D_B = \bar{Q}_A.\bar{Q}_B Q_C + Q_B.\bar{Q}_C$

$D_C = \bar{Q}_C$

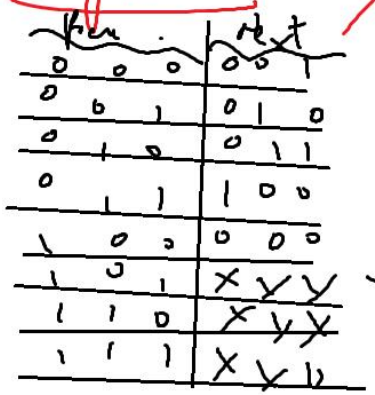
Step 4: Implement the counter

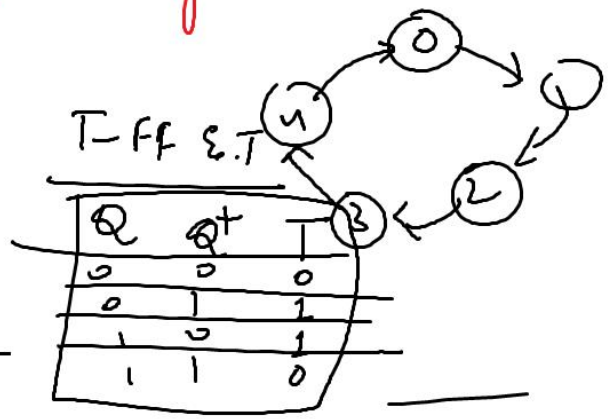


Signal names

Implemented diagram:

Assignment: 1) Design a Mod-5 sync counter using T ff

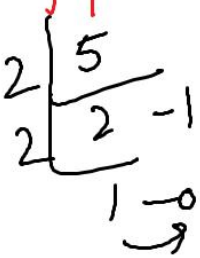




$= 5 = n$

$n = 3 \rightarrow$ ff

$2^3 \geq$ if combinations

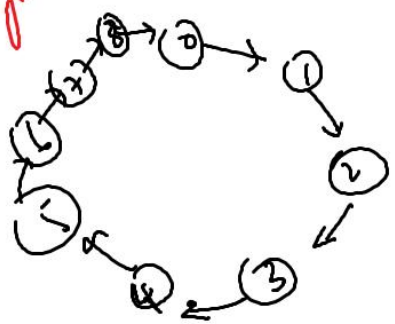


$5_{(10)} = \underbrace{101}_{(2)}$

SR
D
JK
T

JK

2) Design a mod-9 sync counter using SR-ff



$N = 9_{(10)} = \underbrace{1001}_{4\text{-bits}}{}_{(2)}$

$n = 4 \Rightarrow$ 4 ff's.

$2^4 = 16$ //

X X 4 X 1

S R - ff Excitation table

Q	Q^+	S	R	State Transition
0	0	0	X	
0	1	1	0	
1	0	0	1	
1	1	X	0	

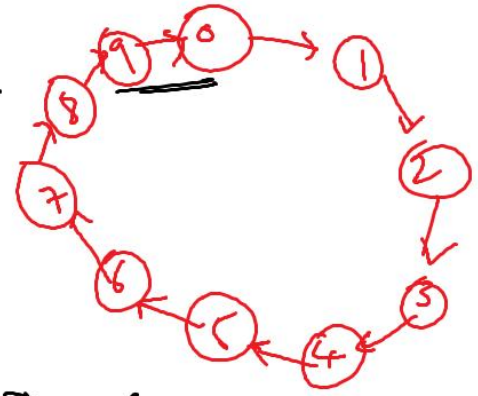
Design a syn. decade counter using D-ff

10 : 0 → 9 ⟷ 0

↳ mod-10 counter

$10_{(10)} = 1010_{(2)}$ (4-bits) $\Rightarrow 2^4 = 16 \Rightarrow$ i/p combinations

2 | 10
2 | 5 - 0
2 | 2 - 1
1 - 0

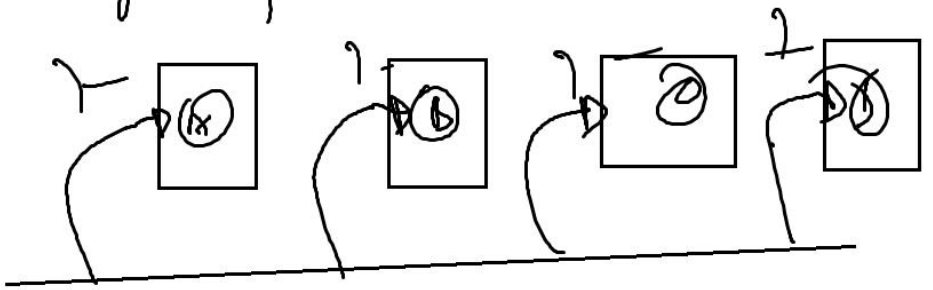


present state				next state				D-ff i/ps			
Q_A	Q_B	Q_C	Q_D	Q_A^+	Q_B^+	Q_C^+	Q_D^+	D_A	D_B	D_C	D_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X

⇒ D-ff Excitn table

⇒ k-map : $D_A = 1$ $D_B = 1$ D_C ? D_D

⇒ logic diagm.



Design a counter with the sequence 0, 1, 3, 7, 6, 4, 0

Soln.

JK

① 6 states are there $= N = 6$ ✓

$G_{(10)} = 110_{(2)}$ — 3-bits

$n = 3$

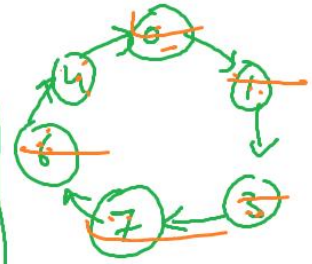
② $n = 3$ bits - 3 ff

$= 2^3 = 8$ states as ff

$2^n \geq 6$

$2^2 \geq 6 \Rightarrow 4 \not\geq 6$

$2^3 \geq 6 \Rightarrow 8 \geq 6$



③

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

④

	Present state			Next state			FF inputs					
	Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	0	1	0	X	0	X	1	X
1	0	0	1	0	1	1	0	X	1	X	X	0
2	0	1	0	X	X	X	X	X	X	X	X	X
3	0	1	1	1	1	1	1	X	X	0	X	0
4	1	0	0	0	0	0	X	1	0	X	0	X
5	1	0	1	X	X	X	X	X	X	X	X	X
6	1	1	0	1	0	0	X	0	X	1	0	X
X	1	1	1	1	1	0	X	0	X	0	X	1

⑤ k-map simplification

$J_A = Q_B$ $\quad K_A = \overline{Q_B}$

$J_B = Q_C$ $\quad K_B = \overline{Q_C}$

$J_C = \overline{Q_A}$ $\quad K_C = Q_A$

J_A

Q_A \ $Q_B Q_C$	00	01	11	10
0	0	0	1	X
1	X	X	X	X

$\Rightarrow Q_B$

⑥ Implement / Realization / ckt diagram.

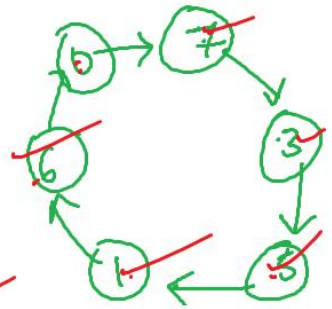
Assign: Design a counter (Synch) for the sequence

7, 3, 5, 1, 6, 0

6 → 3 → 1 → 6

JK

Soln.



$N = 6$

$n = 3 \Rightarrow 110_{(2)} \Rightarrow 2^3 = 8$ combⁿ

K-map.

$J_A = 1$ ✓

$K_A = 1$ ✓

$J_B = 1 \Rightarrow \bar{Q}_A$ ✓

$K_B = 1 \Rightarrow \bar{Q}_A + \bar{Q}_C$ ✓

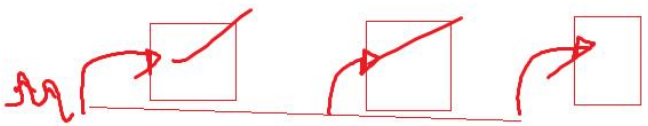
$J_C = 1 \Rightarrow \bar{Q}_B$ ✓

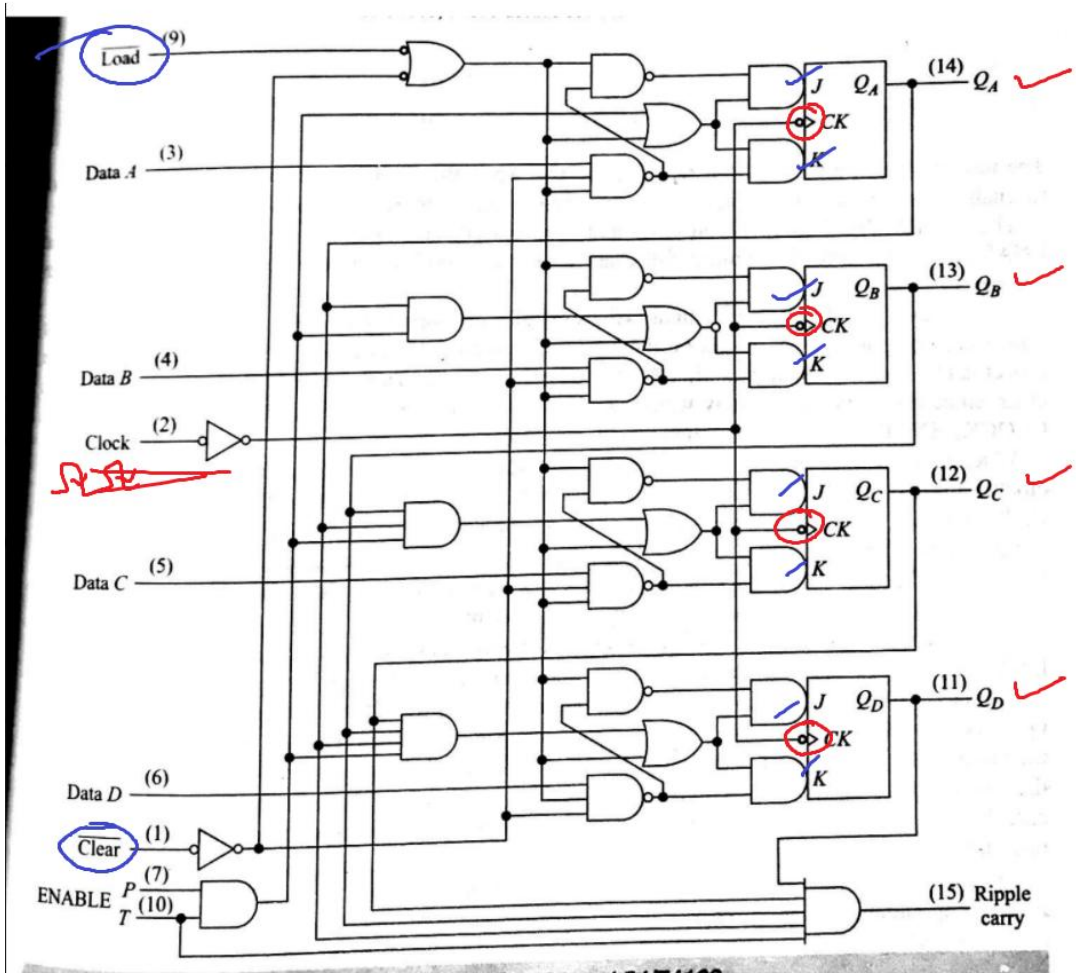
$K_C = 1 \Rightarrow \bar{Q}_A \cdot \bar{Q}_B$ ✓

Present state			Next-state			J-K-ff-i/ps.					
Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	1	1	1						
0	0	1	1	1	0						
0	1	0	x	x	x	x	x	x	x	x	x
0	1	1	1	0	1						
1	0	0	x	x	x	x	x	x	x	x	x
1	0	1	0	0	1						
1	1	0	0	0	0						
1	1	1	0	1	1						

3

8





NT

4-bits

$2^4 \Rightarrow$ i/p combination $= 16$

Clear = Reset

$\overline{\text{Clear}}$

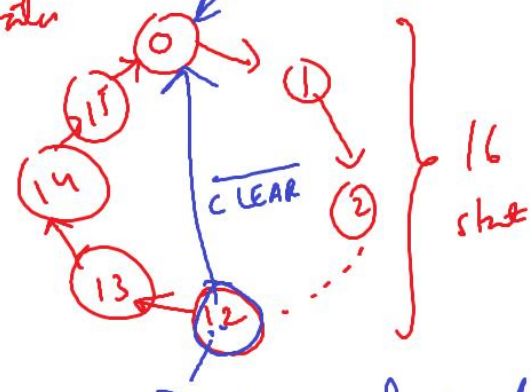
16-states

0 0 0 0

1 1 1 1

$N = 16$

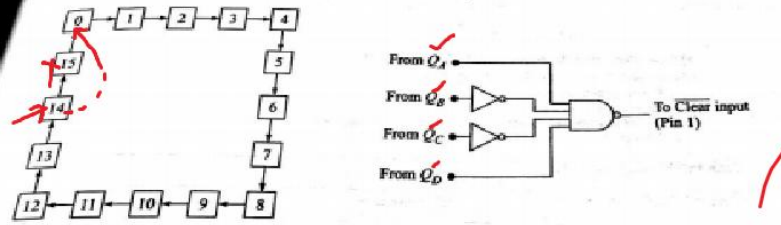
$n = 4$



Decoding gate : D $\Rightarrow$ desired state

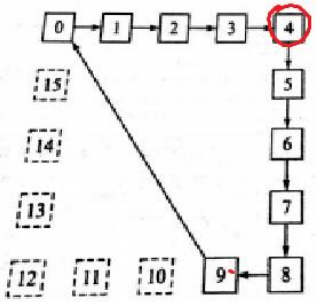
$\overline{\text{LOAD}}$ = loaded with the next state

from 0000 up to the maximum desired count and then clear back to 0000. This is the technique that can be used to construct a counter that has any desired modulus.



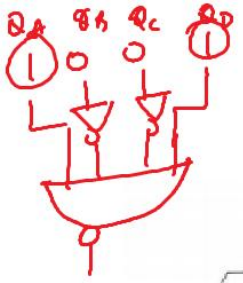
(a) Mod-16 counter state diagram

(b) Gate to decode count 9 (1001)

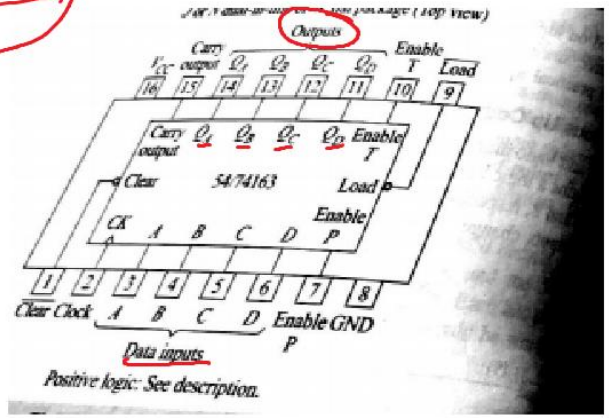


(c) Modified state diagram for Mod-10 counter

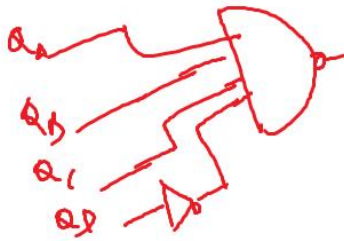
Fig. 10.26



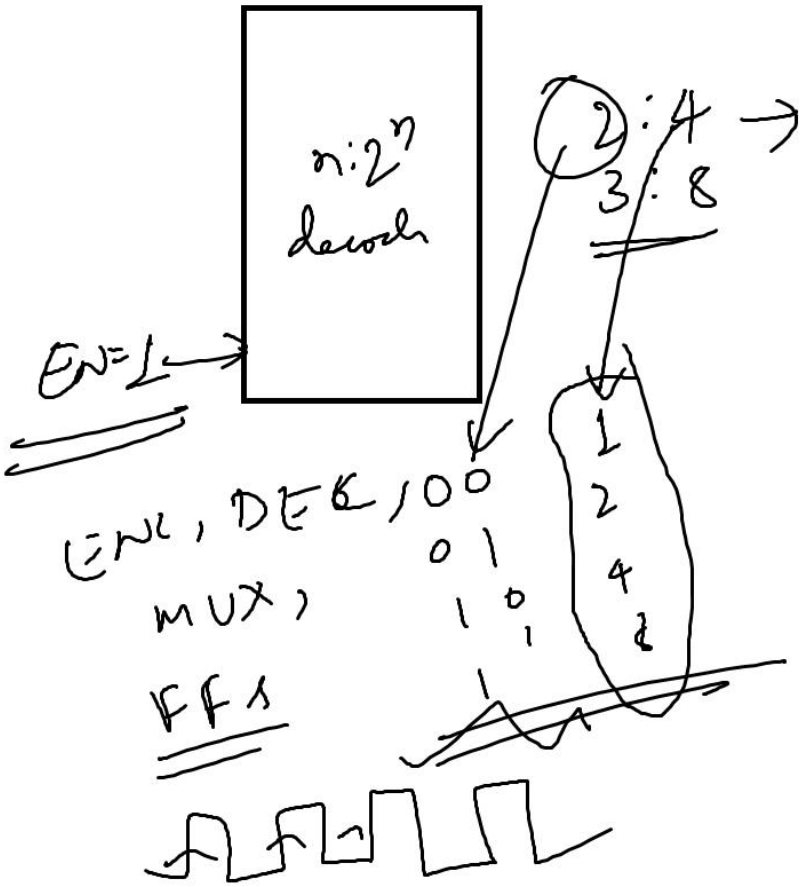
= 1001 = 9(10)



Gate to decode count 14 (1110)



14 = (1110)



00 01 10 11

0 1 2 3

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

1 2 4 8

Structural model : AND, OR, NOT, NAND....

data-flow model : Operators (&), (|), (~)

behavioural model : always (@ pos)

⋮ ⋮

Unit-5 : Moore Model & Mealy model

In Synch(or) clocked sequential circuits, clocked-FF are used as memory elements, which change their individual states in synchronism with the periodic clock signal.

∴ The change in states of FF & change in the state of the entire ckt occurs @ the transⁿ of the clock signal.

The states of the o/p of the F-F for the Seq ckt gives the state of the seq ckt

Present state : Status of all state vari, @ some time (t), before the next clock edge

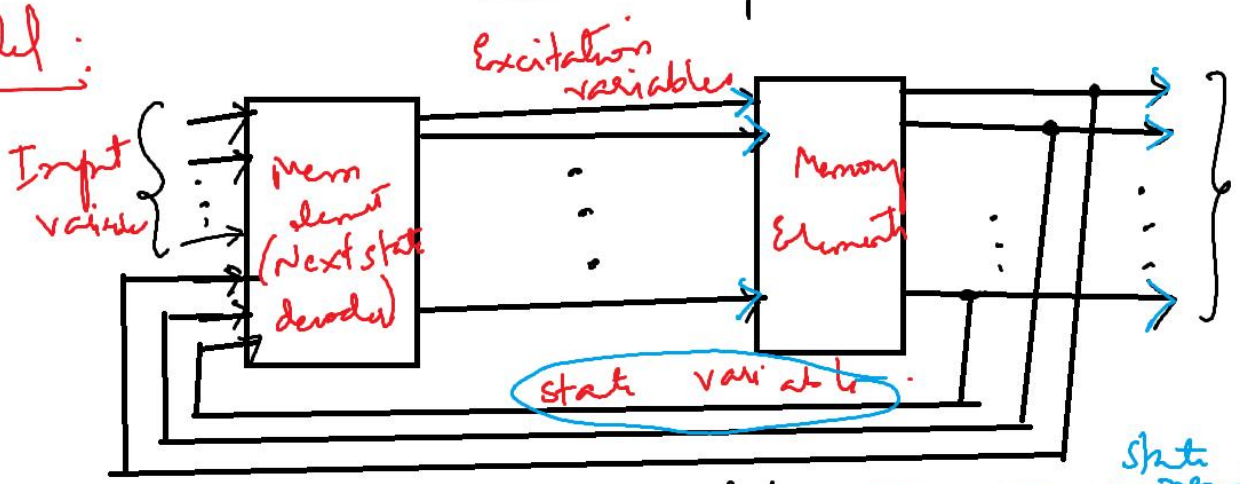
Next state : the status of all state vars, at some time (t+1)

Synchronous (or) clocked Seq ckts represents 2 models.

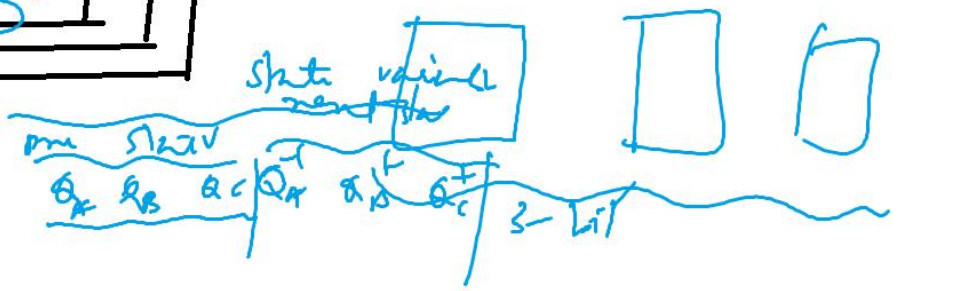
(1) Moore model : The o/p depends only on the present state of the FFs.

(2) Mealy model : The o/p depends on BOTH the present state of the FFs & the on the I/ps.

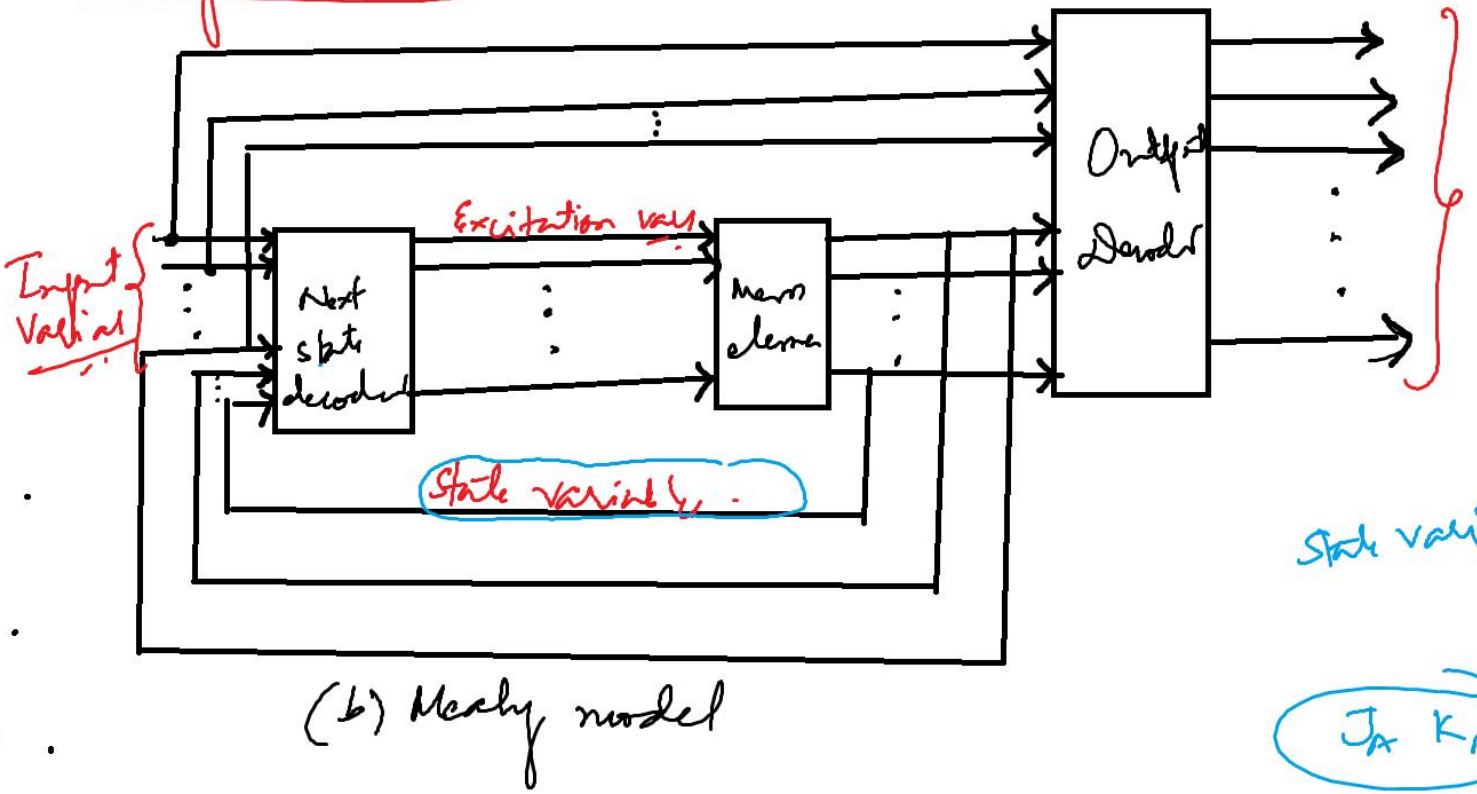
Moore model :



(a) Moore model



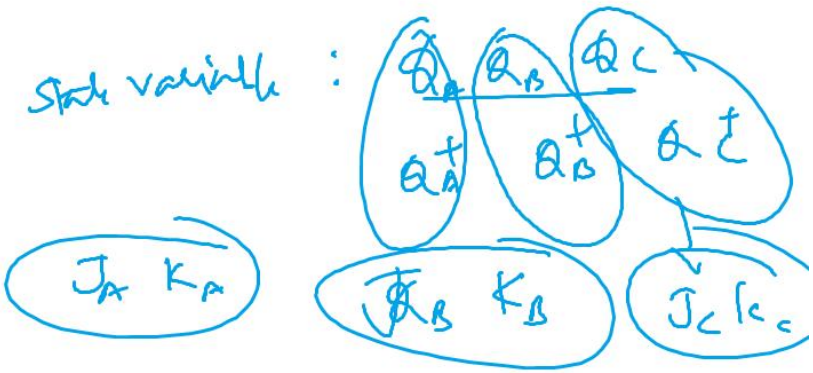
(2) Mealy model :

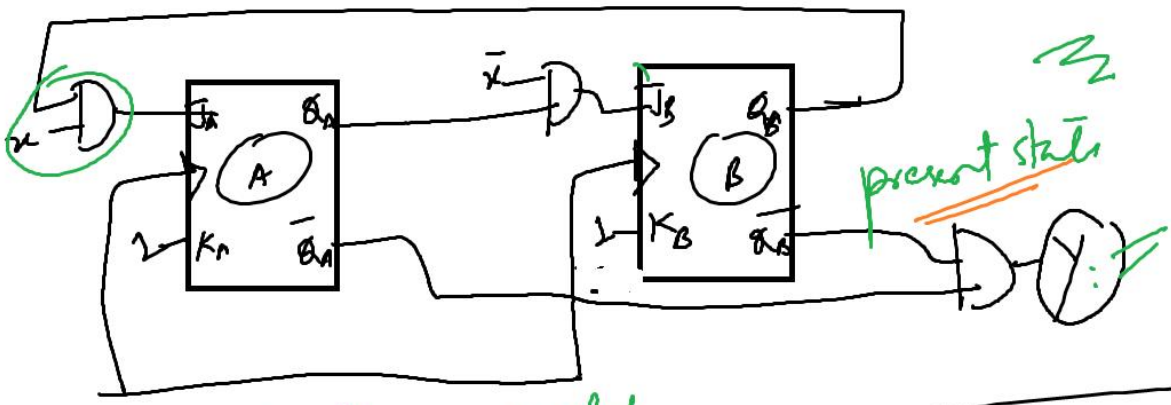


(b) Mealy model

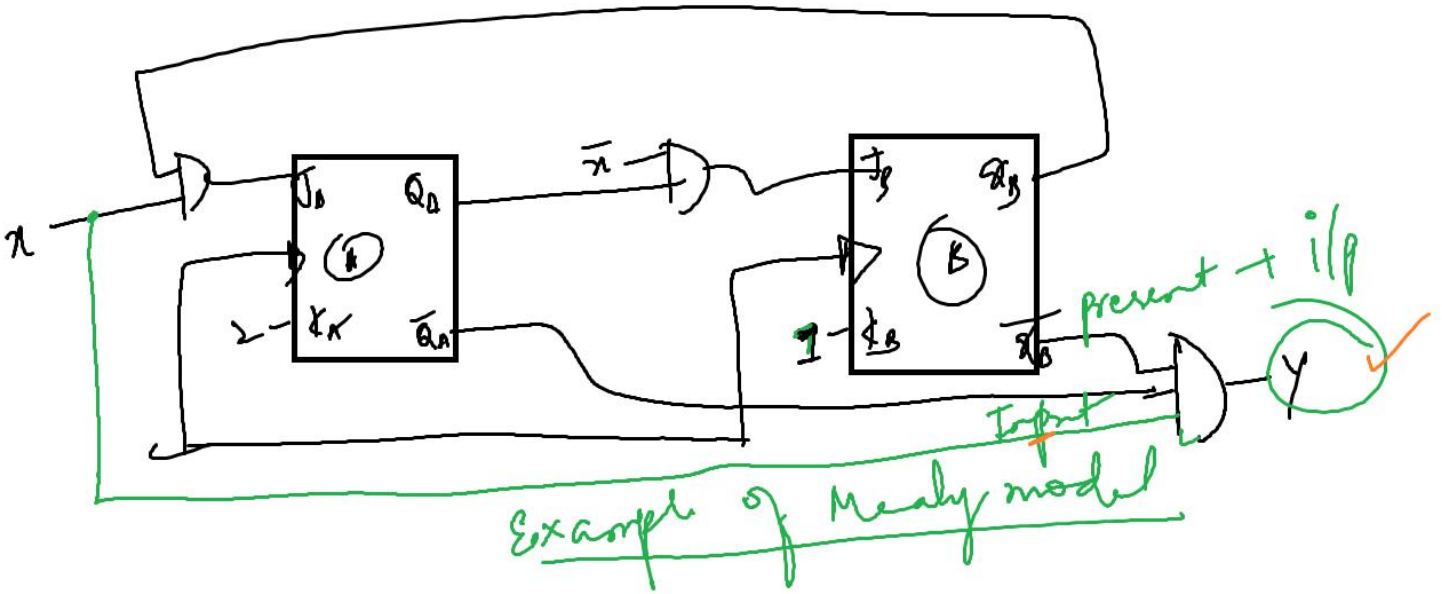
Op variables.

State variable :



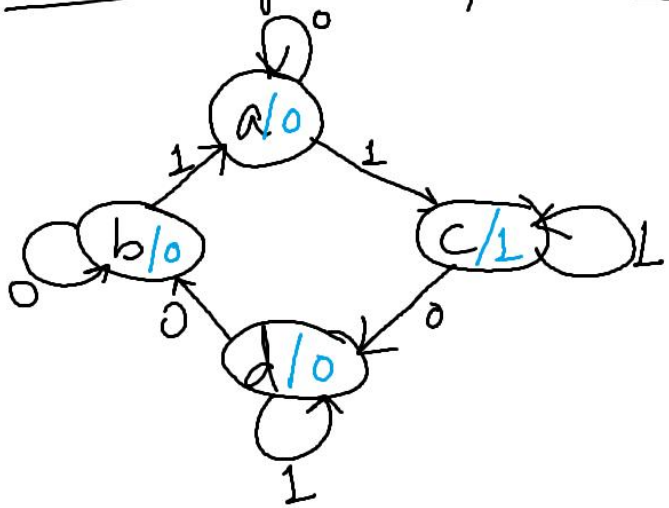


Example of Moore - model



Example of Mealy model

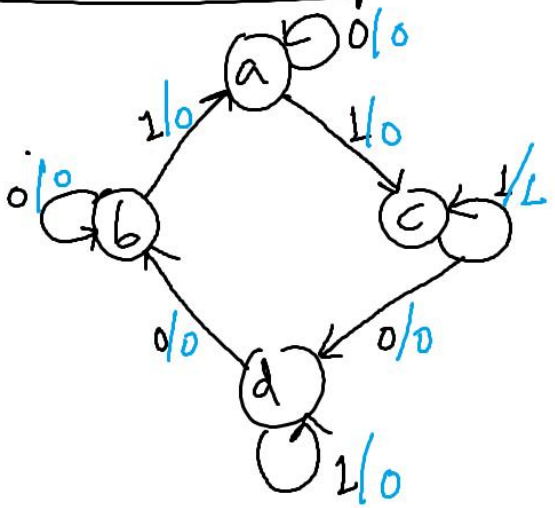
State diagram for Moore model & Mealy model



(a) Moore Model

I/p variables : 0 & 1

O/p vars : 0 & 1



(b) Mealy model

I/p variables : 0 & 1

o/p vars : 0 & 1

Notation of state m/c :

a, b, c, d = state

⟶ ⇒ on i/p 0 & L

Moore model :

Node ⇒ x/y

x ↓ state

y → o/p

Mealy model :

Node ⇒ state

arrow ⇒ x/y

x ↓ i/p

y → o/p

State Table :

Mealy model

Present state	Next state		Output = Y	
	x=0	x=1	x=0	x=1
a	a	c	0	0
b	b	a	0	0
c	d	c	0	1
d	b	d	0	0

Mealy

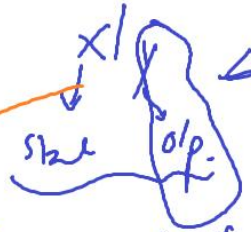
x = i/p variable

Moore :

Present state	Next state		output
	x=0	x=1	Y
a	a	c	0
b	b	a	0
c	d	c	1
d	b	d	0

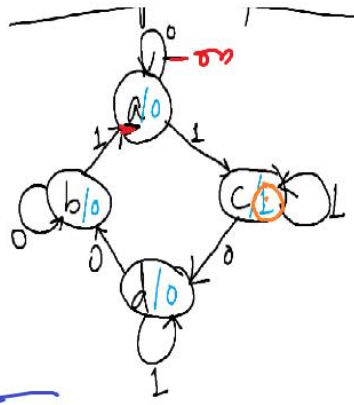
a, b, c, d ⇒ STATES

Node:

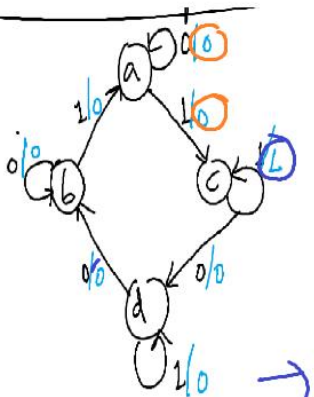


TRANSITION TABLE

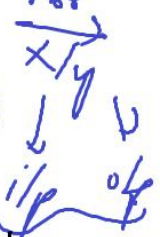
Present state	Next state				Output	
	x=0		x=1		x=0	x=1
A	B	A	B	A	B		
a	0	0	0	0	1	0	0
b	0	1	0	1	0	0	0
c	1	0	1	1	1	0	1
d	1	1	0	1	1	1	0



(a) Moore Model



(b) Mealy model



$2^x = 2^1 = 2$ STATES i/p o/ps

$2^2 = 4$

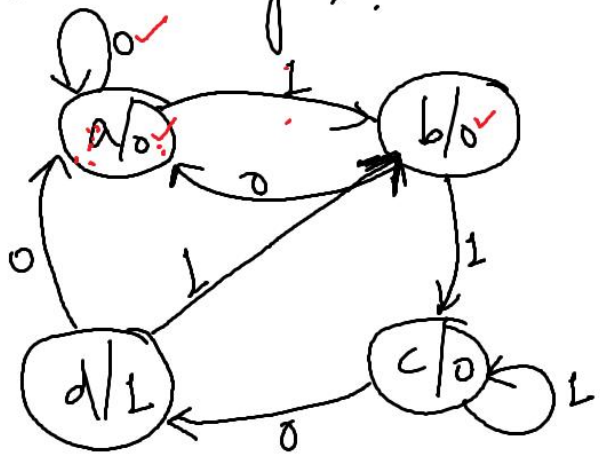
How many states = 4

State & state variable

4 possible combinations

a, b, c, d

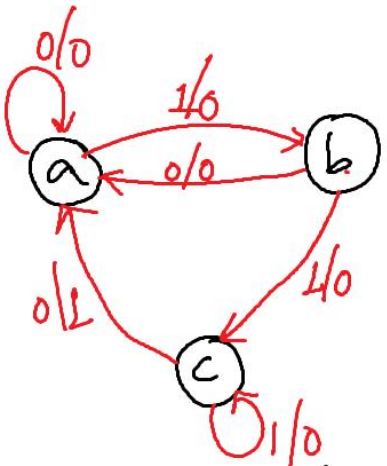
Write a state table & state transition table for the given state transition diagram:



Moore model := (4)

0, 1.

Problem statement

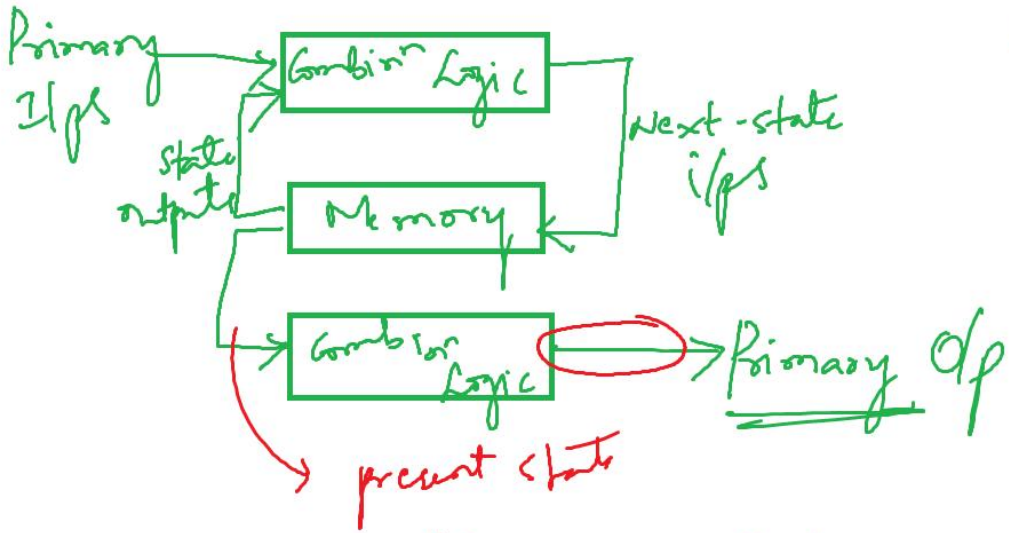


Mealy model : (3)

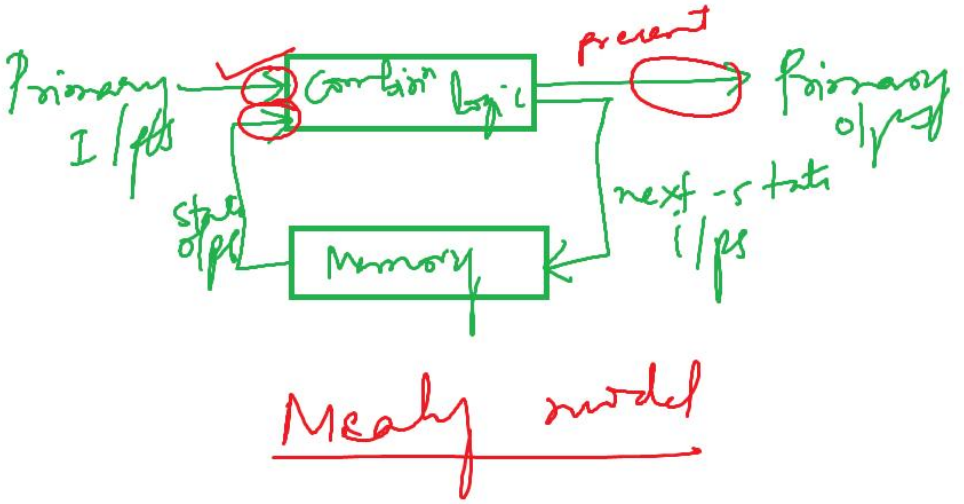
Unit - 5 : Contd...

05/01/2021

Model selection: Moore & Mealy model.



Moore model



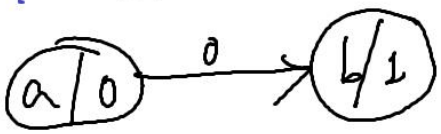
Mealy model

Problem: Design a sequential detector that receives binary data stream @ its i/p, X and signals when a combination '011' arrives @ the i/p by making its o/p Y high, which otherwise remains low

Consider, data is coming from the left.

Soln: i.e., the first bit identified is 1, second 1, third 0 from the i/p sequence : Use Moore model (or) Mealy model

Moore model:



Node: s/Y → o/p

present state

state

Arrow line → i/p

Mealy model: Y

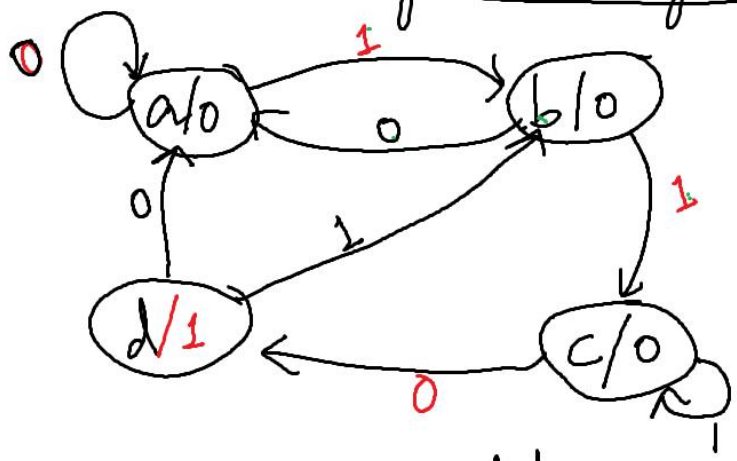
present state i/p

Node: state

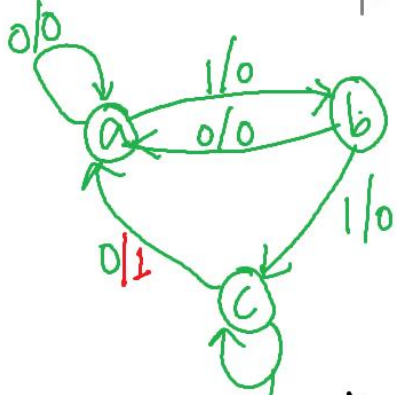
Arrow line: i/o

Problem: Design a sequential detector that receives binary-data-stream @ its i/p, X and signals when a combination '011' arrives @ the ' ' its o/p Y high. which otherwise remains low

State transition diagram using moore model:



Moore model = more no of states



Mealy machine

state = n-bit + 1

3-bit + 1 = 4 states = 011

3-bits

011

d c b

Y=0 01

1=Y = 011

0 = Y ✓

1 ✗ or

Y=0 11

Y=0 111 011

Y=1 0110

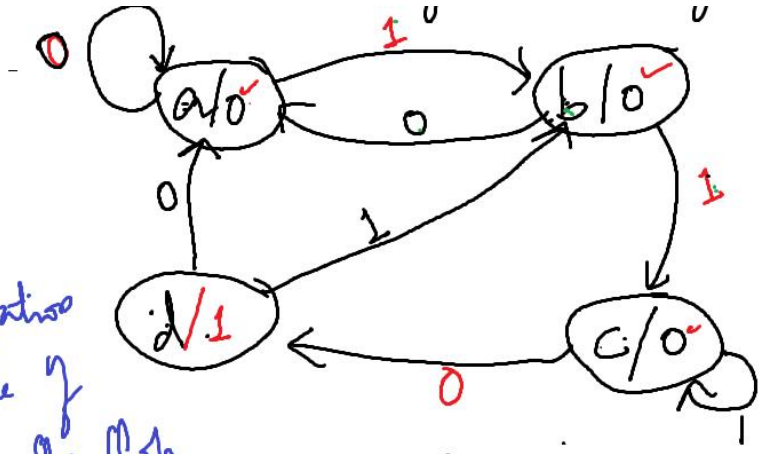
State Synthesis table for Moore Model:

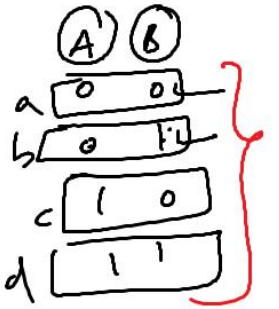
	Present state A	Present state B	Present I/p X	Next state A^f	Next state B^f	Output Y	J_A	K_A	J_B	K_B
a	0	0	0 ✓	0	0	0				
			1 ✓	0	1	0				
b	0	1	0 ✓	0	0	0				
			1 ✓	1	0	0				
c	1	0	0 ✓	1	1	0				
			1	1	0	0				
d	1	1	0	0	0	1				
			1	0	1	1				

Flip-flop I/ps

Excitation table of JK flip flop

A	B	A^f	B^f





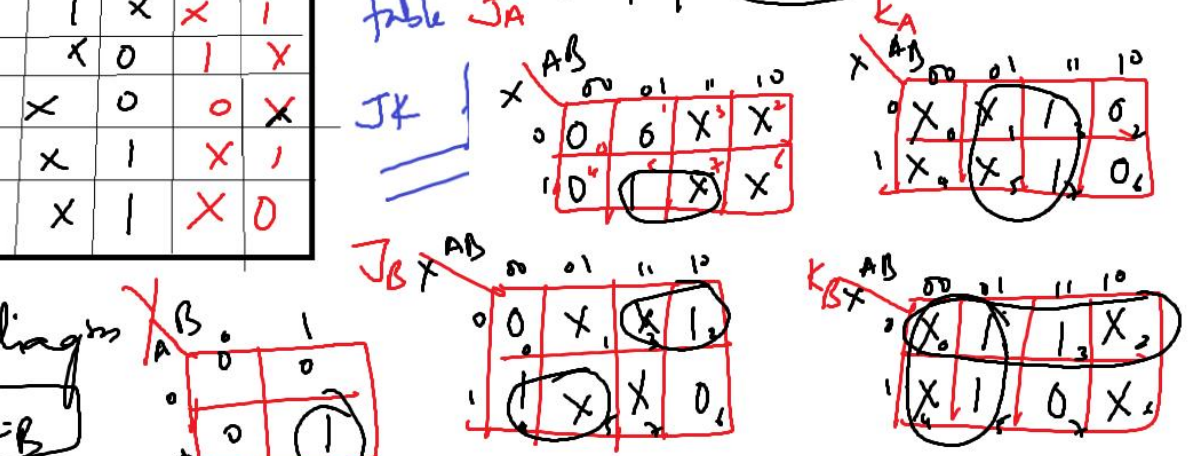
State synthesis table for Moore Model:

Present state A	Present state B	Present I/P X	Next state A^+	Next state B^+	O/P Y	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	1	X
0	1	0	0	0	0	0	X	X	1
0	1	1	1	0	0	1	X	X	1
1	0	0	1	1	0	X	0	1	X
1	0	1	1	0	0	X	0	0	X
1	1	0	0	0	1	X	1	X	1
1	1	1	0	1	1	X	1	X	0

Flip-flop I/ps

Excitation table JK

X	A	B	J_A	K_A	J_B	K_B
0	0	0	0	X	0	X
1	0	0	0	X	1	X
0	0	1	0	X	X	1
1	0	1	1	X	X	1
0	1	0	X	1	0	X
1	1	0	X	1	0	X
0	1	1	X	1	1	X
1	1	1	X	1	X	0



Design "equations" & draw ckt diagram

$\therefore J_A = X \cdot B$ $\quad K_A = B$

$\therefore J_B = X\bar{A} + \bar{X}A = X \oplus A = J_B$ $\quad K_B = \bar{X} + \bar{A}$

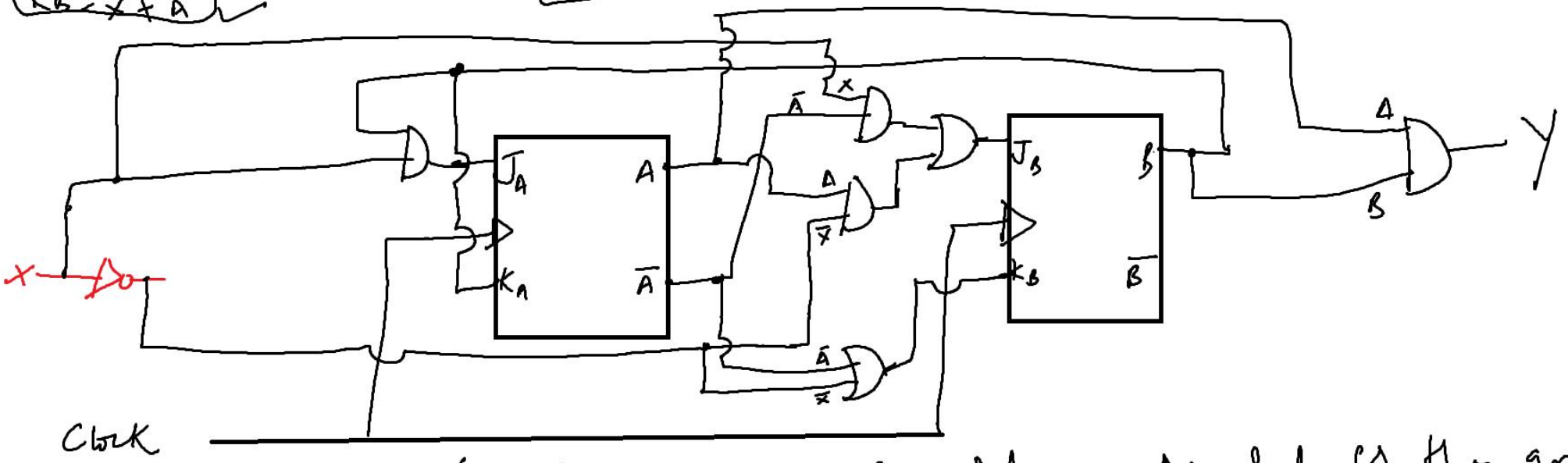
$\therefore Y = A \cdot B$

$J_A = X \cdot B$ $K_A = B$

$J_B = X\bar{A} + \bar{X}A = X \oplus A = J_B$

$K_B = \bar{X} + \bar{A}$

$\therefore Y = A \cdot B$

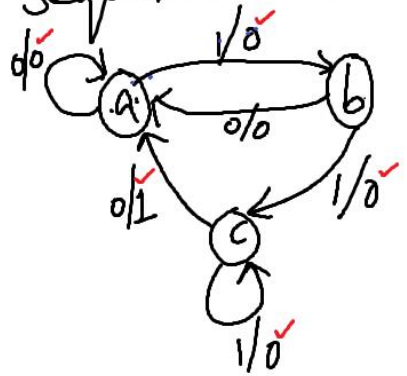


Circuit diagram for Moore Model for the given problem

011 ← left.

Same problem: Sequence detector

$Q^3 = 8$



011 ← left. using Mealy model.

3-bits = 3-state in Mealy

	A	B
a	0	0
b	0	1
c	1	0
d	1	1

State transⁿ diagram of Sequen detector (Mealy)

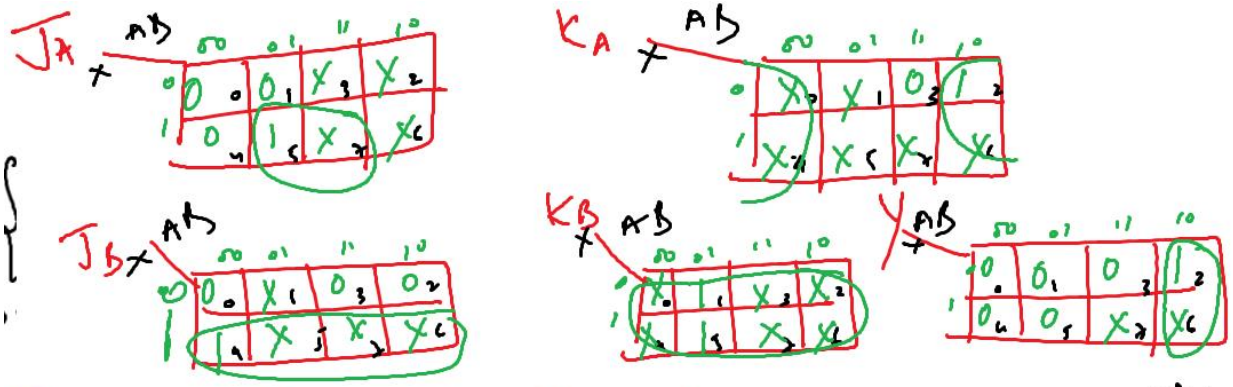
Q	Q+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State synthesis table : Mea

	Present state A	Present state B	Present I/p X	Next state A+	Next state B+	Present O/p Y	J_A	K_A	J_B	K_B
a	0	0	0	0	0	0	0	X	0	X
	0	0	1	0	1	0	0	X	1	X
b	0	1	0	0	0	0	0	X	X	1
	0	1	1	1	0	0	1	X	X	1
c	1	0	0	0	0	1	X	1	0	X
	1	0	1	1	0	0	X	0	0	X

I/ps for k-map : X, A, B. ∴ J_A K_A

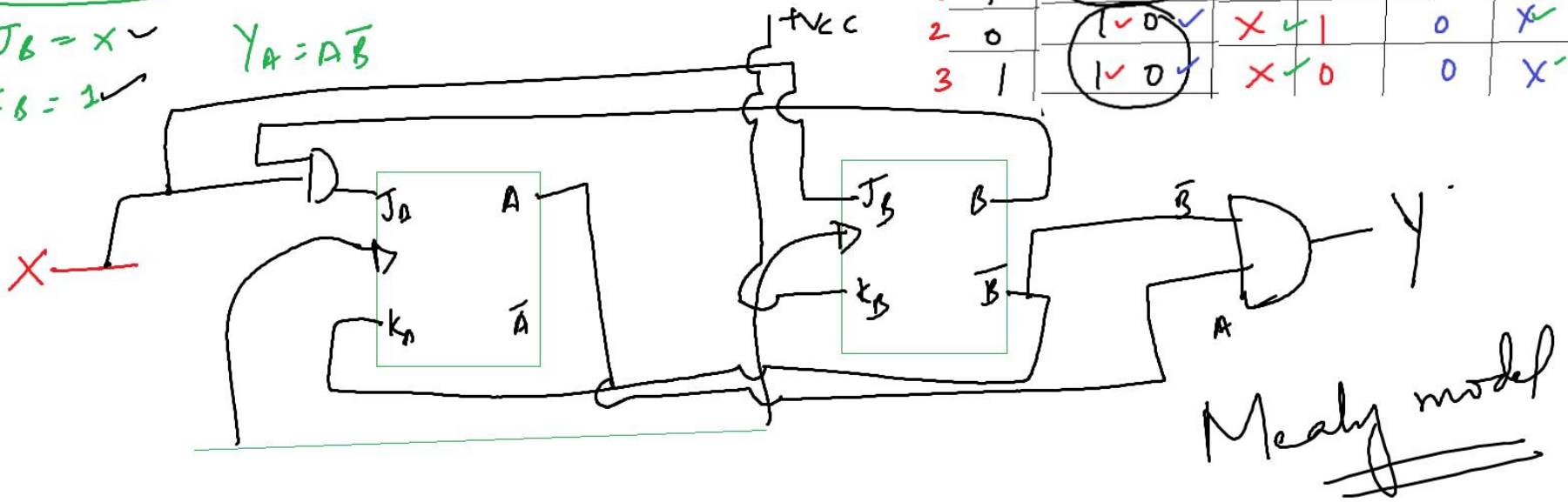
Y = J_B K_B



	Present input X	Present state A	Present state B	J_A	K_A	J_B	K_B	Next output Y
0	0	0	0	0	X	0	X	0
4	1	0	0	0	X	1	X	0
1	0	0	1	0	X	X	1	0
5	1	0	1	1	X	X	1	0
2	0	1	0	X	1	0	X	1
3	1	1	0	X	0	0	X	0

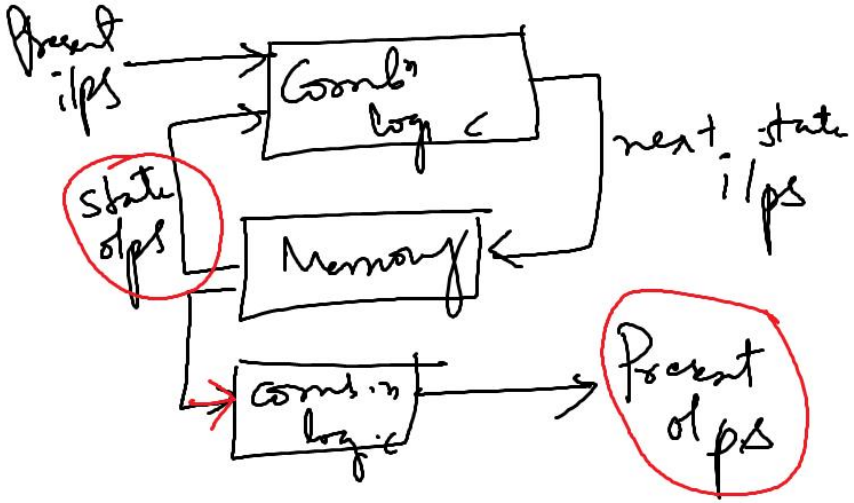
$J_A = XB$ $\quad J_B = X$ $\quad Y_A = A\bar{B}$

$K_A = \bar{B}$ $\quad K_B = 1$

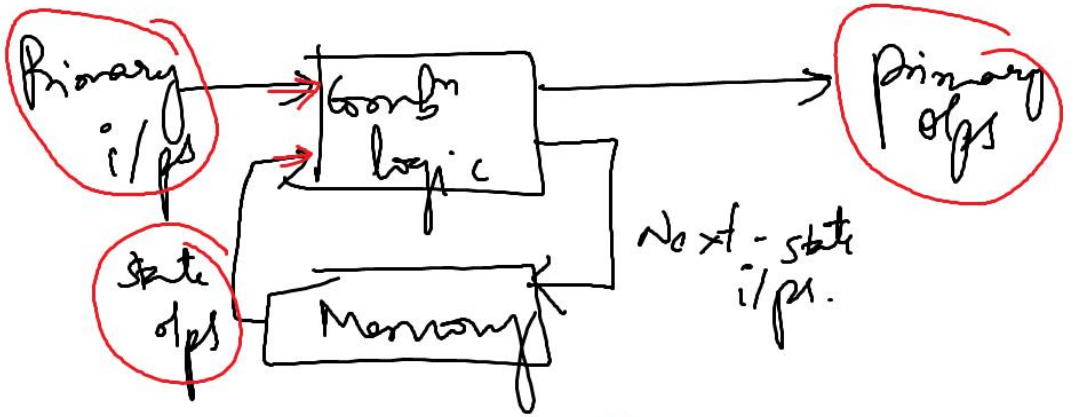


Mealy model

Design of Seq. Seq. ckt → Moore model
Mealy "

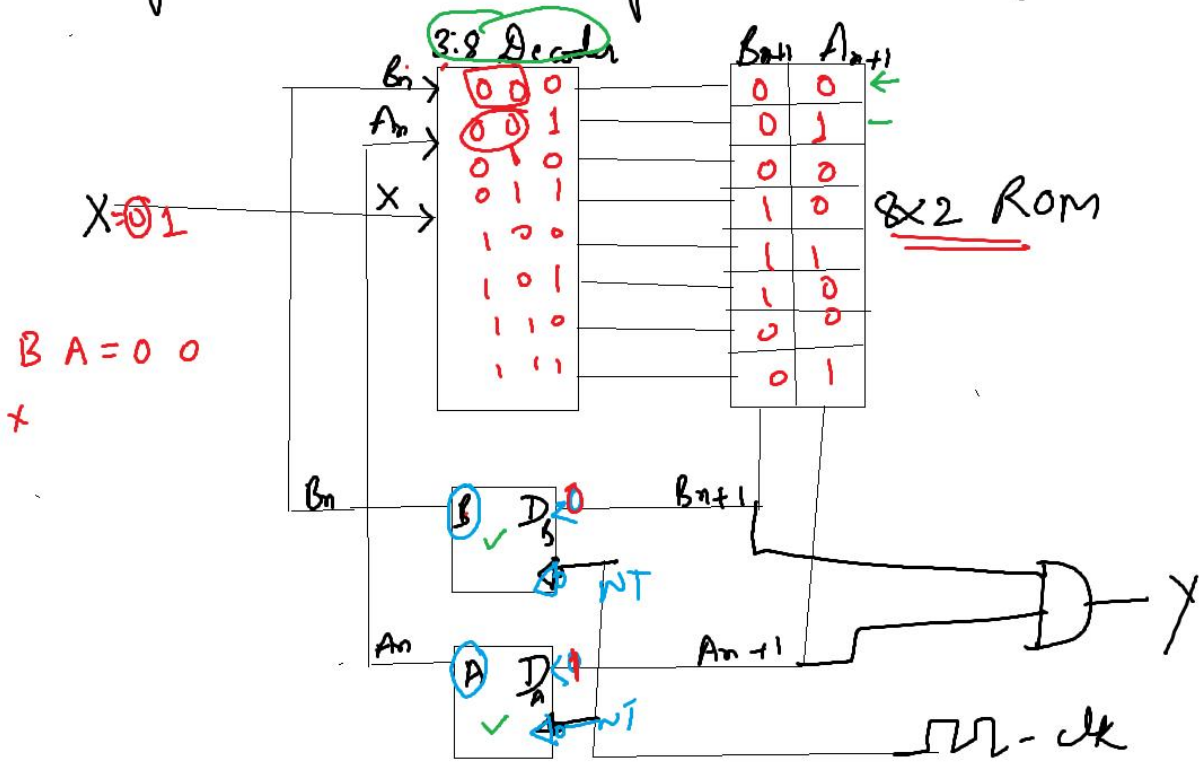


Moore — More no of state



Mealy — less no of states

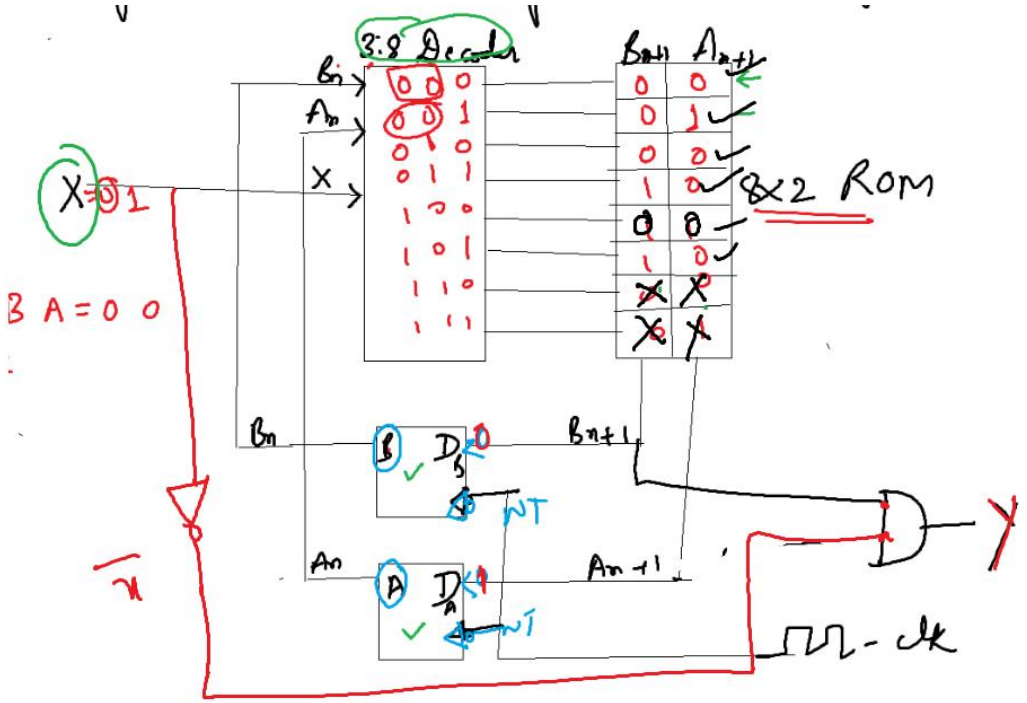
Implementation using Read Only Memory (ROM): o/p of ROM is available immediately

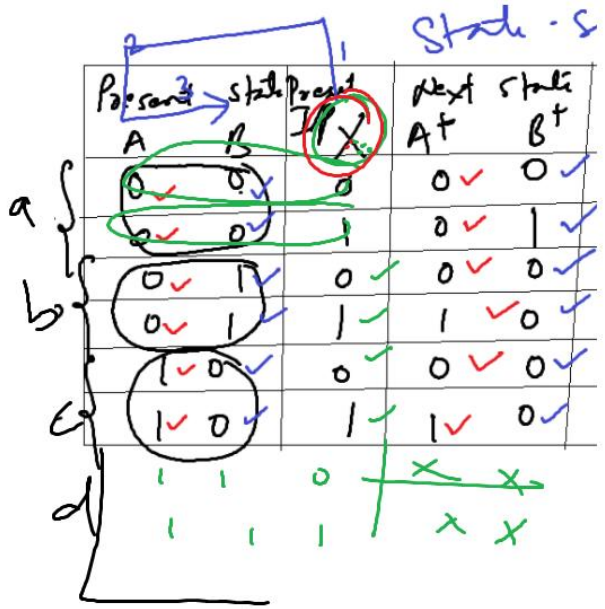


State synthesis table for Moore

ROM-based implementation of Seq. Detector

011 ← 11.2



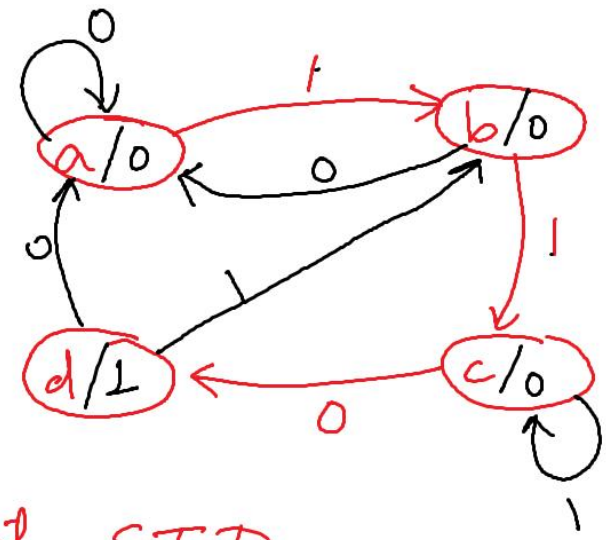


Mealy Model : $Y = \bar{X} \cdot B_n$

Present i/p + present state = present o/p $\Rightarrow Y = \bar{X} \cdot B_n$

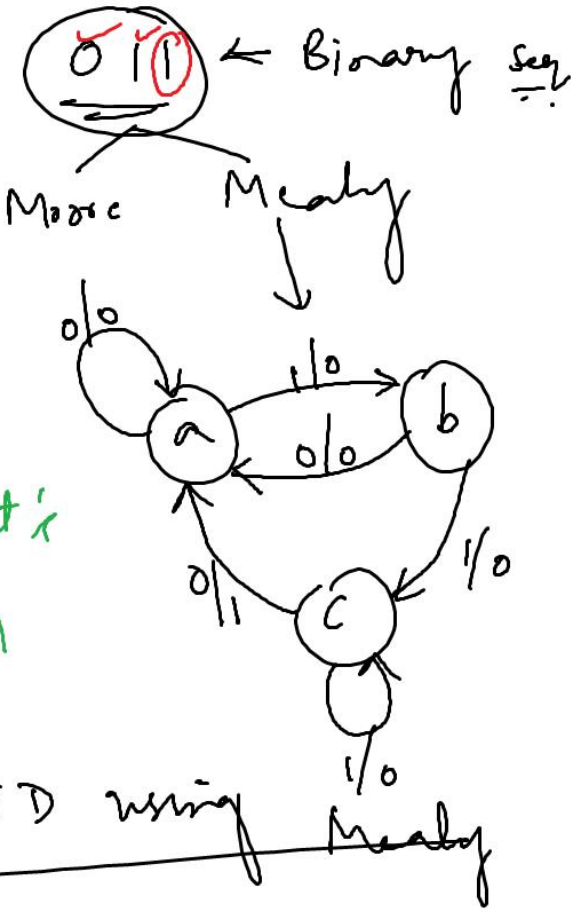
ASM: Algorithmic State M/c

STD using Moore:



Assignment: Construct ST Diagrams using Moore & Mealy to detect 101 and 010

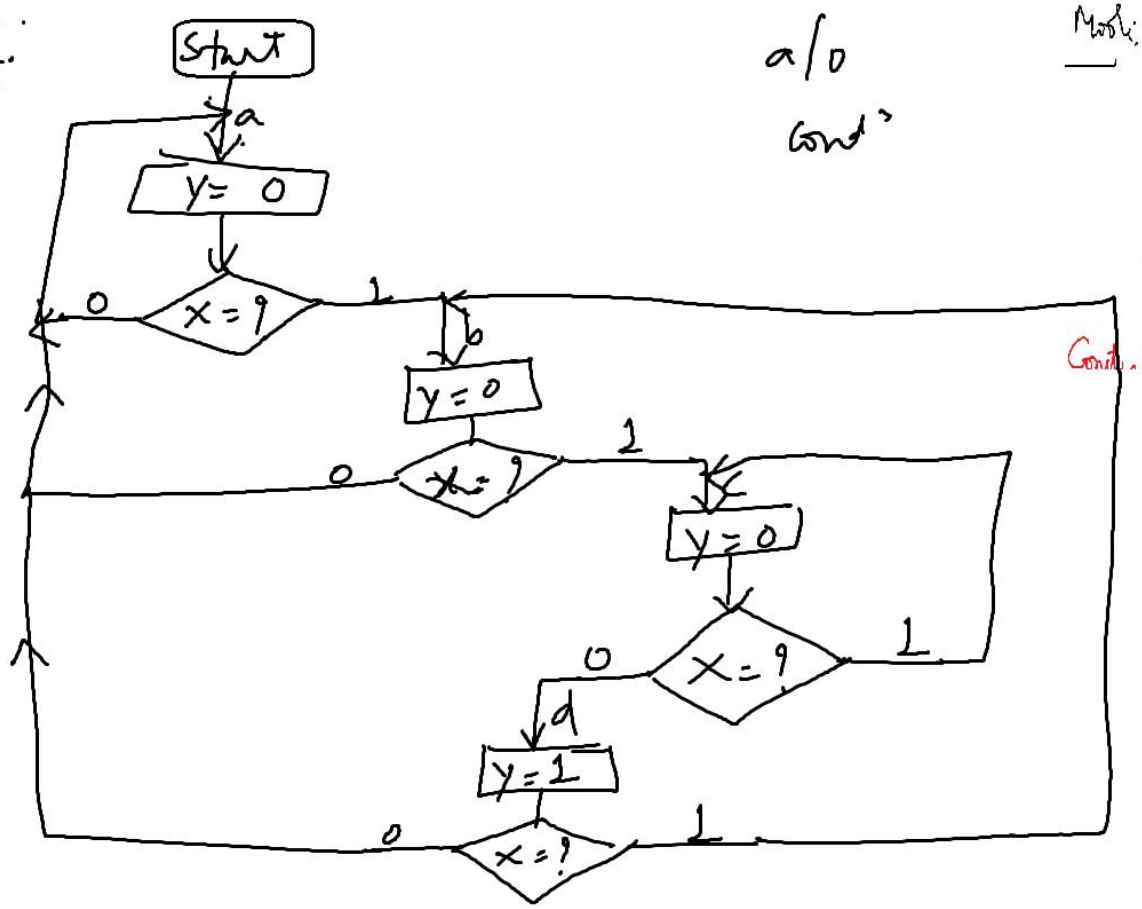
: 1 1 0



For this STDs, let's write ASM

STD using Mealy

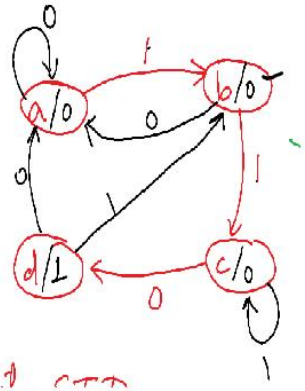
Algorithmic State M/C:



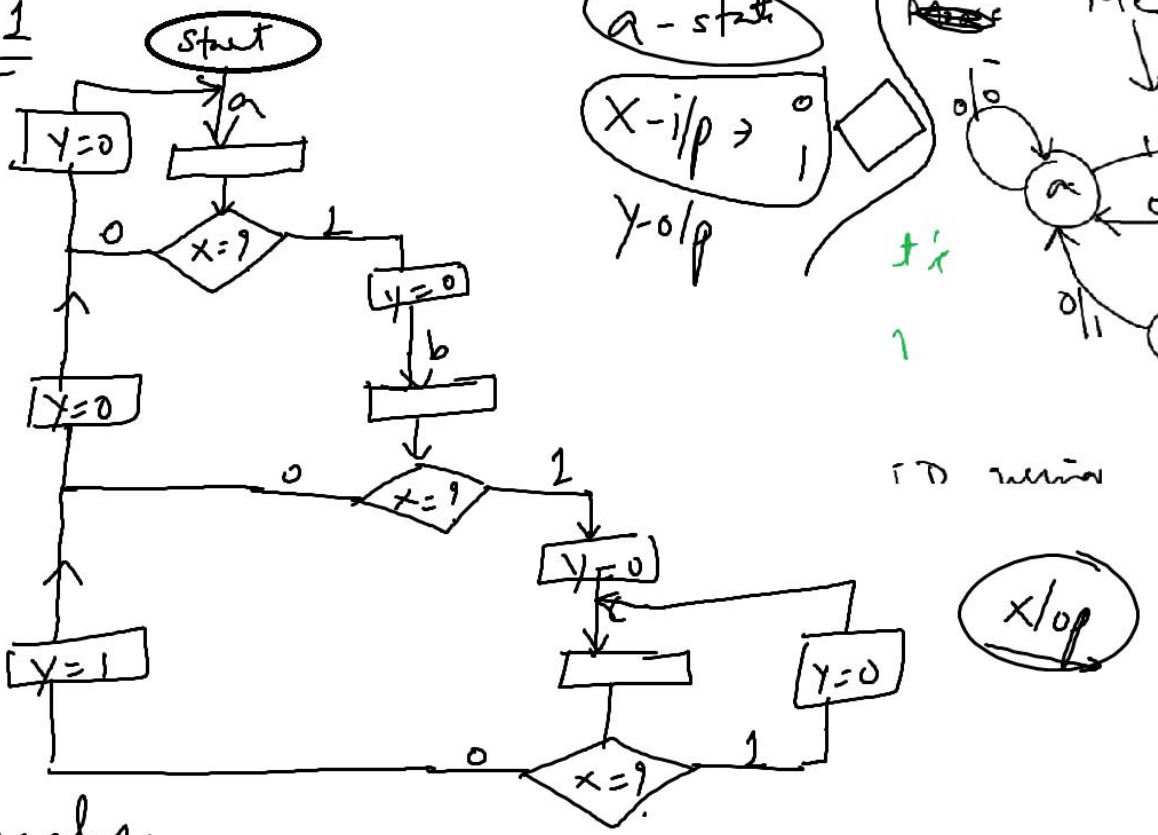
a/0

Cont'

Model:



ASM for Mealy : 011

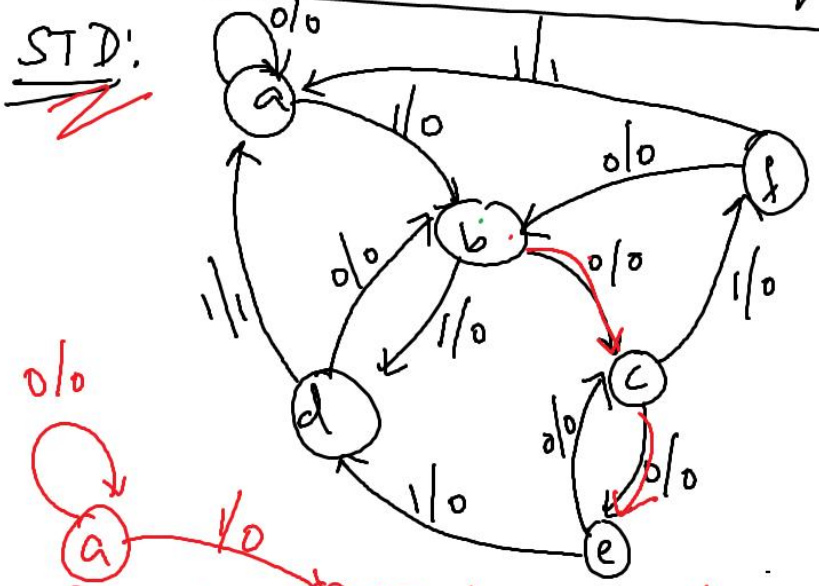


X/op

ASM using Mealy

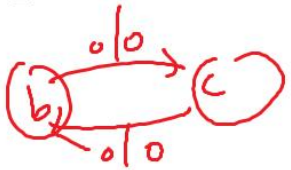
STATE Reduction Technique : (Mealy)

STD:



b = e

d = f



Row-Elimination Table

Present state	Next state x=0	Next state x=1	Present o/p x=0	Present o/p x=1
a	a	b	0	0
✓b	c	d	0	0 ✓
c	e	f	0	0
✓d	b	a	0	1
✓e	c	d	0	0 ✓ X- removed
✓f	b	a	0	1 — X clmn

6-states

(a) Original table.

Present state	Next state x=0	Next state x=1	Present o/p x=0	Present o/p x=1
a	a	b	0	0
✓b	c	d	0	0
✓c	b	d	0	0
d	b	a	0	1

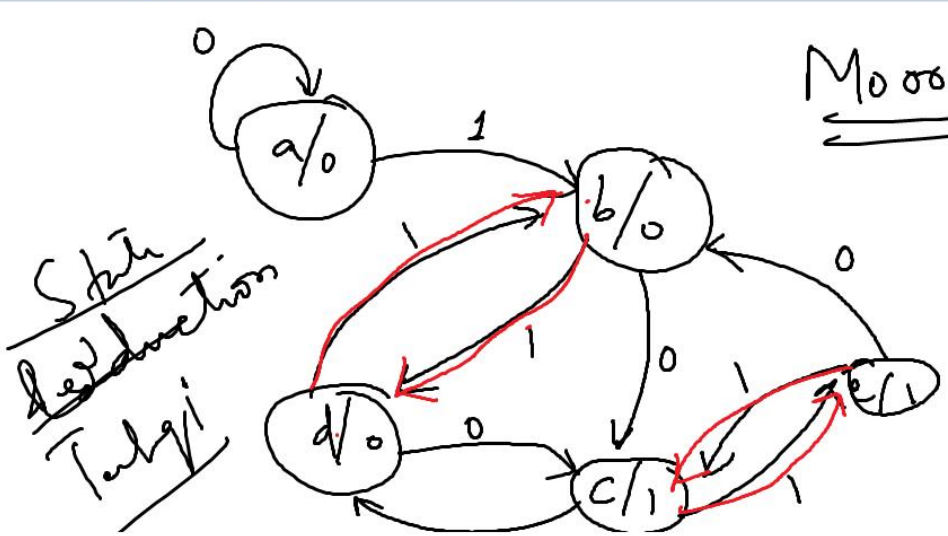
∴ b = c

P.S	N.S x=0	N.S x=1	P.O/P x=0	P.O/P x=1
a	a	b	0	0
b	b	d	0	0
d	b	a	0	1

Untitled - Paint

File Home View

Paste Cut Copy Clipboard | Select Crop Resize Rotate Image | Tools | Brushes | Shapes | Outline Fill | Size | Color 1 Color 2 Colors | Edit colors | Edit with Paint 3D



State reduction table

Moore soln's

P.S	N. S x=0	N. S x=1	P. o/p
a	a	b	0
√b	c	d	0
c	d	e	1
√d	c	b	0
e	b	c	1

∴ b = d

original table

S2; Apply - Reduction - check for redundant rows.

Since b = d, retains b, eliminate d

P.S	N. S x=0	N. S x=1	P. o/p
a	a	b	0
b	c	b	0
√c	b	c	1

S3 - Q = 0

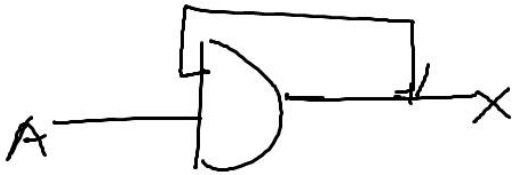
meet.google.com is sharing your screen. Stop sharing Hide

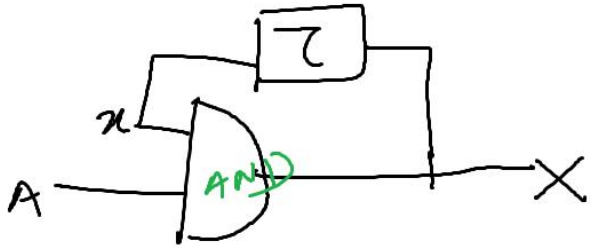
1894 × 888px 100% 4 new notifications

Type here to search ENG 13:48 06-02-2021

Analysis of Asynchronous Seq ckt

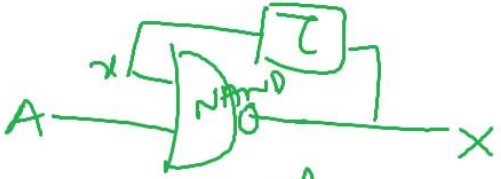
AND gate:

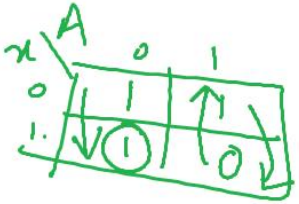




AND gate with prop delay

NAND gate:

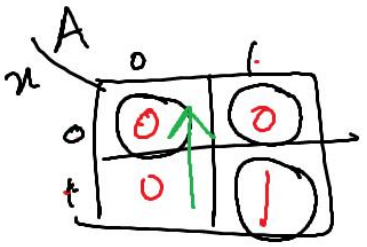




- oscillation

$X = x$ stable state

$X \neq x$ unstable.



horizontally / vertically.

Un - stable