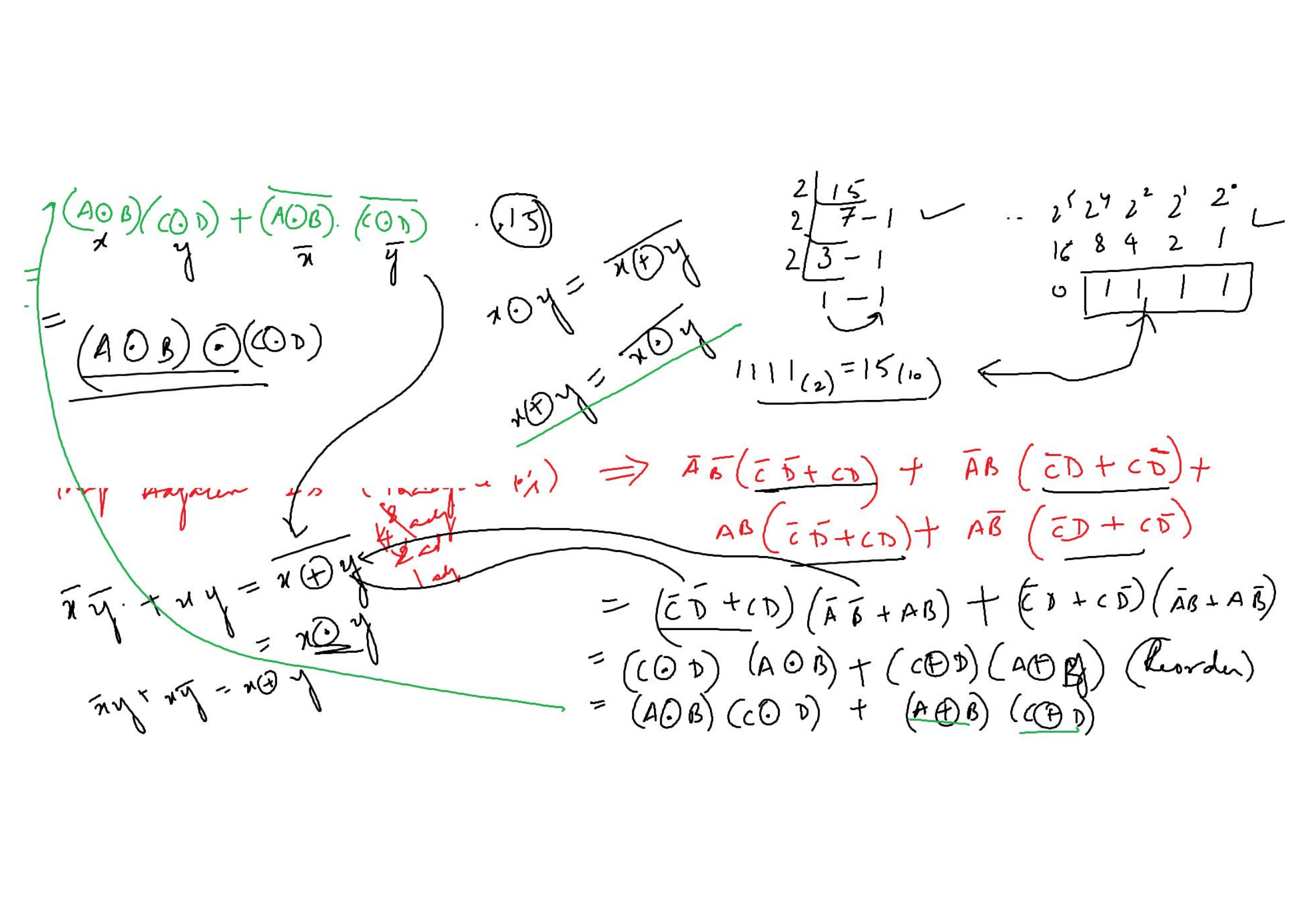
1) Show that

$$\Sigma m(0, 3, 5, 6, 9, 10, 12, 15) = A \odot B \odot C \odot D$$

2) Reduce the following f's using k-map technique and implement using the gates

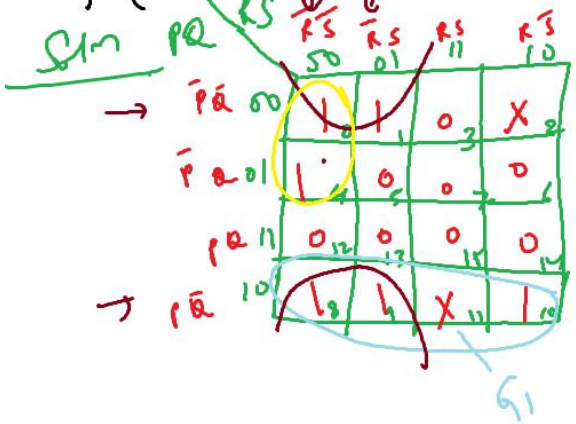
i) $f(P, Q, R, S) = \Sigma m(0, 1, 4, 8, 9, 10) + d(2, 11)$

ii) $f(A, B, C, D) = \Pi M(0, 2, 4, 10, 11, 14, 15)$


(A⊙B)(C⊙D) + (A⊙B)·(C⊙D)
x y x̄ ȳ
= (A⊙B)⊙(C⊙D)
x⊙y = x⊕y
x⊕y = x⊙y
x̄ȳ + xy = x⊕y
= x⊙y
x̄y + xȳ = x⊕y
(15)
2 | 15
2 | 7 - 1
2 | 3 - 1
1 - 1
2⁵ 2⁴ 2³ 2² 2¹ 2⁰
16 8 4 2 1
0 1 1 1 1
1111(2) = 15(10)
⇒ Ā B̄ (C̄ D̄ + CD) + ĀB (C̄D + CD̄) +
AB (C̄ D̄ + CD) + AB̄ (C̄D + CD̄)
= (C̄ D̄ + CD)(Ā B̄ + AB) + (C̄D + CD̄)(ĀB + AB̄)
= (C⊙D)(A⊙B) + (C⊕D)(A⊕B) (Reorder)
= (A⊙B)(C⊙D) + (A⊕B)(C⊕D)

k-map & Realize/Implement using gates

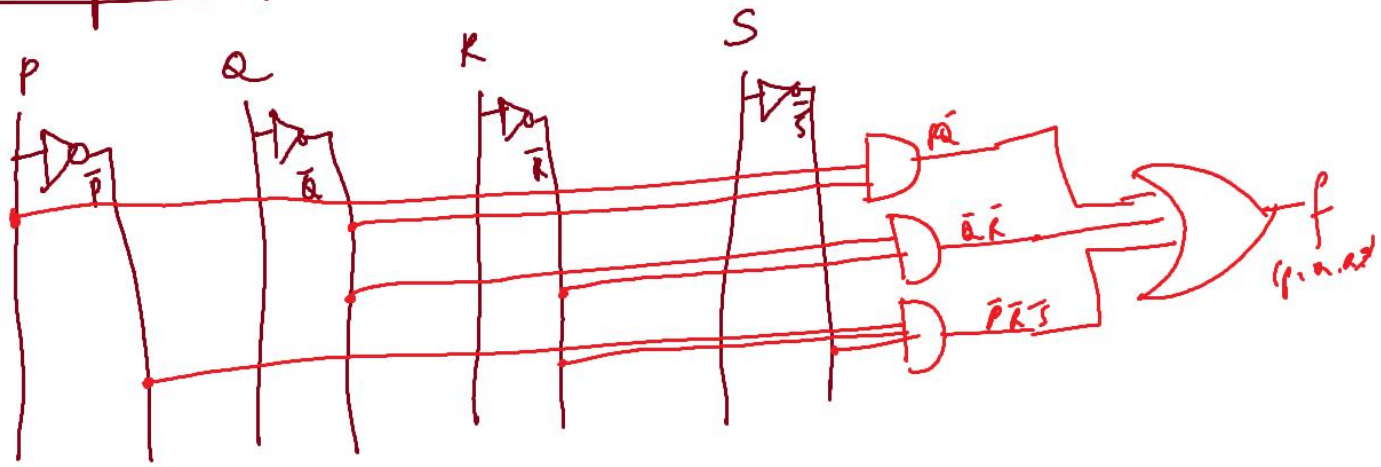
$f(P,Q,R,S) = \Sigma m(0,1,4,8,9,10) + d(2,11)$



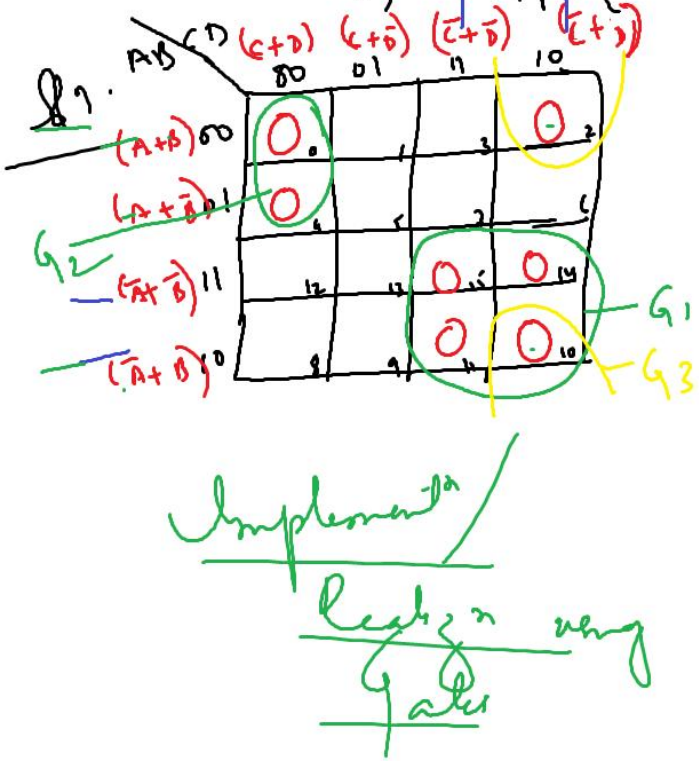
The simplified B.E = $G_1 + G_2 + G_3$

$\bar{S} \cdot B.E = P\bar{Q} + \bar{Q}\bar{R} + \bar{P}\bar{R}\bar{S}$

Implementation / Realization



$f(A, B, C, D) = \prod M(0, 2, 4, 10, 11, 14, 15)$



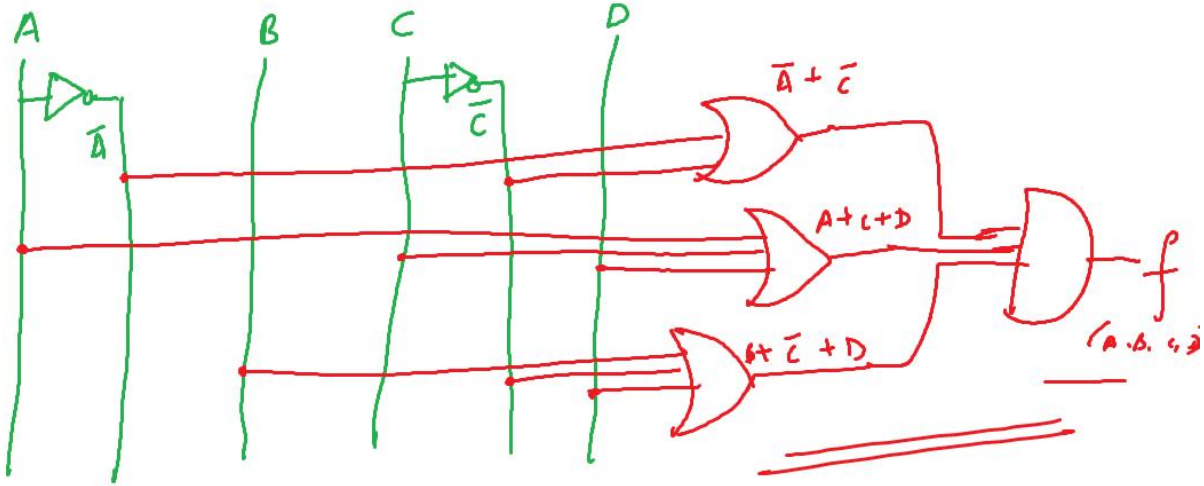
Implementation Realization using gates

16 & 4

$G_1 = \bar{A} + \bar{C}$ $G_2 = A + C + D$ $G_3 = B + \bar{C} + D$

Reduced B.E $= G_1 \cdot G_2 \cdot G_3$

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



Quine - McCluskey Minimization technique (minterms) - SOP

Quine-McCluskey method => 5

Simplify the following Boolean fn using

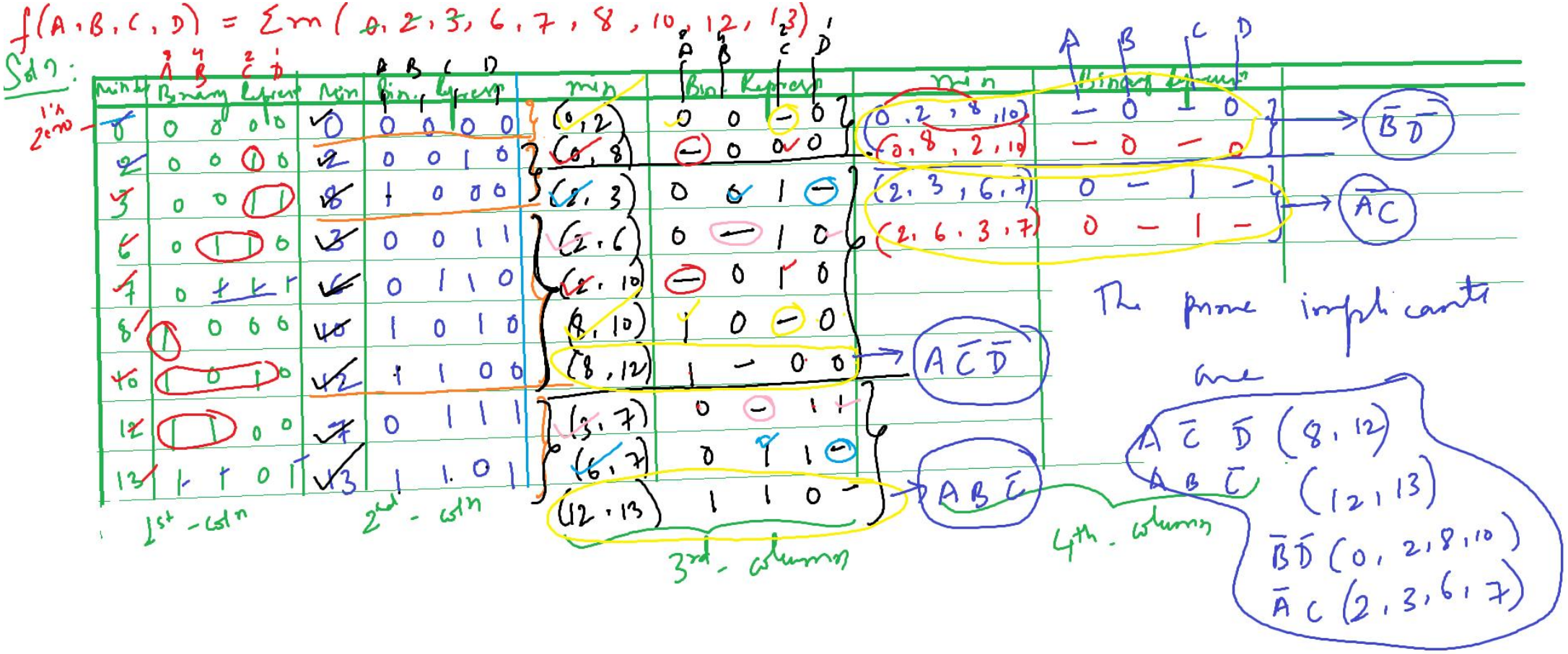
$f(A, B, C, D) = \Sigma m(0, 2, 3, 6, 7, 8, 10, 12, 13)$

Sol:

1. List all the minterms in binary form
2. Arrange the minterms according to no. of 1's
3. Compare each binary no with every term in the adjacent next higher category and if they differ only by 1-posn, then put a check mark and copy the term in the next column with hyphen (-) in the position that they differed.
4. Apply the same process (as in step 3) until there is no elimin^n of literals.

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

Sol?:



Step 5: List all the prime implicants

Prime Implicants	Bin Representn
$A\bar{C}\bar{D}$ (8, 12)	1 – 0 0
$AB\bar{C}$ (12, 13)	1 1 0 –
$\bar{B}\bar{D}$ (0, 2, 8, 10)	– 0 – 0
$\bar{A}C$ (2, 3, 6, 7)	0 – 1 –

Step 6: Select the min no. of prime implicants which MUST cover ALL the minterms

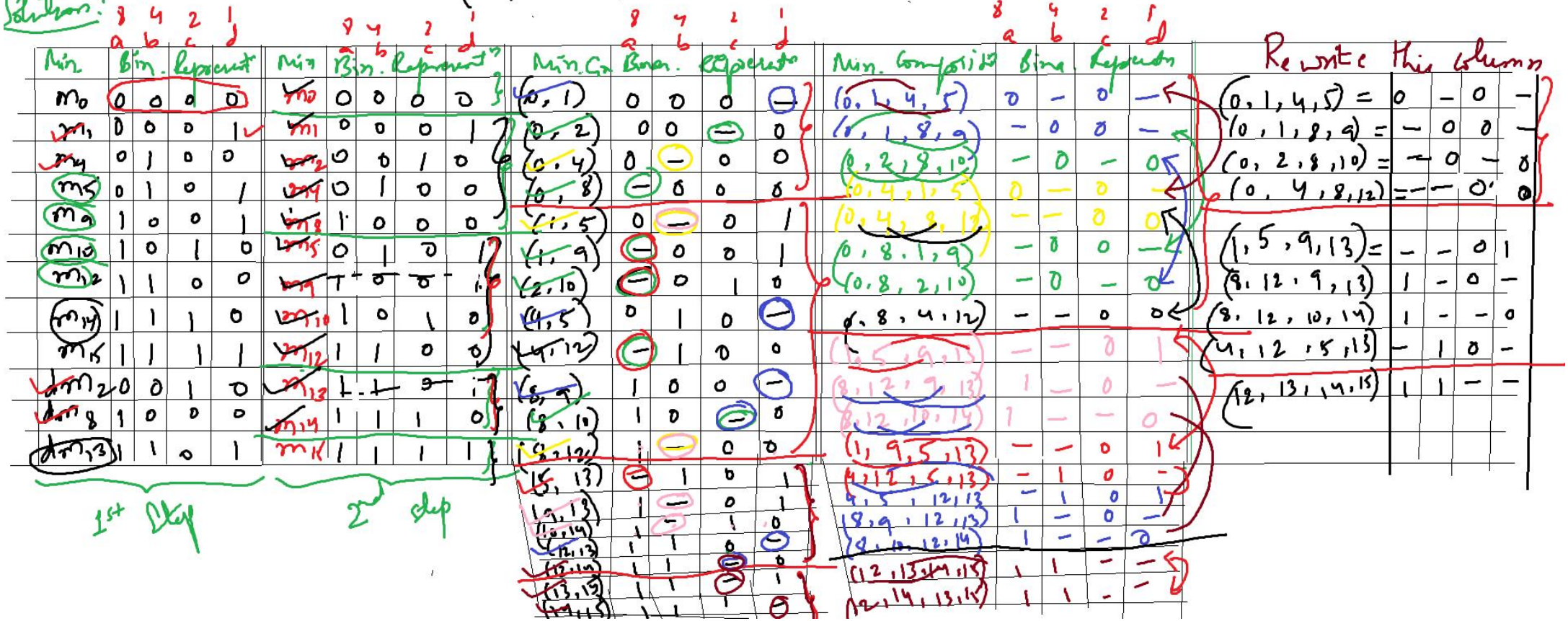
Prime Implicants	m_0 wt-1	m_2 wt-2	m_3 wt-3	m_6 wt-4	m_7 wt-5	m_8 wt-6	m_{10} wt-7	m_{12} wt-8	m_{13} wt-9
$A\bar{C}\bar{D}$ (8, 12)						⊙ x		⊙ x	
$AB\bar{C}$ (12, 13)								⊙ x	⊙
$\bar{B}\bar{D}$ (0, 2, 8, 10)	⊙	⊙ x				⊙	⊙		
$\bar{A}C$ (2, 3, 6, 7)		⊙ x	⊙	⊙	⊙				

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

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

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

Solution:



Rewrite this column

	a	b	c	d
(0, 1, 4, 5) =	0	0	0	–
(0, 1, 8, 9) =	0	0	0	–
(0, 2, 8, 10) =	0	–	0	–
(0, 4, 8, 12) =	0	–	0	–
(1, 5, 9, 13) =	–	–	0	1
(8, 12, 9, 13) =	1	–	0	–
(8, 12, 10, 14) =	1	–	–	0
(4, 12, 5, 13) =	–	1	0	–
(12, 13, 14, 15) =	1	1	–	–

$\bar{b}\bar{d}$

$a\bar{d}$

ab

Step 1: Prime Implicants

min. comp	Binary Rep
$\bar{b}\bar{d}$ = (	– 0 – 0
$a\bar{d}$ = (	1 – – 0
ab = (	1 1 – –
$\bar{c}$ = (✓	0 –

Minterms Comp^n	Binary Representn
(0, 1, 4, 5, 8, 12, 9, 13)	– – 0 –
(0, 1, 8, 9, 4, 12, 5, 13)	– – 0 –
(0, 4, 8, 12, 1, 5, 9, 13)	– – 0 –

Step 2

(0, 1, 4, 5, 8, 12, 9, 13)

a b c d

= – – 0 –

↓ $\bar{c}$

Step 1: Select min. prime – ALL minterms

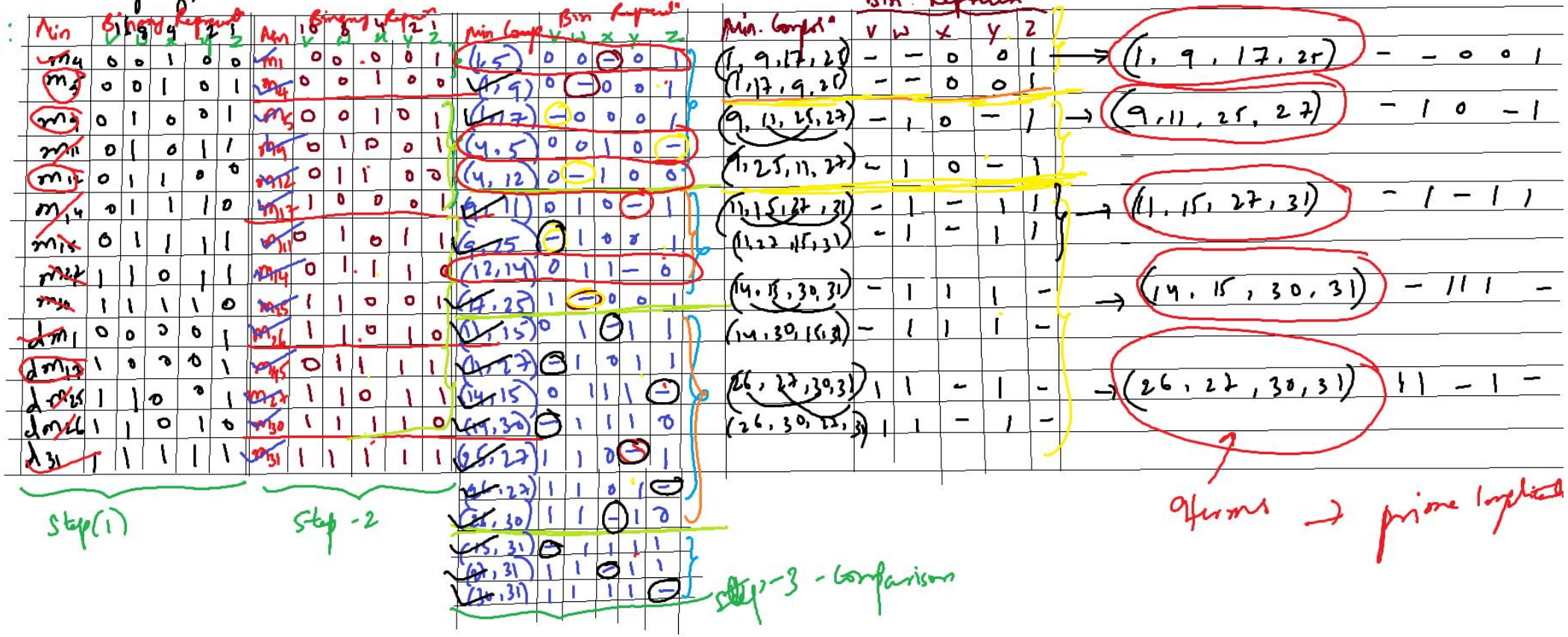
Prime Implicants	m_0	m_1	m_4	m_5	m_9	m_{10}	m_{12}	m_{14}	m_{15}	dm_2	dm_8	dm_{13}
✓ $\bar{b}\bar{d}$ = (0, 2, 8, 10)	⊙					⊙				⊙	⊙	
✓ $a\bar{d}$ = (8, 12, 10, 14)						⊙	⊙	⊙		⊙	⊙	
✓ ab = (12, 13, 14, 15)							⊙	⊙	⊙			⊙
✓ $\bar{c}$ = (0, 1, 4, 5, 8, 12, 9, 13)	⊙	⊙	⊙	⊙	⊙		⊙		⊙		⊙	⊙

$f(A, B, C, D) = \bar{c} + ab + \bar{b}\bar{d}$ → final simplified Boolean Expr^n using Quine Mc.-cluskey.

Obtain all the prime implicants of the function

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

Soln:



Prime Implicants	Binary Representation (v w x y z)
(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 − ✓

$\Sigma m(4, 5, 9, 11, 12, 14, 15, 27, 30) + d_c(1, 17, 25, 26, 31)$

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

$= (0\text{-}001) + (11\text{-}1\text{-})$

Last Step:

→ Prime Implicants Selection Chart:

Prime Implicants	m_4 (1)	m_5 (2)	m_9 (3)	m_{11} (4)	m_{12} (5)	m_{14} (6)	m_{15} (7)	m_{27} (8)	m_{30} (9)	dm_1 (10)	dm_{17} (11)	dm_{25} (12)	dm_{26} (13)	dm_{31} (14)
(1, 5) = $\bar{v}\bar{w}\bar{y}z$		⊙								⊙				
(4, 5) = $\bar{v}\bar{w}x\bar{y}$	⊙	⊙												
(4, 12) = $\bar{v}x\bar{y}\bar{z}$	⊙				⊙									
(12, 14) = $\bar{v}wx\bar{z}$					⊙	⊙								
✓ (1, 9, 17, 25) = $\bar{x}\bar{y}z$			⊙							⊙	⊙ ✓	⊙		
(9, 11, 25, 27) = $w\bar{x}z$			⊙	⊙				⊙				⊙		
(11, 15, 27, 31) = wyz				⊙			⊙	⊙						⊙
(14, 15, 30, 31) = wxy						⊙	⊙		⊙					⊙
✓ (26, 27, 30, 31) = vwy								⊙	⊙				⊙ ✓	⊙

Simplify using Quine McCluskey Tabulation algorithm,

$V = f(a, b, c, d) = \Sigma(2, 3, 4, 5, 13, 15) + \Sigma d(8, 9, 10, 11)$

Solution

Min	a	b	c	d
m_2	0	0	1	0
m_3	0	0	1	1
m_4	0	1	0	0
m_5	0	1	0	1
m_{13}	1	1	0	1
m_{15}	1	1	1	1
d_8	1	0	0	0
d_9	1	0	0	1
d_{10}	1	0	1	0
d_{11}	1	0	1	1

step (1)

Min	a	b	c	d
m_2	0	0	1	0
m_3	0	0	1	1
m_4	0	1	0	0
m_5	0	1	0	1
m_8	1	0	0	0
m_9	1	0	0	1
m_{10}	1	0	1	0
m_{11}	1	0	1	1
m_{13}	1	1	0	1
m_{15}	1	1	1	1

step (2)

Min. comp	a	b	c	d
(2, 3)	0	0	1	–
(2, 10)	–	0	1	0
(4, 5)	0	1	0	–
(8, 9)	1	0	0	–
(8, 10)	1	0	–	0
(9, 11)	–	0	1	1
(5, 13)	–	1	0	1
(9, 11)	1	0	–	1
(6, 13)	1	–	0	1
(10, 11)	1	0	1	–
(11, 15)	1	–	1	1
(13, 15)	1	1	–	1

step (3) comp

Min. comp	a	b	c	d
(2, 3, 10, 11)	–	0	1	–
(2, 10, 3, 11)	–	0	1	–
(8, 9, 10, 11)	1	0	–	–
(8, 10, 9, 11)	1	0	–	–
(9, 13, 11, 15)	1	–	–	1
(9, 11, 13, 15)	1	–	–	1

step (4) - comp

Min. comp	a	b	c	d
(2, 3, 10, 11)	–	0	1	–
(8, 9, 10, 11)	1	0	–	–
(9, 13, 11, 15)	1	–	–	1

$\bar{b}c$

$a\bar{b}$

ad

Final Expn $= \bar{a}b\bar{c} + \bar{b}c + a\bar{b} + ad$

Prime Implicants

(4, 5) $= \bar{a}b\bar{c}$

(5, 13) $= bc\bar{d}$

(2, 3, 10, 11) $= \bar{b}c$

(8, 9, 10, 11) $= a\bar{b}$

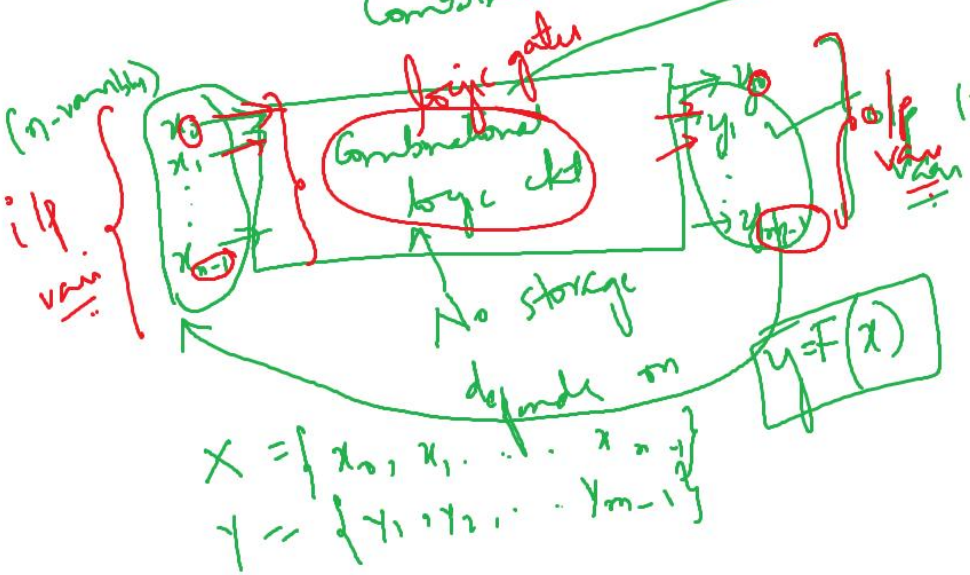
(9, 13, 11, 15) $= ad$

	1	2	3	4	5	6	7	8	9	10
	m_2	m_3	m_4	m_5	m_{13}	m_{15}	d_{m8}	d_9	d_{10}	d_{11}
(4, 5)			⊙	⊙						
(5, 13)				⊙	⊙					
(2, 3, 10, 11)	⊙✓	⊙							⊙	⊙
(8, 9, 10, 11)							⊙✓	⊙	⊙	⊙
(9, 13, 11, 15)					⊙	⊙✓		⊙		⊙

Unit – 2

Data processing circuits → collection of logic gates

- Combinational
- Sequential

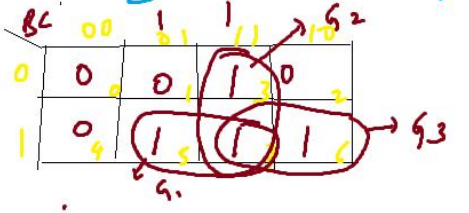


Ex: A combinational logic ckt with 3 i/p vars that will produce a logic-1 when > one i/p vars are = 1. Derive the TT. & implement the ckt.

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

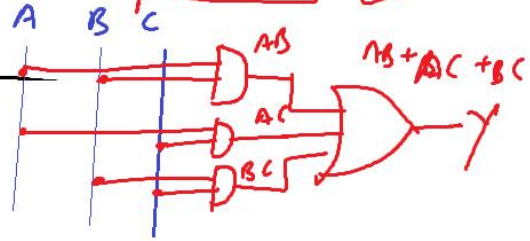


$Y = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

$Y = AC + BC + AB$

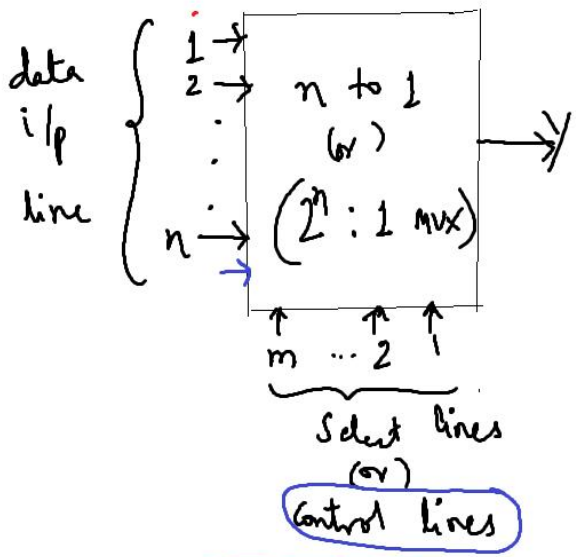
$\bar{Y} = \overline{AB + AC + BC}$

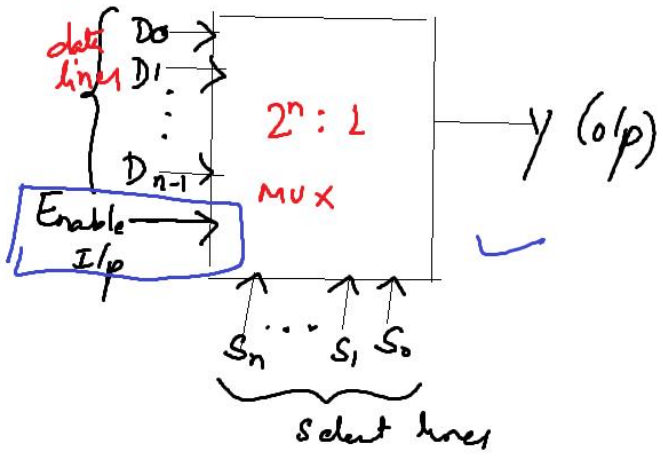
Implementation:



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

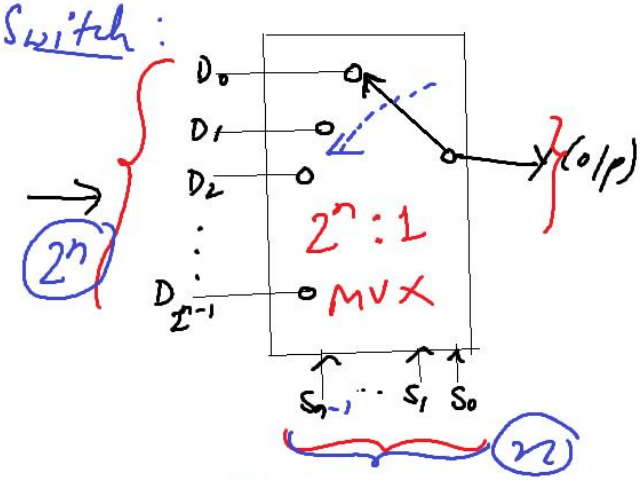
↳ data selector - many to 1 -





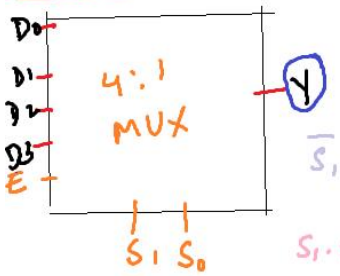
Block diagram of 2^n : 1 MUX

Equivalent ckt



Equivalent ckt.

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

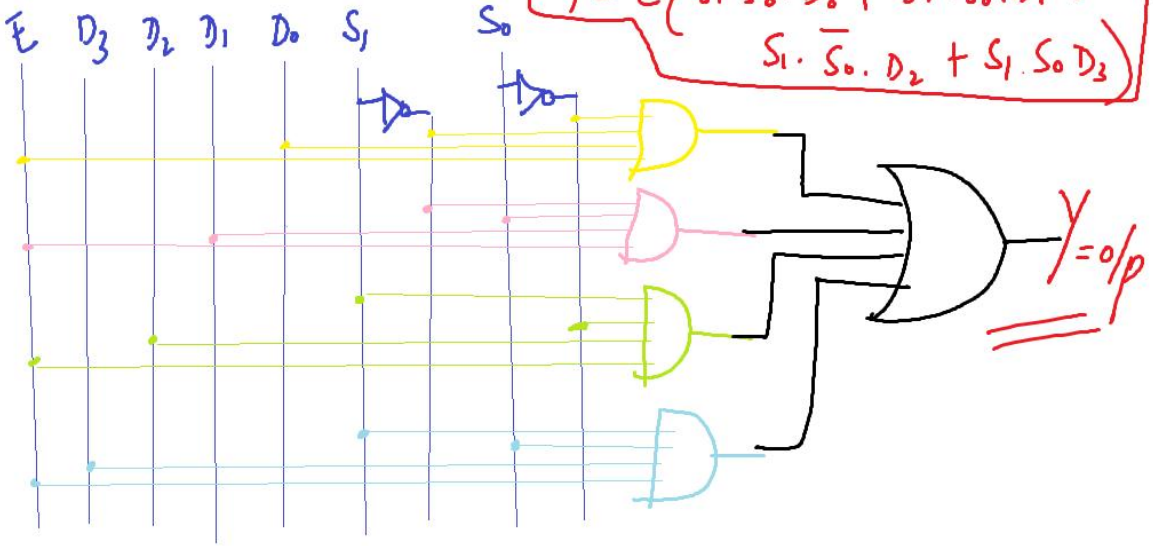
$\bar{S}_1 \bar{S}_0 . D_0 E$

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

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

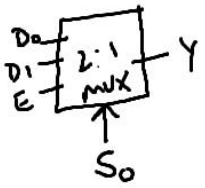
$S_1 . S_0 . D_3 E$

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



2:1 MUX:

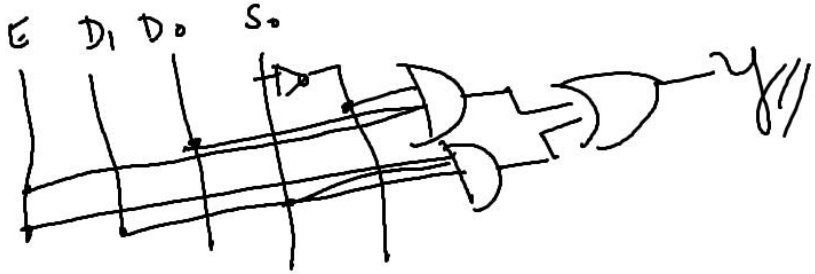
Select line = 1
data lines = 2



f^n table / Truth table

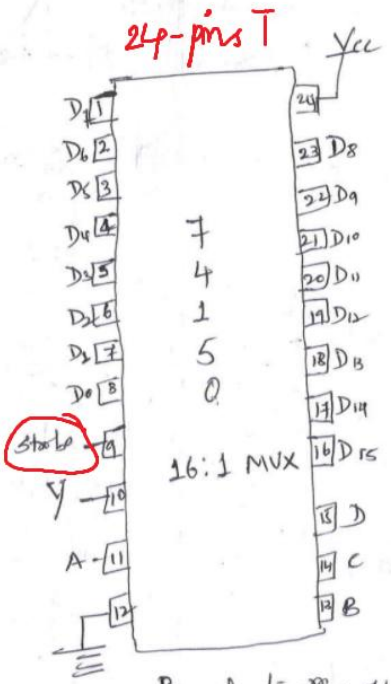
E	S_0	Y
1	0	D_0
1	1	D_1

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



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

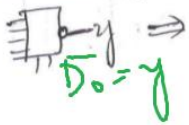
74150 Truth Table



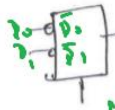
Pinout diagram of 74150

can't be determined

Bubbles on Signal line:



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



⇒ 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

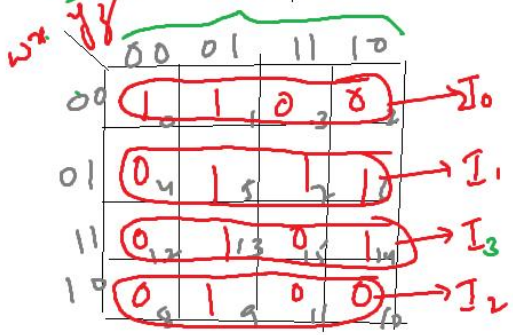
Realization of 4 variable f using 4:1 MUX

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

Soln:

4-variables → 2^4 → 16:1 MUX

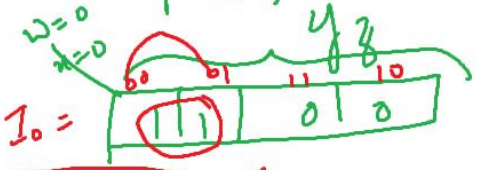
2 variable → 2^2 → 4:1 MUX



Common - constant literals → select lines

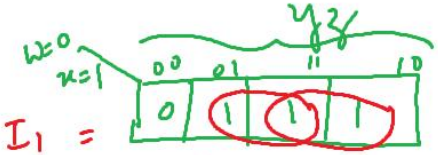
varying → data lines (yz)

Submaps: for I_0, I_1, I_2 and I_3



$I_0 =$

$I_0 = \bar{y}$ ✓

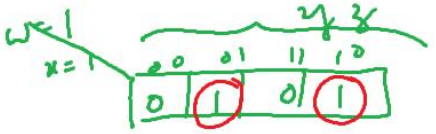


$I_1 =$

$= z + y$

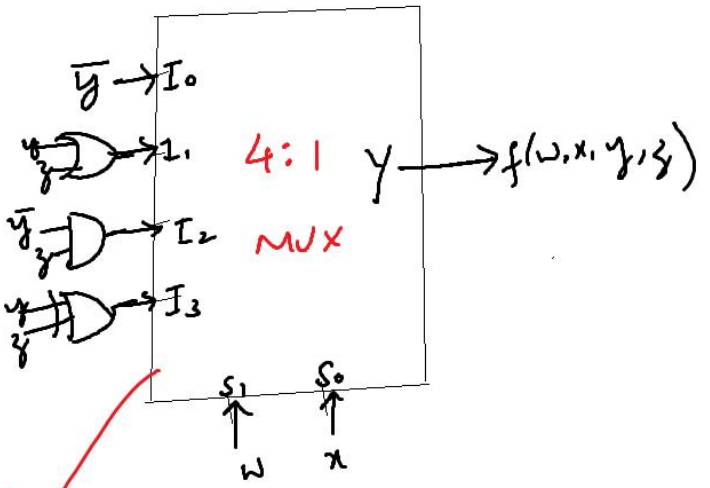
$I_1 = y + z$ ✓

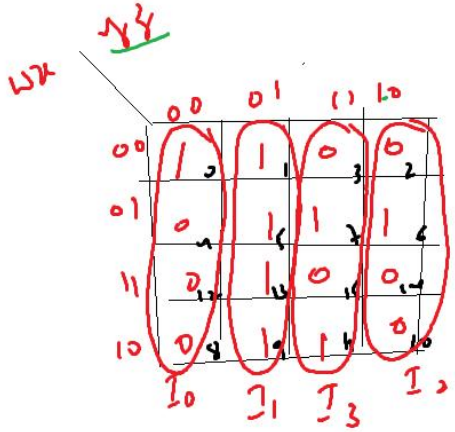
$I_2 = \bar{y}z$ ✓

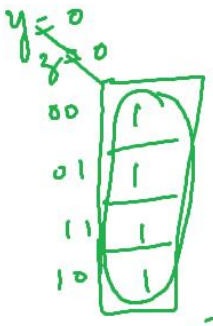


$I_3 = \bar{y}z + y\bar{z} = y \oplus z$

$I_3 = y \oplus z$







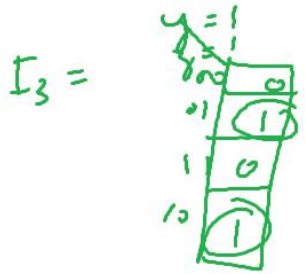
$= w\bar{x} + \bar{w}x + wx + w\bar{x}$

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

$1 + 0 = 1$

$I_1 = 1$ ✓

$I_2 = \bar{w}x$ ✓

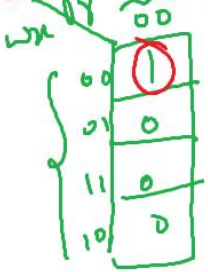


$\bar{w}x + w\bar{x}$

$I_3 = w \oplus x$ ✓

Submaps for I_0, I_1, I_2, I_3

$I_0 = y'z'$ Select lines

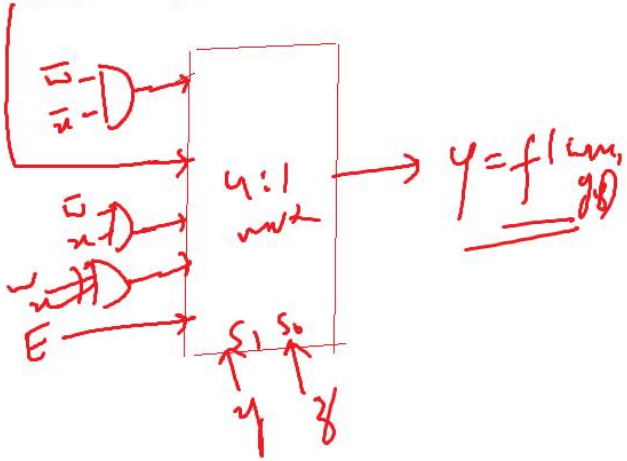


→ wx → data lines

yz → select lines

$I_0 = \bar{w} \cdot \bar{x}$ ✓

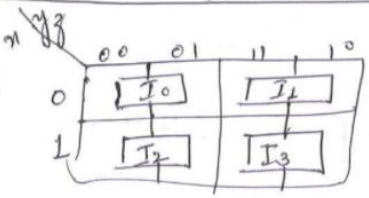
Vcc = logic 1



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

2 ways:

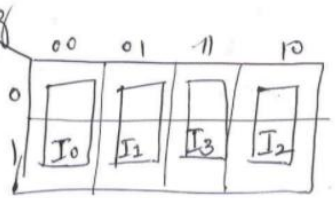
①



⇒ xy – select lines

z – data line

②

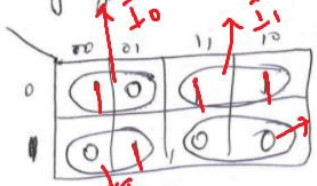


⇒ yz – select lines

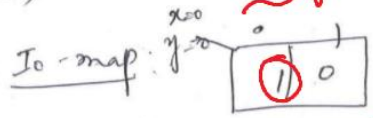
x – data line

Eg: First method:

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

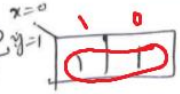


I_0 – map:



$= \bar{z}$ ✓

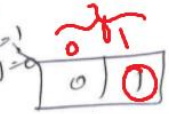
I_1 – map:



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

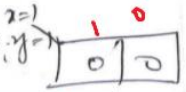
$= 1$ ✓

I_2 – map:



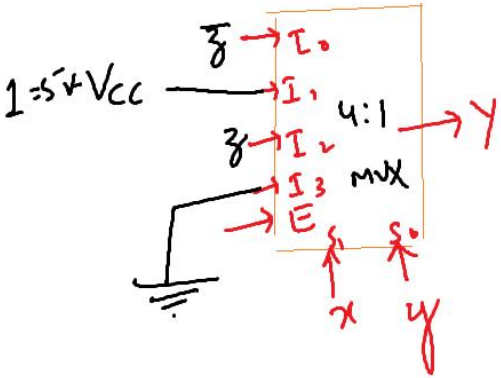
$= z$ ✓

I_3 – map:



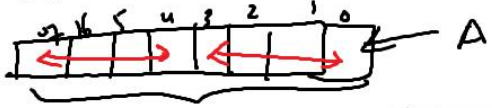
$= 0$ ✓

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



1 bit = 0 or 1

1 byte = 8-bits



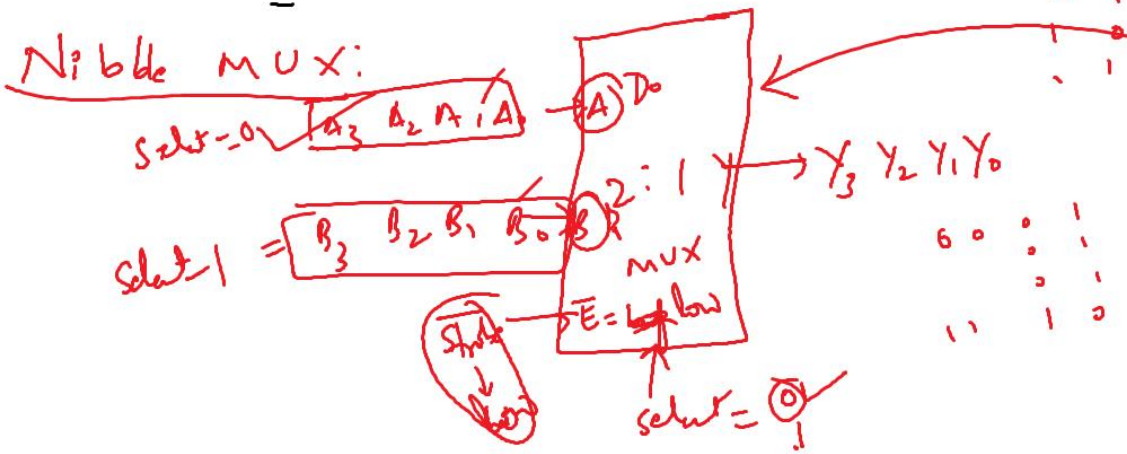
1 word = 2-bytes =

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

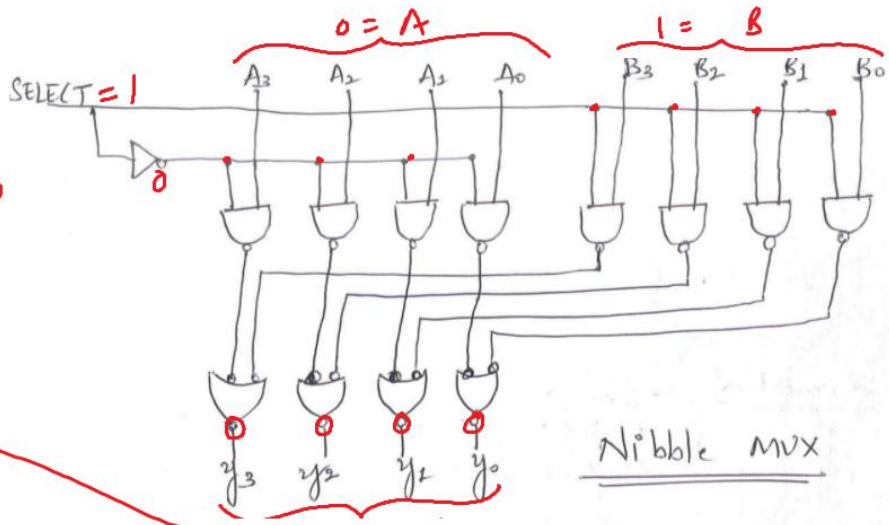
0 0

0 1

1 0

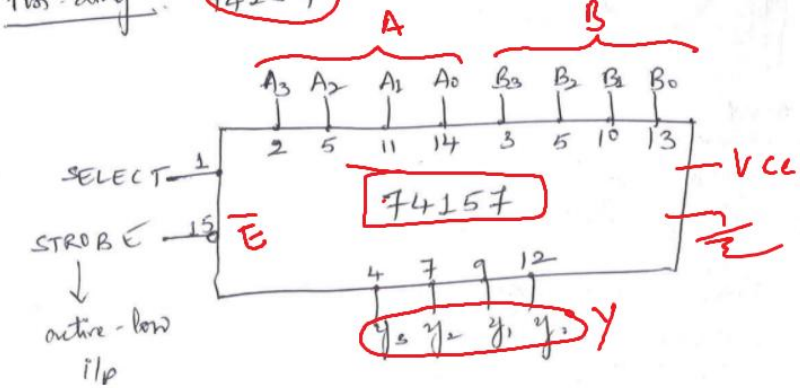
1 1

A < 0 / 1



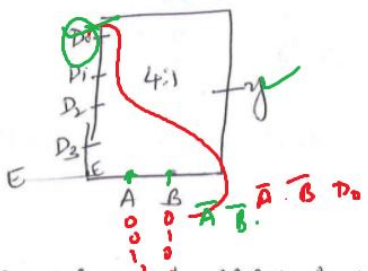
Nibble MUX

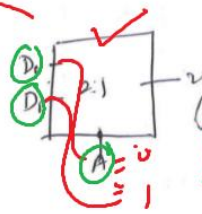
Pin-diagram: 74157



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

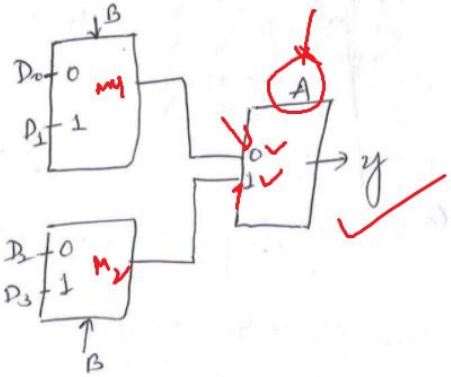
The logic eqn of 4:1 MUX can be re-written as

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

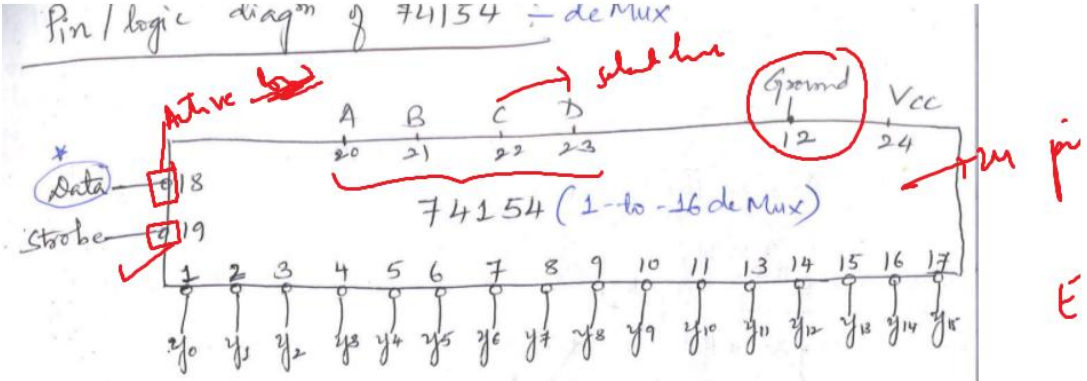
B_1 B_2

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;

Combinational ckts: Adders & Subtractors

MUX, deMUX

Half Adder: 2-bit Adder

2 binary i/ps: Augend and addend bits

2 binary o/ps: Sum and carry

x_i	y_i	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

3
11

0 0 1 1 2
0 1 0 1 10

Carry: x_i y_i

	0	1
0	0	0
1	0	1

$\therefore C = x_i \cdot y_i$

Sum: x_i y_i

	0	1
0	0	1
1	1	0

$\bar{x}_i y_i + x_i \bar{y}_i$

$\therefore$ Sum $= x_i \oplus y_i$

Full Adder:

Half Adder:

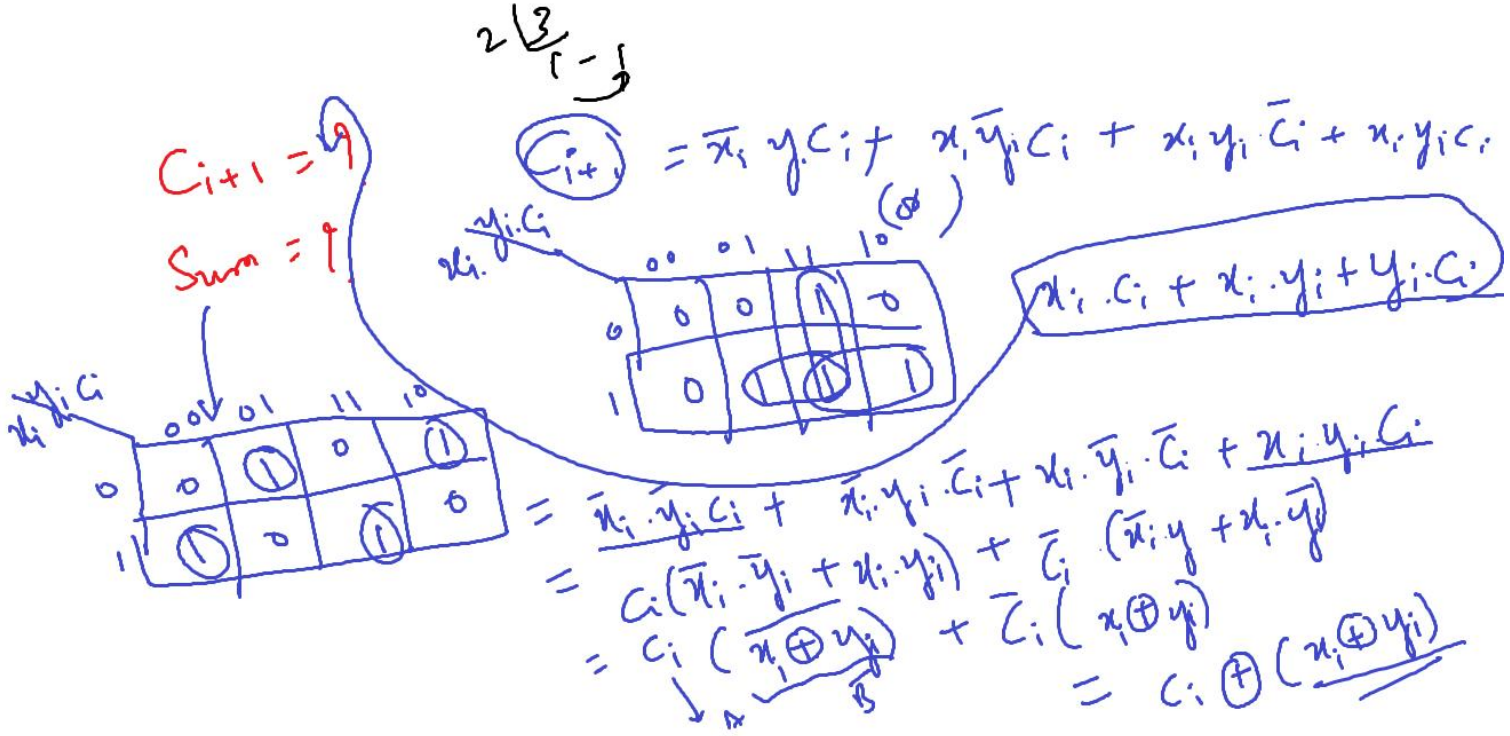
→ 3 i/ps — x_i . y_i C_i

2 - o/ps — Sum

Carry bit C_{i+1}

x_i	y_i	C_i	C_{i+1}	Sum
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

Logic diagram.



Logic diagram for half adder and full adder

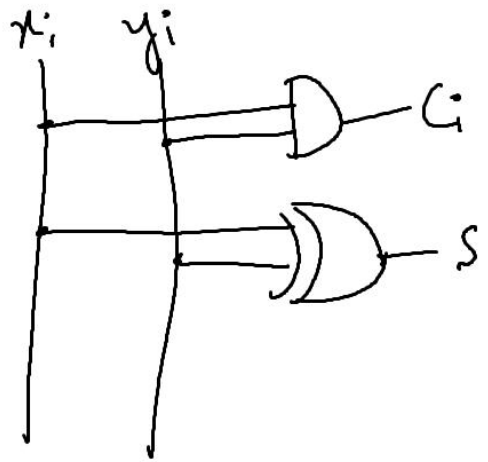
$C_i = x_i \cdot y_i$

$S = x_i \oplus y_i$

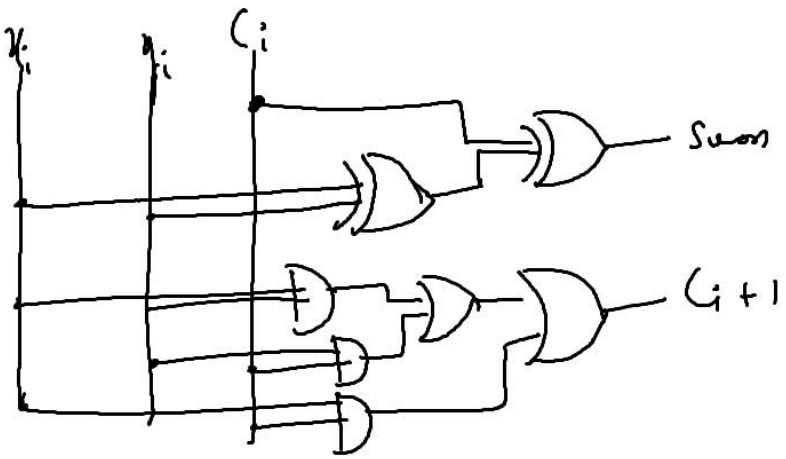
$Sum = C_i \oplus (x_i \oplus y_i)$

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

ilps:



ilps:



Subtractor CC

- HS — 2-bits — 2 bits
- FS — 3-bits — 2 bits

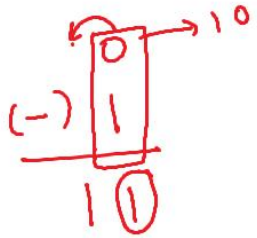
Binary Adder & Binary subtractor

Binary half subtractor : $(x_i - y_i)$

2 binary i/ps : Minuend — x_i
Subtrahend — y_i

2 binary o/ps : Difference — d
borrow — b

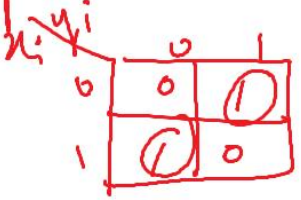
x_i	y_i	d	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



1
(-) 0
01

0
(-) 0
0

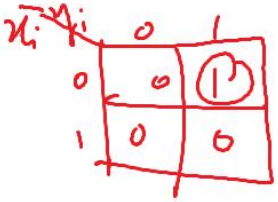
k-map : Diff



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

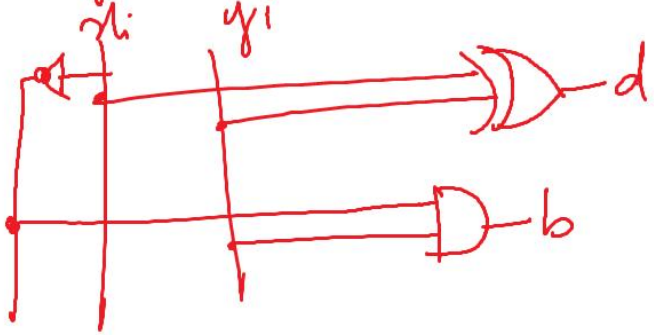
$= x_i \oplus y_i$

Borrow:



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

Logic diagram:

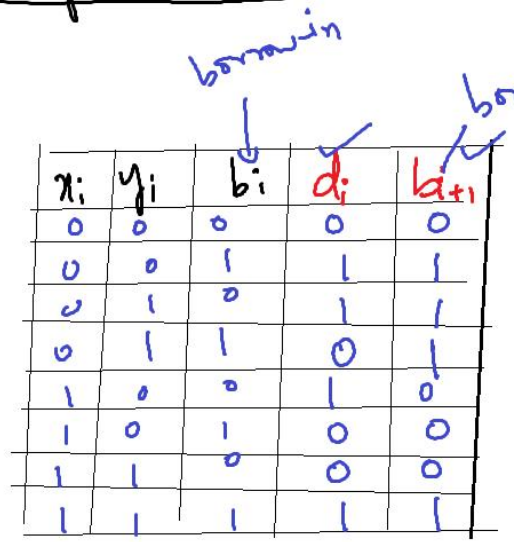


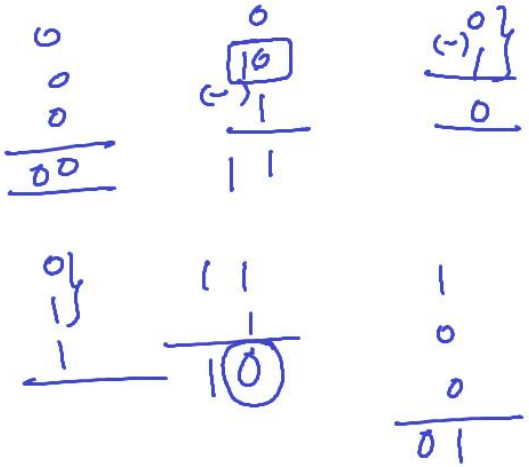
Binary Full Subtractor : 3 i/ps — 3 binary i/ps — minuend - x_i
2 o/ps — 2 binary o/ps. — Subtrahend - y_i
borrow-in from previous

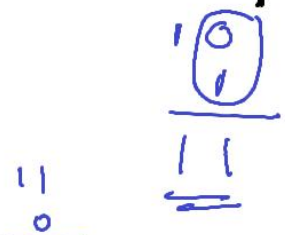
borrow-in

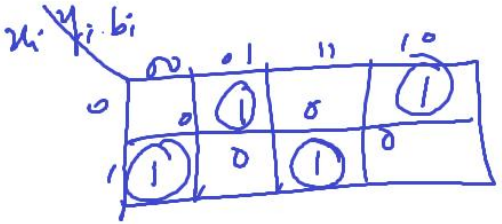
borrow-out

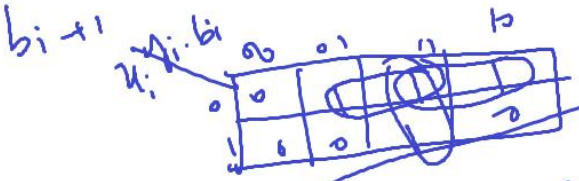
x_i	y_i	b_i	d_i	b_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1











$b_{i+1} = \bar{x}_i b_i + \bar{x}_i \cdot y_i + y_i \cdot b_i$

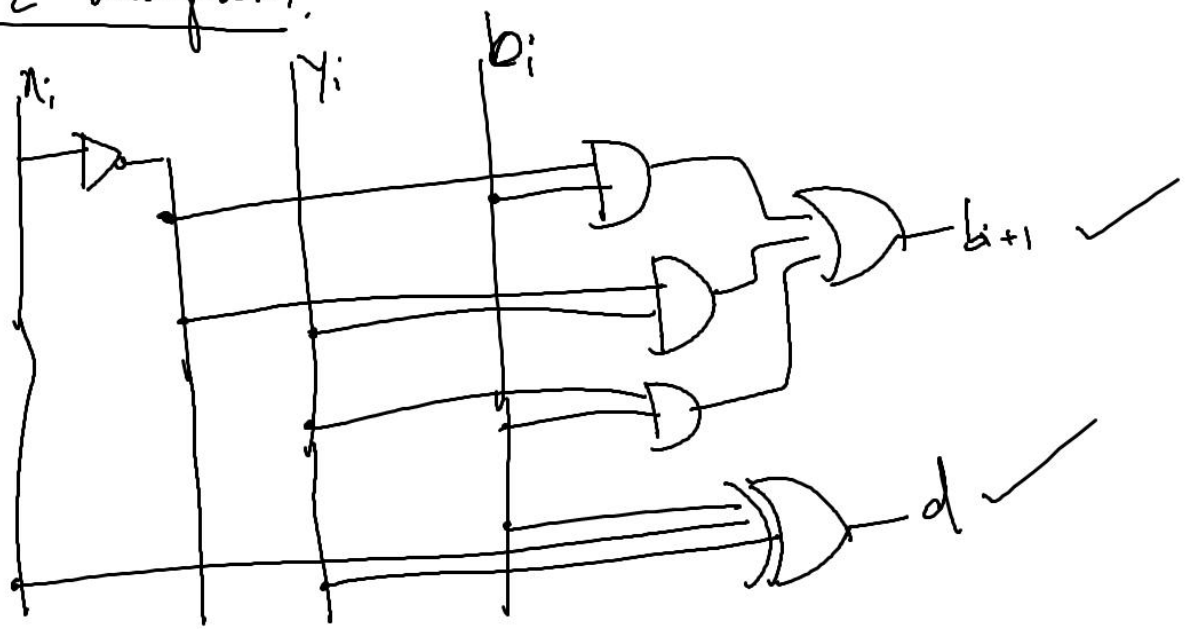
$d_i = \bar{x}_i \bar{y}_i b + \bar{x}_i \cdot y_i \bar{b}_i + x_i \cdot \bar{y}_i \cdot \bar{b}_i + x_i y_i b_i$

$= b_i(\bar{x}\bar{y} + xy) + \bar{b}_i(\bar{x}y + x\bar{y})$

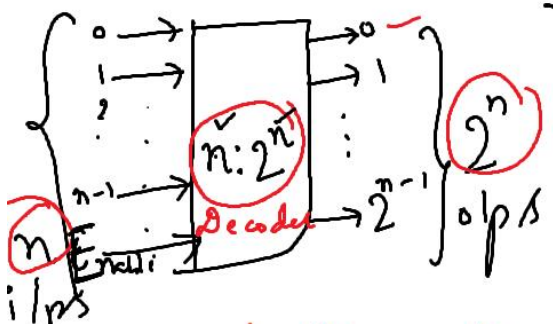
$= b_i(\overline{x \oplus y}) + \bar{b}_i(x \oplus y)$

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

Logic Diagram:



Decoder: Combinational L.C. which has multiple-i/ps and multiple-o/ps (logic ckt) which converts the coded i/p's into coded o/ps where i/p-o/ps codes are different.



General structure of the decoder

Binary decoder: $n - 2^n$ ($2^3 - 8$, 4) $\Rightarrow$

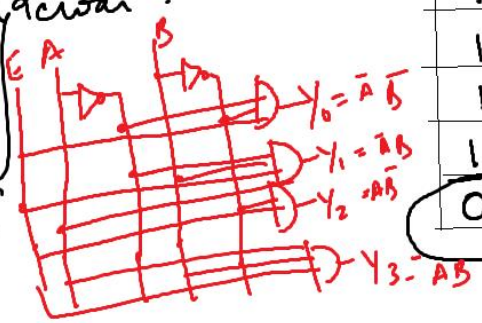
- 2 to 4
- 3 to 8
- 4 to 16
- 5 to 32
- ⋮
- n to 2^n

decoder.

n-bit binary i/p code

2 : 4 decoder

i/p are decoded



i/ps			o/ps				
E	A	B	Y_3	Y_2	Y_1	Y_0	
1	0	0	0	0	0	1	$= \bar{A}\bar{B}$
1	0	1	0	0	1	0	$= \bar{A}B$
1	1	0	0	0	0	0	$= A\bar{B}$
1	1	1	1	0	0	0	$= AB$
0	X	X	0	0	0	0	

n: 2^n decoder : 3 : 8 Decoder : 74138.

$3 : 2^3$

3 : 8

accepts 3 bin. i/ps

8 - individual active LOW o/ps

$\overline{E} = 0$, active low enable

$m = \overline{M}$ (or) $M = \overline{m}$

$\Sigma m(0,4,6)$

ABC

$\overline{m_0} = \overline{A}\cdot\overline{B}\cdot\overline{C}$ → $M_0 = \overline{\overline{A}\cdot\overline{B}\cdot\overline{C}}$

0 0 0

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

$M_0 = A + B + C$

5 1 1

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

$\Rightarrow \overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} = (\overline{A} + B + C) = M_4$

$f_1(A,B,C) = \Sigma m(0,4,6)$ D

$f_2(A,B,C) = \Sigma m(0,5)$ D

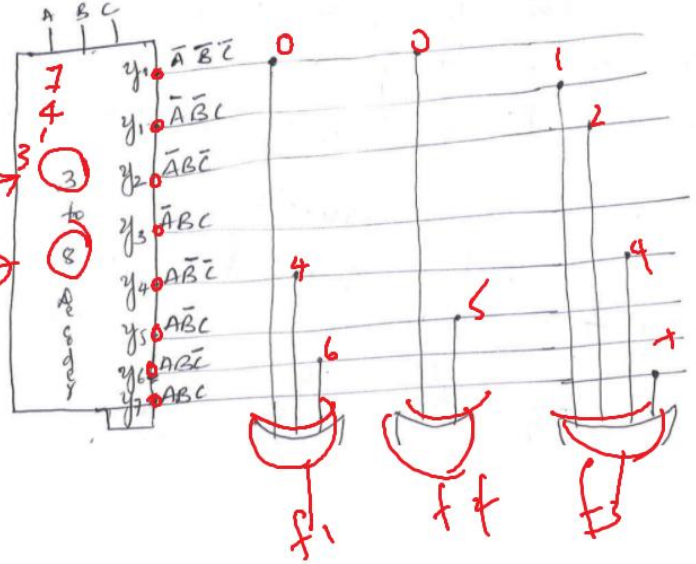
$f_3(A,B,C) = \Sigma m(1,2,3,7)$ D

Since @ the decoder o/p, we'll get all the r terms, we use them as shown to get boolean f^n s.

C.V = 1

U.V = 0

$M_0 \cdot M_4 \cdot M_6$



BCD to Decimal decoder (nibble - 4 bits)

Binary Coded Decimal

Decimal no 429 ⇒ BCD: 4 2 9

0100 0010 1001

BCD : 1000 0101 0010 0111 ⇒ Decimal

8 5 2 7 ← = 8527(10)

Convert the decimal no 563 into BCD. (each digit – 4bits)

0101 0110 0011

1) BCD

0101 0111 0101 = 575(10)

MSB LSB

Decimal

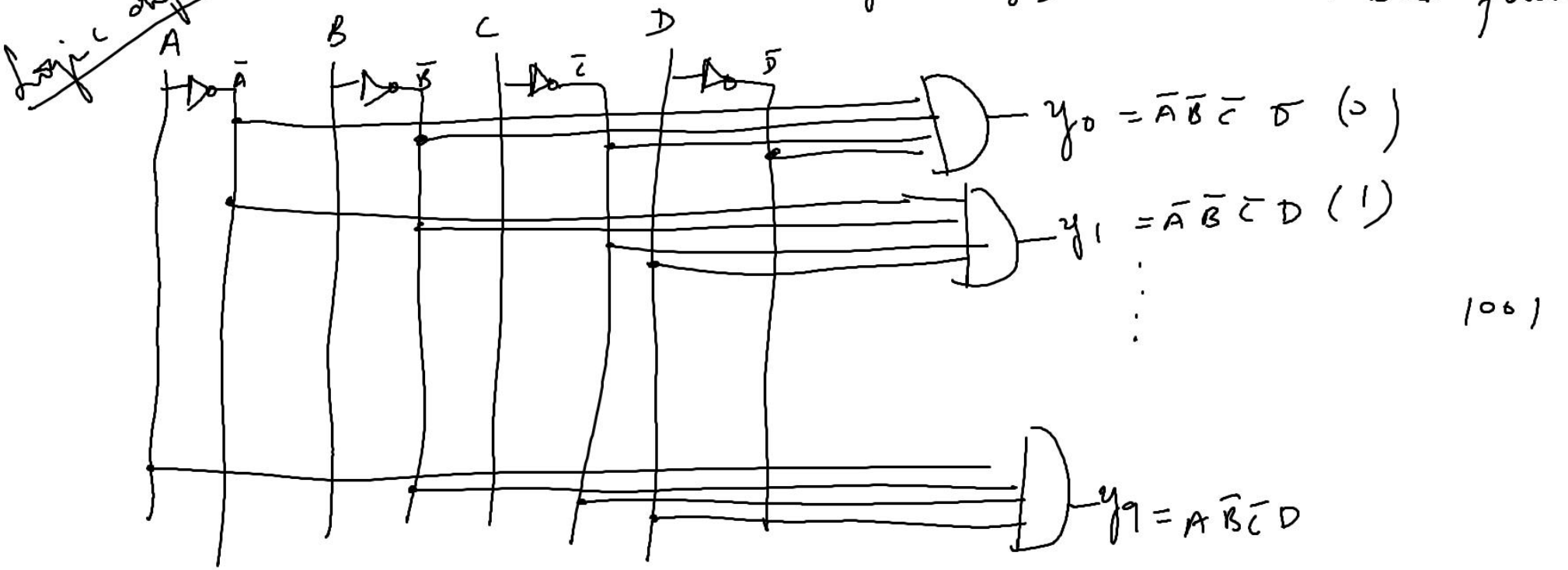
Group 4-bits together

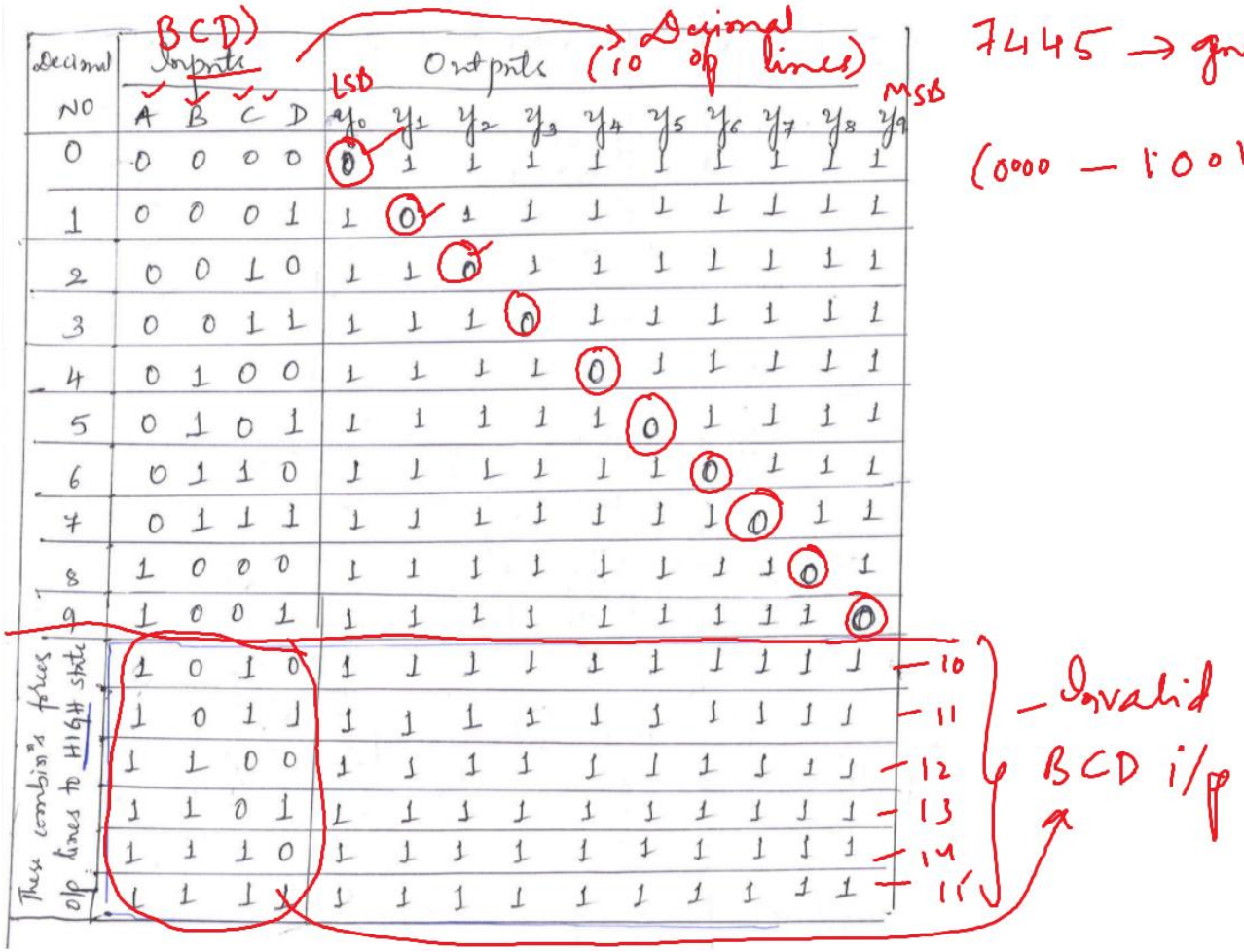
BCD to Decimal Decoder (7445)

$2^4 = 16 \rightarrow \underbrace{0 \ldots 9}_{10 \ldots 15}$ — can NOT exist in BCD form.

↳ 0000 — 1001 (0 through 9) $(y_0 \ldots y_9)$

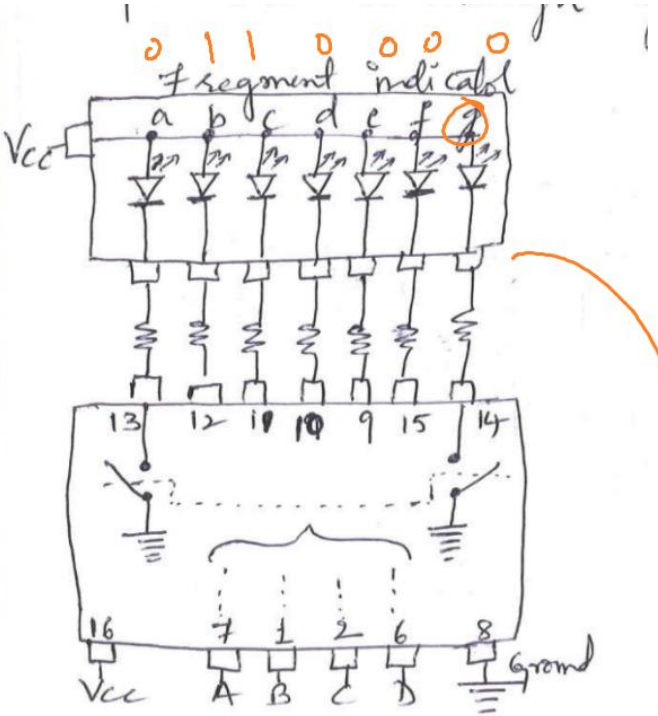
Logic diagram

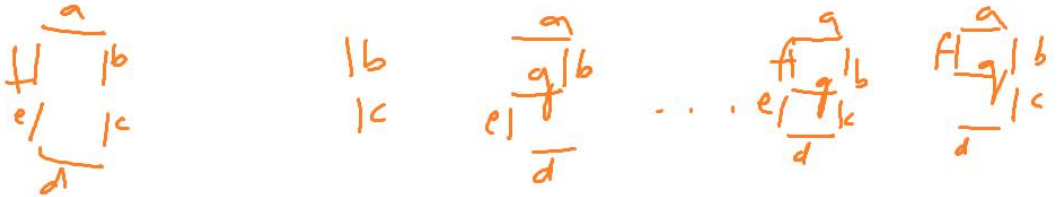




7445 → generates active LOW o/ps

(0000 – 1001)





1 0 0 1 1 1 1

0 0 0 0 → Decimal no in BCD

0 0 0 1

0 0 1 0

1 0 0 0

1 0 0 1

Logic design using decoders: $(n : 2^n)$

minterms — product terms

Boolean fn — sum of minterms — SOP

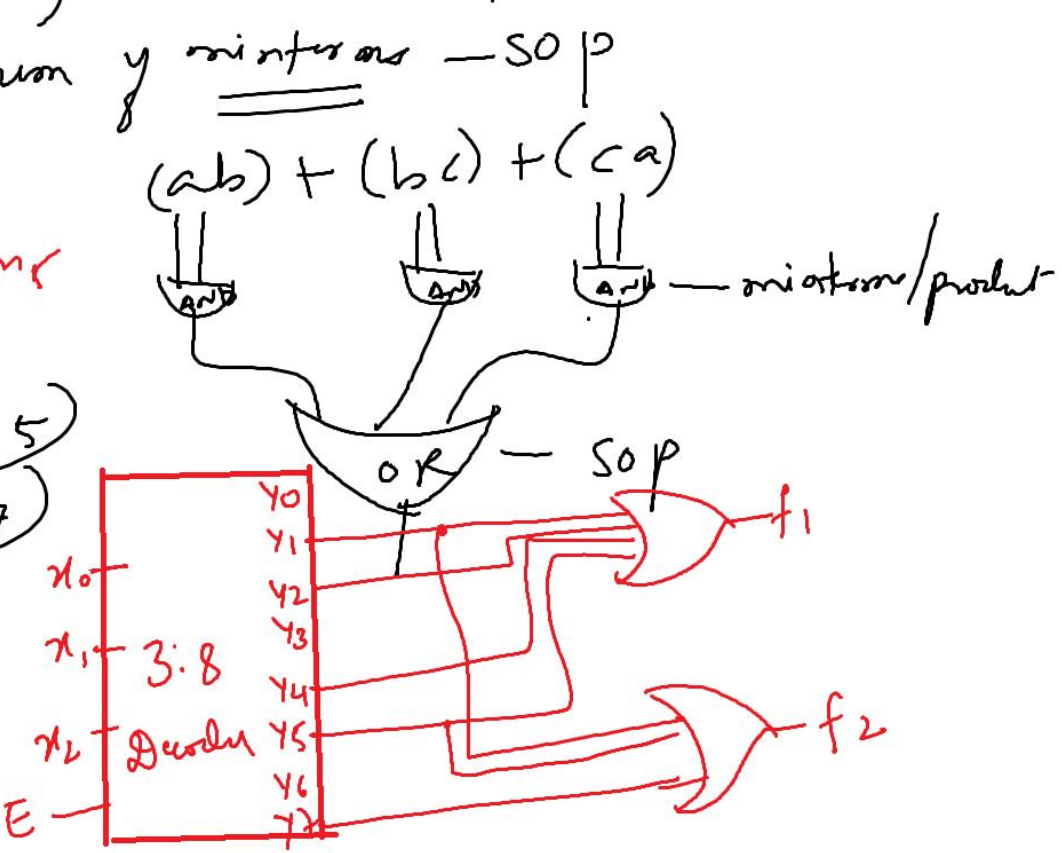
74138 — LOW o/p's

SOP = $m_1 + m_2 + m_4 + m_5$

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

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

Realize decoder circuit. (Default: Active HIGH o/p's)



Realize the decoder ckt for given 2 f^n's

$f_1(c,b,a) = \sum m(0,1,3,4,5,6)$

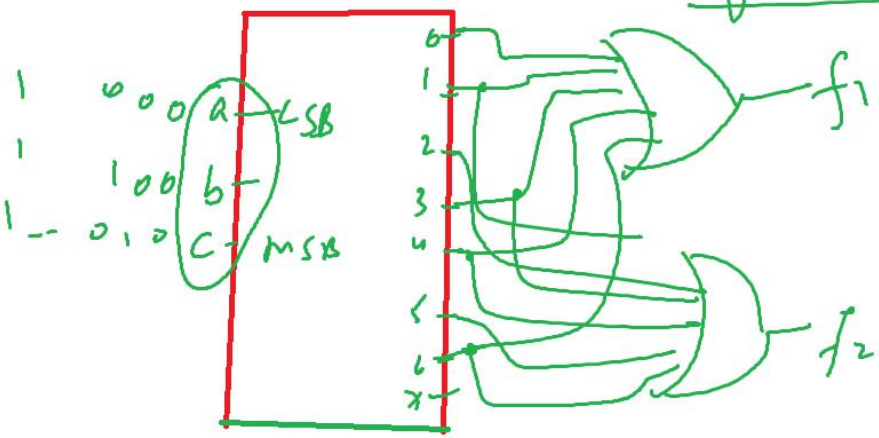
$f_2(c,b,a) = \sum m(1,2,3,4,6)$

MSB LSB

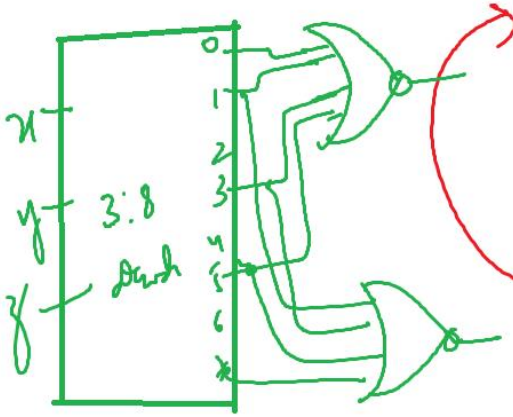
Default o/p: HIGH o/p

$f_1(z,y,x) = \pi M(0,1,3,5)$

$f_2(z,y,x) = \pi M(1,3,6,7)$



$m = \overline{M}$



$m = \overline{(m_1 + m_2 + m_3)}$

$\overline{m_1 + m_2 + m_3}$

$(\overline{m_1}) \cdot (\overline{m_2}) \cdot (m_3)$

$M_1 \cdot M_2 \cdot M_3$ } POS

max max max

Active HIGH o/ps ← minterm (SOP) — OR
maxterm (POS) — NOR

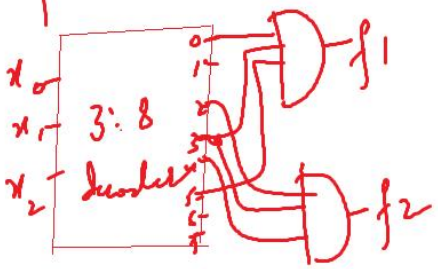
Active LOW o/ps (74138) ← minterm (SOP) — NAND
maxterm (POS) — AND

n : 2^n → NAND gates

Realize decoder (LOW o/ps) fn given 2 fns:

1) $f_1(x_2, x_1, x_0) = \pi M(0,3,5)$

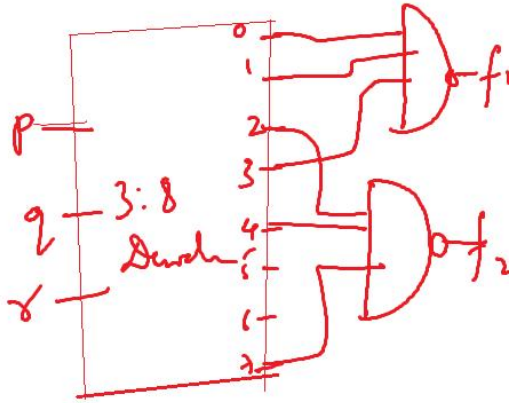
$f_2(x_2, x_1, x_0) = \pi M(2,3,4)$



2) Realize active low o/p decoder fn given 2 fns

$f_1(r,q,p) = \Sigma m(0,1,3)$

$f_2(r,q,p) = \Sigma m(2,4,7)$



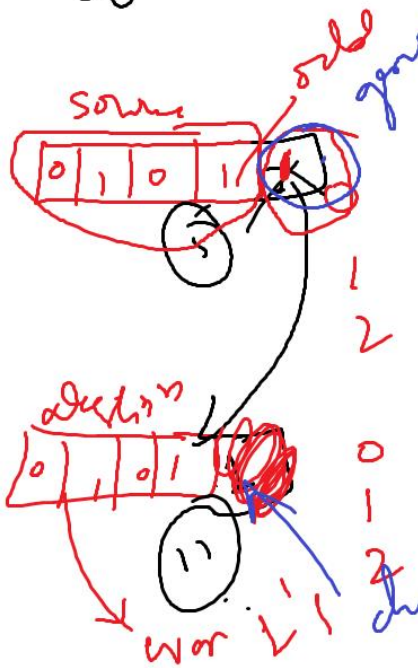
Encoders

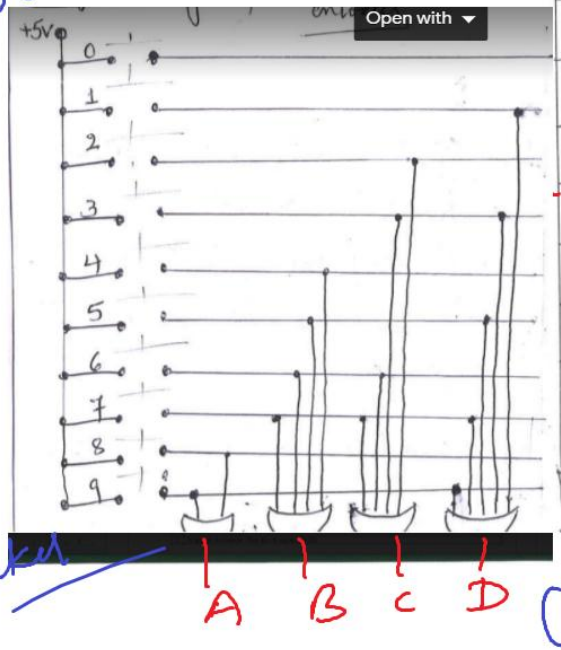
$2^n : n$

Parity

Decoder

$n : 2^n$





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
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				
7	1	1	1	1	1	1	1	0	1	1				
8	1	1	1	1	1	1	1	1	0	1				
9	1	1	1	1	1	1	1	1	1	0				

10 - Inputs

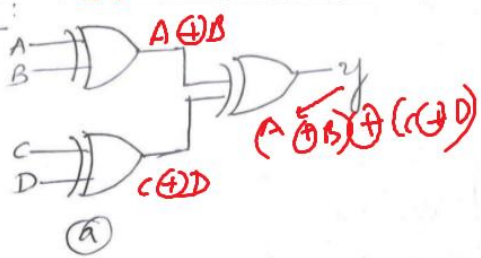
Outputs 4

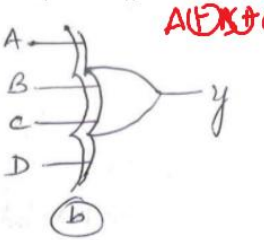
High o/p

Low o/p

Four Inputs

0 0 1 1 | 1





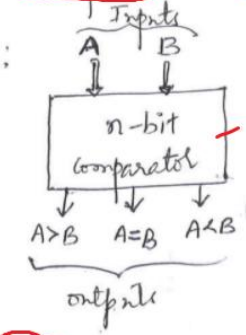
Four i/ps X-OR gate

Magnitude comparator

Open with

Def^n: A comparator is a special combinational circuit designed primarily to compare the relative magnitude of 2-binary nos.

Block diag^m:



It receives two n-bit nos A and B as inputs and the outputs are $A>B$, $A=B$ and $A<B$. Depending upon the relative magnitude of the 2 nos, one of the o/ps will be 'high'.

⇒ (1) Design 1-bit comparator using basic gates.

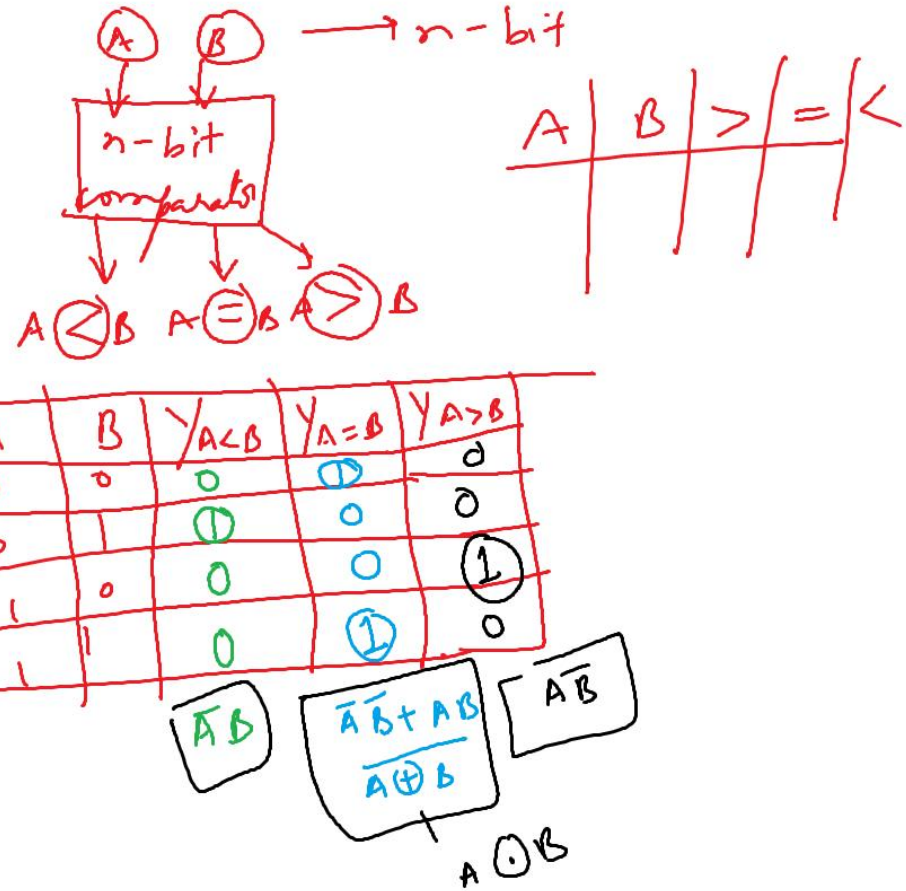
Sol^n: Consider two 1-bit nos A and B.

Inputs		Outputs		
A	B	$Y_{A>B}$	$Y_{A=B}$	$Y_{A<B}$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

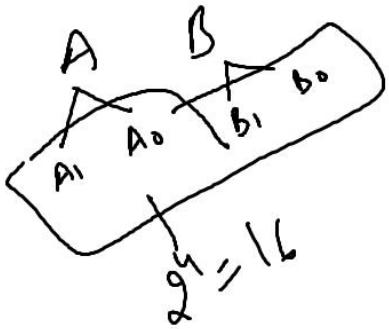
$Y_{A>B}$:- k-map

A \ B	0	1
0	0	0
1	(1)	0

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



2-bit comparator:



Inputs				Outputs		
A		B		<	>	=
A_1	A_0	B_1	B_0	$A<B$	$A>B$	$A=B$
0	0	0	0	0	0	1
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	0	0	1
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	0	0	1

$A = B$

$\Downarrow$

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

Programmable array logic / logic array: (PAL / PLA)

Programmable logic devices (PLDs): ICs

h/w programming

ARRAY of AND & OR gates

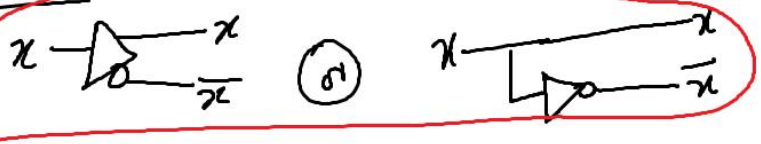
LOGIC Devices

Combinational Sequential

NOT programming

General structure:

Buffer / Inverter / NOT gate:



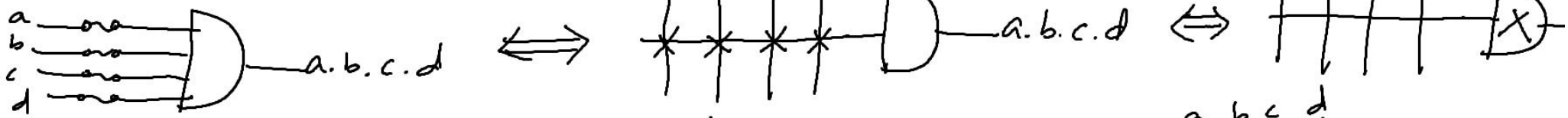
3 types of PLD:

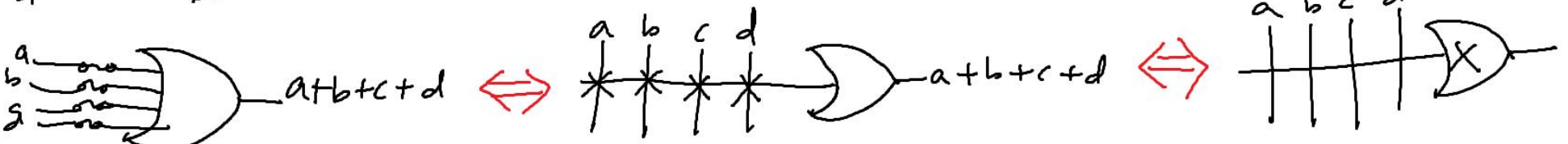
	AND-array	OR-array
1) PROM	Fixed	Programmable
2) PLA	Programmable	Programmable
3) PAL	Programmable	Fixed

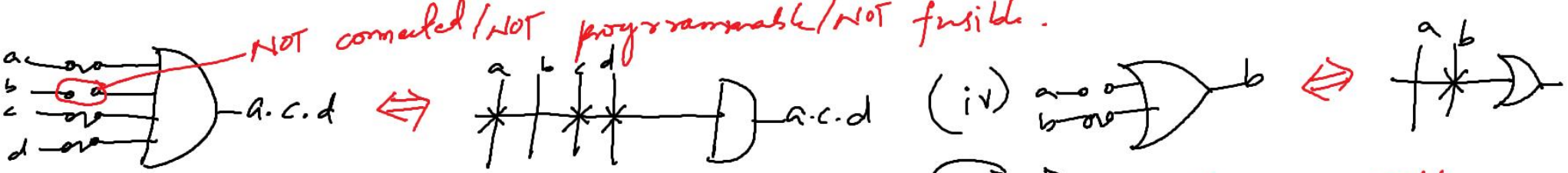
In PLDs, connection to each gate can be modified
i.e., (connect each i/p to fuse)

(x – fusable/programmable)

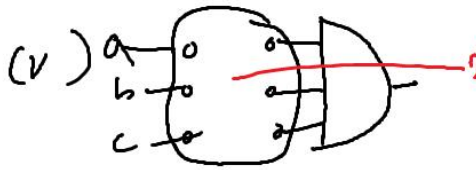
Notations:

i) 

ii) 

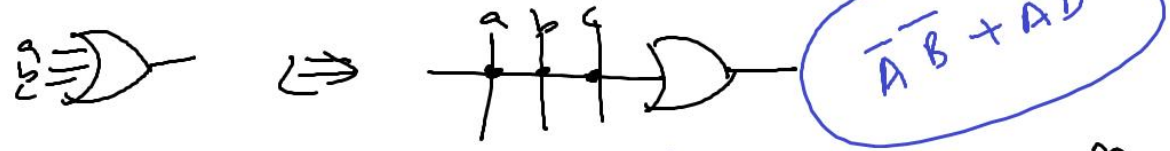
iii) 

(iv)

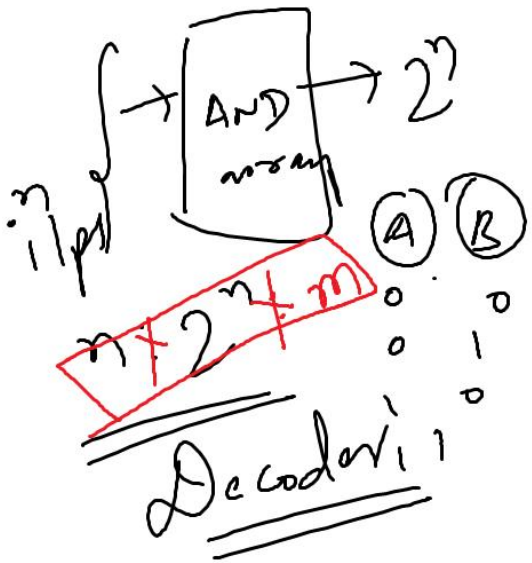
(v) 

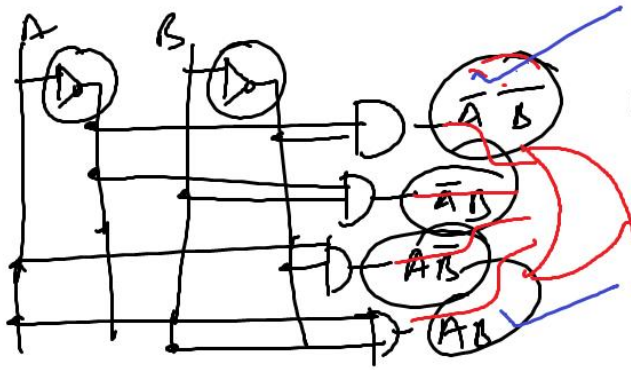
Imp ⚠ : 1) cross @ intersection ⇒ fusible link which is intact
2) lack of cross ⇒ fuse/connection is blown (or) No connection exist
3) Junction dot ⇒ Not fusible

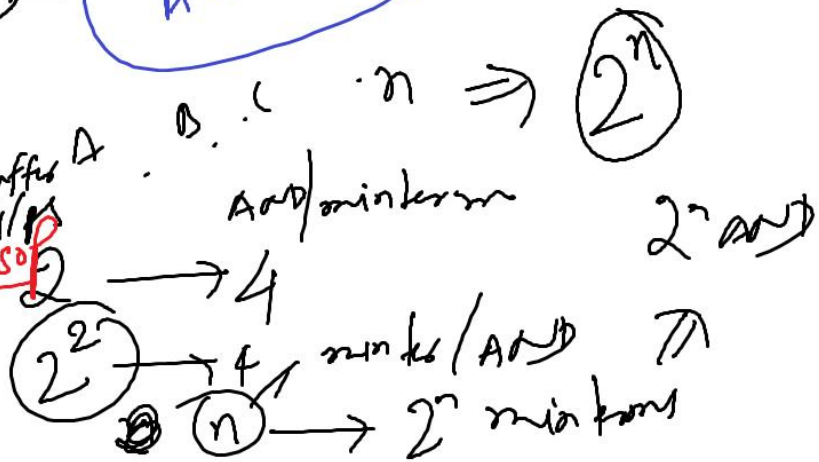
Non/not fusible inputs:



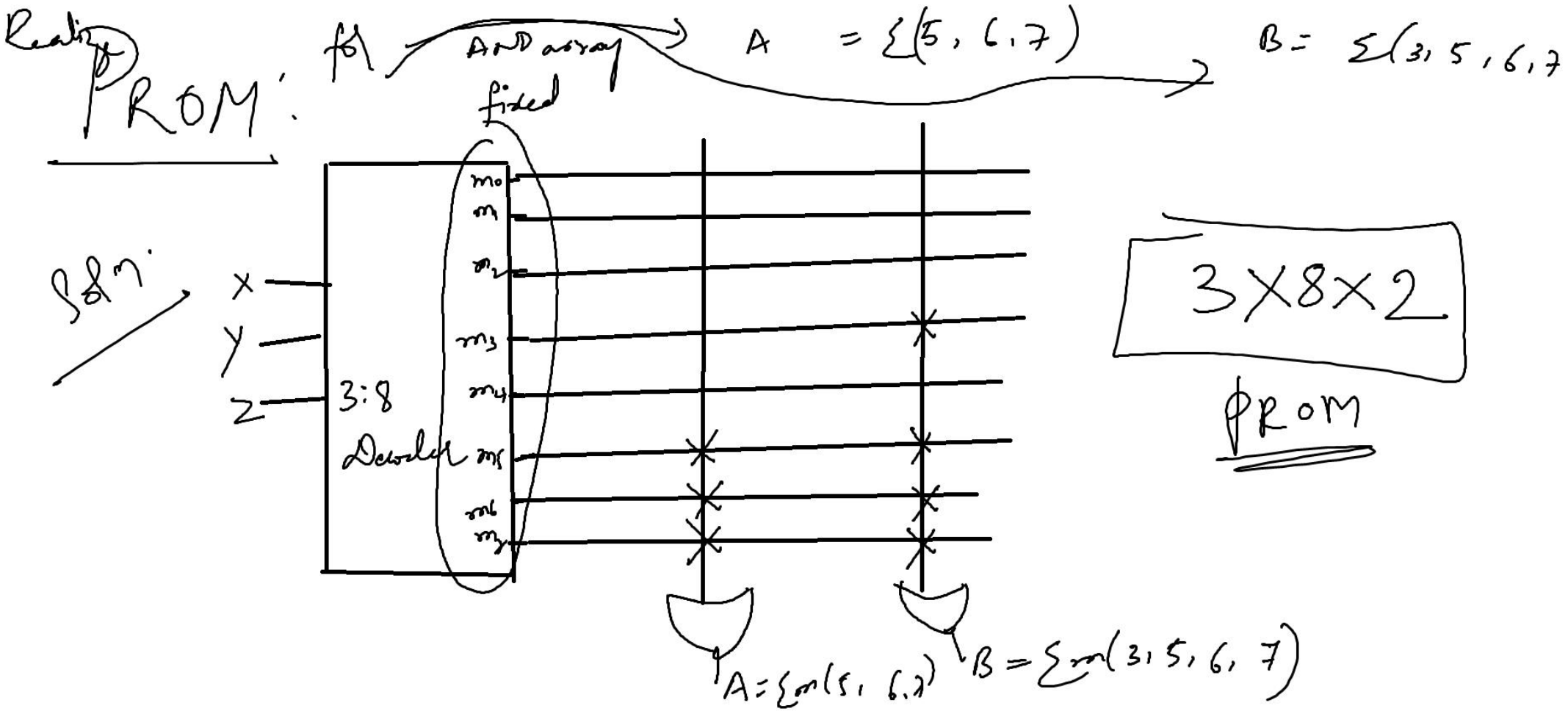
X - programmable
• - non fusible non-programable







Read only PROM:

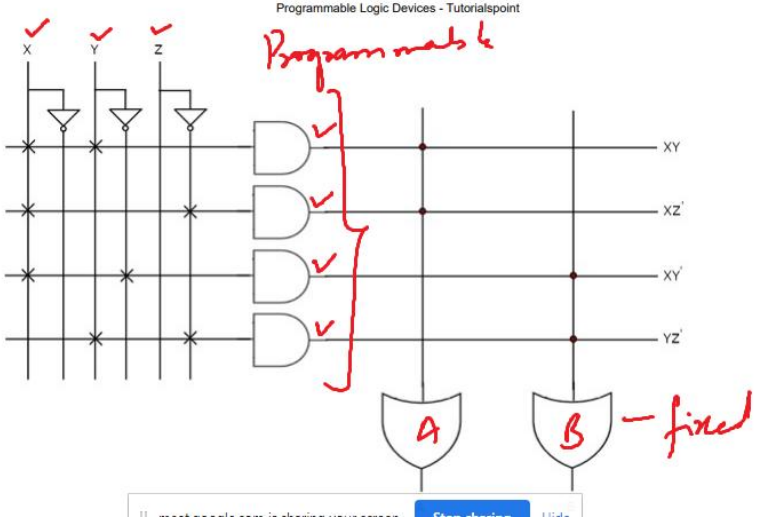


Let us implement the following **Boolean functions** using PAL.

$$A = XY + XZ'$$

$$B = XY' + YZ'$$

The given two functions are in sum of products form. There are two product terms present in each Boolean function. So, we require four programmable AND gates & two fixed OR gates for producing those two functions. The corresponding **PAL** is shown in the following figure.



PAL :

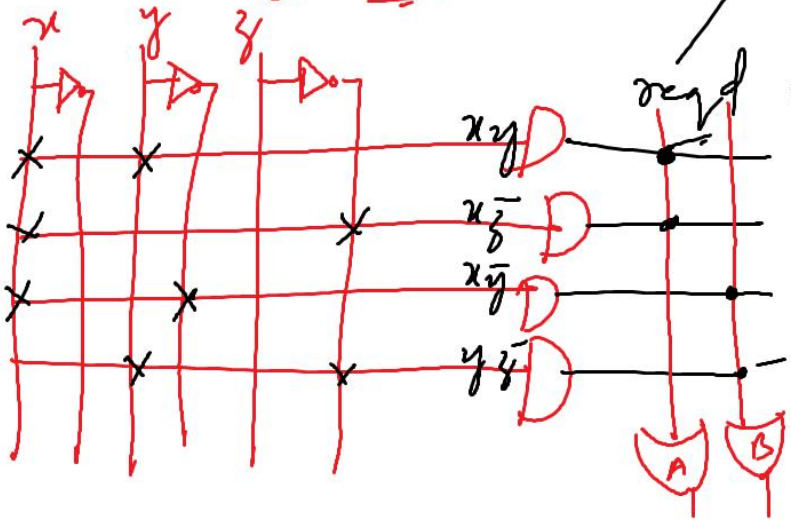
$A = xy + x\bar{z}$ → 2 minterms

$B = x\bar{y} + y\bar{z}$ → 2 minterms

xyz — 3 variable — 2^3 — 8 combination

For 3 - variable

8 - minterms/product terms



required minterms are programmable.

Example

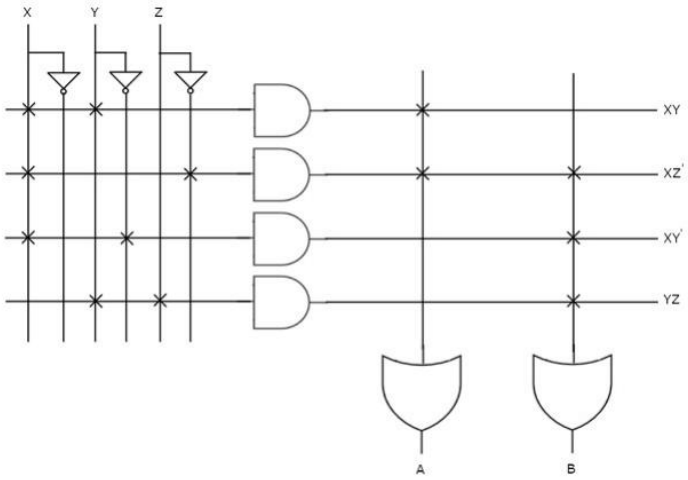
Let us implement the following **Boolean functions** using PLA.

$$A = XY + XZ'$$

$$B = XY' + YZ + XZ'$$

The given two functions are in sum of products form. The number of product terms present in the given Boolean functions A & B are two and three respectively. One product term, $Z'X$ is common in each function.

So, we require four programmable AND gates & two programmable OR gates for producing those two functions. The corresponding **PLA** is shown in the following figure.

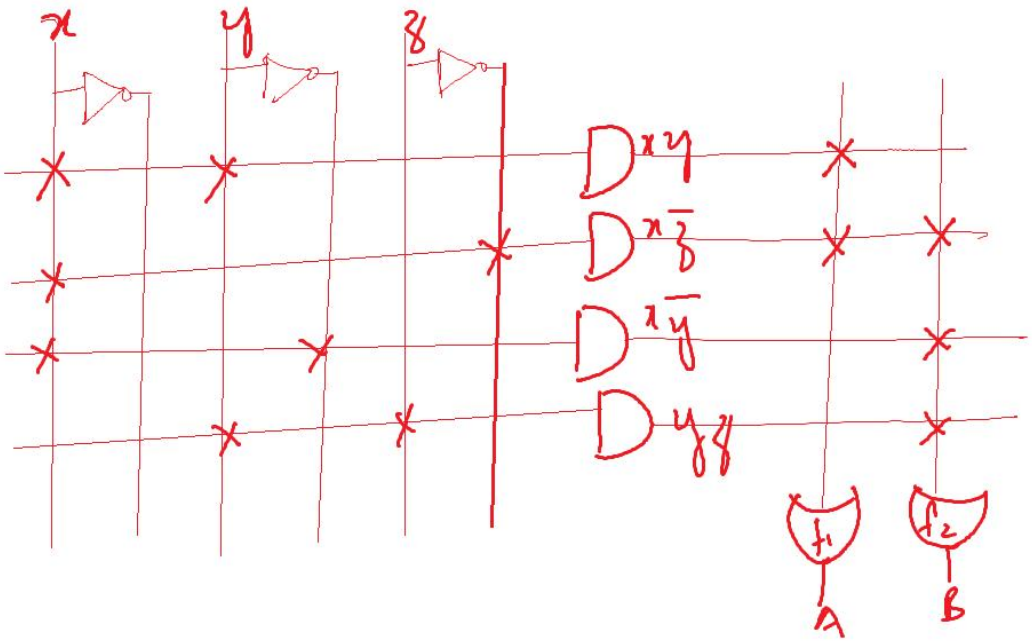


PLA : Both AND arrays and OR-arrays are programmable

3 vari

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

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



1) Realize the PROM fct given 2 f^ns

$f_1(A,B,C) = \sum m(0,1,2,5,7)$

$f_2(A,B,C) = \sum m(0,1,2,4,6)$

2) Realize the PLA fct given 2 f^ns

(i) D

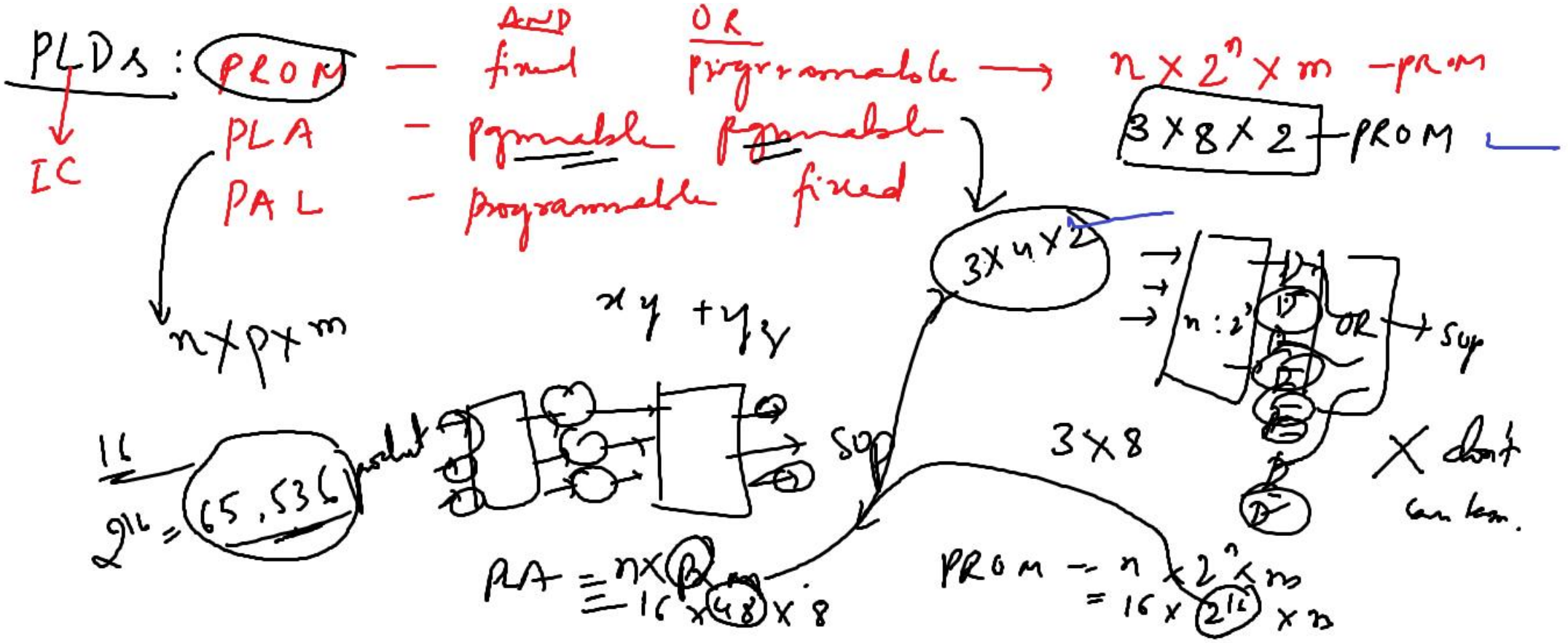
Ex: $f_1(x,y,z) = \sum m(0,1,3,4)$

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

3) Realize the PAL fct given 2 f^ns

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

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

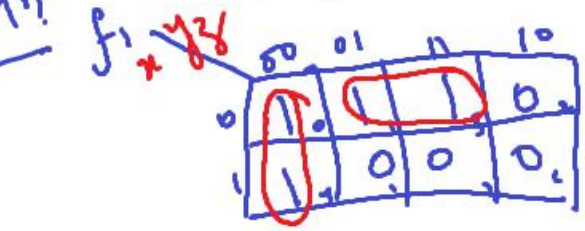

PLDs :
PROM — fixed — programmable → n × 2^n × m – prom
AND
OR
IC
PLA – programmable programmable
PAL – programmable fixed
3 × 8 × 2 PROM
n × p × m
xy + yz
3 × 4 × 2
n : 2^n
OR
Sop
IC
65,536
product
3 × 8
PLA ≡ n × p × m
≡ 16 × 48 × 8
PROM = n × 2^n × m
= 16 × 2^16 × m
× don't care term.

Realize PLA using 2 fns

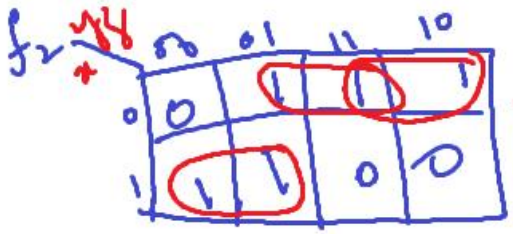
i) $f_1(x,y,z) = \Sigma m(0,1,3,4)$

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

Soln

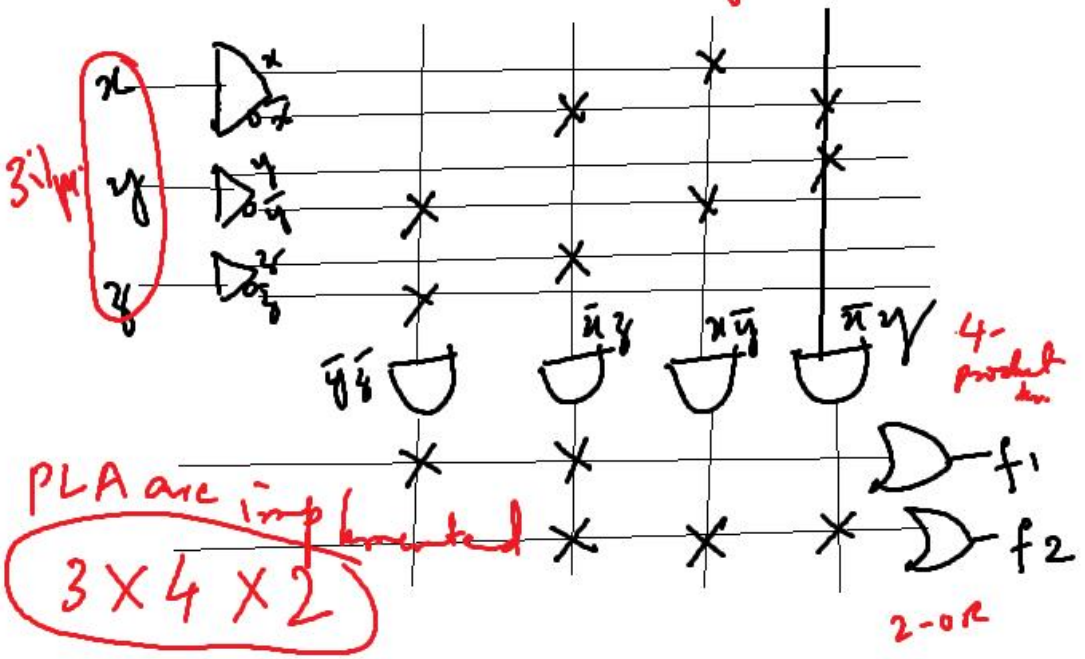


$= \bar{y}\bar{z} + \bar{x}z$ ✓



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

— both AND & OR arrays are Programmable



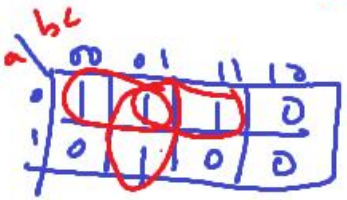
PLA are implemented

3 × 4 × 2

② Realize/Implement PLA (3×4×2) for the following boolean fn.

$f_1(a,b,c) = \sum m(0,1,3,5)$ $\quad f_2(a,b,c) = \sum m(3,5,7)$

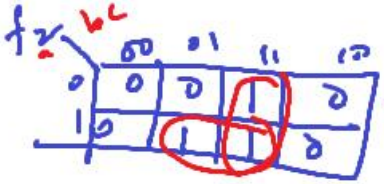
Soln : f_1



$= \bar{a}\bar{b} + \bar{a}c + \bar{b}c$

No common product

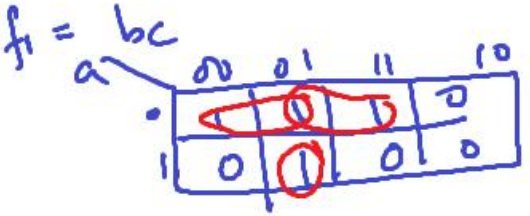
f_2



$= ac + bc$

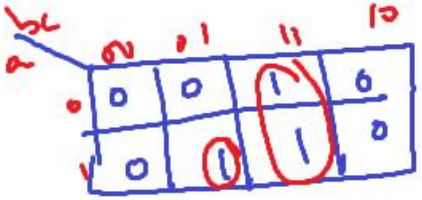
$3 \times (5) \times 2$ → 4

$f_1 = bc$



$= \bar{a}\bar{b} + \bar{a}c + a\bar{b}c$

$f_2 = bc$



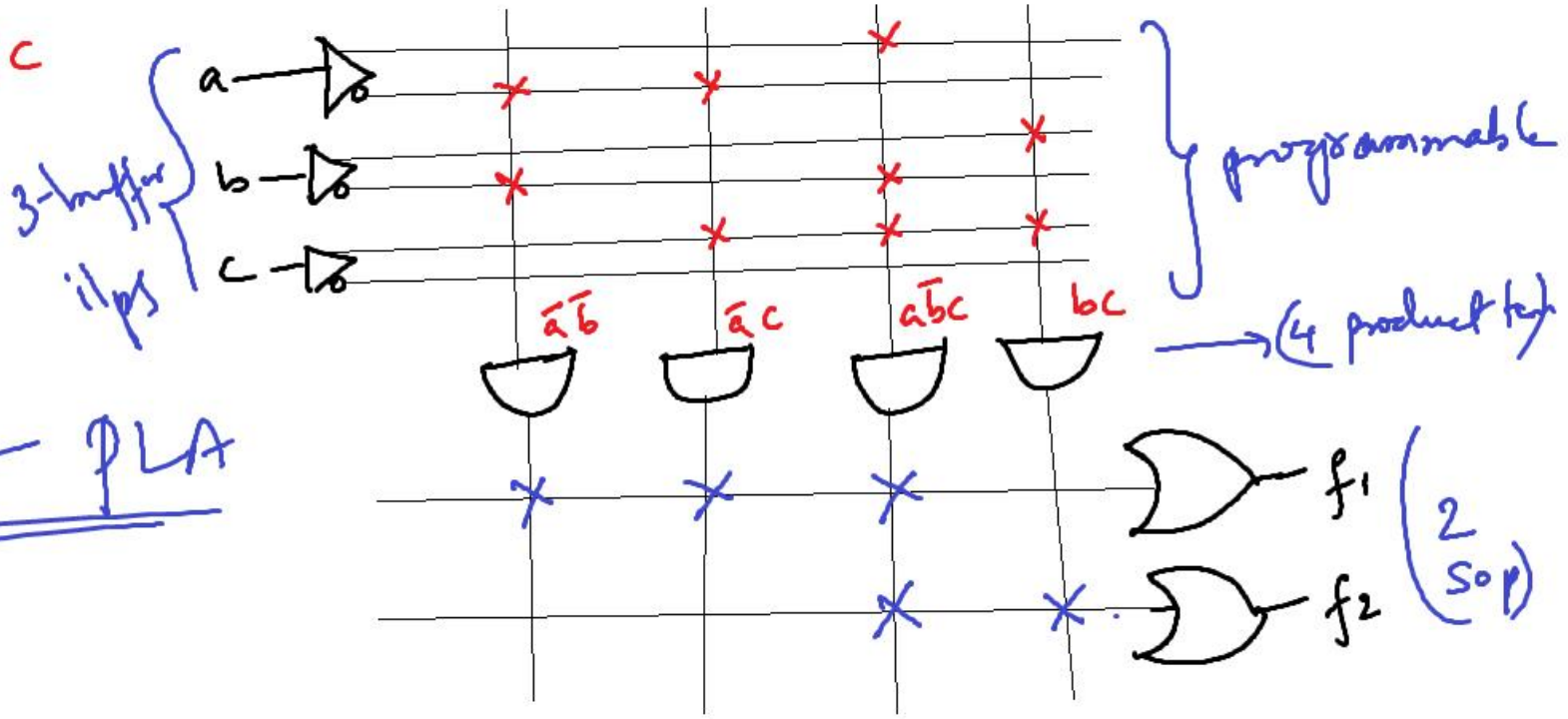
$= bc + a\bar{b}c$

$\therefore f_1 = \bar{a}\bar{b} + \bar{a}c + a\bar{b}c$

$f_2 = bc + a\bar{b}c$

$f_1 = \bar{a}\bar{b} + \bar{a}c + a\bar{b}c$

$f_2 = bc + a\bar{b}c$



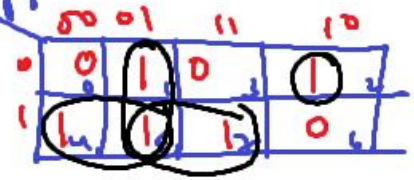
3 × 4 × 2 — PLA

PAL: Programmable Array Logic (AND-array is programmable, OR-array is fixed)

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

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

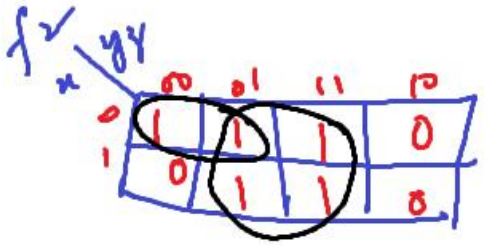
Soln: f_1



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

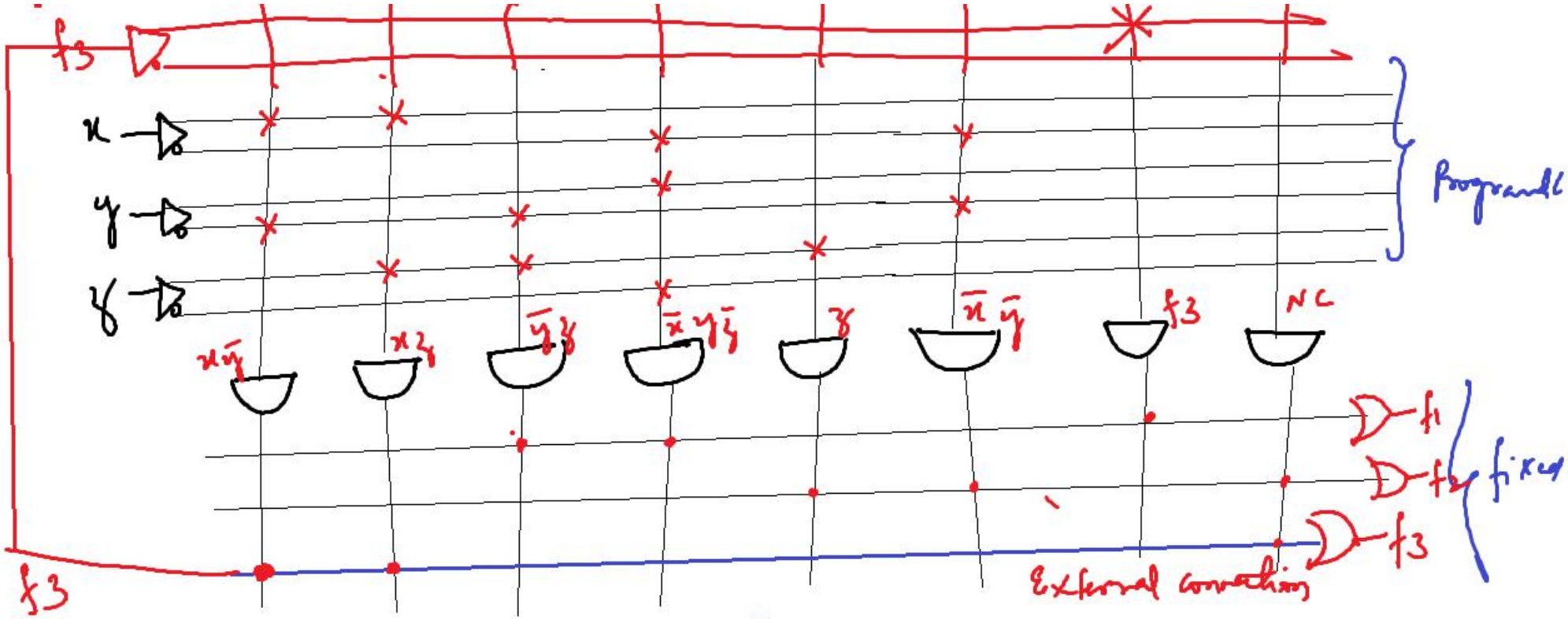
4 product terms

f_2



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

2 product terms



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

(max 4 product terms only 3 terms are realizable in PAL)

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

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

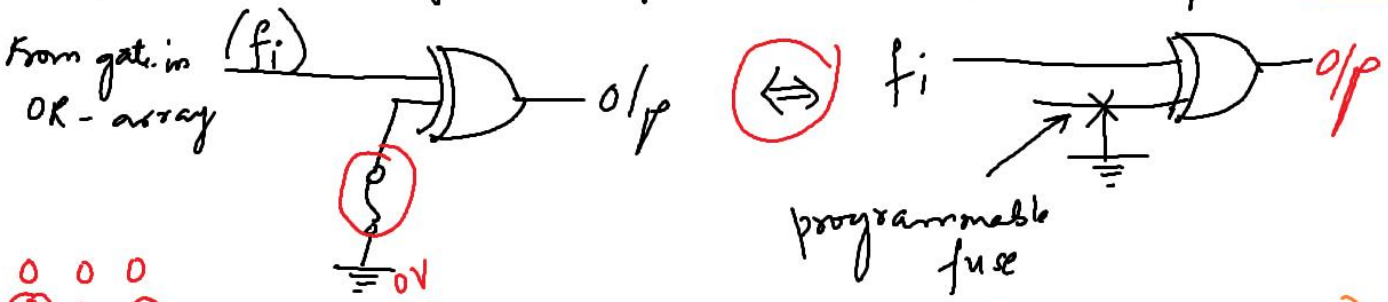
$f_2 = z + \bar{x}\bar{y}$ + NC

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

The inform? stored in the mem-elements @ any given time defines the present state of the Seq ckt. The present state and external i/ps determine the outputs and the next state of the Seq ckt. Thus we can specify the Seq ckt by the TIME SEQUENCE of external i/ps, internal states (present & next states), & o/ps. Eg: Counters & Registers are common examples of Seq ckt.

Comparison:	Combinational ckt	Sequential ckt
O/ps of vars	- all times depend on the combination of i/p vars	The o/p vars depend NOT only on the present i/p vars but also depend upon the past history of these vars
Memory Unit	- NOT req'd	Req'd to store the past history of i/p vars
Speed	- Faster, ∵ delay b/w i/p & o/p is due to propogation delay of gates	Slower
Design	- Easy to design	Comparatively harder to design

For greater flexibility, PLA provide true o/p (or) complemented o/p. USE EX-OR gates



Fuse intact = ground = logic 0

$f_i \oplus 0 = f_i$ (same as i/p)

Fuse blown $\Rightarrow$ +V = logic 1

$f_i \oplus 1 = \overline{f_i}$ (complemented o/p)

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

Implement PLA for the following functions:

$f_1(x, y, z) = \Sigma m(1, 2, 3, 7)$

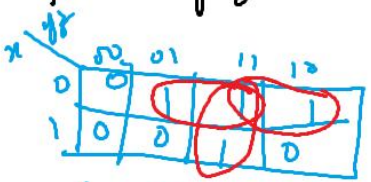
$f_2(x, y, z) = \Sigma m(0, 1, 2, 6)$

$\overline{f_1} = \Sigma m(0, 4, 5, 6)$

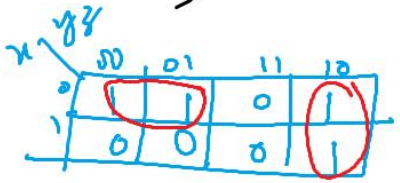
$\overline{f_2} = \Sigma m(3, 4, 5, 7)$

Complemented o/p: $\overline{f_1}$ and $\overline{f_2}$ (since NO common product terms)

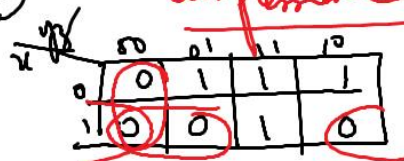
Sol^n:



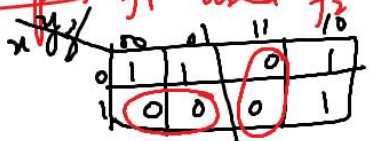
$f_1 = \bar{x}y + \bar{x}y + yz$



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



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



$\overline{f_2} = yz + x\bar{y}$

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

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

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

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

Compare: f_1 → f_2 ✗, $\bar{f_1}$, $\bar{f_2}$

f_2 → f_1 ✗, $\bar{f_1}$, $\bar{f_2}$

Case i : f_1 and $\bar{f_2}$ Ex-OR gate:

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

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

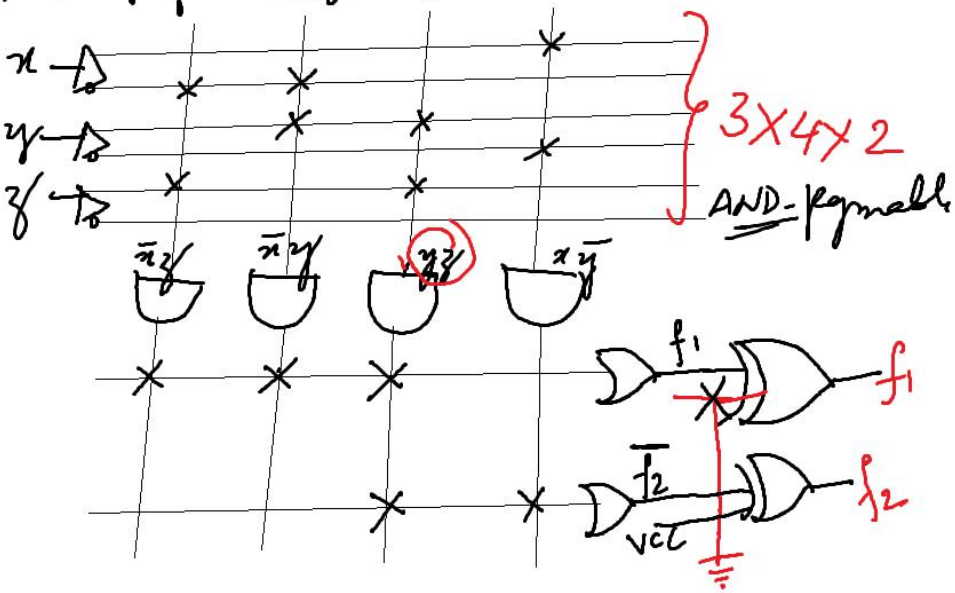
$f_i \oplus 0 = f_i \Rightarrow f_1$

$f_i \oplus 1 = \bar{f_i} \Rightarrow f_1$

$\bar{f_i} \oplus 0 \text{ (GND)} = \bar{f_i}$

$\bar{f_i} \oplus 1 \text{ (VCC)} = \overline{\bar{f_i}} = f_i$

+5V = logic 1

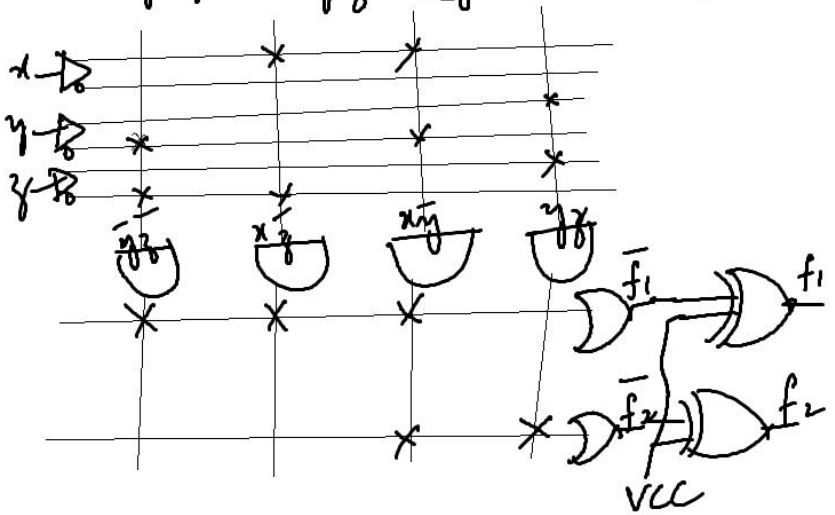


Case (ii) : $\bar{f_1}$ and $\bar{f_2}$

$\bar{f_1}(x,y,z) = \bar{y}\bar{z} + xz + x\bar{y}$

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

total 4 product



Common way of specifying the connections in PLA — PLA table

Case i: f_1 and $\bar{f}_2$ Ex-OR gate:

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

$\bar{f}_2(x,y,z) = yz + x\bar{y}$ ⇒ Ex-OR ⇒ f_2

PLA table: case (i)

Product terms

	Inputs			outputs	
	x	y	z	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

4-rows

T/C

1 – true value

0 – complement

Inputs: 0 → complemented form

1 → uncomplemented form

– → NO connections

Outputs: 1 → connection exists thru AND-gate & OR-gate

– → NO connection exists

Case (ii): $f_1 \to f_2$ and $\bar{f}_2$

f_1 – Ex-OR

f_2 – Ex-OR $\bar{f}_2$

$\bar{f}_1(x,y,z) = \bar{y}\bar{z} + xz + x\bar{y}$

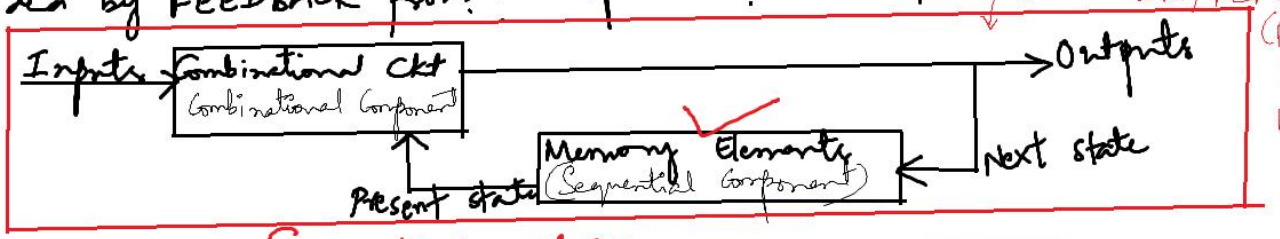
$\bar{f}_2(x,y,z) = yz + x\bar{y}$

total 4 products

Product terms	Inputs			outputs	
	x	y	z	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

Unit -3 : Flip-flops

There are many applications in which digital o/ps are reqd to be generated in accordance with the sequence in which the input signals are received. This reqt can NOT be satisfied using a combinational logic system. These applications req o/p to be generated that are NOT only depend on the present i/p cond^{n}s but also depend up on the past history of these i/ps. The past history is provided by FEEDBACK from the o/p back to the i/p.

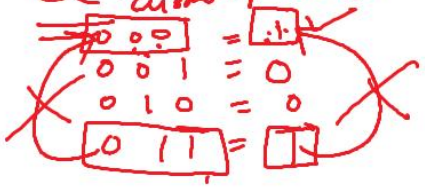


Block diagram of sequential ckt / FSM (Finite State M/c)

Sequential ckt

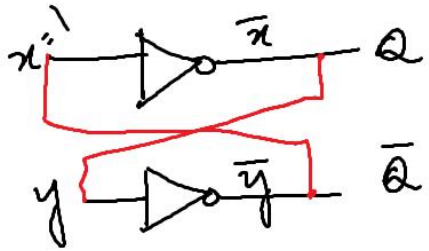
CC + Memory

CC: current i/p combⁿ



SC: Odd parity gen

The basic bistable element:



2 i/ps: x & y (two stable conditions (or) States)

2 o/ps: Q & $\bar{Q}$ → Normal o/p → Complemented o/p

2 cross-coupled NOT gates – o/p of 1st NOT gate serves as i/p to 2nd NOT gate & o/p of 2nd NOT gate serves as i/p to 1st NOT gate → feedback

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

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

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

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

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

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

i.e., $Q = \bar{x} = y = 0$ & $\bar{Q} = \bar{y} = x = 1$

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

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

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

Positive logic: o/p

o/p

Q = 1 → 1-state (SET)

Q = 0 → 0-state (RESET)

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

Amount of time a device stays in met state is unpredictable due to ckt noise.

∴ Must be avoided

$Q = 0$ (or) $Q = 1$
$\bar{Q} = 1$ $\bar{Q} = 0$ } valid

Latches: storage device
↳ o/p immediately responds to change in i/p-lines.

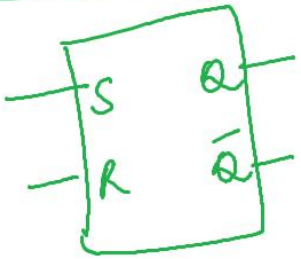
i) The SET & the RESET latch (SR Latch):

→ 2 cross coupled NOR gates

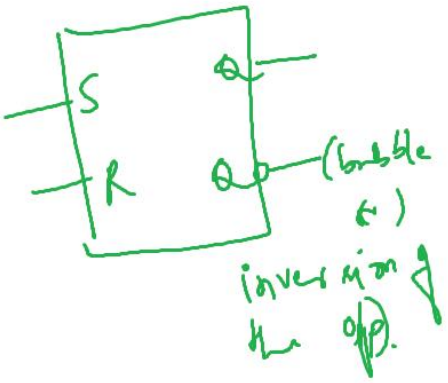
→ Two i/ps (i) S-set
(ii) R-Reset

→ two o/ps: (i) Normal o/p $Q = 0$ $\bar{Q} = 1$
(ii) $Q = 1$ $\bar{Q} = 0$

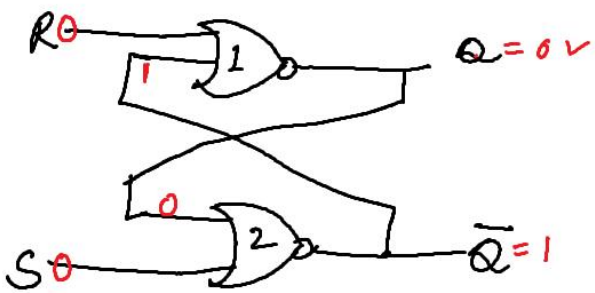
Block diagram:



(or)



SR Latch:



2 i/p: S R Q_n

1) 0 0 No change
2) 0 1 Reset
3) 1 0 Set
4) 1 1 ?

TT of NOR

	NOR
0 0	1
0 1	0 ✓
1 0	0 ✓
1 1	0 ✓

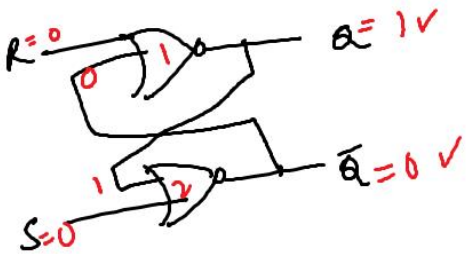
When 1 of the i/p ⇒ o/p = 0.

Case 1: S = 0 and R = 0. (Initially assume $Q = 0$)

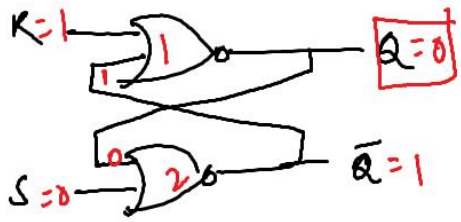
$\bar{Q} = 1$ when $Q = 0$ → stable ⇒ No change

S = 0 and R = 0 (Initially assume $Q = 1$)

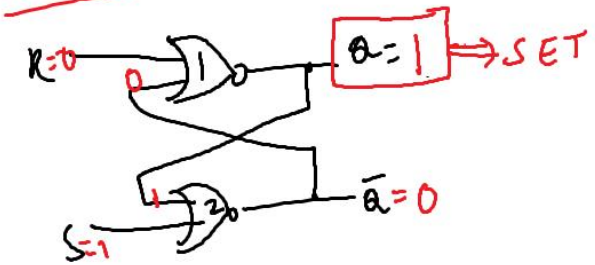
$\bar{Q} = 0$ when $Q = 1$ → stable ⇒ No change



Case 2: S = 0 and R = 1



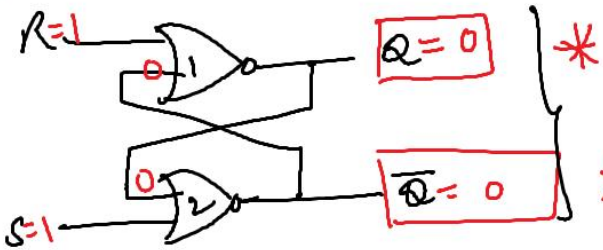
Case 3: S = 1 and R = 0



SR contd...

Case (iv) : S = 1 and R = 1

NOR
0 0 → 1
1 0 → 0
0 1 → 0
1 1 → 0



* Unable to determine ⇒ unpredictable behaviour.

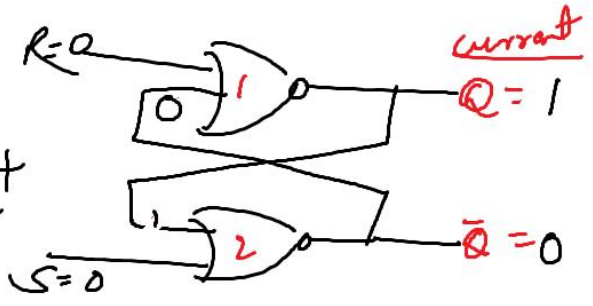
S	R	Q
0	0	No change
0	1	Reset
1	0	Set
1	1	* → Unpredictable

forbidden i/p condn

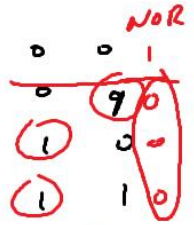
Q = 0 ⇒ Q̄ = 1
Q = 1 ⇒ Q̄ = 0
} Q & Q̄ are complementary.

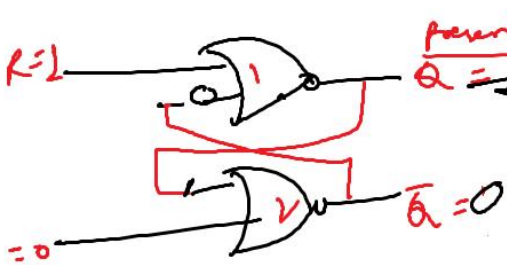
In this case S = 1 & R = 1, Q & Q̄ are NOT complementary (Metastable state).

S	R	Q	o/ps
0	0	Q	present state = Q
0	1	0	
1	0	1	next state = Q^+
1	1	*	forbidden state



Q^+ 1 Q_n Q_{n+1}





Q = 0 , $\bar{Q}$ = 1 ⇒ stable state

Q = 1 , $\bar{Q}$ = 0 ⇒ stable state

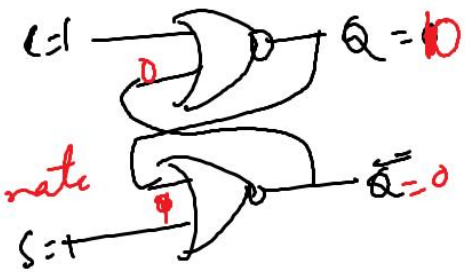
S = set
R = Reset

S	R	present Q	next state Q^+	
0	0	0	0	
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	set
1	1	0	1	

S	R	Q	Q^+	
1	1	0	0	*
1	1	1	0	

Indeterminate

Q & $\bar{Q}$ forbidden state



Q & $\bar{Q}$ = 0

S	R	O/p Q
0	0	Q
0	1	0
1	0	1
1	1	*

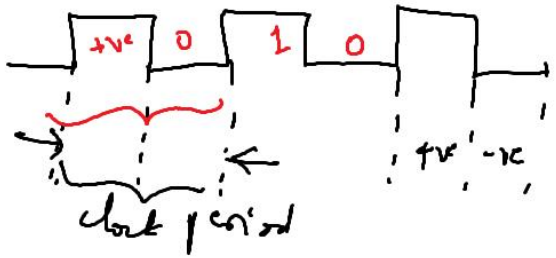
JK flip flop:

S	R	current state Q	Q^+ → Next state	
0	0	0	0	Q
0	0	1	1	Q
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	set
1	0	1	1	set
1	1	0	0 → Q & $\bar{Q}$ = 0	Invalid
1	1	1	0 → Q & $\bar{Q}$ = 0	Invalid

forbidden

Disadvantage of SR flip flop.

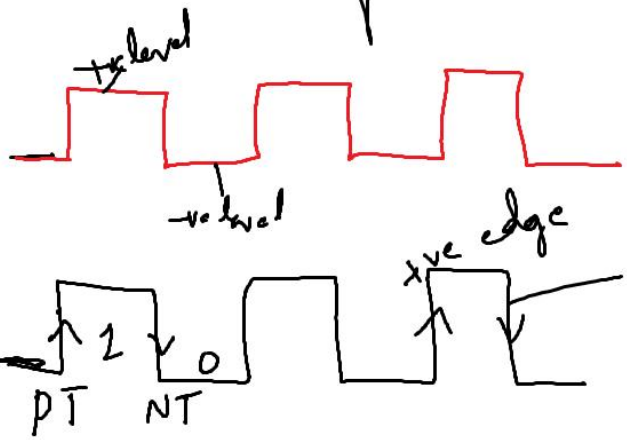
Clock Signal : is a particular type of signal that oscillates b/w a HIGH & LOW state & it is utilized to co-ordinate the actions of the circuit(ckt). It is produced by the Clock-generator.

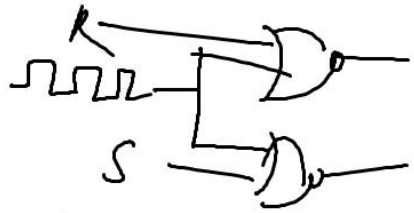


Digital signal — Square wave

Analog " — Sine-wave

Synchronous w.r.t clock :





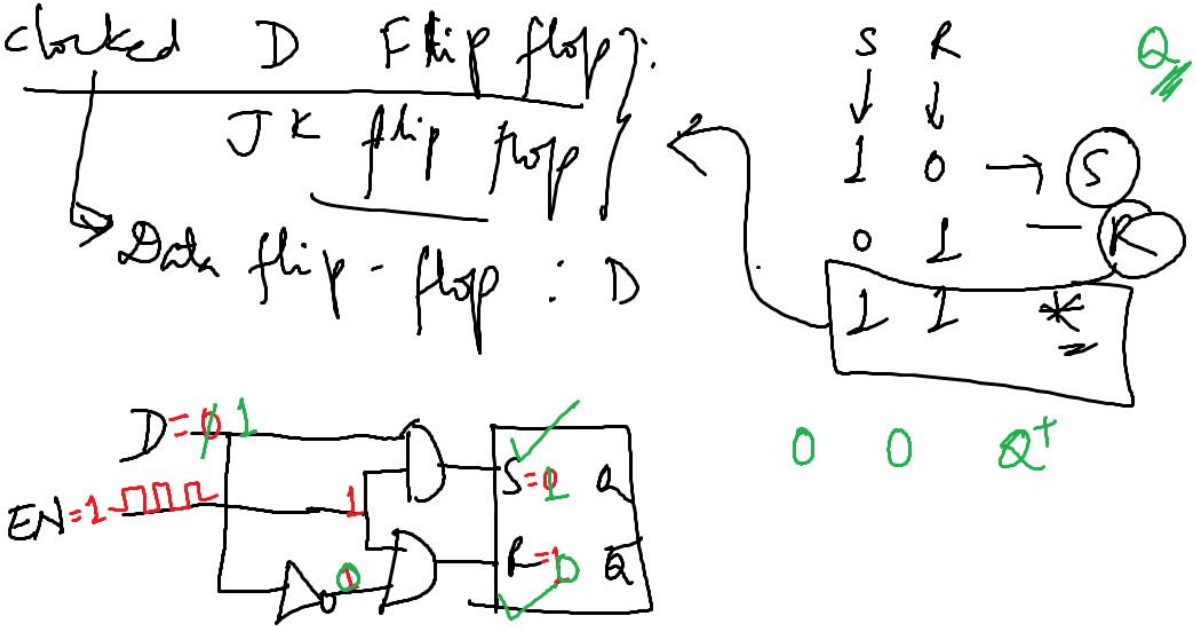
clocked SR FF

S	R	EN	Q	$\bar{Q}$
X	X	0	Q	$\bar{Q}$
0	0	1	Q	$\bar{Q}$
0	1	1	0	1
1	0	1	1	0
1	1	1	?	?

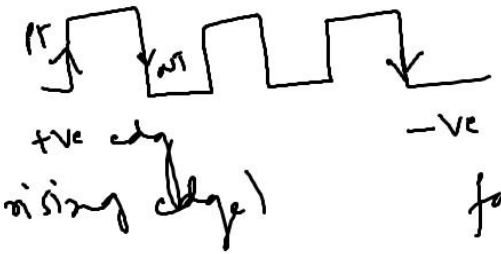
clocked D Flip flop:

JK flip flop

Data flip-flop : D

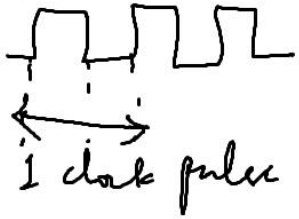


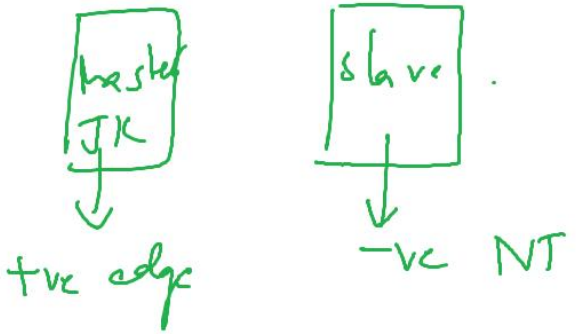
EN	D	Q^+
0	X	Last state (Q)
1	0	0
1	1	1



+ve level
-ve level

+ve edge triggered = PT
-ve edge " = NT





JK flip flop: Overcome the shortcoming of SR forbidden state.

Edge-triggered flip flop: +ve edge trigger JK ff

: PT JK ff.

Various representn of FF:

① Characteristic eqn

② FSM – state transn diagram (finite state m/c)

③ Excitation table.

① Characteristic eqn : (J.K)

3 vari J, K, Q → Q^+

Q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

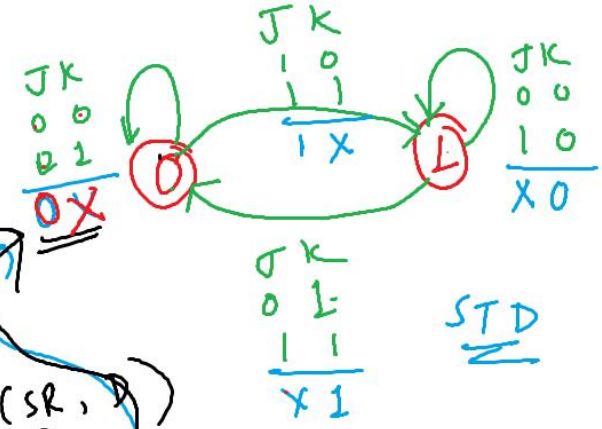
$Q^+ = \bar{Q}J + Q\cdot\bar{K}$

$\therefore$ $Q^+ = J\cdot\bar{Q} + \bar{K}\cdot Q$

J	K	present o/p Q	next o/p Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

SR: $S + \bar{R}Q$

D: D

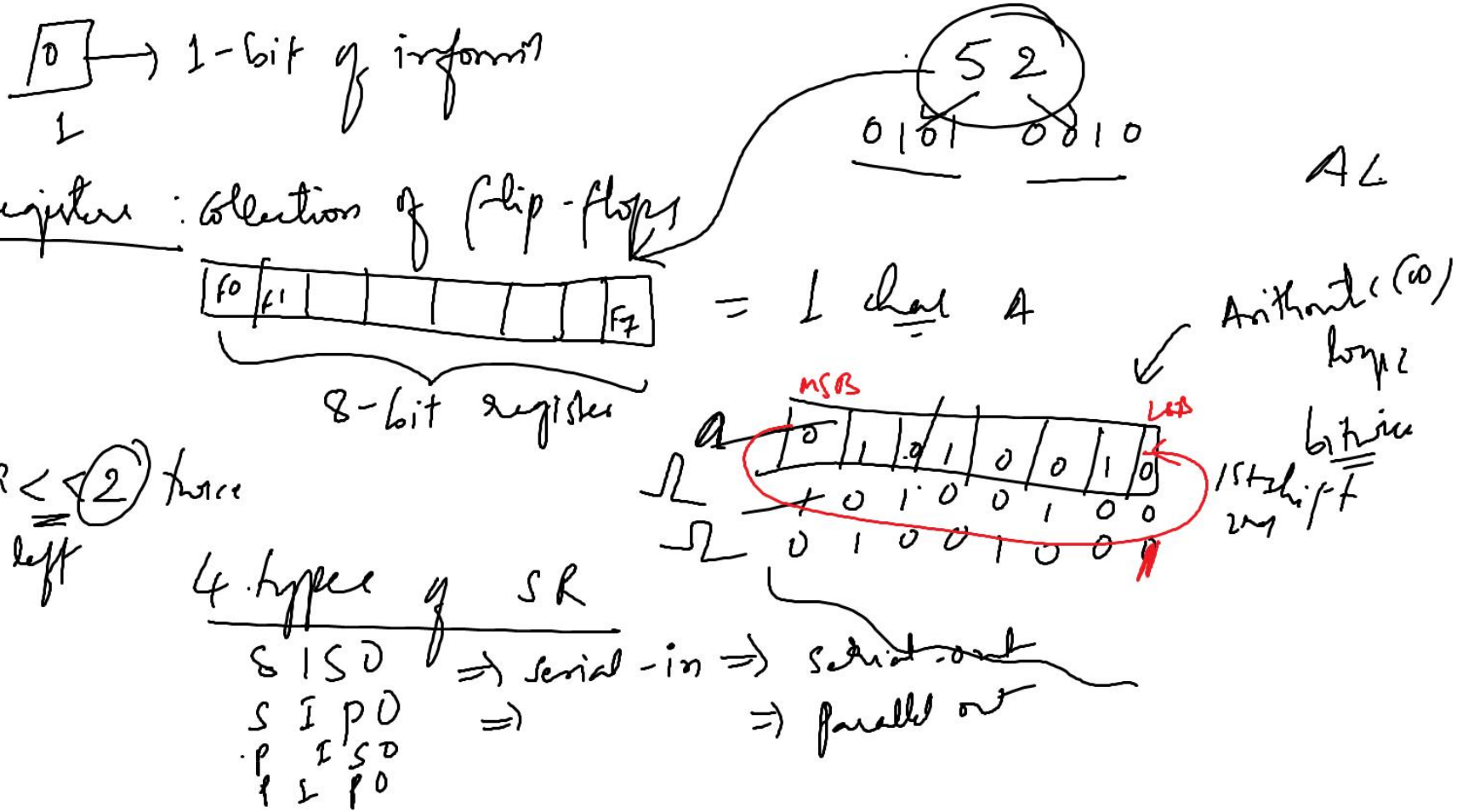


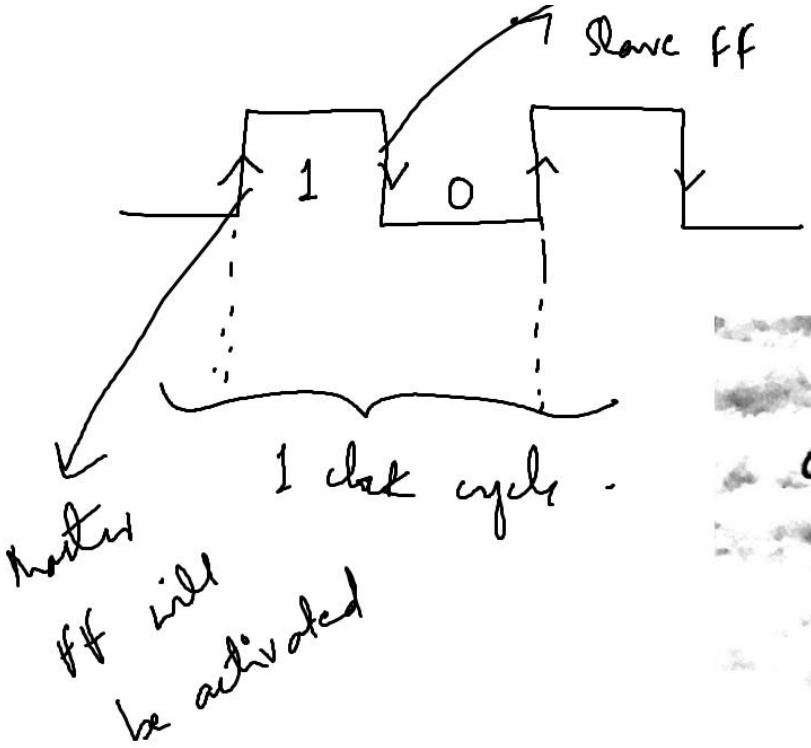
② State Transn diagram

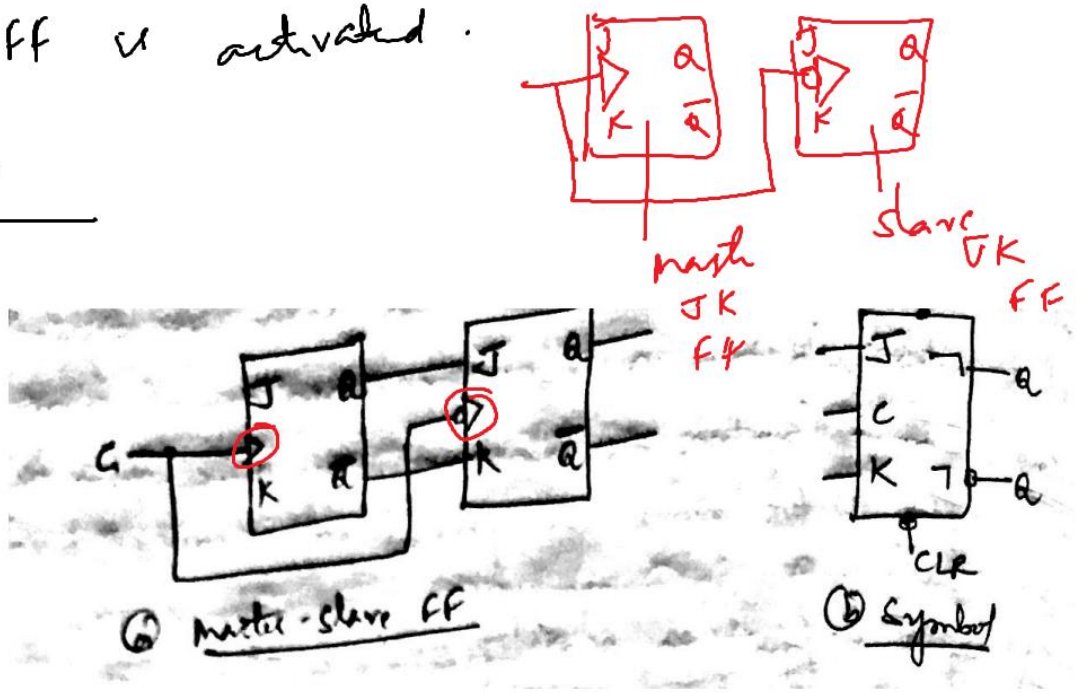
③ Excitation table

(SR, D)

Q	Q^+	J	K	S	R	D
0	0	0	X	0	X	0
0	1	1	X	1	0	1
1	0	X	1	0	1	0
1	1	X	0	X	0	1

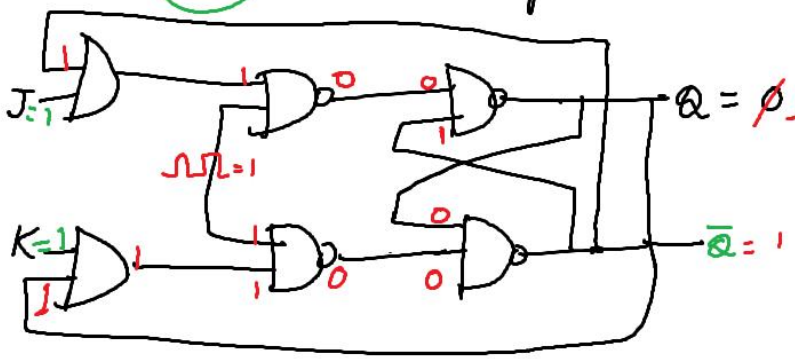

0
1
1-bit of inform
52
0101 0010
AL
Register : Collection of flip-flops
F0 F1 F7
8-bit register
= 1 char A
Arithmetic (or) logic bitwise
MSB
LSB
a
0 1 0 1 0 0 1 0
+ 0 1 0 0 1 0 0
0 1 0 0 1 0 0 0
1 Shift
a = a << 2 times
left
4 types of SR
SISO ⇒ serial-in ⇒ serial-out
SIPO ⇒ ⇒ parallel out
PISO
PIPO


Slave FF is activated.
1
0
1 clock cycle.
Master FF will be activated


J
Q
K
Q̄
master JK FF
slave JK FF
C
CLR
ⓐ master-slave FF
ⓑ Symbol

JK flip flop : $\overline{S}\overline{R}$ → using NAND

SR

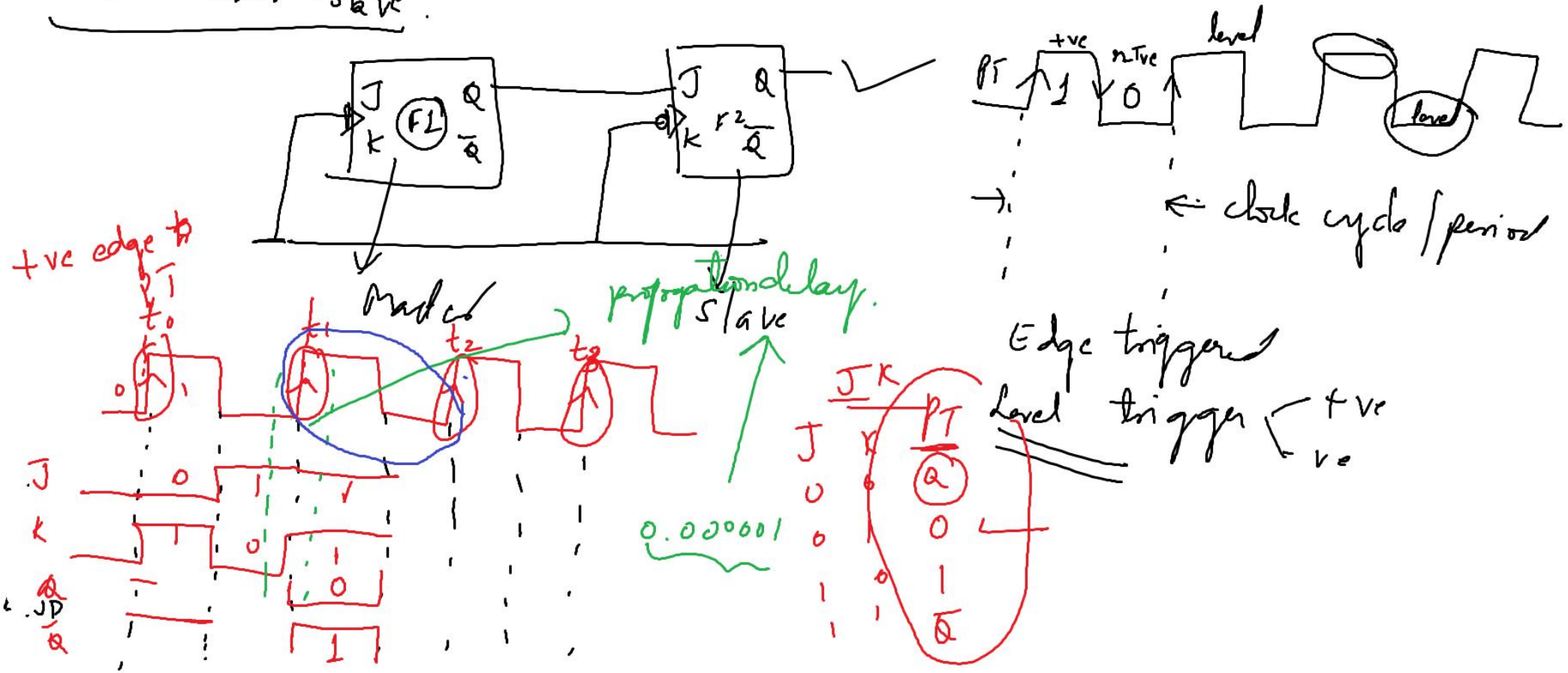


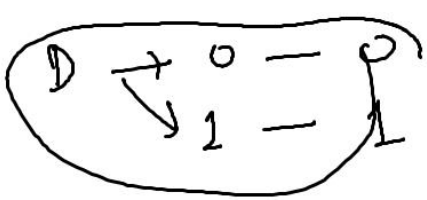
Q = 1 = toggled

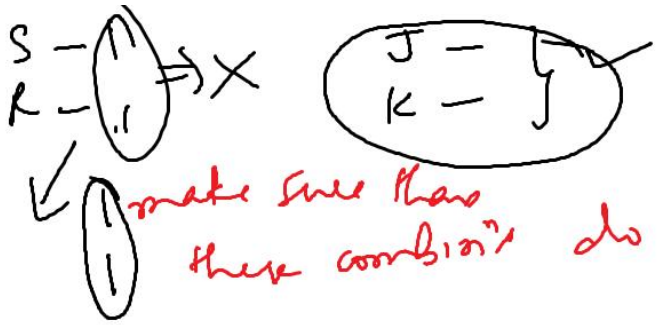
J	K	Q	Q^+	
0	0	0	0	Q (last state)
0	0	1	1	
0	1	0	0	0 - reset
0	1	1	0	
1	0	0	1	1 - set
1	0	1	1	
1	1	0	1	toggled $\overline{Q}$
1	1	1	0	

J=1, K=1, Q=0 what is Q^+ = ? 1

J K Master-Slave:



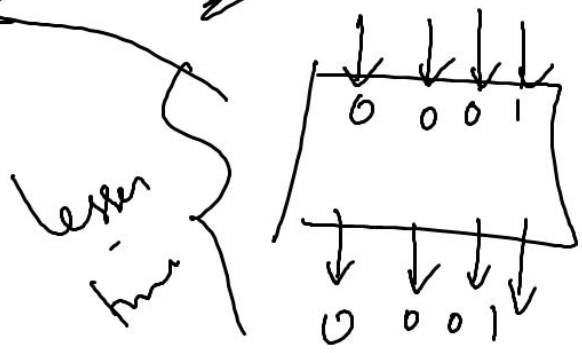

D
0
1
0
1


S
R
X
J
K
make sure these
these combin" do NOT exist


F1
f2
f3
f4

NIBBLE

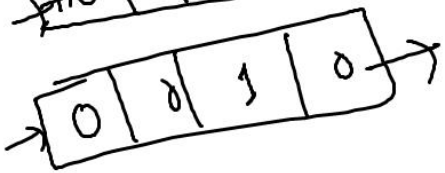

0
0
1
0

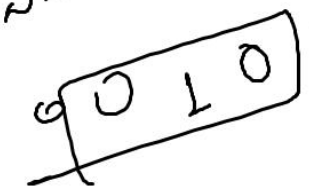

parallel in
0 0 0 1
lower nibble
parallel out
0 0 0 1

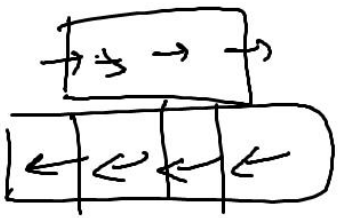

serial-in
0

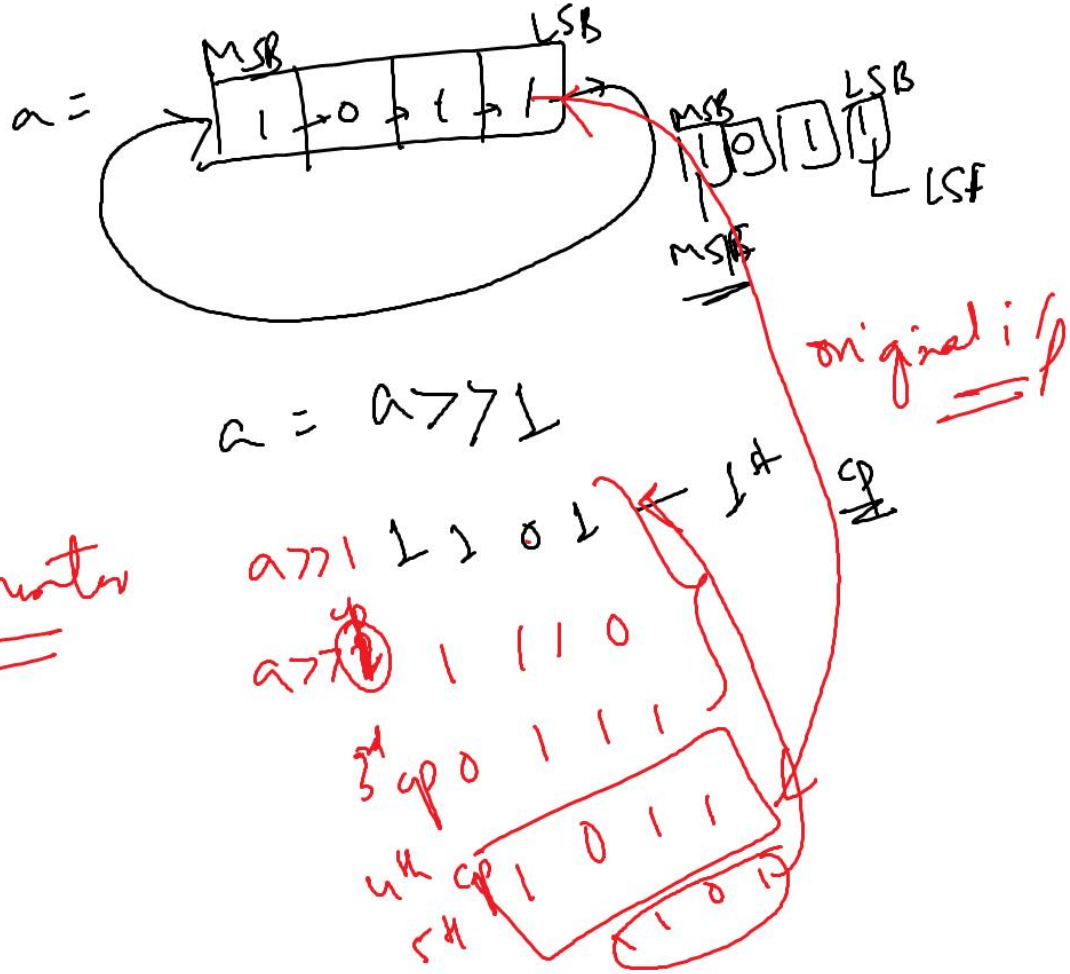

serial-in
1
0


serial-
0
1
0


0
0
1
0


0 1 0
serial out

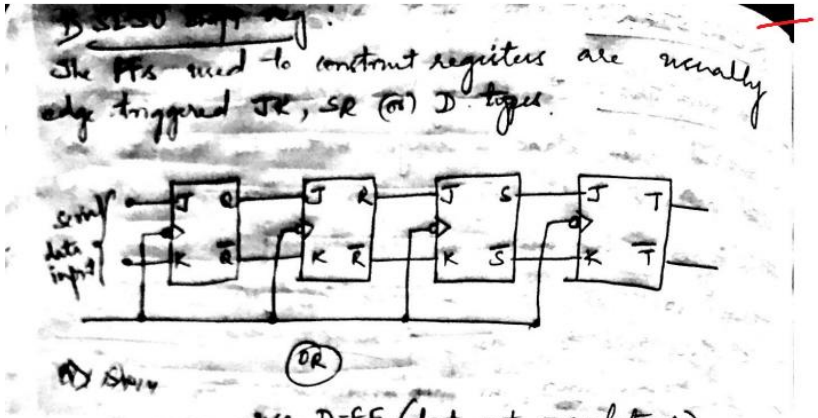




Ring ◎

↓

Ring counter



use D-FF (but not mandatory)

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

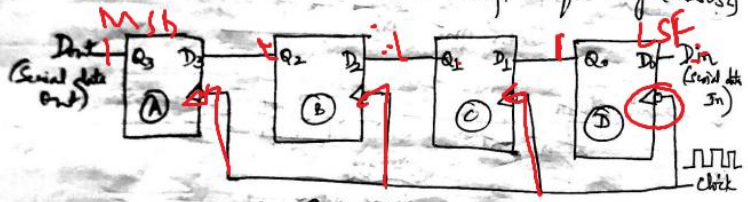
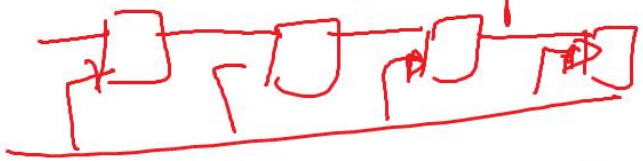


fig: Shift-left reg

D flip flop

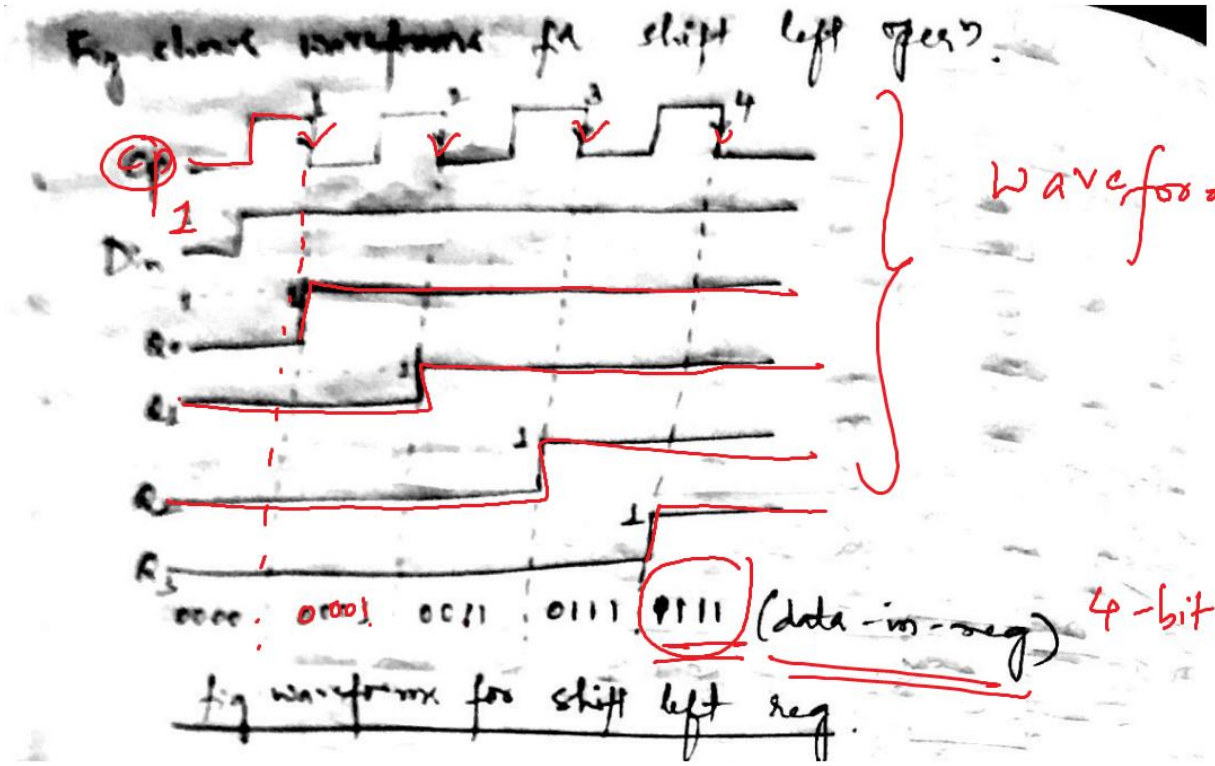
Serial-In ⇒ Serial-out

4-FF → 4-bit shift reg



Din ⇒ Data i/p

	D_3	D_2	D_1	D_0	
Din	Q_3	Q_2	Q_1	Q_0	
	↓	↓	↓	↓	
	0	0	0	0	cl
	0	0	0	1	↓
	0	0	1	1	↓
	0	1	1	1	↓



waveforms – timing chart

given

4-bit reg

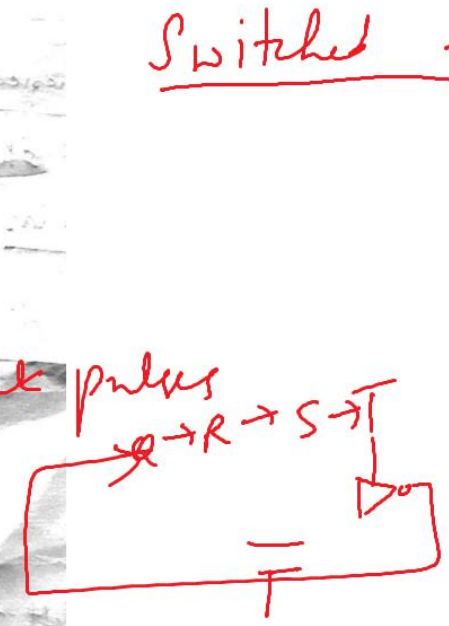
Switched tail / twisted tail / Johnson counter.

clock	Flip-Flops 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

(9-1) 8 clock pulses

8 clock pulses

fig (b): State Table of Johnson counter



4-FFs ⇒ 8 clock cycl.

N- FFs ⇒ 2N clock pulses.

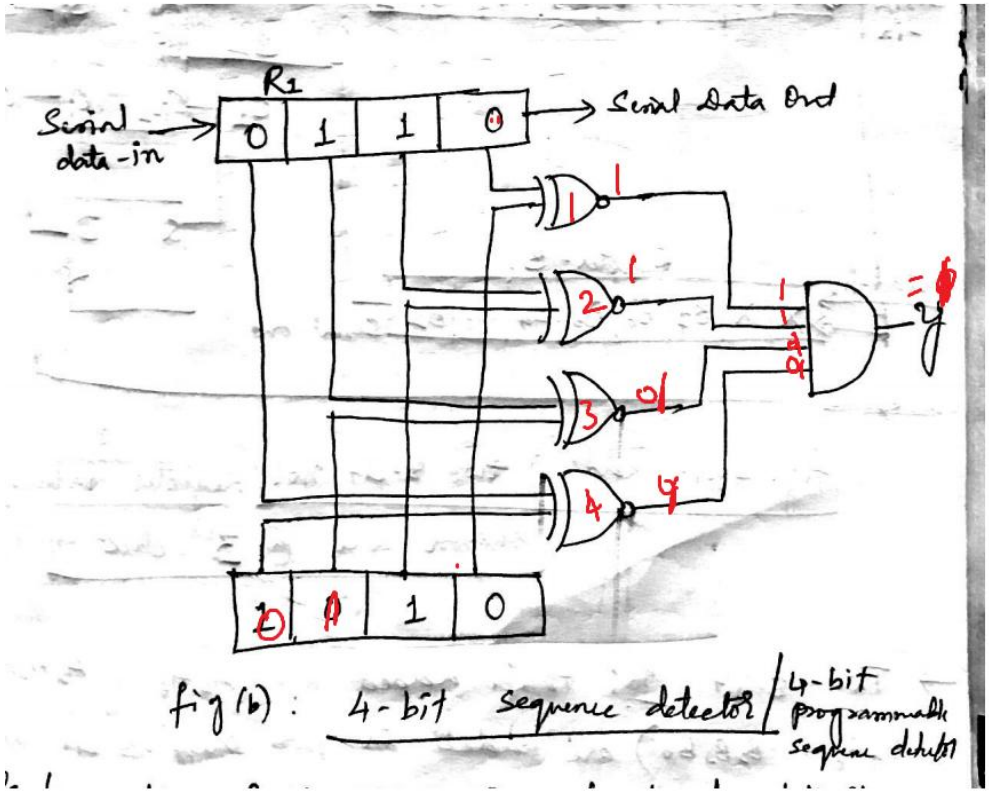


fig (b): 4-bit sequence detector / 4-bit programmable sequence detector

R1 0 1 1 0

R2 1 0 1 0 Data to be stored in reg

EX-OR gate EX-NOR

0 0 → 0 → 1

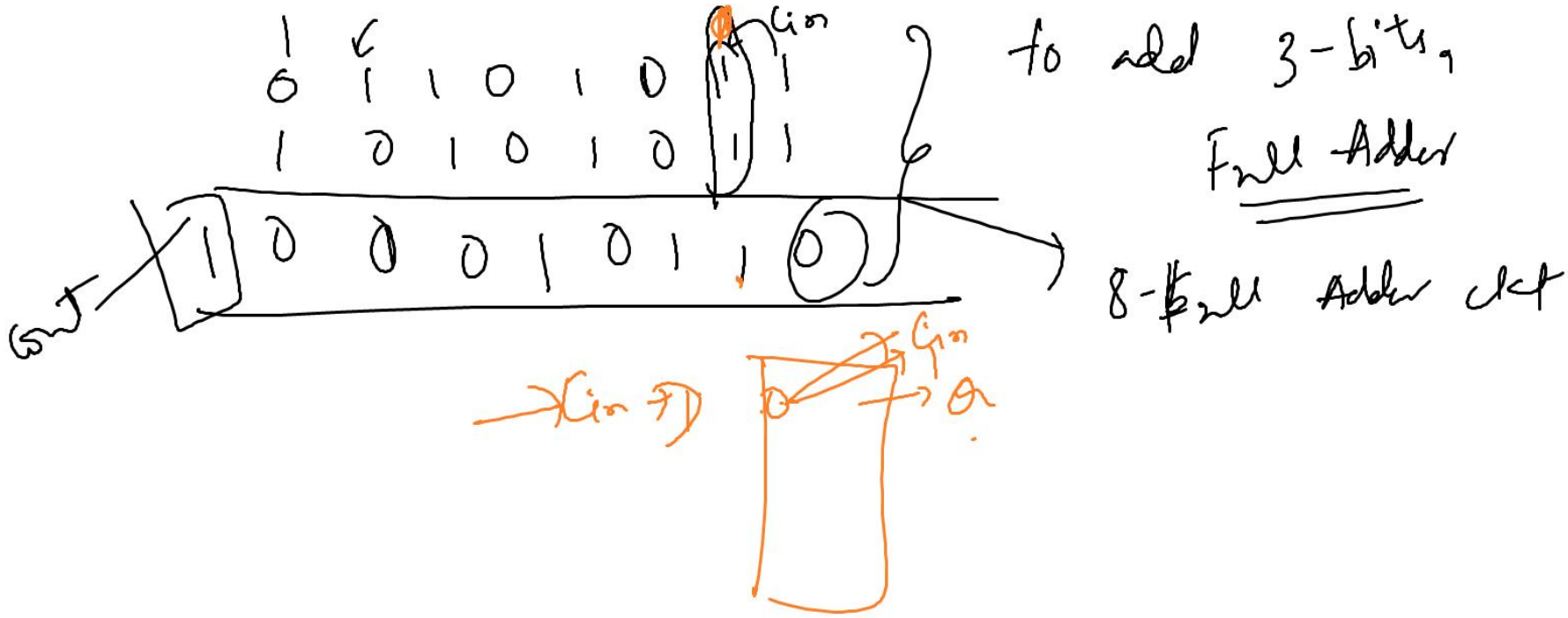
0 1 → 1 → 0

1 0 → 1 → 0

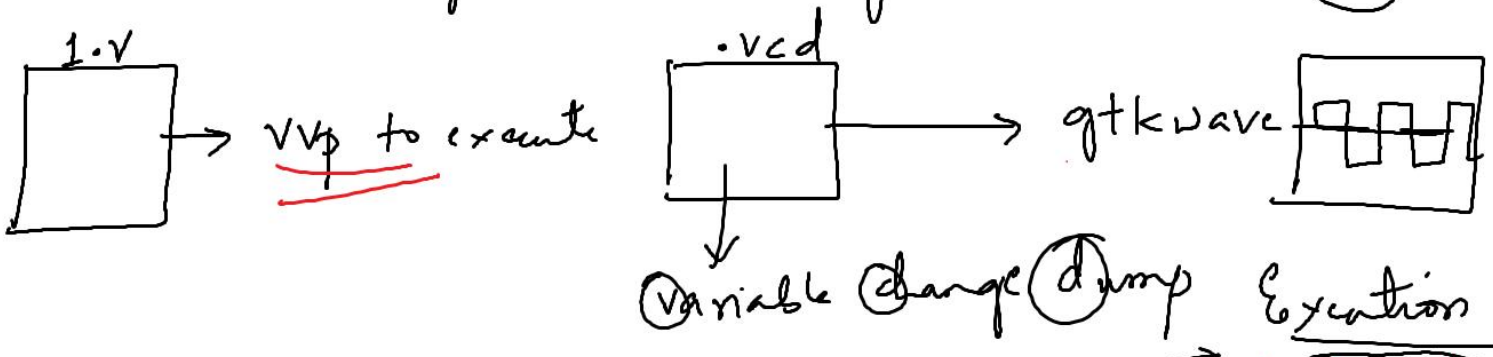
1 1 → 0 → 1

Both i/ps are equal = 1

All are 1 = 1 in AND


Cin
to add 3-bits,
Full Adder
8-Bit Full Adder ckt
Cout
Cin
Cin
Co

icarus : (i)verilog ⇒ icarus verilog
- verilog simulation & synthesis



1.v ⇒ cd (change directory)
⇒ cd iverilog
⇒ c:\iverilog\: cd bin
⇒ c:\iverilog\bin:

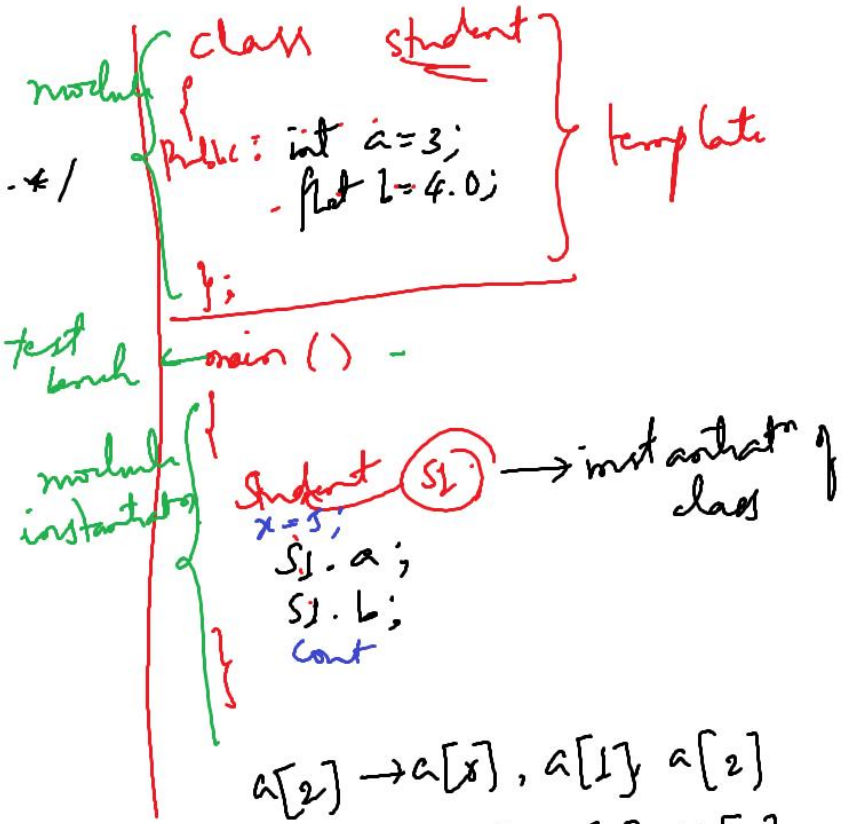
Execution of iverilog:

1) iverilog -o some-file 1.v
2) vvp some-file
test.vcd → in your source prgm only.
3) gtkwave test.vcd

Writing verilog programs: AND, OR, NOT opers

```
module gates (input a, b, output [2:0] y);
    assign y[2] = a & b;  //AND   multiline /*.....*/
    assign y[1] = a | b;  //OR    comments.
    assign y[0] = ~a;     //NOT
endmodule

module gates_tb;
    wires [2:0] y;
    reg a, b;
    gates aon(a, b, y);
    initial
    begin
        $dumpfile("gate_test.vcd");
```



a[2] → a[0], a[1], a[2]

y[2:0] → y[2], y[1], y[0]

```
$dumpvars(0, gates_tb);
  a = 1'b0;
  b = 1'b0;
  #50; //time delay - 50 units of time
  a = 1'b0;
  b = 1'b1;
  #50;
  a = 1'b1;
  b = 1'b0;
  #50;
  a = 1'b1;
  b = 1'b1;
  #50;
end
endmodule
```

ns

a	b	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Syntax:

<size>'<radix><value>

NOT:

1 operand

~a ~b

size : 1-bit

radix : base : binary : b

value : 0 / 1

a → 0 / 1

$dumpvars(<levels>, <module>);

```
module gates(     → 0
endmodule

module univ( );
:
endmodule

module FF();
:
endmodule
```

0, TB main()

1, TB main()

2, TB main()

```
add()
sub()
mul()

main()
{
  struct s1;
  s1.add();
  s1.sub();
  s1.mul();
}
```

TC / C++ → IDE

change directory

c:\Users\bmsce> cd .. (parent)

c:\Users> cd ..

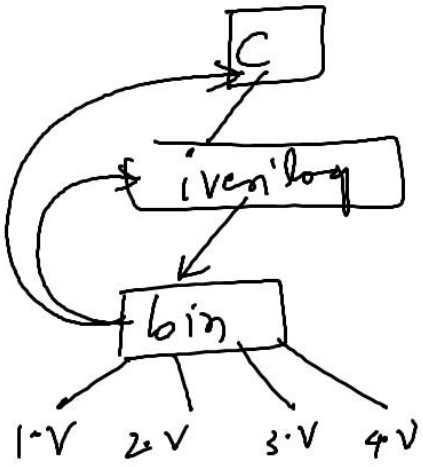
c:\> cd iveriloj

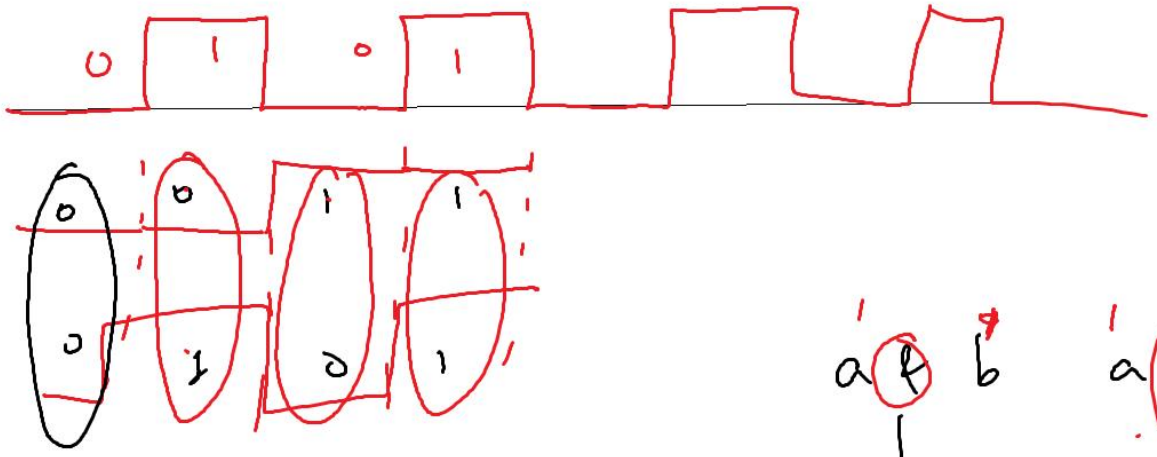
c:\iveriloj> cd bin

c:\iveriloj\bin>

c:\TC\bin\ I.C

c:\iveriloj\bin





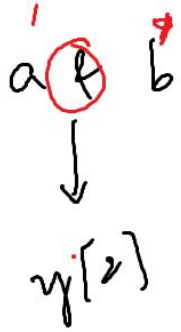
y[2:0]: y[2] y[1] y[0]

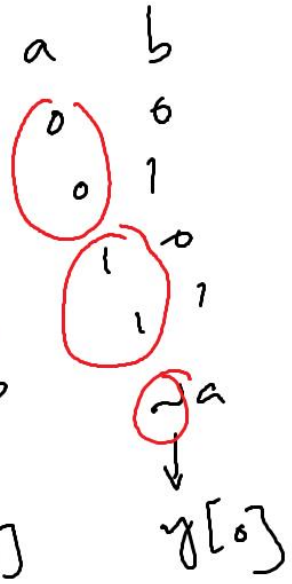
= 001

= 010

= 011

= 110





Counters :

FF → registers → Counters → is also a register capable of counting the no. of clock pulses arriving @ its input.

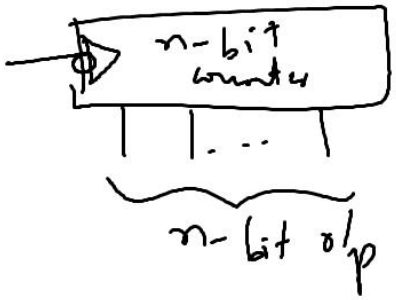
↓

group of FF.

↓

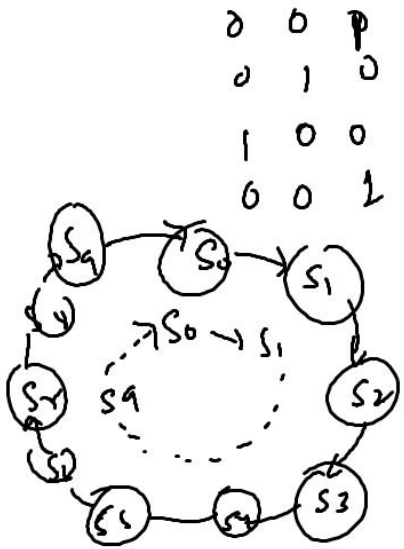
storing & shifting the data

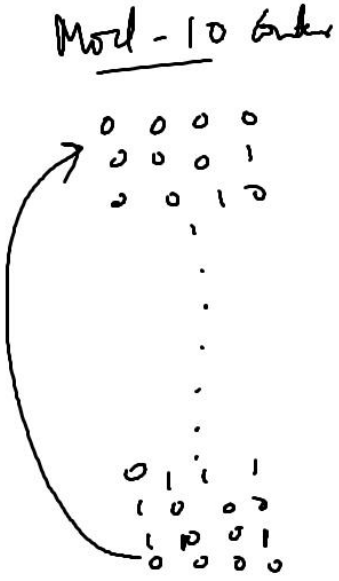
(a) FF – n-bit counter



(b) NT – n-bit counter

Mod-10 Counter





The n-bit binary counter has n-flip flops and has 2^n distinct states of o/p

Eg. 2-bit bicounter has 2-FFs and 2^2-distinct o/p state
- 4 o/p states
00, 01, 10, 11

IIIrd 3-bit binary counter has 3-FFs — 2^3 - o/ps (0, 1, 2, 3)
8-o/ps 000, 001, ... 111
(n=3) 0 .... $2^n - 1$

2 types 1) Synchronous counter
2) Asynchronous ..

Synch: When a counter is clocked such that each FF in the counter is triggered @ the SAME-time.

Async: Each FF is connected to the clock i/p of NEXT-HIGHER-ORDER-FF

Differences/comparison

Asynchronous counter	Synchronous counter
1) O/p of First FF drives the clock for the next FF	1) No connection b/w the O/p of 1st FF & clock i/p of the next FF
2) All the FFs are NOT clocked simultaneously	2) All the FFs are clocked simultaneously
3) Speed: Slow (propagation time delay)	3) Fast
4) Logic ckt is very simple even for more no. of states	4) Design involves complex logic ckt as no of states increase.
5) Cost is LESS	5) Cost is MORE

Modulus of Counter: Total no. of counts (or) states a counter can indicate is called as Modulus.

Eg: Modulus of a 4-stage counter would be $(2^4) = 16$

$2 \mid 16$
$2 \mid 3 - 0$
$1 - 1$

0000 (2)
⋮
1111 (2).

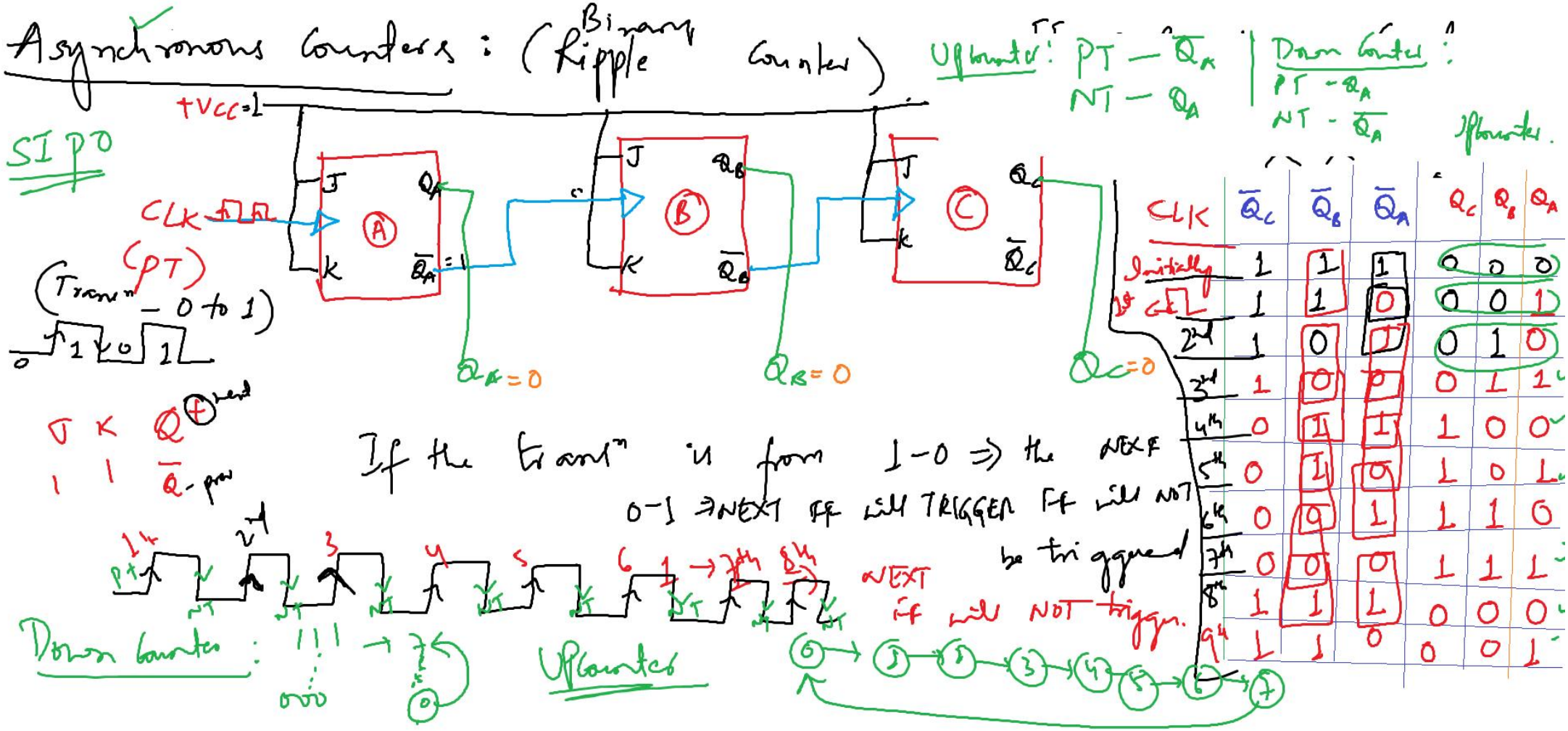
Eg: Mod-6 counter ⇒

0	0	0	0
0	0	1	⋮
0	1	0	⋮
0	1	1	⋮
1	0	0	⋮
1	0	1	↙

Mod-4 counter : 0 → 3 ⇒

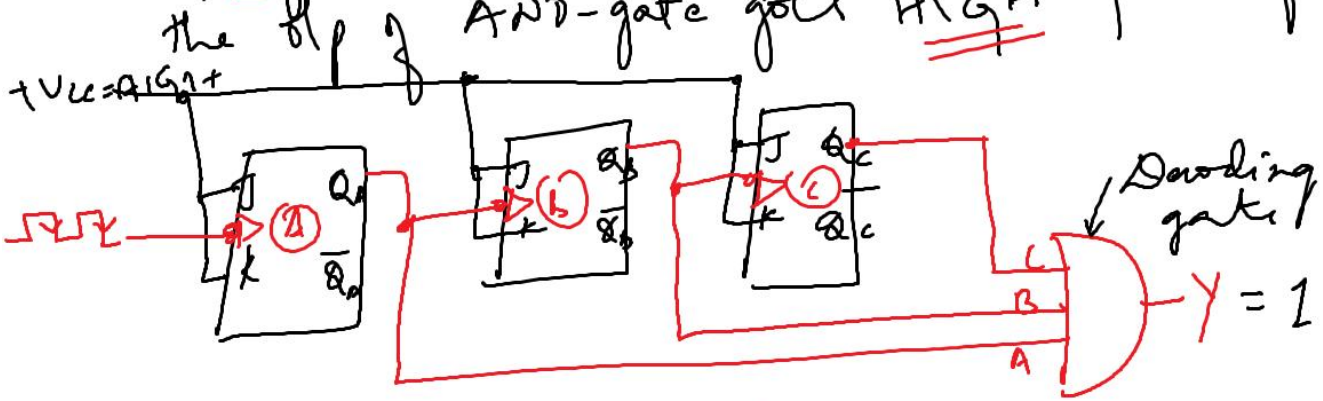
0	0
0	1
1	0
1	1

Asynchronous Counters : (Binary Ripple Counter)



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

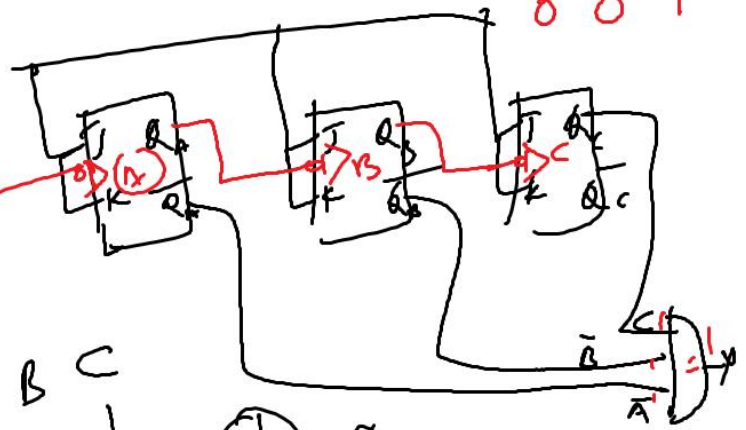
→ O/ps of the counter are connected to the AND-gate as i/ps and the O/p of AND-gate goes HIGH for a particular state.



C = 1 , B = 1 and A = 1

(a) Decoding gate to indicate the state $7_{(10)} = 111_{(2)}$

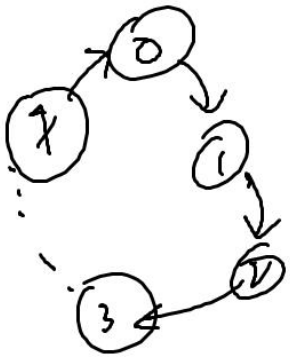
A B C
1 1 1



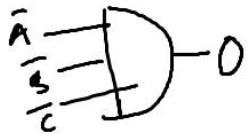
(b) Decoding gate to indicate state $4_{(10)} = 100_{(2)}$

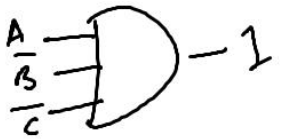
III by, we can connect corresponding o/ps to decoding gate i/ps to indicate the desired state

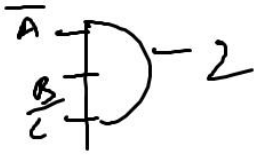
3-bit (3-FF)

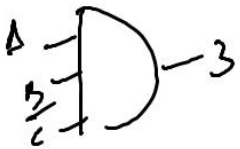


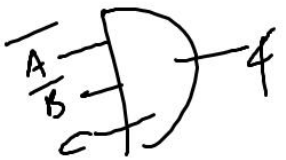
	C	B	A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

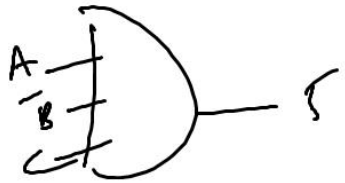


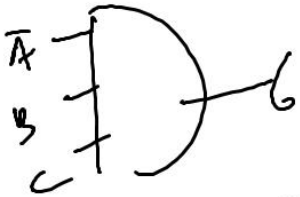


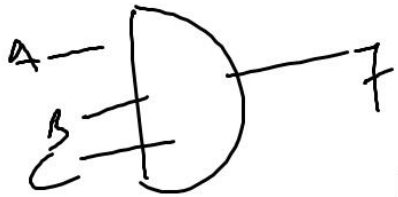






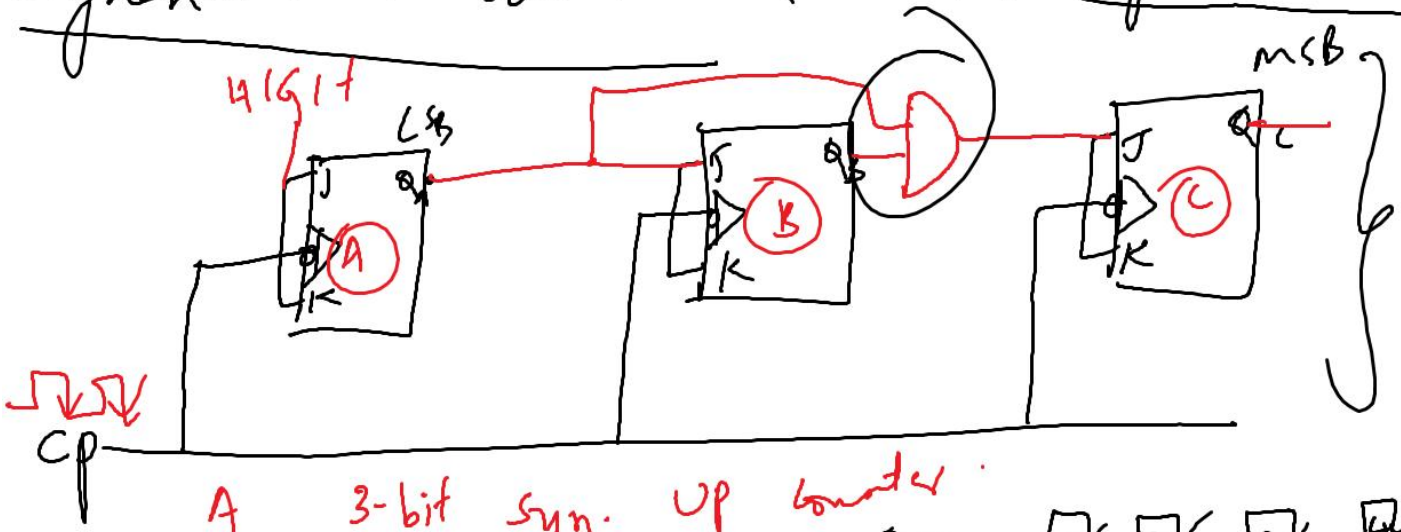






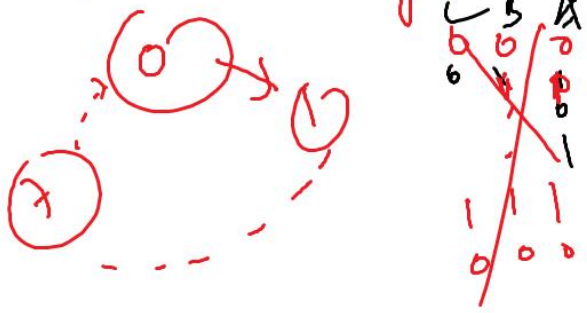
Decoding gate : 1. can design Synchronous counter

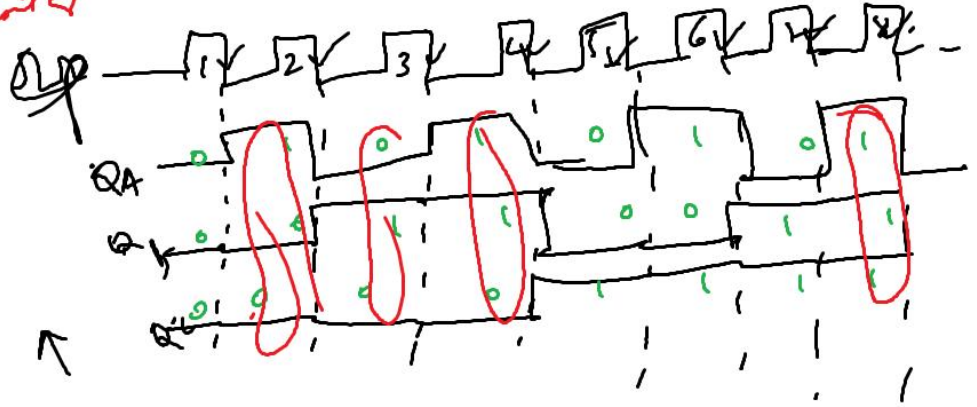
Synchronous counters : (3-bit) Synchronous Binary Up counter :

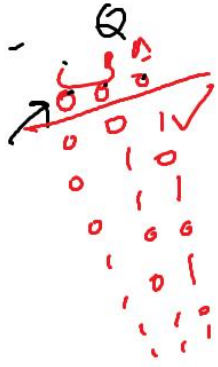


working of this, we should know the Design of a synchronous counter.

A 3-bit syn. Up counter







Design of Synchronous Counters:

1) Determine the no of FFs reqd.
   If $n \rightarrow$ no. of FF, then 3-FFs - 3-bits
   $N \rightarrow$ no of states (2^3 - states)
   i.e., $2^n \geq N$ in the counter

2) Choose the type of the FF to be used

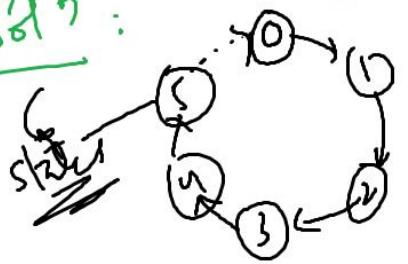
3) Using the excitation table for selected FF, determine the excitations table for the counter.

4) Use K-map (or) any other simplification method to derive the FF f^ns.

5) Draw the logic diagram of reqd counter.

Eg: Design a synchronous Mod-6 counter using clocked JK ff:

Sol^n:



→ How many bits are reqd to represent (6)?

$2^n \geq N$ — no of state.

$8 \geq 6$

$2^3 \geq 6$

$\therefore n = 3$

2 | 6
2 | 3 - 0
1 - 1

$110 = 3$-bits

② JK - FF is choosen

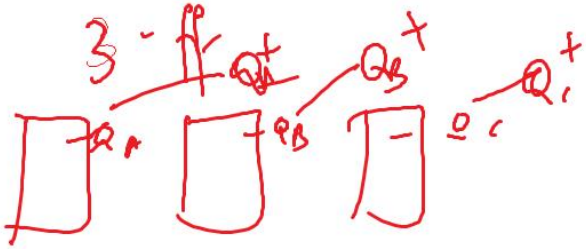
③ Excitation table of JK-ff, det Exc^n table of Counter (Trans^n Table)

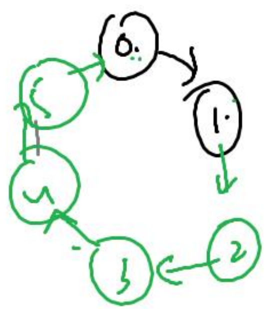
Pres Q	Next Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

state trans^n diag^m → Ex. table (counter) → JK

Present state			Next state			Flip-flop 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	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

3 - ff





④ Use K-map as simplic^n method.

FF

J_A

Q_A \ Q_BQ_C	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$\therefore J_A = Q_B \cdot Q_C$

K_A

Q_A \ Q_BQ_C	00	01	11	10
0	X	X	X	X
1	0	1	X	X

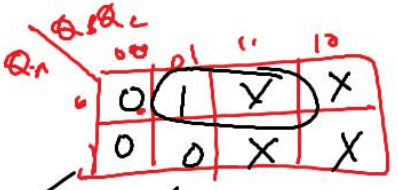
$\therefore K_A = Q_C$

	Present state			Next state			Flip-flop 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	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

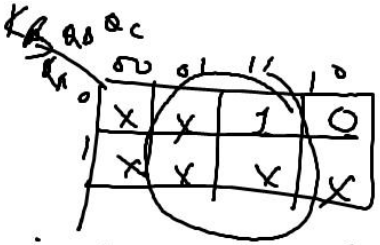
$J_A = Q_B Q_C$

$K_A = Q_C$

J_B =

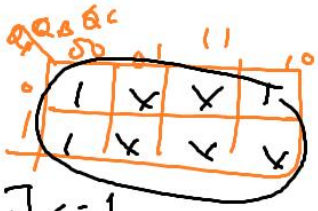


$\therefore J_B = \bar{Q}_A \cdot Q_C$ (Reading gate)



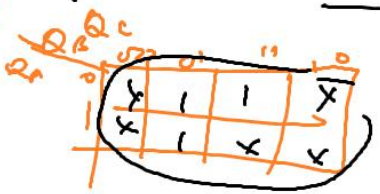
$\therefore K_B = Q_C$

J_C =



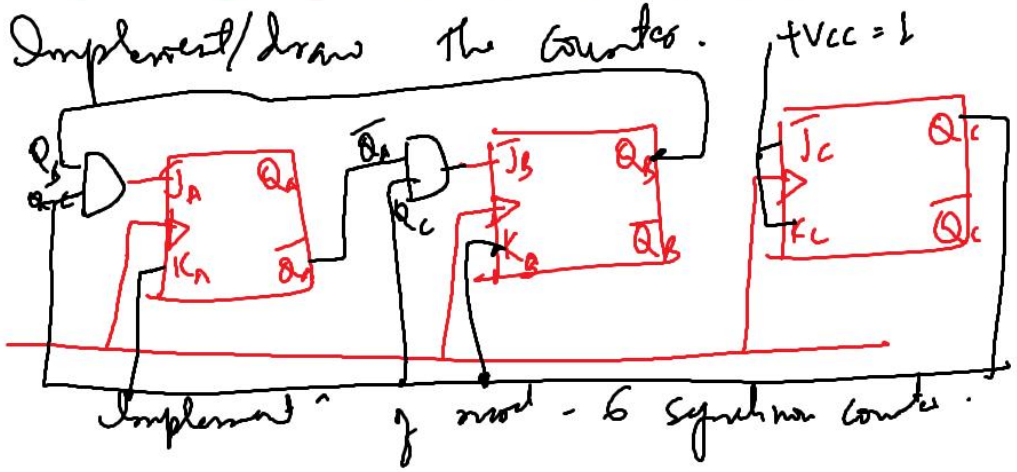
$\therefore J_C = 1$

K_C



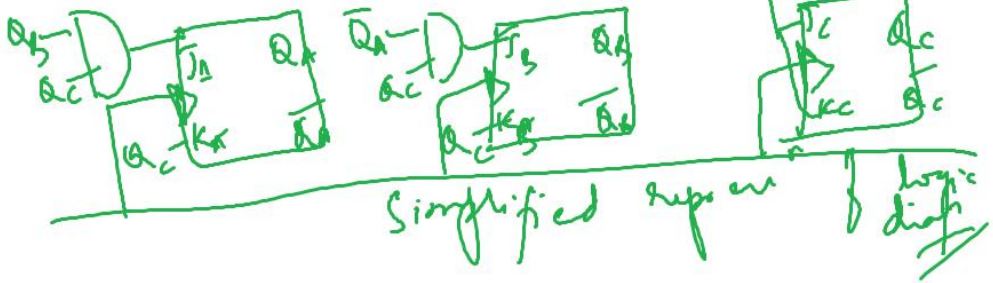
$\therefore K_C = 1$

(5) Implement/draw the counter.



Implement of mod-6 synchronous counter.

To avoid crossing of lines - represent the counter by signal names



Design of a Synchronous Mod-6 counter using clocked T flip flop

1) Finding the no of ff's ⇒ mod-6

3-bits = 3-ff

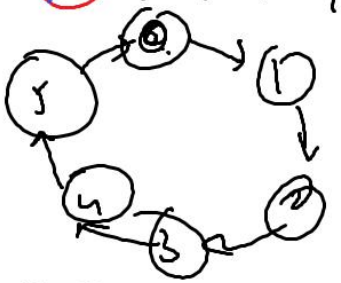
$\therefore n = 3$

$N = 6$

$2^n \geq N$

$2^3 \geq 6$

$8 \geq 6$



2|6
2|3-0
1-1

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

2) Def. the Excitable table of a T-ff

Q	Q^+	
0	0	0
0	1	1
1	0	1
1	1	0

state transn diagm of T-ff

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

D-ff Excitn table

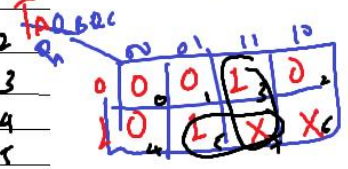
3) Def. Excitable Table for Counter (T-ff)

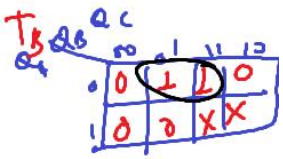
Transn table of mod-6 counter

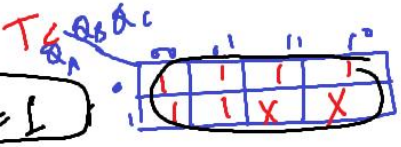
Present state			Next State			T-flip-flop i/ps			
Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	T_A	T_B	T_C	
0	0	0	0	0	1	0	0	1	0
0	0	1	0	1	0	0	1	1	1
0	1	0	0	1	1	0	0	1	2
0	1	1	1	0	0	1	1	1	3
1	0	0	1	0	1	0	0	1	4
1	0	1	0	0	0	1	0	1	5
1	1	0	X	X	X	X	X	X	6
1	1	1	X	X	X	X	X	X	7

4) K-map: ff f^n's.

$T_A = ?$ $T_B = ?$ $T_C = ?$



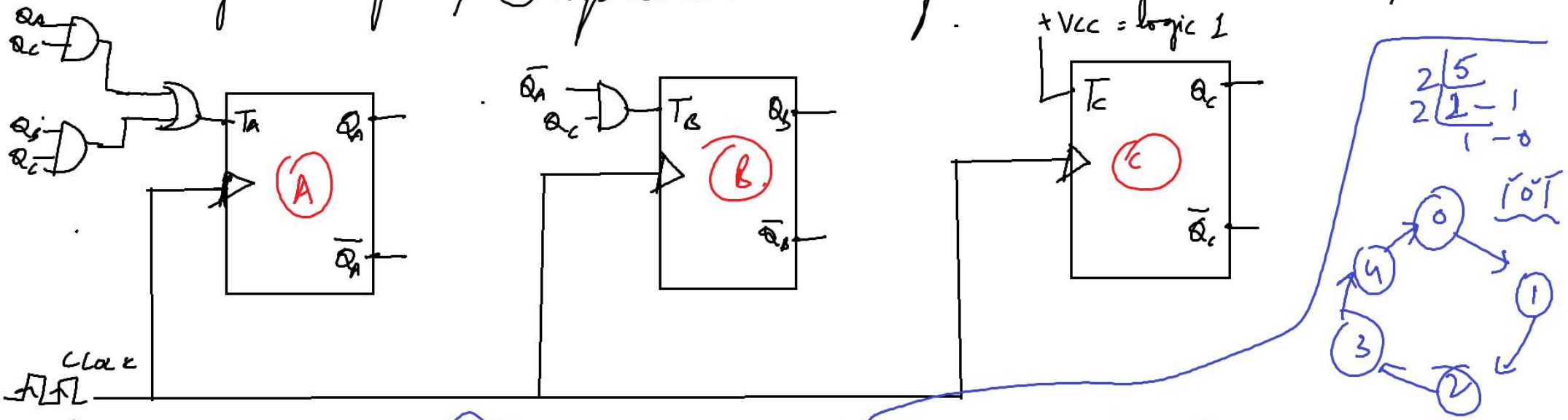


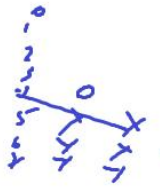


$\therefore T_A = Q_A \cdot Q_C + Q_B \cdot Q_C$ $T_B = \bar{Q}_A \cdot Q_C$ $T_C = 1$

$T_A = Q_A \cdot Q_C + Q_B \cdot Q_C \quad T_B = \bar{Q}_A \cdot Q_C \quad T_C = 1$

3) Draw a logic diagram / Implementation of mod-6 counter using T-ff





Assignment: Design a mod-5 counter using D-ff

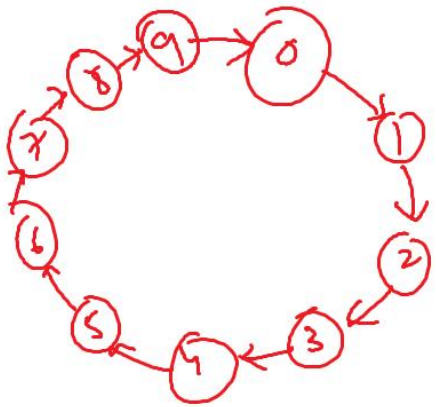
3-ff ⇒ $2^3 = 8$ possible 5-states ⇒ 0, 1, 2, 3, 4, 0, 1 ....

(2) Design a synchronous Decade-counter using D-ff

Decade-counter → 10 ⇒ mod-10 (0 → 9)

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

⇒ $10_{(10)} = 1010_{(2)}$ = 4-bits = 4-ff ⇒ $2^4 = 16$ combin?



D-ff Excit" table

JK → T-ff

JK → toggled-o/p → T-ff

D-ff

SR

(3) Design a Synch. mod-4 counter using SR ff

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

Soln

1) Determining the no. of FF.. = 3 ffs are reqd

2) FF to be used : JK

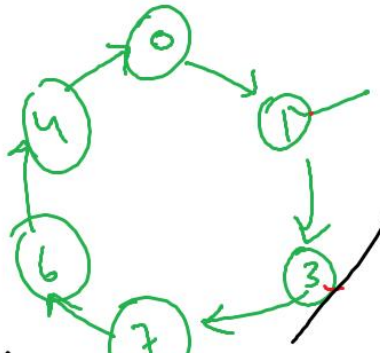
3) Dd Exci table of JK

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

$2 \mid 7$
$2 \mid 3 - 1$
$1 - 1$

4) Dd. Excitn tab / Transn table of given counter

Present state			Next state			JK - flip-flop ips					
Q_A	Q_B	Q_C	$Q_A{}^{t+1}$	$Q_B{}^{t+1}$	$Q_C{}^{t+1}$	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	0	X	X	X	X	X	X	X	X	X
0	1	1	1	1	1	1	X	X	0	X	0
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	1	0	0	X	0	X	1	0	X
1	1	1	1	1	0	X	0	X	0	X	1



J_A, K_A
J_B, K_B
J_C, K_C

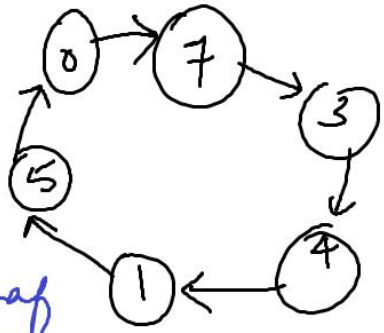
using signal - name

5) K-map - simplic"

6) Implemen" / Logic diagm.

Design a Sync counter using JK to count the following sequence.

7, 3, 4, 1, 5, 0, 7 ...



Soln: (1) 3-ffs (2) JK (3) JK Excitn table

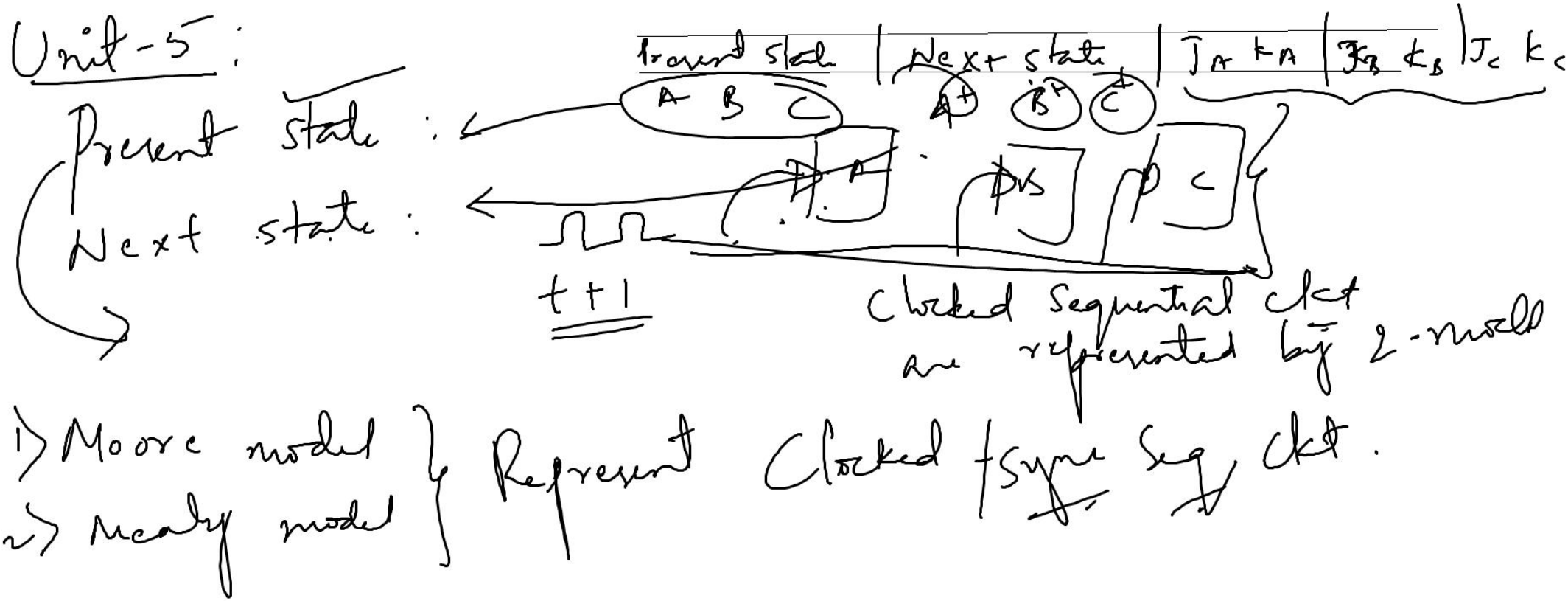
(4) Excitable for counter

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	0	1						
0	1	0	X	X	X	X	X	X	X	X	X
0	1	1	1	0	0						
1	0	0	0	0	1						
1	0	1	0	0	0						
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	0	1	1						

(5) k-map FF-i/ps f^n's

(6) Implementation diagm using signal-names.

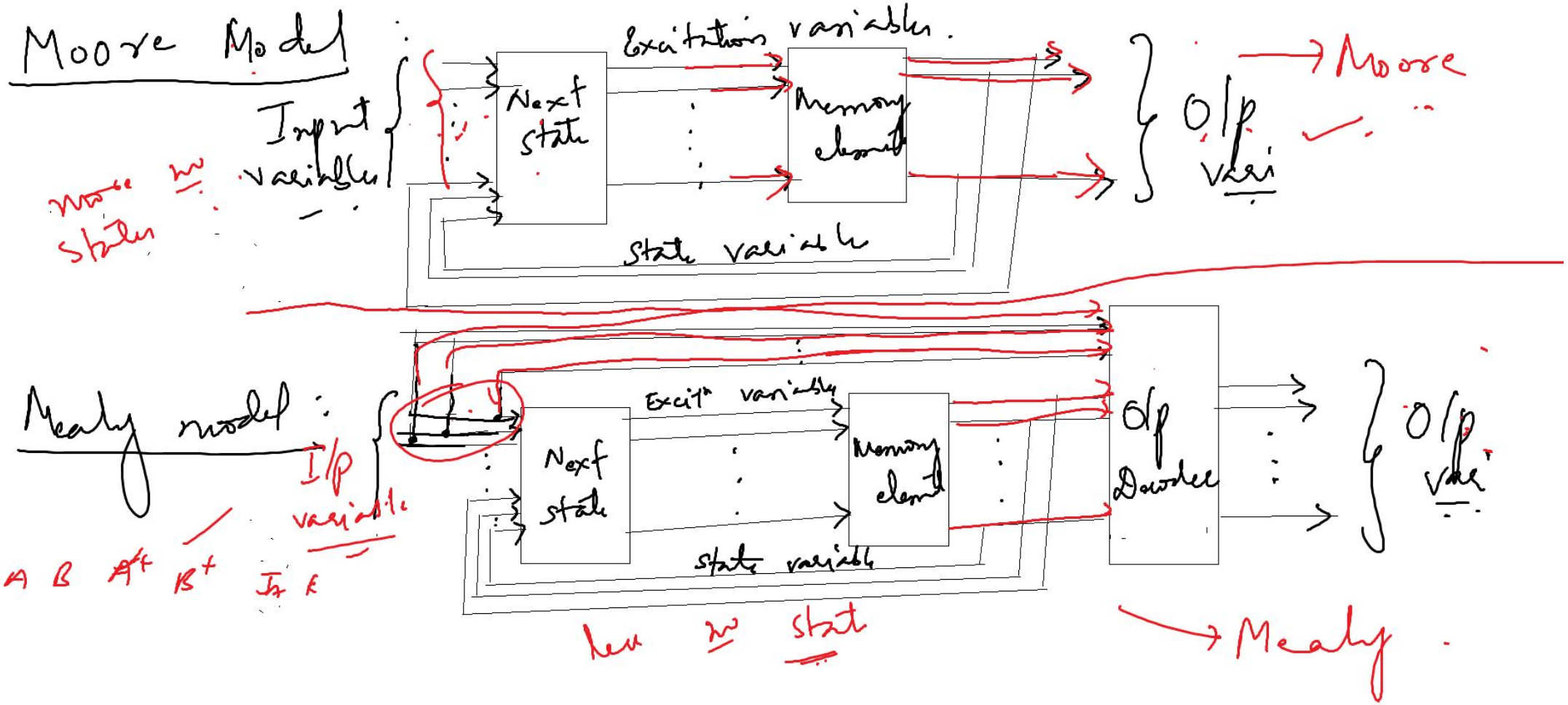
Unit-5:



Clocked Sequential ckt are represented by 2 models

1) Moore model
2) Mealy model

} Represent Clocked Sync Seq. ckt.

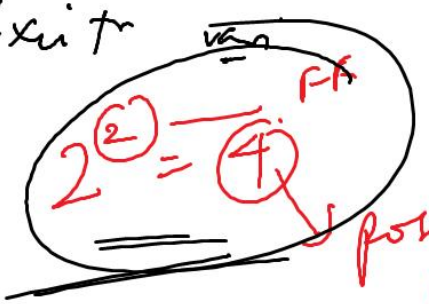

Moore Model
Input variables
Next state
Excitation variables
Memory element
State variable
O/p var
Moore
Mealy model
I/p variable
Next state
Excit" variable
Memory element
State variable
O/p Decoder
O/p var
Mealy

State M/c: I/p vari, O/p vari, state vari, Exitr vari

State & state vari: $2^x = y$

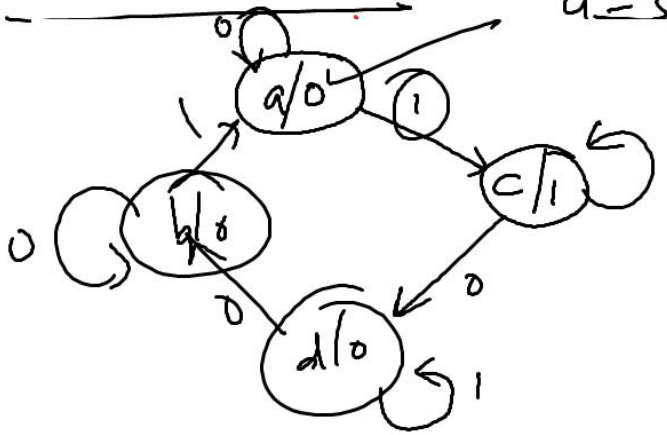
$x \rightarrow$ no. of FF.

$y =$ max no of possible state.



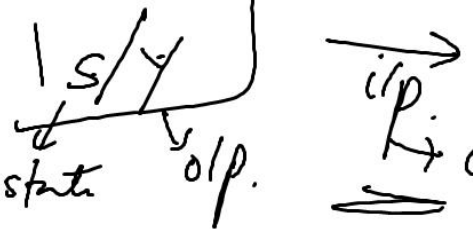
State diag in: Moor model

Nodes: state with o/p.



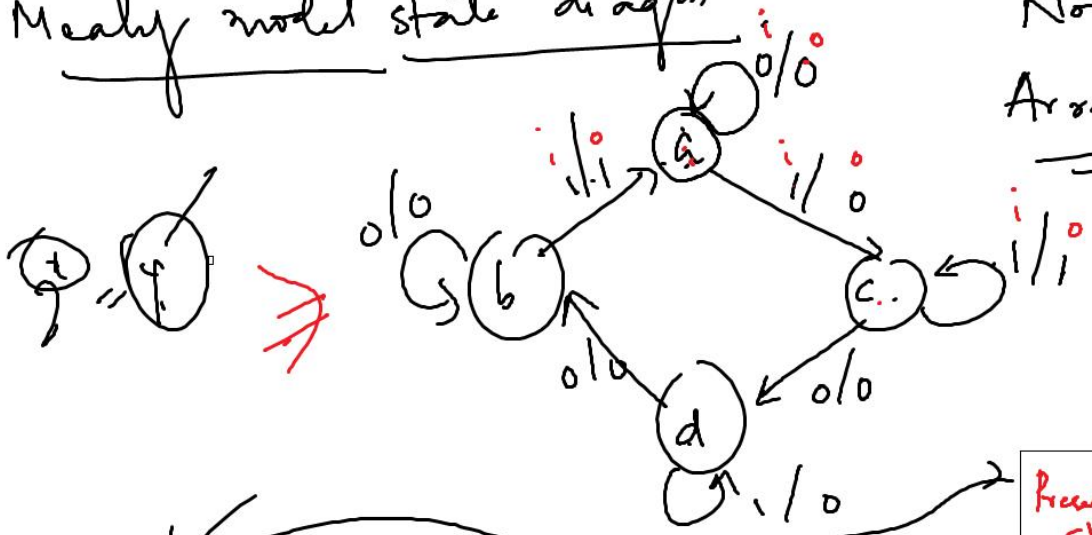
4 states

Each state: Node:



a/

Mealy model state diagram :



Node : State

Arrow : i/o

Moore : S/Y i/p.

Mealy : state i/o

X → i/p.

State table :

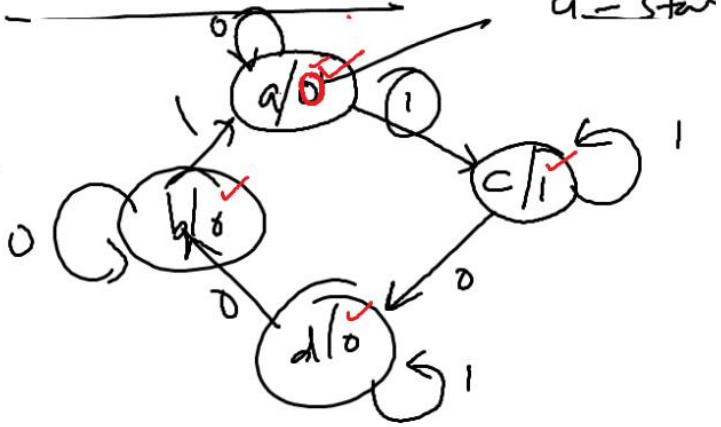
Transition table

Present state A	Present state B	Next state X=0	Next state X=1	output X=0	output X=1
0	0	0	0	1	0
0	1	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1

	Present state	Next state X=0	Next state X=1	output X=0	output X=1
	A B	A B	A B		
0 0 ←	a	a	c	0	0
0 1	b	b	a	0	1
1 0	c	d	c	0	1
1 1	d	b	d	0	0

Mealy model state diagram

Moore model.



4 - state

State table

P.S	N.S x=0	N.S x=1	o/p
A B	AB	AB	y
a	a	c	0
b	b	a	0
c	d	c	1
d	b	d	0

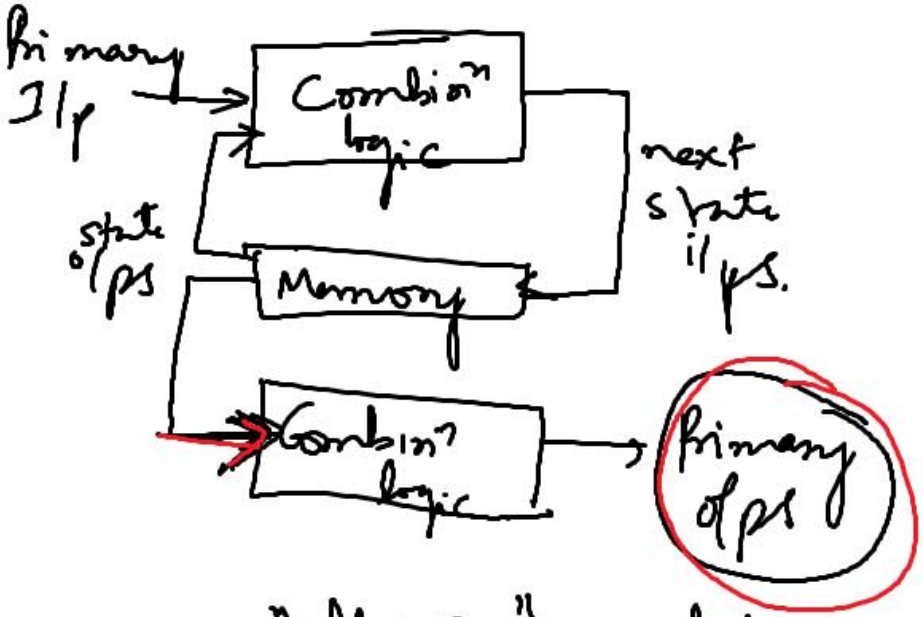
Right

left

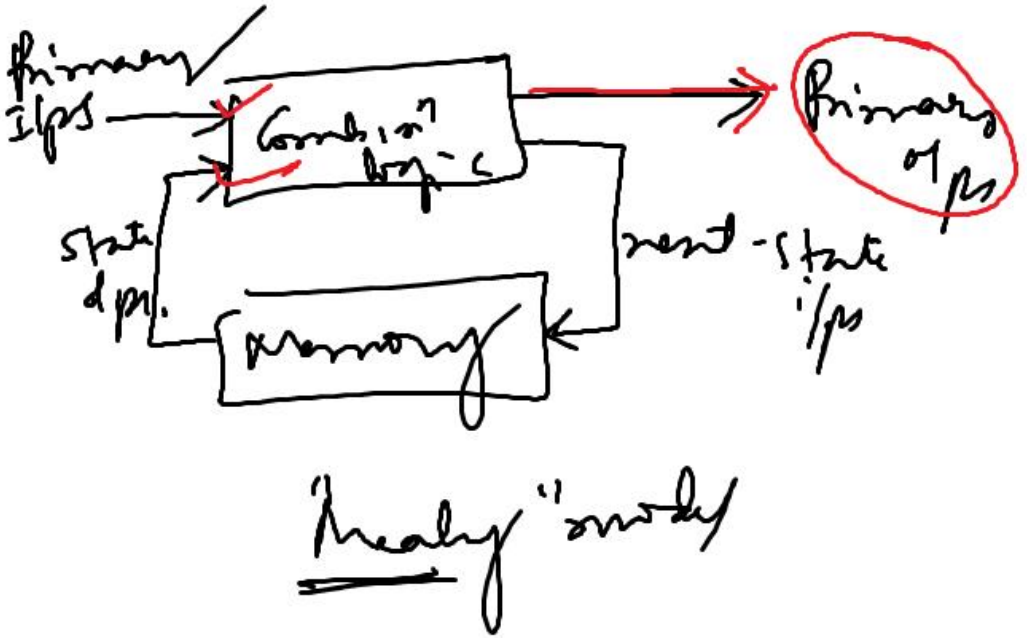
Sequence Detector :

Moore } mod
Mealy }

Moore & Mealy

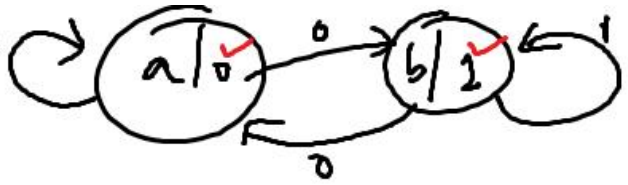


"Moore" model.

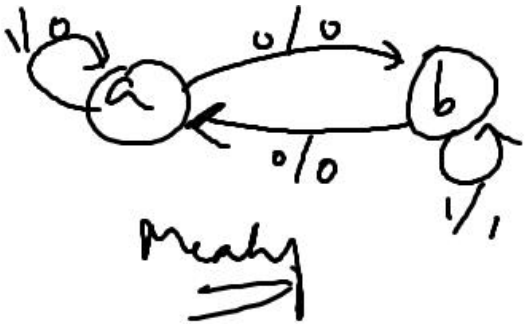


"Mealy" model

State trans^{n} diag^{m} :



Moore



Mealy

Eg: Sequence detector

Binary Seqn $\overleftarrow{0\,1\,1}$ ← left

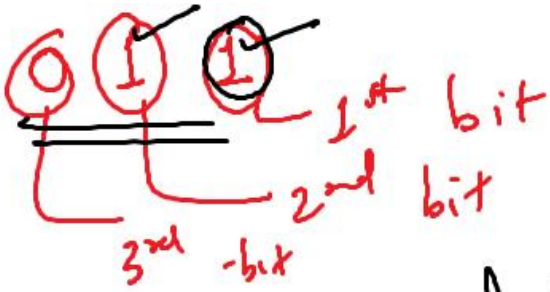
$x = y$

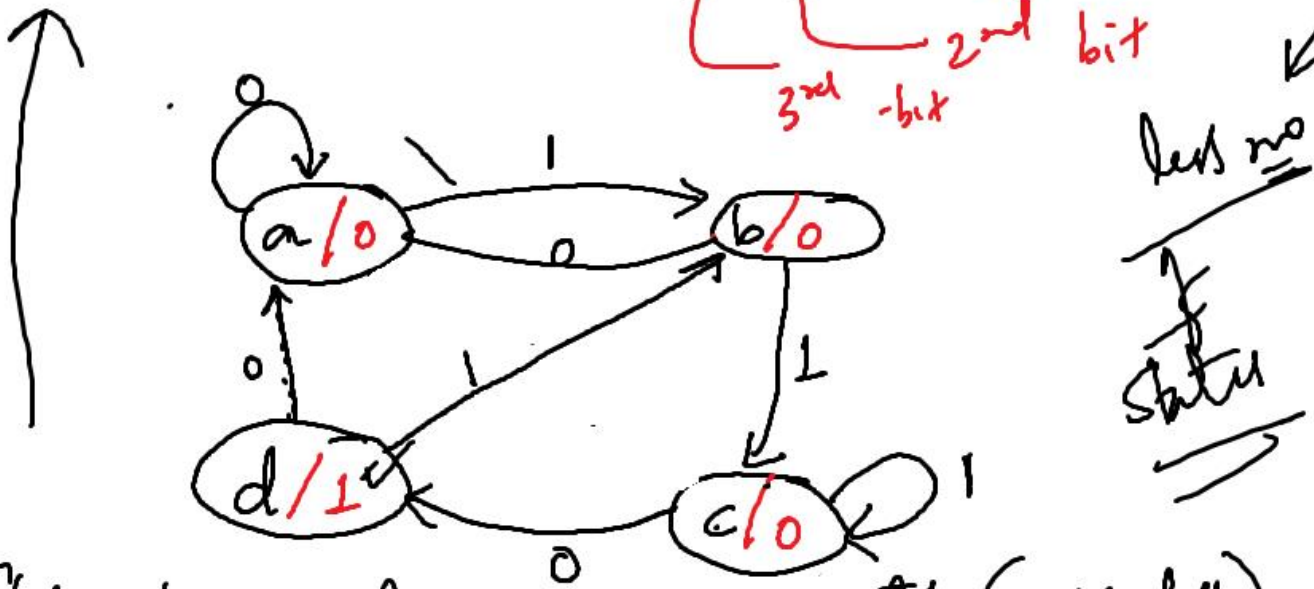
— LHS RHS —

right $x = y \leftarrow$ left

Design a Seq detector that receives a binary data stream @ its i/p-X, & signals when a combin^{n} 0 1 1 arrives @ the i/p by making its o/p Y=HIGH, which otherwise remains (Y=0) LOW. Consider the data is coming from LEFT.
i.e., 1st bit to be detected is 1, 2nd .... 1, 3rd .... 0 from i/p seq.

Moore model :

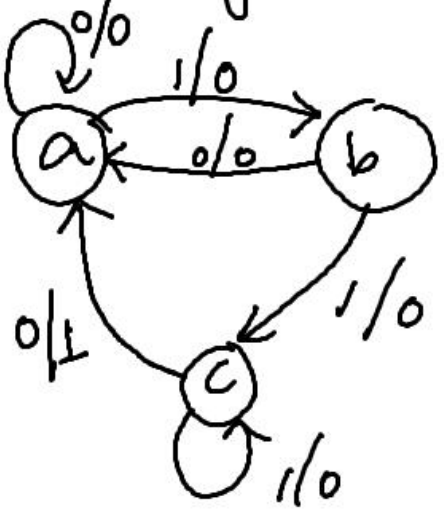




Next no

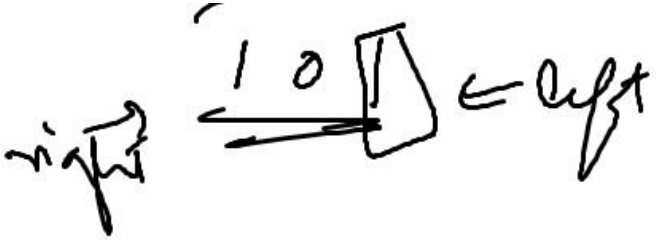
Present

Mealy model

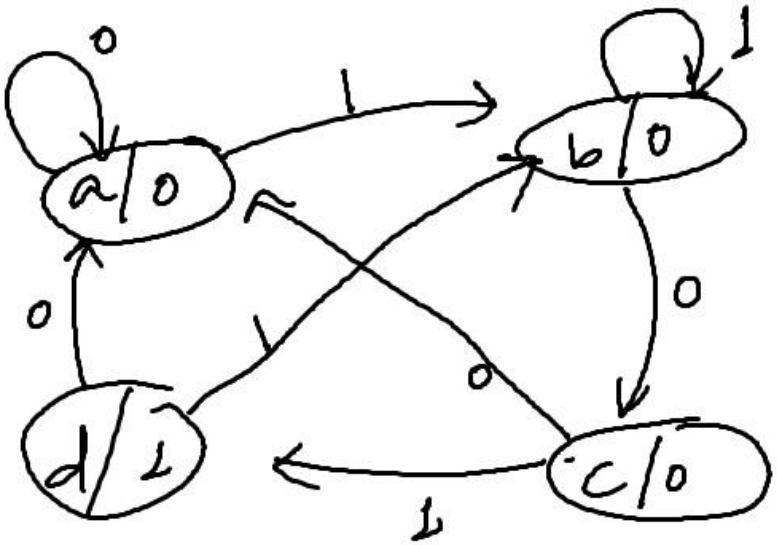


State trans^{n} diagr^{m} of Seq detector (0 1 ← left)

Mealy model



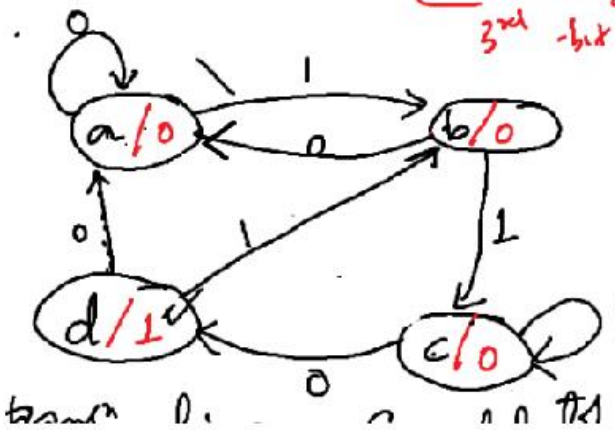
(a)



1 0 1 0 1 1 1 0 0

→ 0 1 0 ← left

State transition diagram:



Present state B_n	Present state A_n	I/p X	Next state B_{n+1}	Next state A_{n+1}	O/p Y	J_B	K_B	J_A	K_A
a 0	0	0	0	0	0	0	X	0	X
	0	1	0	1	0	0	X	1	X
b 0	1	0	0	0	0	0	X	X	1
	1	1	1	0	0	1	X	X	1
c 1	0	0	1	1	0	X	0	1	X
	0	1	1	0	0	X	0	0	X
d 1	1	0	0	0	1	X	1	X	1
	1	1	0	1	1	X	1	X	0

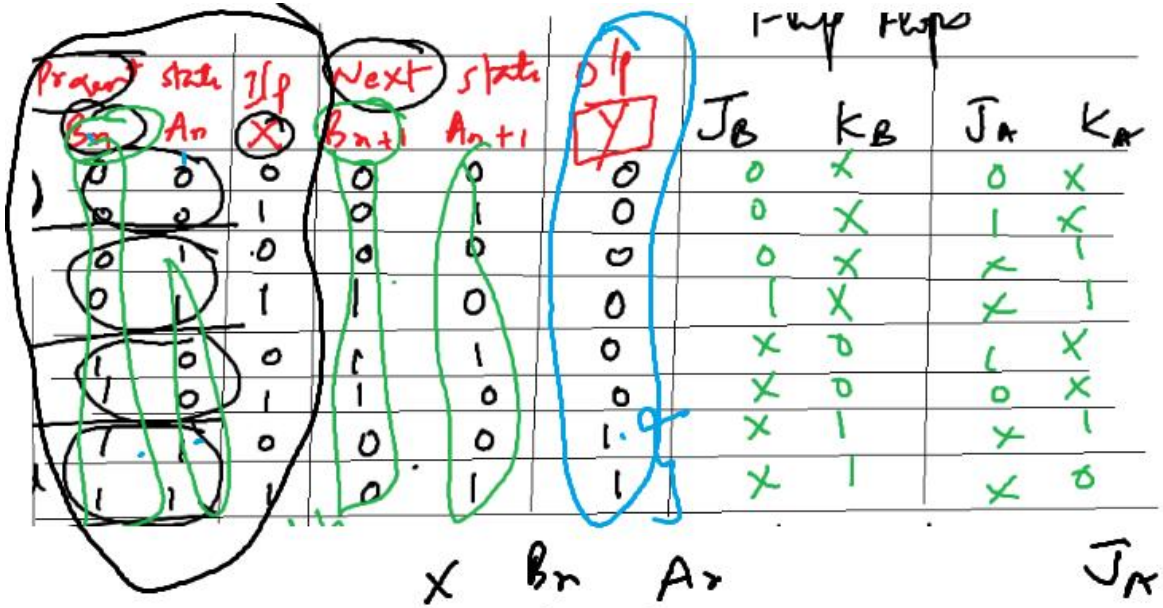
Flip Flops

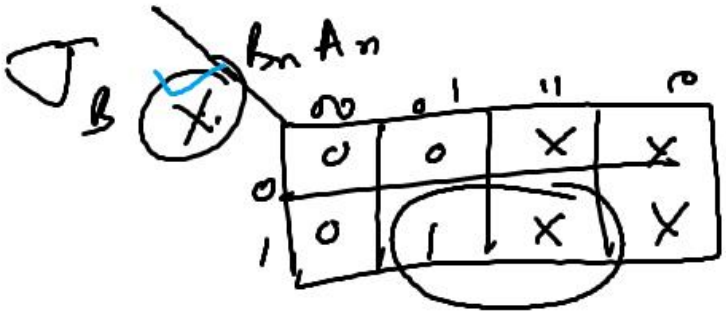
a — 0 a
b — 0 1 ✓
c — 1 0 ✓
d — 1 1 ✓

State assignment table

Excita table:

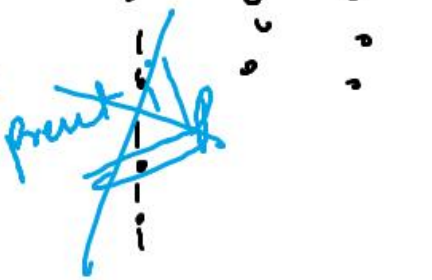
Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0





$J_B = X A_n$

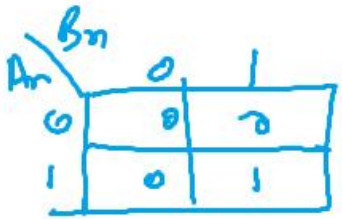
Moore



$J_A = \bar{X} B_n + X \bar{B}_n$ $K_B = A_n$

$K_A = \bar{X} + \bar{B}_n$

$Y_n = A_n . B_n$



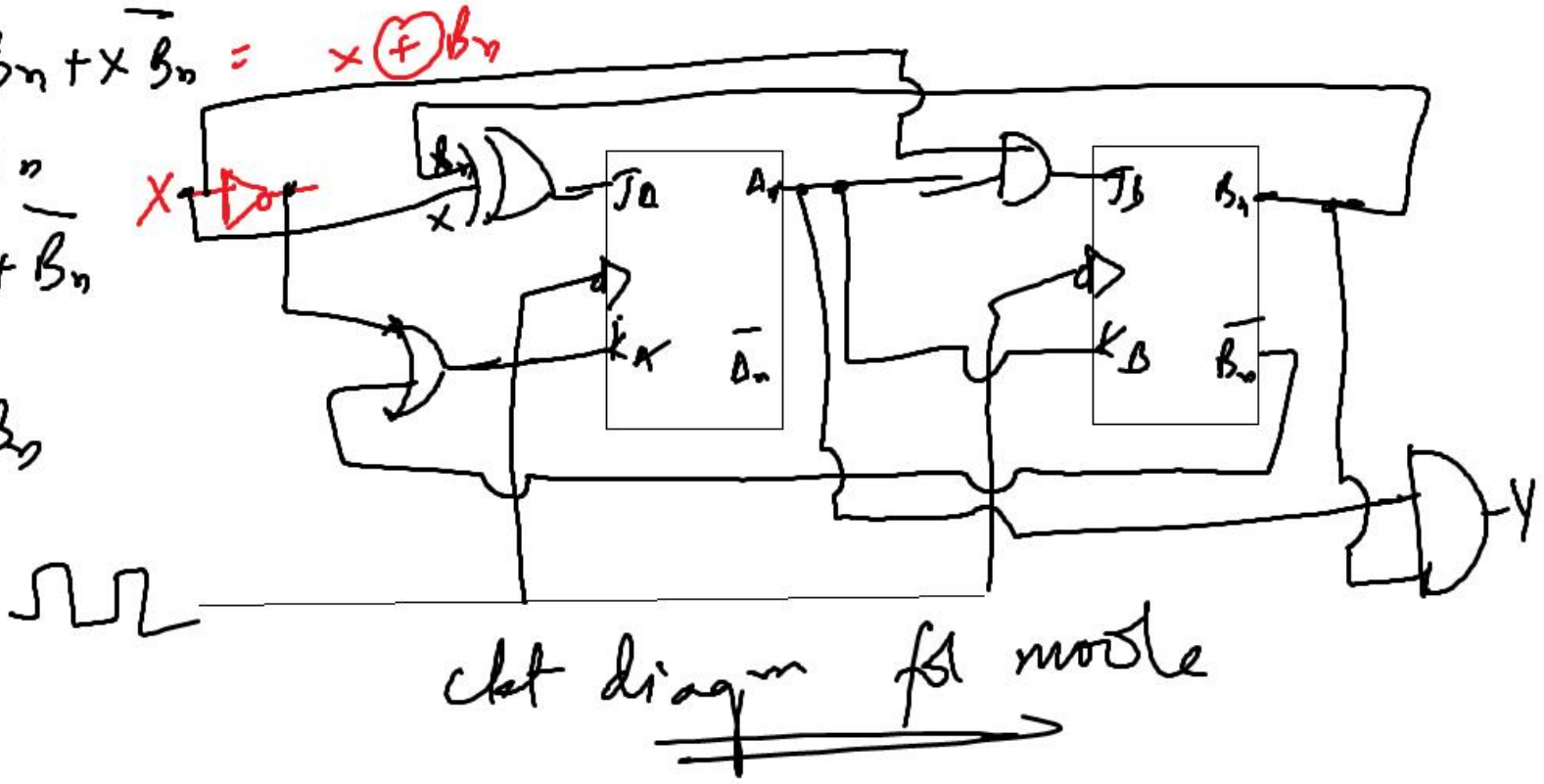
Moore: $J_A = \bar{x} B_n + x \bar{B}_n = x \oplus B_n$

$J_B = x \cdot A_n$

$K_A = \bar{x} + \bar{B}_n$

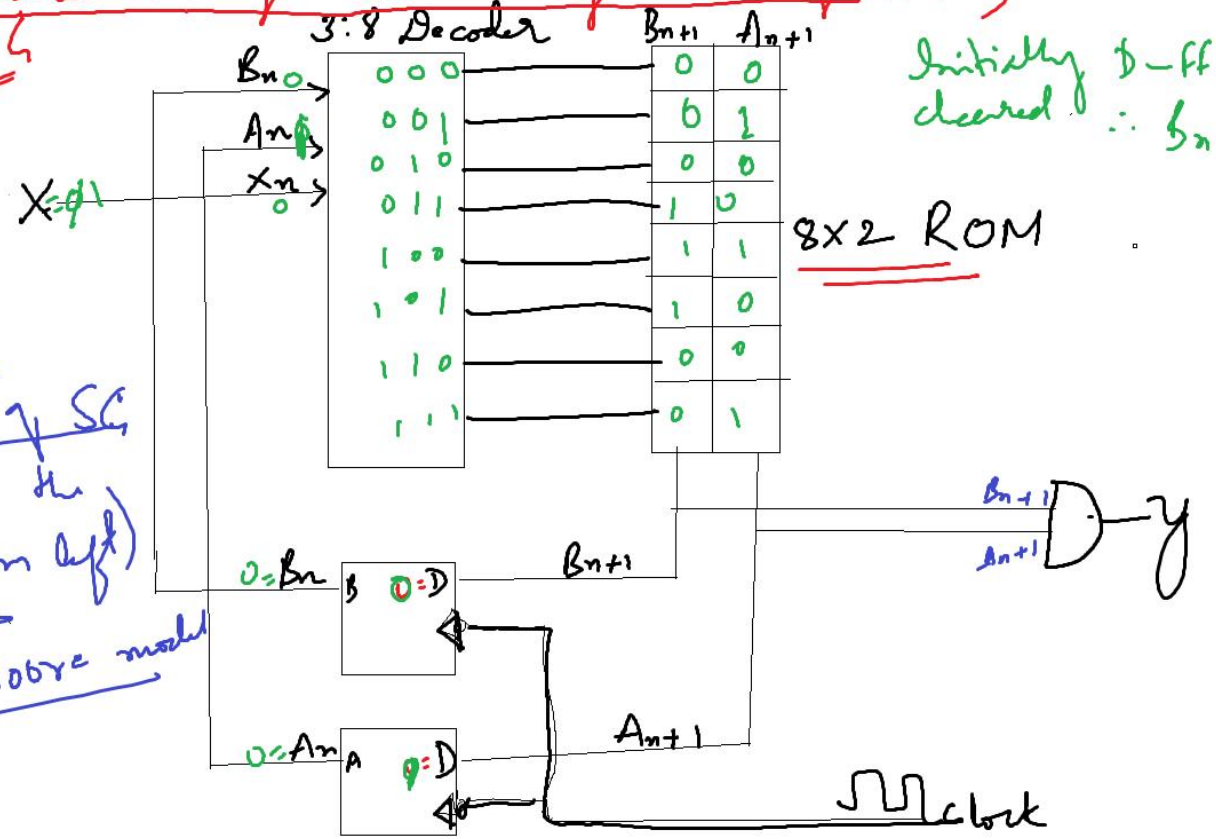
$K_B = A_n$

$Y = A_n \cdot B_n$



ckt diagram for moore

Implementation using Read Only Memory (ROM)



Initially D-ffs are cleared ∴ $B_n = 0$ & $A_n = 0$ $B_n = 0$

$A_n = 0$

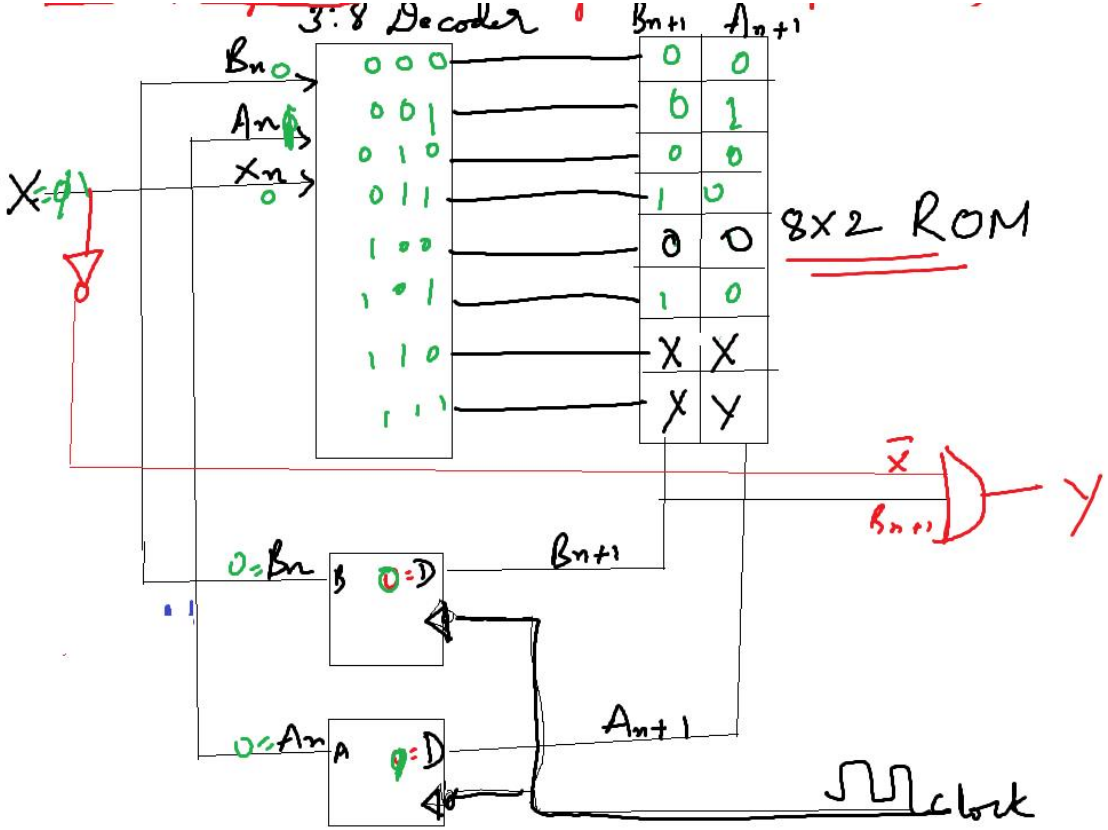
X = 0 (or) 1

BAX = 000 is selected

BAX =

Seq detected 011←

Implementing SC that detects the seq 011 (from left) using Moore model



Mealy model: 3 - state

pre i/p + present state

6 - combinations

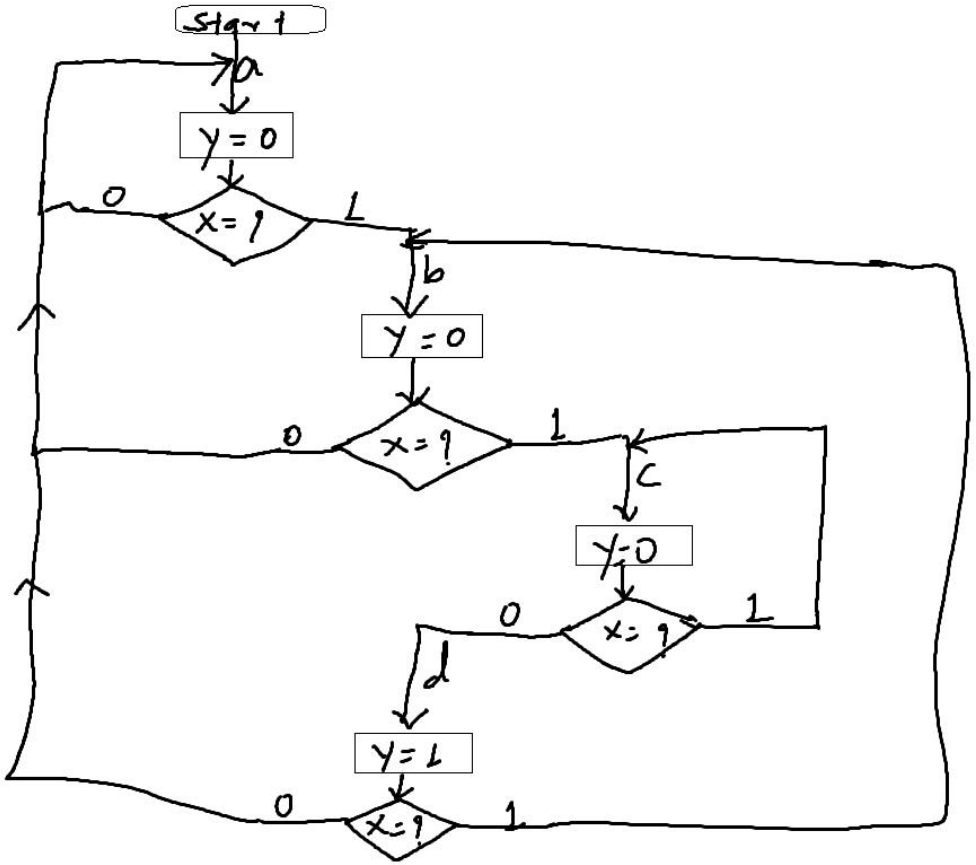
$Y = \bar{X} \cdot B_n$

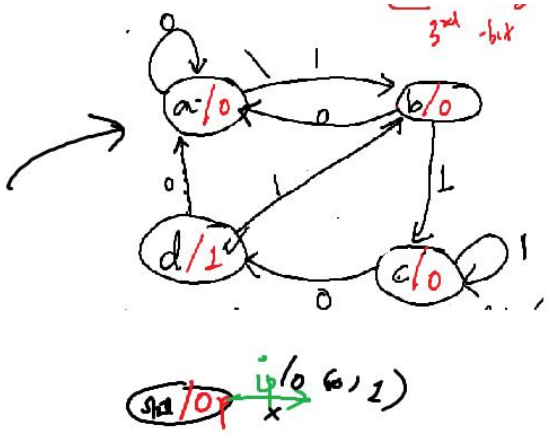
ROM - based implementation

sequence - detector (011 left)

using Mealy - model

ASM: Algorithmic State Machine :

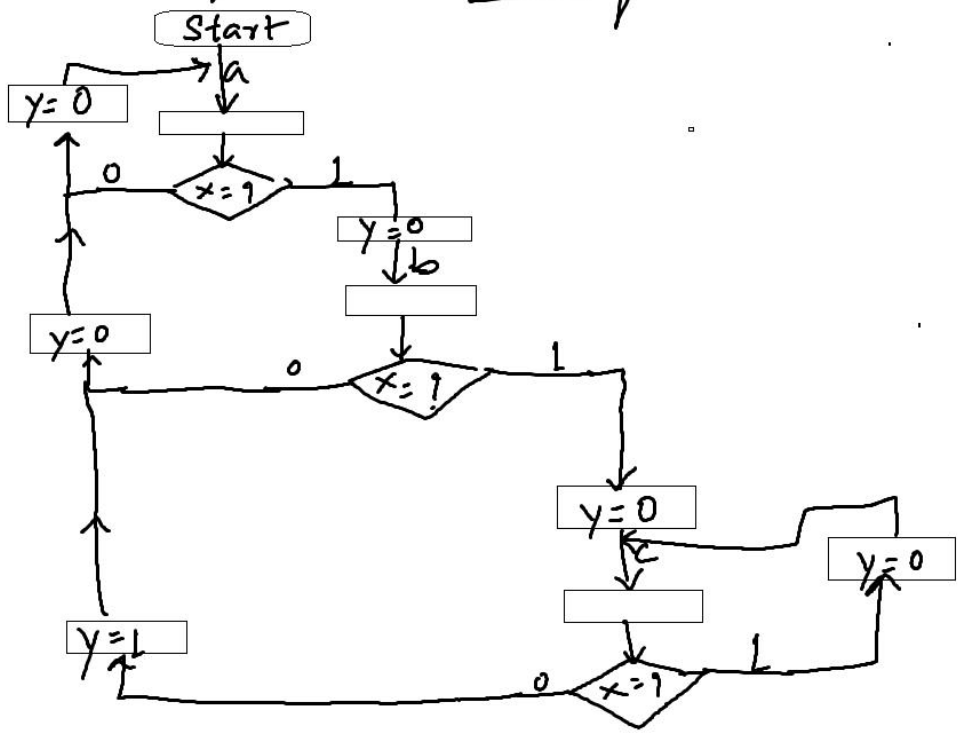




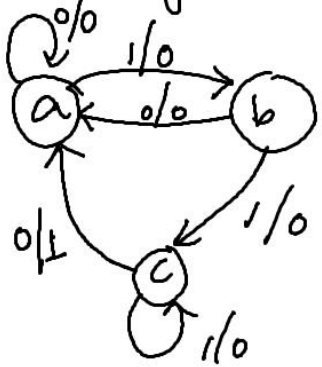
ASM chart of seq. detector

problem: Moore model

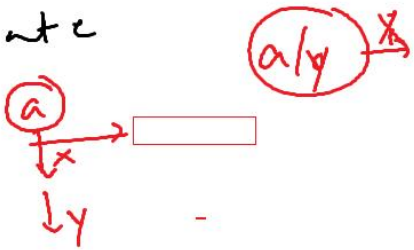
ASM chart of seq detector problem : Mealy model



Mealy model

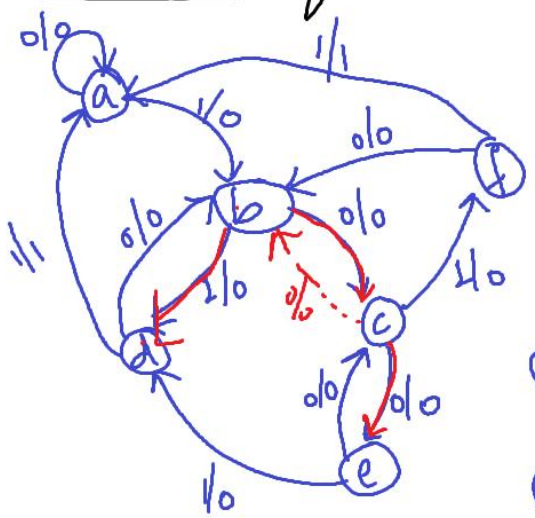


State



State Reduction Technique: Row-elimination method (Mealy model)

STD using Mealy:



S1:

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√	a	0
f	b	a	0	1

remove row → (√e)

next state ≡ present state ⇒ o/p ≡ state are eq.

original table

b ≡ e, remove 4th row

d ≡ f, remove 5th row

S4:

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

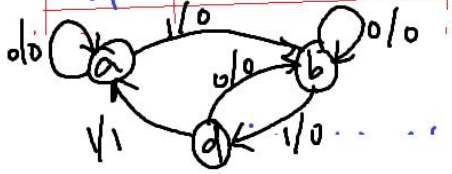
b ≡ c remove

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	b	d	0	0
d	b	a	0	1

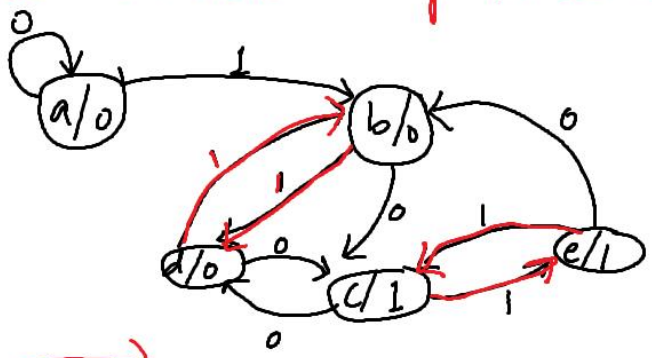
S2 ≡ S3:

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

Final Simplified STD ⇒



Reduce State-trans^n diag^m (Moore model) : row-elim^n meth.



(a)

Present State	Next state x=0	Next state x=1	Present o/p
→ a	a	b	0
→ √b	c ✓	d ✓	0 ✓
c	d	e	1
√d	c ✓	b	0 ✓
e	b	c	1

Original tab b

∴ b = d ∵ b on 1 ⇒ next state d
d on 1 ⇒ . . . b
} o/p = 0

(b)

Present State	Next state x=0	Next state x=1	Present o/p
→ a	a	b	0
→ b	c ✓	d ✓	0 ✓
√c	b	e	1 ✓
√e	b	c	1 ✓

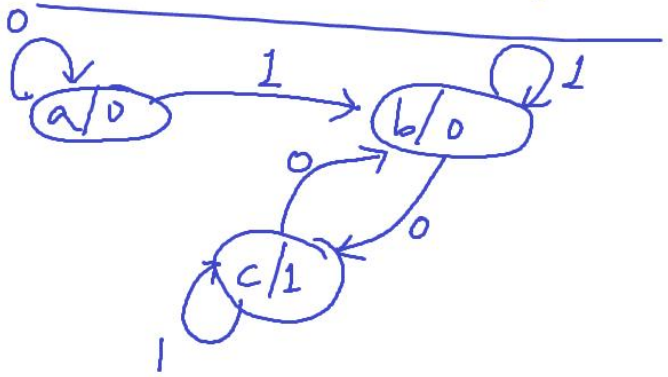
∴ c = e ,
∵ c on 1, NS = e
e on 1, NS = c
} o/p = 1

c = ✗

(c)

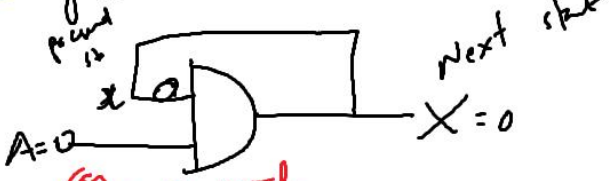
Present State	Next state x=0	Next state x=1	Present o/p
→ a	a	b	0
→ b	c ✓	d ✓	0 ✓
√c	b	c	1 ✓

Final reduced STD



Asynchronous Seq ckt: Analysis of Asynch. Seq ckt

① AND gate



(a) AND gate

o/p $X = 1$, when $A = 1$, $x = 1$

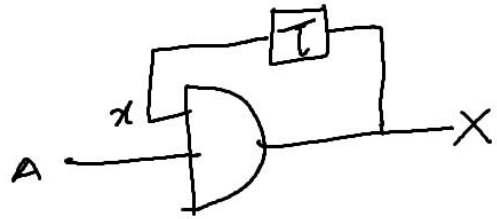
$X = 0$ when $A = 0$, $x = 0$

$x = 0$ $A = 0$, $X = 1$

$x = 0$ $A = 0$, $X = 0$

Unstable - stable state

$0 \rightarrow (0)$



(b) AND gate with propagation delay.

deadlock

$X = x.A$

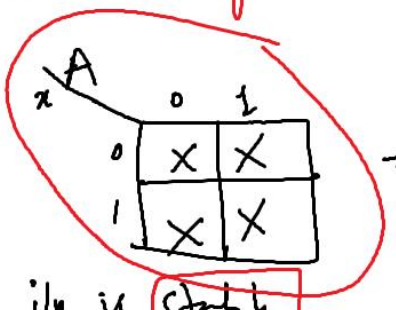
$\Downarrow$

$X = x$, the i/p is stable

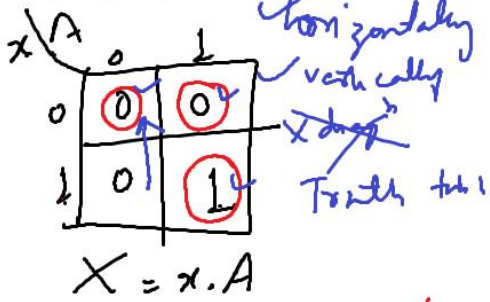
$X \neq x$, ... unstable

$1 \rightarrow (1)$

$0 \rightarrow (1)$ oscillation



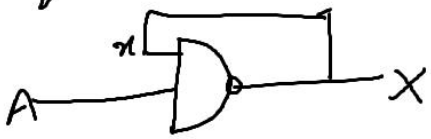
$\Rightarrow$



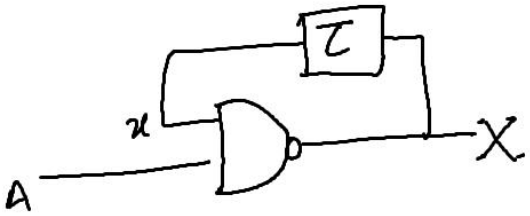
stable - enable 0^{th}

$1 \rightarrow 0$ $1 \rightarrow 0$

Analysis of NAND :



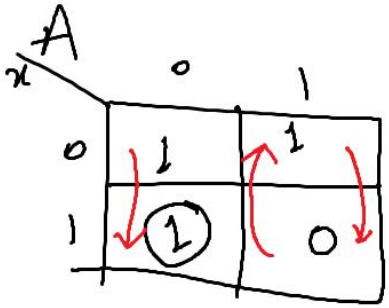
(a) NAND gate :



(b) NAND gate with propogation delay

		NAND
0	0	1
0	1	1
1	0	1
1	1	0

(c) Truth table



make transⁿ from i) Unstable to stable

1 → (1) (a) 0 → (0)

horizontally / vertically

2 Unstable : oscillations

(0 - 1) & (1 - 0)