DIGITAL LOGIC

[Min / 3 , Avg / 5.5 , Max / 8]

→ Logic Functions.

→ Minimization.

→ Design and Synthesis of Combinational Circuits.

LOGIC FUNCTIONS

* Basic Properties of Switching Algebra:

- Boolean var. → 0, 1
- Operators
  - → OR (+)
  - → AND (.)
  - → NOT (¯)

- AND

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

- OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

binary

- NOT

x	$\bar{x}$
0	1
1	0

→ unary

* Basic Properties

1. Idempotency

$x \cdot x = x$
$x + x = x$
$x + 1 = 1$, $x + 0 = x$
$x \cdot 0 = 0$, $x \cdot 1 = x$

2. Commutativity

$x + y = y + x$
$x \cdot y = y \cdot x$

3. Associativity [LA = RA]

LA RA

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

4. Complementation

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

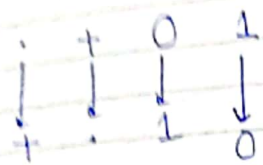
5. Distributivity

$x \cdot (y + z) = xy + xz$
$x + yz = (x + y) \cdot (x + z)$

. distributive over +

Substitute x = 0, 1 for Fast checking

* If we interchange :



(all from complementation).

Dual : Principle of duality.

SA
* SE → helps in designing electronic circuit.

eg: $x + \bar{x}yz + x\bar{z}$

- 6 literals
- 3 terms

* Properties for simplifying Switching Exp.

1 Absorption :

→ $x + xy = x$
$x \cdot (x + y) = x$

→ $x + \bar{x}y = x + y$
$x \cdot (\bar{x} + y) = xy$ ∵ $x\bar{x} = 0$

* Consensus Theorem :

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

↳ redundant (can remove)

$a + c = a + b$
$a \cdot c = a \cdot b$ ← Inverse & cancell. not allowed in SA.

* De-Morgan's Law [Complementation]

→ $\overline{x \cdot y} = \bar{x} + \bar{y}$

→ $\overline{x + y} = \bar{x} \cdot \bar{y}$

eg. $f(a, b, c \cdots z, 0, 1 \cdots, +)$

comp. → $\bar{f}(\bar{a}, \bar{b}, \bar{c} \cdots \bar{z}, 1, 0, +, \cdot)$

dual → $f_d(a, b, c \cdots z, 1, 0, +, \cdot)$

∴ $\bar{f} \neq f_d$

$\bar{f}$ ⟶ f_d
complement the literals.

eg: $\overline{x+y} = xy$.

$\bar{f}$ ↓

xy $(\bar{f})$

$\bar{x}\cdot\bar{y}$ (f_d) ↑ dual.

$f = \bar{x}+\bar{y}$ VI

· is distr. over + ⟶ Maths / SA
+ is distr. over · ⟶ SA

* Switching Functions ← SE ← SA.

- SE act as functions.

i/p ⟶ mapped ⟶ o/p.

eg: $a+b=f$

a	b	f
0	0	0
0	1	1
1	0	1
1	1	1

f ⟶ boolean funct^n.

$f(a,b) = a+b$ → 2 var. function.

- Every function can be represented as canonical form.

* Canonical SOP

- Min Term: Product Term which contains each of n var. as factors either in com/uncom. form.
- Sum of all minT = canonical SOP / disjunctive NF.
  ↓ 1
  0

· $a+bc+ac$ ⟶ SOP

$abc + \bar{a}bc$ ⟶ canonical SOP.

$f = \Sigma(1,2,4,6)$ → $\Sigma(0,1,2,3)$: AND.
↳ compact rep.

* Canonical POS

→ 0.0

· Max Term: Sum term which contains each of n var.

Product of all maxT = can. POS / conjunctive NF.

eg: $f = (a+b+c)\cdot(a+\bar{b}+\bar{c})$

$f = \Pi(000, 011)$

$\therefore \boxed{\Pi(0,3)}$ → compact rep.

$\therefore$ f_1 (funct^n in 3 var)

$\Sigma(2,3,5,6,7)$ $\quad$ $\Pi(0,1,4)$.

∴ [Canonical SOP & can. POS are 2 rep. of Same Funct^n]. VI

* Standard circuit → all terms.

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

$\Rightarrow \bar{x}y(z+\bar{z}) + (x+\bar{x})(y+\bar{y})z + xyz$.

* Functional Properties

→ Many exp. represent same funct^n but can. SOP/POS: unique.

→ 2 funct^{n}s: equivalent, if [SOP/POS terms are identical] VII

* Number of funct^{n}s

Q If n boolean variables, how many boolean funct^{n}s?

2^n raws → 0 / 1.

n: 2 2 2 2

$\therefore 2^{2^n}$ funct^{n}s

⇒ K-ary variables (each var = K val.)

How many n-ary funct^{n}s? (each fun. → n val.)

Soln:

n: K K K.

$\therefore K^n$ raws/inputs.

K^n $\begin{matrix}0\\1\\\vdots\\m\end{matrix}$

$\therefore \boxed{m^{K^n}}$ functⁿs ✓

Q How many boolean functⁿs are possible with 3 boolean var. such that there are 3 minTerms exactly?

raws $= 2^3 = 8$. ∵ Total minT = 8,

$\therefore {}^8C_3$. Ans

· atmost 3 $= {}^8C_0 + {}^8C_1 + {}^8C_2 + {}^8C_3$

* K variables (boolean var.) 'm' miniterm functⁿs, How many?

$\boxed{{}^{2^K}C_m}$

Select m raws out of 2^K raws.

* Neutral Functⁿ VI

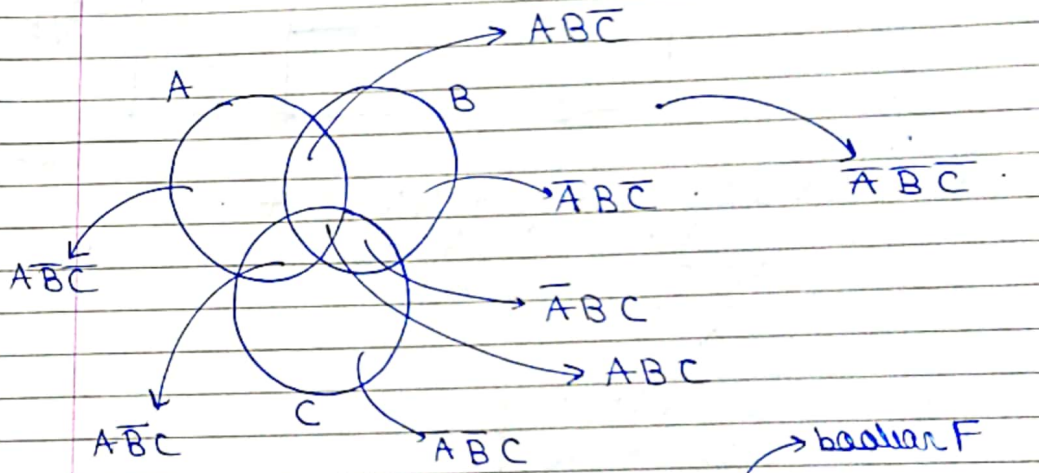
· No. of minTerms = no. of maxTerms

if n variables (BV)

$\boxed{{}^{2^n}C_{2^n-1}}$

→ No. of NF possible with n BV.

* Venn Diag. Representation



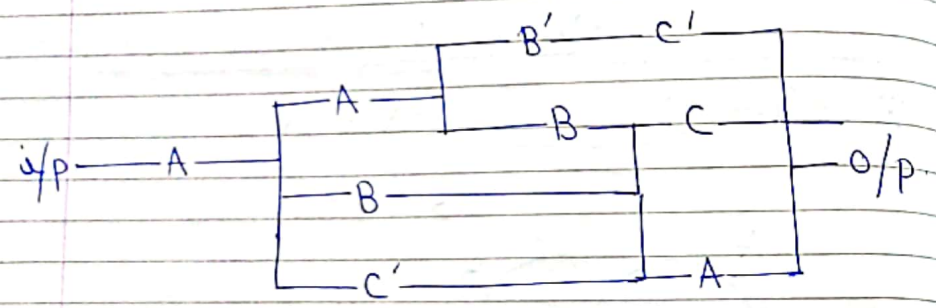
∴ No. of i/p combⁿ = 8 (raws) ∵ 3 var. F → boolean F

n V → 2^n Regions.

∴ [2^{2^n} ways to Shade the VD]

→ Only BF can be rep. → Shade
                      → No Shade

* Contact Repr.



f = (AAB'C' + AABC + AABA + ABC + ABA + AC'C + AC'A)
                                  X

Series → AND
Parallel → OR

* Nested Funct^n:

f(f(x+y, y), z)

* NAND → Not AND. (↑)

A	B	A ↑ B
0	0	1
0	1	1
1	0	1
1	1	0

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

1. $A \uparrow A \neq A$
2. $A \uparrow B = B \uparrow A$ (Is commutative)
3. $A \uparrow (B \uparrow C) \neq (A \uparrow B) \uparrow C$ (not associati

* NOR : not OR (↓)

A	B	A ↓ B
0	0	1
0	1	0
1	0	0
1	1	0

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

1. $A \downarrow A \neq A$.
2. $A \downarrow B = B \downarrow A$. (Is commutative)

3. $A \downarrow (B \downarrow C) \neq (A \downarrow B) \downarrow C$.

→ Not associative.

* EX-OR

- exclusive OR
- modulo 2 sum.

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

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

1. $A \oplus A \neq A$. $\quad A\overline{A} + \overline{A}A \Rightarrow 0$.
2. $A \oplus B = B \oplus A$
3. $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

(Both associative & commutative)

→ Subst. A = 0, 1 & check.

$\overline{B \oplus C} = B \odot C$.

1. $(\overline{B \oplus C}) + (B \oplus C) \cdot 0$

$\Rightarrow \overline{B \oplus C} = B \odot C$ (LHS)

* EX-NOR

A	B	$A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

when i/p are same.

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

→ 1 bit comparator.

$$A \odot B = \overline{A}\,\overline{B} + AB$$

1. $A \odot A \neq A$. (identity rule).
2. $A \odot B = B \odot A$ (commutative).
3. $A \odot (B \odot C) = (A \odot B) \odot C$.

VI

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

XOR $\oplus$ → 1, for odd no. of 1s

X-NOR $\odot$ → 1, for even no. of 0s

VI

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

∴ When even inputs:

XOR, XNOR are complements.

When odd inputs:

XOR = XNOR

eg: $A \oplus B \oplus C \oplus D \oplus E = A \odot B \odot C \odot D \odot E$ ← add

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

* 4V K-Map

EXOR, EXNOR → diagonal adjacies.

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

← cyclic code

VI

1 present in odd no of 1s = EXOR.

No. of minT = no. of maxT.

minT filled where no. of 1s odd = EXOR

minT filled where no. of 0s even = EXNOR

* Min no. of NAND/NOR to design:

	NAND	NOR
EXOR	4	5
EXNOR	5	4

* Functional Completeness

- Set of op. is FC, if & only if every SF can be expressed by means of op in it

$\{+, \cdot, -\} \rightarrow$ FC → De-Morgan's law.

$\{+, -\} \rightarrow$ FC

$\{\cdot, -\} \rightarrow$ FC.

→ Set = FC, if we can derive a set which is already FC.

eg: $f(A, B) = \overline{A} + B$ → PFC.

$f(A, A) = \overline{A} + A = 1$

$f(B, B) = \overline{B} + B = 1$.

∴ Kill B.

$f(A, 0) = \overline{A}$ ($^-$ is derived).

$\therefore f(f(A,0),B) = \overline{\overline{A}+B} = A+B$

{+ is derived}

F → NFC.

FC

PFC (take support from 0/1)

* Complement can be extracted from the eqⁿ, not derived.

∴ EXOR ⟶ NFC (eq in NOTES)

$\{\oplus, 1\}$, $\{\odot, 0\}$ VI

$\oplus$ → NOT

$\odot$ → NOT.

∴ (By elim. a var, we can get NOT, ab. not AND/OR)

VI

$$x \oplus y = \overline{x \odot y} = \overline{x} \odot y = x \odot \overline{y} = \overline{x} \oplus \overline{y}$$

$$x \odot y = \overline{x \oplus y} = \overline{x} \oplus y = x \oplus \overline{y} = \overline{x} \odot \overline{y}$$

* Self Dual Functions

$f = f_d$

eg: $f(A,B,C) = AB + BC + CA$

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

$\Rightarrow AB + BC + CA$

∴ A BF is Self Dual if:

(i) It is neutral function

(ii) F does not contain 2 Mutually Excl. Terms

* If n var. : 2^n minTerms (not met: 001, 110)

No. of Pairs = 2^{n-1} (0,7); (1,6); (2,5); (3,4)

⇒ 2 choices in each pair.

∴ $2^{2^{n-1}}$ → Total no. of SDF with n BV.

$n=3$: $2^{2^2} = 16$

VI

$^{2^n}C_{2^{n-1}}$: NF

V

$2^{2^{n-1}}$: SD

AllF.

* Say Dual Functns are closed under complementation.

* Electronic Gates

- 2 Volts {H, L}

1 → HV	1 → LV
0 → LV	0 → HV
Positive Logic System	Negative logic System.

* AND ⟶ OR [duals].
(PLS) (NLS)

* PLS & NLS are dual of each other.

* We follow PLS everywhere.

XOR = 0, No. of 1s are even.
XOR = 1, No. of 1s are odd
XNOR = 0, No. of 0s are odd
XNOR = 1, No. of 0s are even

* MINIMIZATION of BF

- SF can usually be represented by using a no. of exp

- [Min no of terms in SOP exp. provided there is no other such exp with the same no of terms & fewer literals]
↳ minimal exp.

- Irredundant / Irreducible: cant minimise further.

* An irredundant exp is not necessarily minimal, nor minimal exp is always unique.

SE ⟶ 1 minimal SE (Not True).

* K-Map:

- modified form of TT.

[n Var. KMap → 2^n mint : 2^n cells]

xy / z	00	01	11	10
0	0	2	6	4
1	1	3	7	5

→ cyclic code.
[Gray Code]

* Cycle code Headings.

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

→ 4 var K Map.
→ decimal Values.

- Coll. of 2^m cells → SC [Subcube] if each cell is adjacent to m other cells.
- Each SC → (n−m) literals

· n Var K-Map.

yz \ wx	00	01	11	10
00	0	4		8
01	1	5	1	9
11	3	7		11
10	2	6		10

→ $\Sigma(1,5,9,13)$.

→ 2^2.

→ $\bar{y}z$.

4 − 2 = 2 literals.

SOP

[No. of Product Terms = No. of Sub Cubes
No. of literals/Term = Size of SC (n−m)]

VI

* Covering Functions

$g \longrightarrow f$.

- f covers g.
- when g = 1, f also 1
- f contains all mT present in g.

	g	f
0	0	1
1	1	1
2	0	0
3	0	0

g → f

- g : x MinTerms, g is funct^n of n var.

[No. of covering Functions = $2^{2^n - x}$]

eg → (See that eg in Pg 78)

If f covers g.

[g → f]

· g is dominated by f.
· from f, we can derive g.

→ g is an implicant of f

(when g = 1, f = 1)

VI

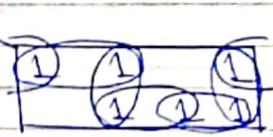
eg: $f(ab+c) = ab + c$
$(f_1)\ (f_2)$

$[f_1 \to f ;\ f_2 \to f]$

- Every SC is an implicant of F.
  - I.

VI

I → SC
PI → can be entirely part of another SC.
EPI → if it covers atleast 1 mT of f which is not covered by any other PI.

* 
  - 6 PI
  - Zero EPI
  - Cyclic K-Map

Pg 81: How to find EPI / PI?

* If all 1s are covered by the EPIs, then its an unique minimal Exp.

VI

Any PI → 2 rep of the same function.

- Its not like that SOP contains more literals than POS / vice-versa. (depends on the funct^n → both popular)

* Don't Cares
  - Some i/ps never occur
  - Some funct^{n}s are not defined for some i/p combinations.
    - unspecified function.
  - Value of f for DC = ϕ. {0, 1}.
  - if, K − DC combinations : f.

f represent → class of 2^K funct^n

* Size of SC ↑ = minimization better.

- our task is to choose a funct^n having minimal rep. out of these 2^K funct^n for minimization.

[See Pg 93 → 2 eqs]

→ See Pg 97 ⇒ PI Chart eg

* [EPIs couldn't cover all 18]
  ↓ we didn't get
  a unique ME

→ Solve Pg 98. (PI chart → Cyclic Func.)

* If function has minT & mutually exclusive term, it can be made into Self Dual.
  → 2^{n-1}

3 ⟶ Sum in Pair : 7	
4 ⟶ Sum in Pair : 15	

PI − EPI = RPI

* Check eg ⟶ Pg 105, 106, 107
  → beautiful Qs.

→ K-Map version.

* Variable Entered Maps (VEM)

- reduce K-Map size

Pg 108 ⟶ 1 example.

⇒ Minimization ⟶ Eg. 110

VI

* If minimal SOP is independent of w, y; POS is also independent of w & y.

* Not true, that [with DCS]

min SOP → f
min POS → f

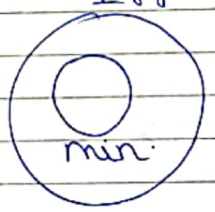
[without DCS, SOP & POS refer to the same funct^n]

VI

∴ when DCS:

SOP ⟶ f_1
POS ⟶ f_2
∴ f not unique.

*



[Example: Pg 119]
→ on PI Chart, Minimal Exps.

VI

VI.

* DCs are never included in the PI Chart
- used in minimization not realization

$1 + \phi = 1$
$0 + \phi = \phi$
$1 \cdot \phi = \phi$
$0 \cdot \phi = 0$

* DESIGN & SYNTHESIS OF COMBINATIONAL CIRCUITS

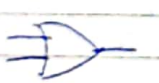
SE ← SA

· ST → appln → Design of digital circuits

[Gates: basic elements]

AND

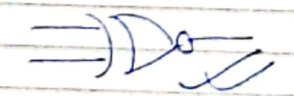
OR



NOT

EXOR

EXNOR

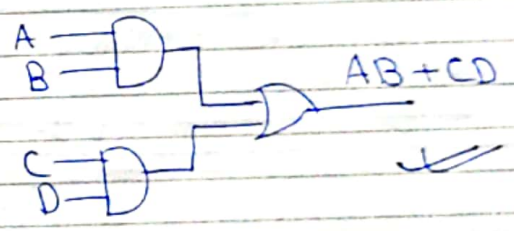


· Design circuit
- → Time ↓
- → Cost ↓

∴ [We need to minimize T → ↓ levels
cost → ↓ No. of gates]

· $f(A,B,C,D) = AB + CD$ (SOP)

→ Sa. AND-OR realizn.



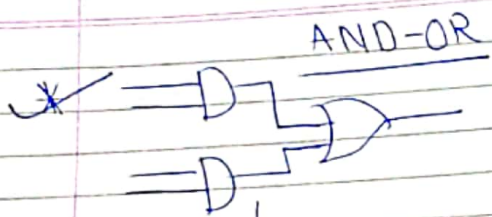
* FAN-in: No. of i/ps to a Gate.
FAN-out: No. of o/ps of Gate.

· 3 i/p gate > 2 i/p Gate.
cost

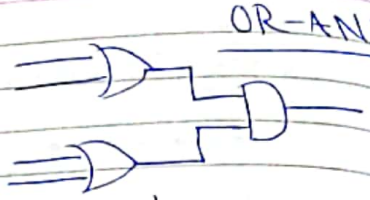
∴ Time ↓ Cost ↑ → universal Gate. (NAND/NOR).

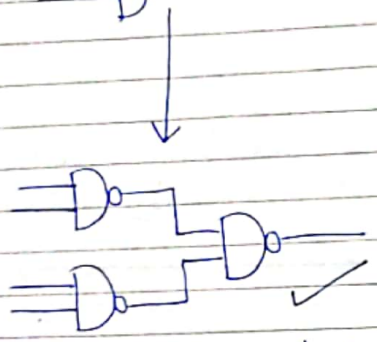
∴ We 1 Single Gate & create all combinational circuits.
→ NAND/NOR

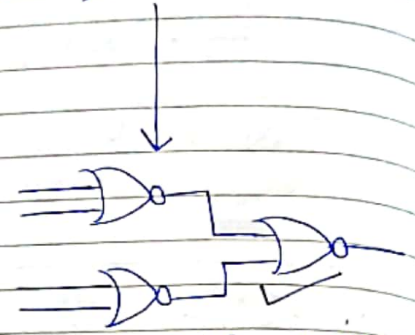
AND-OR



OR-AND







NAND-NAND

NOR-NOR

SOP	⟶ AND-OR	⟶ NAND-NAND
POS	⟶ OR-AND	⟶ NOR-NOR

* We need to minimise no. of Gates

- ⟶ heat prodⁿ ↓
- ⟶ cost / time ↓
- ⟶ Space ↓

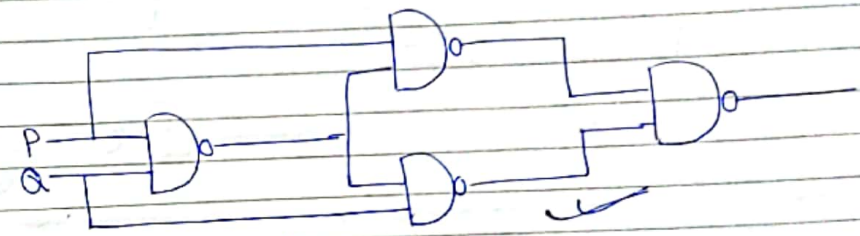
VI

1 example ⟶ Pg 137

1 example ⟶ Pg 139

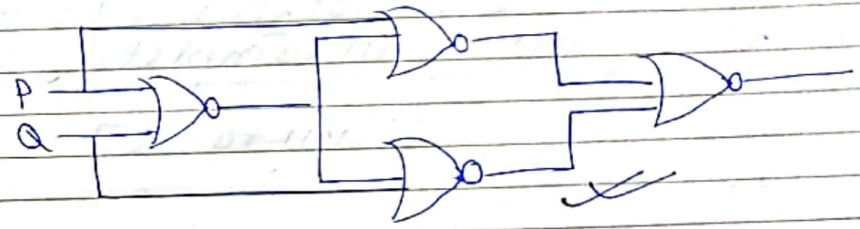
4.

EXOR using NAND Gates



4.

EXNOR using NOR Gates



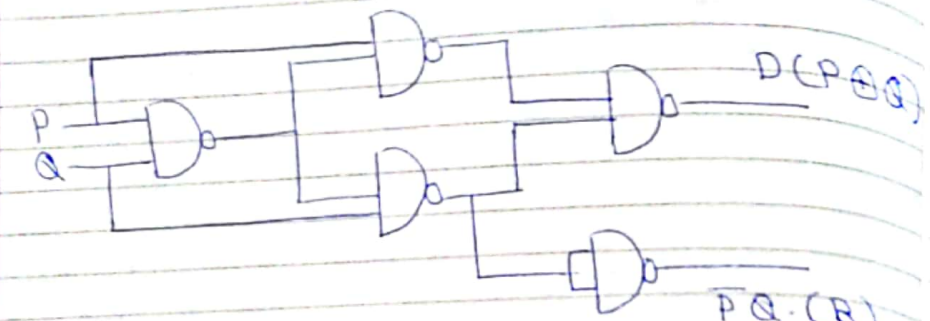
* Half Subtractor

P	Q	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

VI

$D = P \oplus Q$, $B = \overline{P}Q$

$\overline{P\bar{Q}\cdot Q} = (\overline{P}+\overline{\bar{Q}})\cdot Q = PQ$



∴ $2^n \times 1$ Mux

i/p → 2^n lines
bits → n bits
No. of SL → n

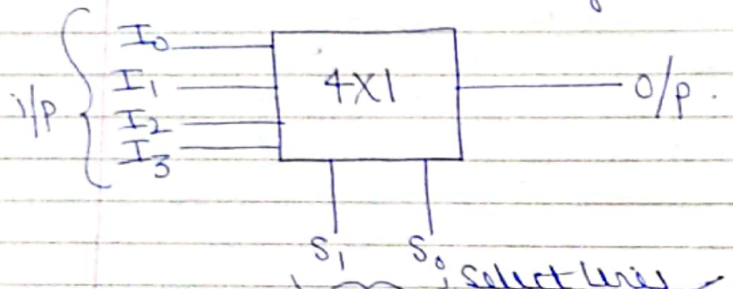
∴ $f = \bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}_0I_2 + S_1S_0I_3$

Char Eqn.

* Intro to Mux:

→ Mux: Electronic Switch that can connect one out of 2^n i/ps to o/ps
→ It is Functionally complete

[Many to one]

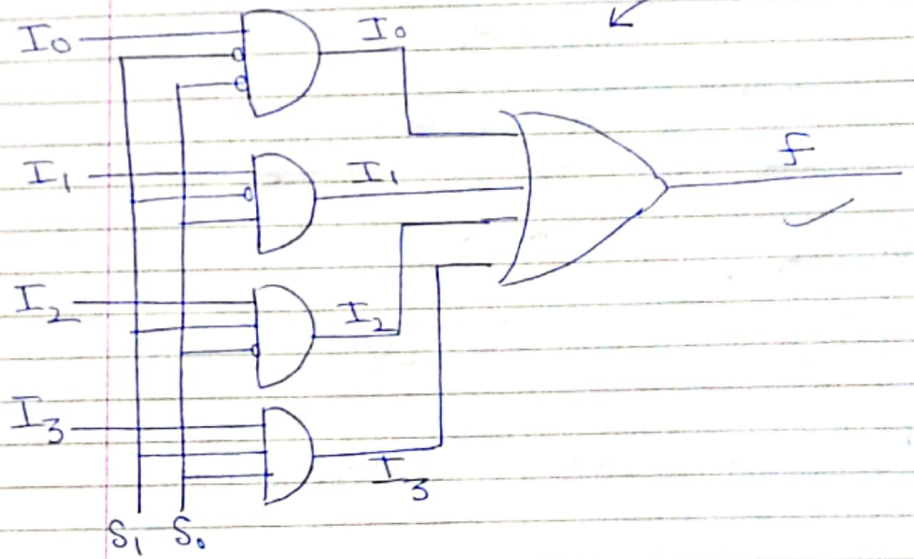


S_1	S_0	o/p
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$\overline{P\bar{Q}\cdot Q}$
$P\bar{Q}+\bar{Q}$
$\overline{\bar{Q}+P}\cdot$
$\overline{P\bar{Q}}\cdot$

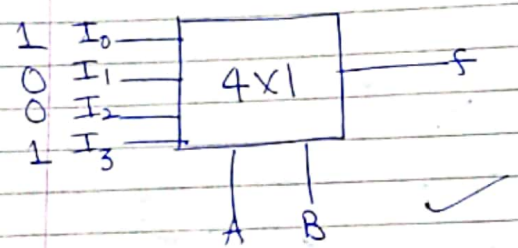
* 4X1 Mux → 4 AND Gates

AND-OR real.



i) I_0 comes out; I_1, I_2, I_3 can't come out

⇒ Mux is Functionally complete.



Q. Implement EXOR using this mux.

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

11 → I_3 ; 00 → I_0

$\therefore I_1 = I_2 = 0 ; I_0 = I_3 = 1$

* $S_n, S_{n-1} \ldots S_1 S_0$

MSB ... LSB

Check → Pg 151 (1 example).

$\therefore$ $2^n \times 1$ mux —can implement→ n var function

→ without any Gates

$\therefore$ By using 4×1 mux ($2^2 \times 1$), we can implement a 3 variable function. (2 var f)

$\therefore$ Using $2^n \times 1$, > n var funct^n can be implemented

→ But maybe extra Gates be required.

* Enable: ON/OFF Switch to mux

$f = E(\bar{I_0} S_1 S_0 + I_1 S_1 S_0 + I_2 S_1 \bar{S_2} + I_3 S_1 S_2)$

{ E = 0; No response
E = 1; (AND Gate react to i/p & give o/ps) }

- Enable not used for function design in industries.

* Inputs of Mux depend on Select lines.

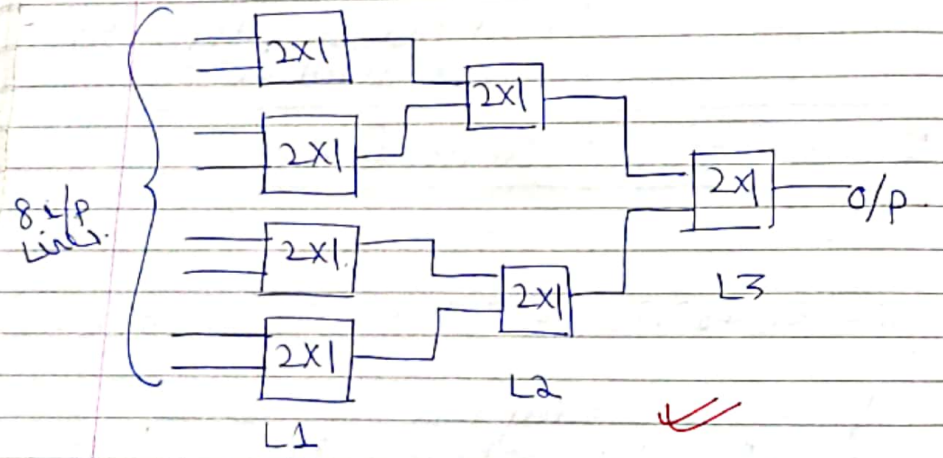
Example → Pg 158

⇒ Cascading Multiplexer

Q → Pg 160.

* Expansion of Mux :

Q 8x1 from 2x1



2x1 mux → 2 lines

1 mux → 2 lines.
1 line → 1/2 Mux.

∴ 8 Lines → 4 Mux (1 Level)

4 lines → 2 Mux (Level 2)

2 lines → 1 Mux (Level 3)

∴ 7 Mux / 3 Level

Q mx1 mux using nx1 mux

1 Mux → N lines
1 line → $(1/N)$ devices

M lines → (M/N) D

$(M/N$ Line) → (M/N^2) D

(M/N^2) line → (M/N^3) D

$\therefore \frac{M}{NK} \leq 1$ (VI)

$\therefore K = \lceil \log_N M \rceil$

$$D = \sum_{K=1}^{\log_N M} \left(\frac{m}{nK}\right)$$

Q. 32×1 using 4×1

$\frac{\log 2^5}{\log 2^2}$

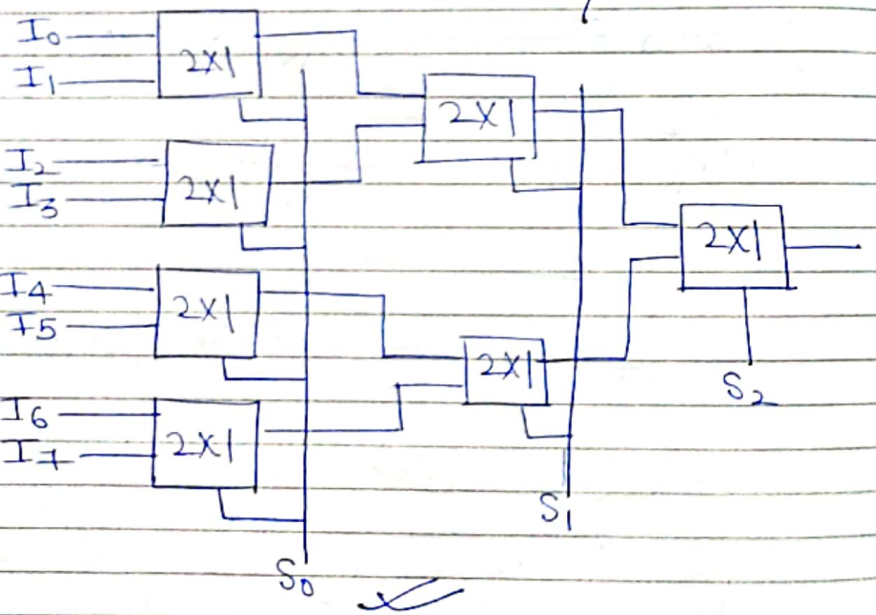
$K = \log_4 32 = \frac{5}{2}$ ✓

$D = \frac{32}{4} + \frac{32}{4^2} + \frac{32}{4^3} = 11$ ✓

8 2 1/2

* It can build larger mux using small mux as building Blocks
  → Scalability of MUX

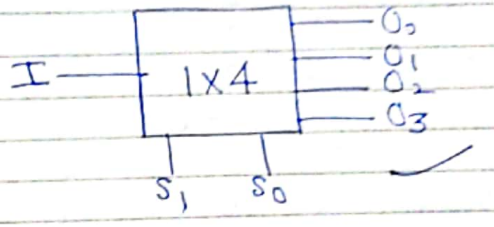
* Assigning Select lines



Order not fixed

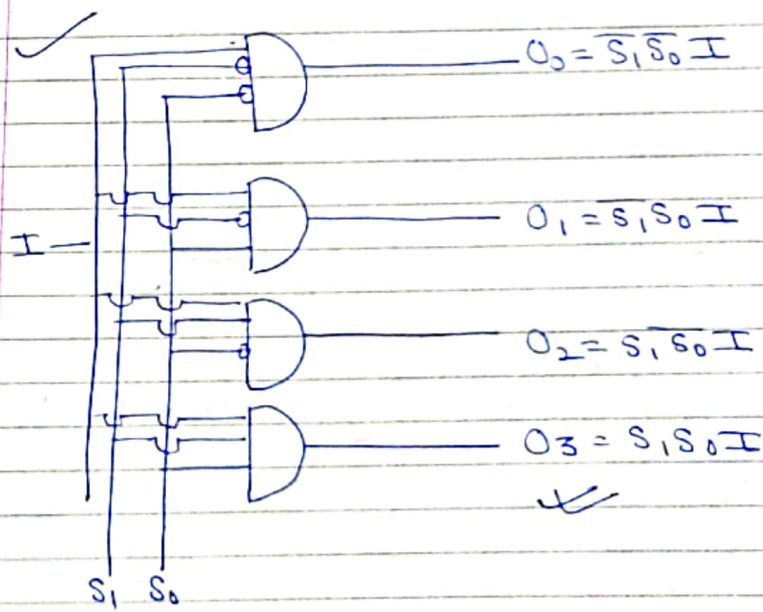
* Demultiplexer

- One to many



2^n AND Gates

- It has 1 i/p, 2^n o/ps where n = No. of SL → 1×2^n Demux.



→ SL: Selects the o/p line.

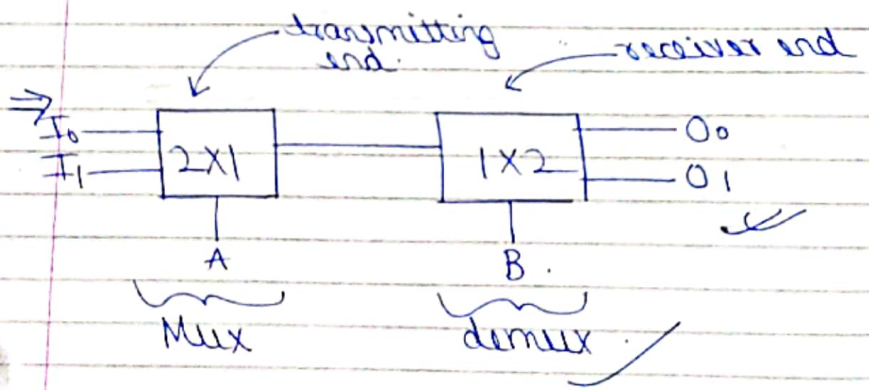
S_1	S_0	O_0	O_1	O_2	O_3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

- Remove OR Gate from Mux
- All i/ps combine to 1 single i/p } Demux

→ No 2 o/p lines will show I at the same time

→ Used in construct^n of Switches.

* Switch:



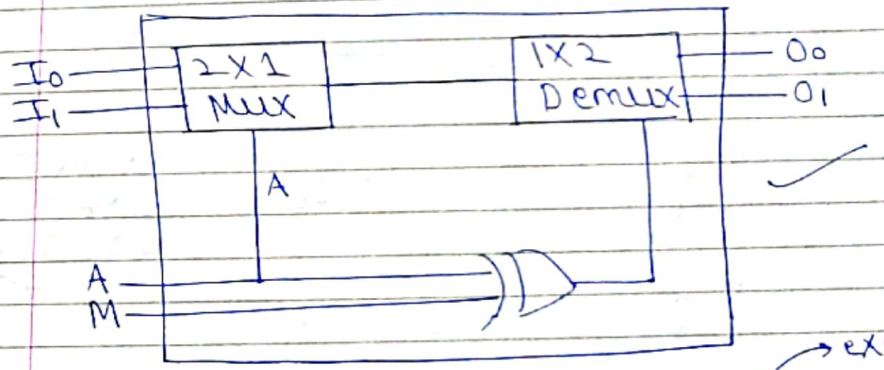
*

A	B	
0	0	I_0 — O_0
0	1	I_0 — O_1
1	0	I_1 — O_0
1	1	I_1 — O_1

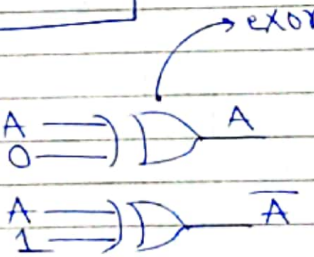
Straight connect^n

Cross connect^n

* Easier imp.

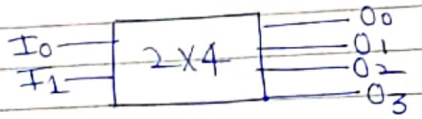


if M=0, SC
M=1, CC.

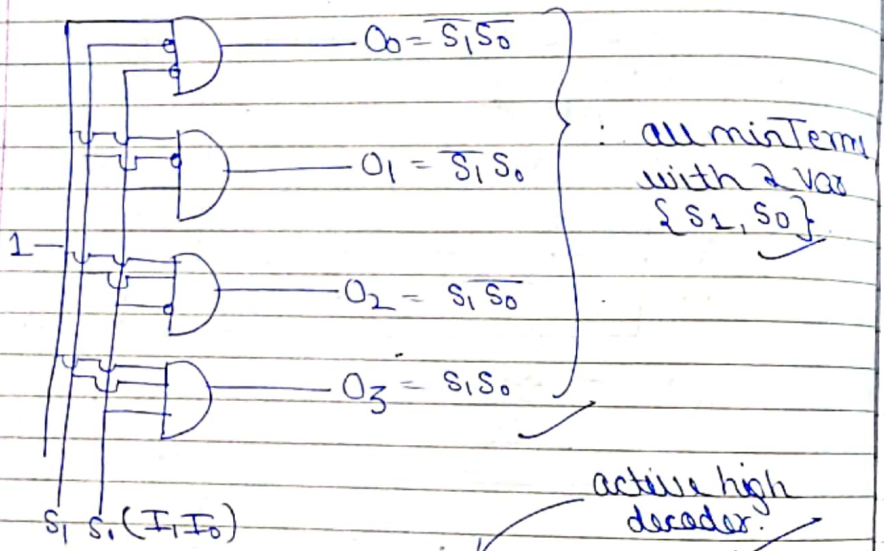


* Decoder

- $n \times 2^n$ Decoder
- No Select Lines



- Demux → Decoder (I=1)



active high decoder

⇒

S1	S0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

: Implement any funct^n in Canonical SOP using OR.

1 Eq → Pg 175

- In case of active low decoder, we NAND Gate outside to realise function.

→ Eq : Pg 175

* See 1 Eq → Pg 176 (Implementing Funt. with Decoder).

active high decoder with OR

↓

Active low decoder with NAND

1 eq → Pg 179

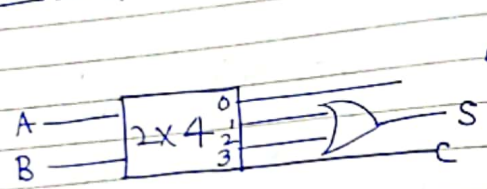
* Decoder converts 1 code to another code.

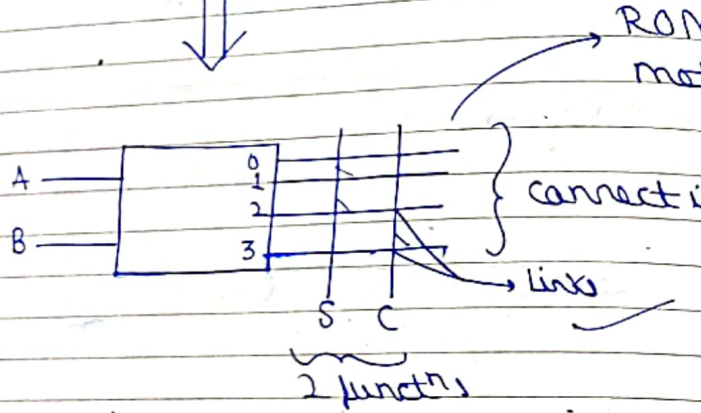
{2421 → Self Complimenting Code}

BCD → excess 3

BCD → 2421 (S=9)

* ROM implementation using Decoder





- horizontal below vertical lines.
- we burn the links we don't want.

ROM = Decoder + Matrix

* Implementing functn's using only decoder

- See 1 eg : Pg 184.

m Functn of n var → n x 2^n decoder
- No: of incoming lines = 2^n.
- No: of outgoing lines = m.

No: of links = $2^n \times m$

$\therefore$ $2^n + m$ → connections.

* Implementing functions using decoder + MUX

→ decrease the no: of connections.
- Links stay constant, but connections ↓

Eg → Pg 187

[No of SL]

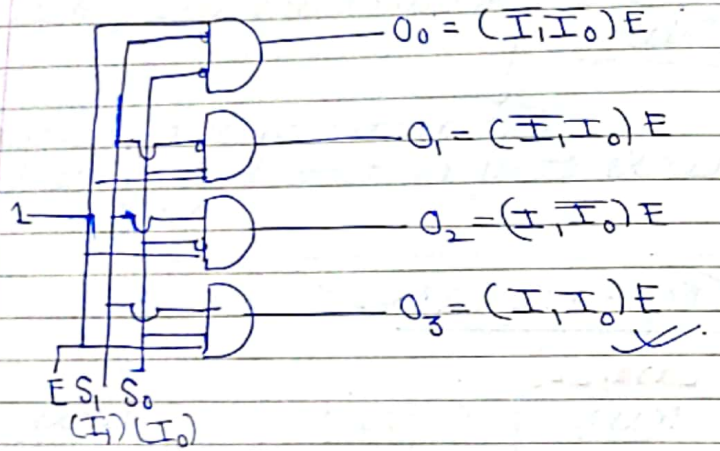
Dec.	MUX	Connect.	Links
10	—	$2^{10} + 1$ = 1025	1024
5	5	25 + 25 = 64	1024
4	6	24 + 26 = 80	1024
3	7	$2^3 + 27$ ⇒ 136	1024

VI

$\therefore$ [If we increase mux capacity ↑, connections reduce ↓ → Not True]

→ optimality should be maintained.

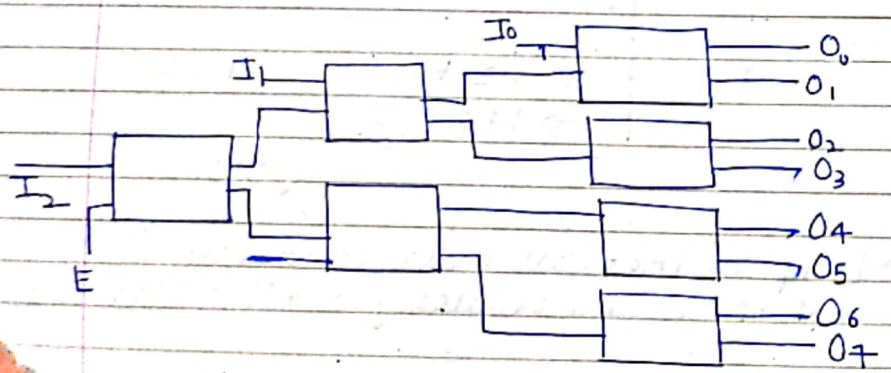
* Enable i/p in decoders are used for expansion of decoders.



if E = 0, o/p = 0
E = 1, o/p depends on S1S0.

Example → Pg 201

* Constructing 3x8 using 1x2 decoder.



(Go in this flow while designing).

→ Decoder decides which o/p line is to be selected.

VI

Eg → Pg 203

* Constructing 6x64 using 3x8

1 D → 8 lines
1 line → (1/8 D).

⇒ 64 lines → 8 D

8 lines → 8/8 = 1 D

Q Construct $m \times m$ decoder using $n \times n$.

1 D → n lines
1 L → (1/n) D

⇒ m lines → (m/n) D

enable

(m/n) L → (m/n²) D

$\therefore \frac{m}{nK} \leq 1$

$\therefore K \geq \log_n m$, $n_D = \sum_{K=1}^{\log_n m} \left(\frac{m}{nK}\right)$

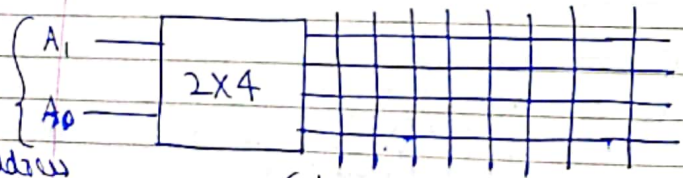
1 eg → Pg 208

* For expansion of decoder in another way:

→ 1 eg → Pg 210

$4(3\times8) + 1(2\times4) = (5\times32)$ decoder

* ROM

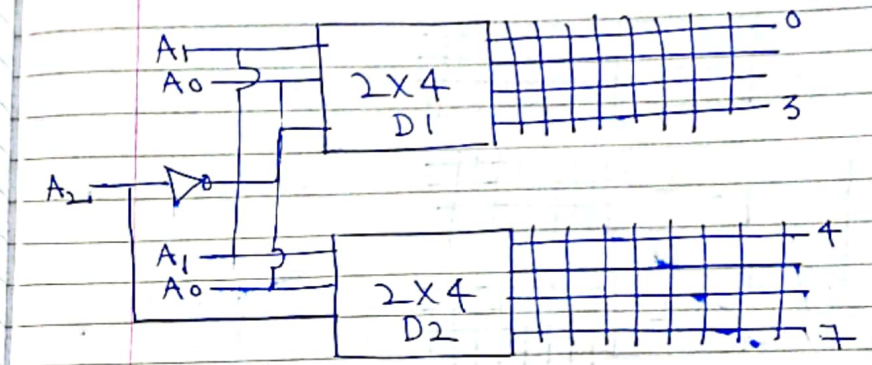


Size of ROM = 4W x 8 bits

Word Addressable

Address Expansion

Convert into 8W x 8 bits



i) $A_2 = 0$, D_1 will be selected.
$A_2 = 1$, D_2.

∴ Increase the word. (no. of words).

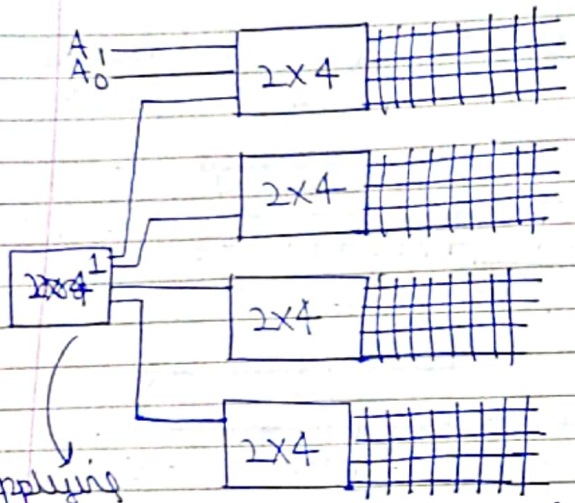
∴ [No. of words / Address Range ↑ Word size constant]

* Expand word size

- Just remove the NOT Gate.

1 dig → Pg 214.

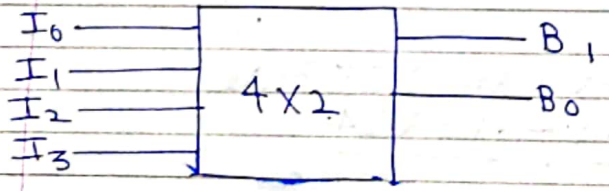
∴ 4W x 32 B



(Address Range → 1 eg: 216).

* Encoders.

- $[2^n \times n]$



∴

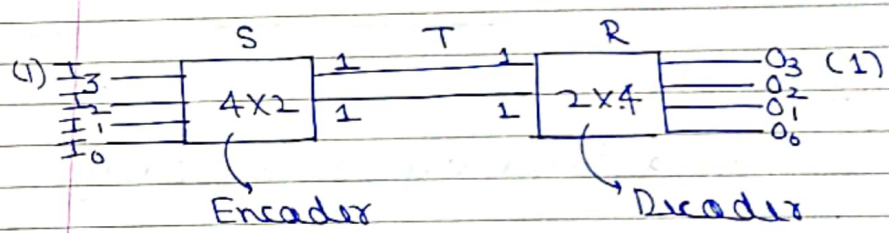
I_0	I_1	I_2	I_3	B_1	B_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

- at any period, one i/p is activated.

VI

Decoder: Select one of the o/p lines
Encoder: Select one of the i/p lines

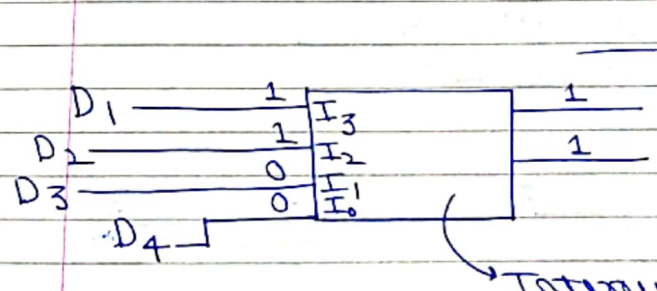
- Convert 1 NS to another NS.



- Lossless compression.

- Non-Priority Encoders: doesn't support simult. i/p activation.

* Priority Encoders → used in interrupt servicing.



∵ $I_3 > I_2 > I_1 > I_0$. [Priority]

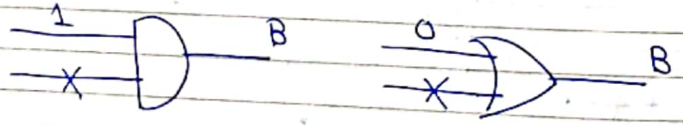
I_0	I_1	I_2	I_3	B_1	B_0
ϕ	ϕ	ϕ	1	1	1
ϕ	ϕ	1	0	1	0
ϕ	1	0	0	0	1
1	0	0	0	0	0

Characteristic Table.

Hazards

- Temporary (Unwanted Delays).
- Permanent → SC(1).
  - OC(0)

- To test whether lead is working
  - Single fault analysis [make o/p dependent on that particular path].
  - → Test Vector



AND/NAND → 1
OR/NOR → 0

S-a-0 / S-a-1.

- We doubt that f shows always 0/ not.
- Use Test Vectors

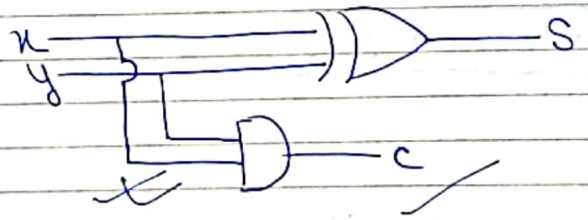
Eg → Pg 225.

Half Adder

- 2 i/p → 2 o/p circuit.
- used to add 2 bits.

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = x \oplus y$, $C = xy$



* Full Adder

- Add 3 bits

x	y	z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

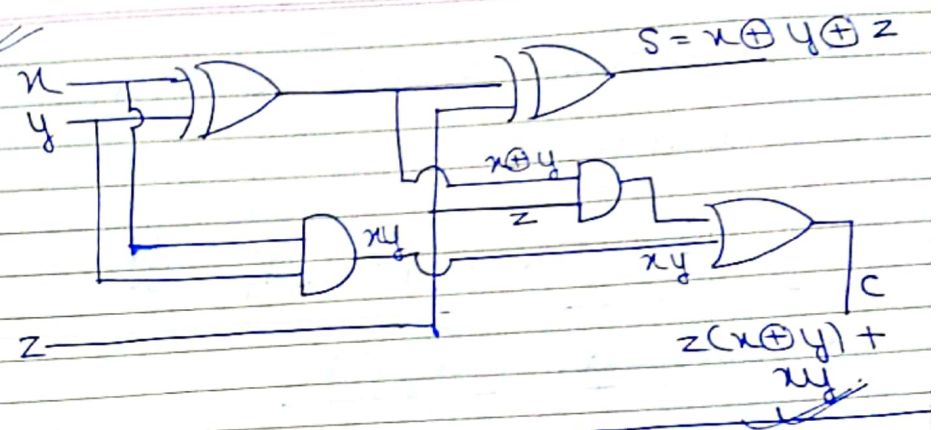
$S = x \oplus y \oplus z$

$C = xy + yz + xz$

$\therefore$ $S = x \oplus y \oplus z$
$C = z(x \oplus y) + xy$

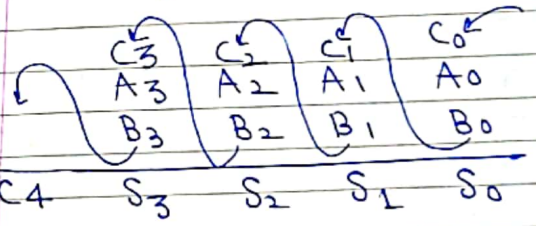
C → any two of 3 bits should be 1.

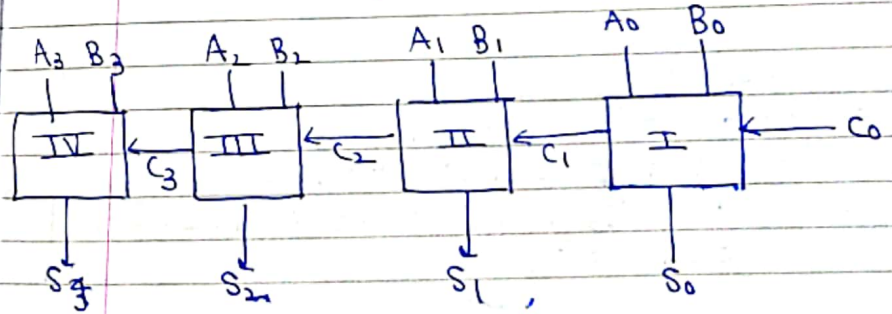
$\therefore$ 2HA → FA



* Ripple carry Adder

- Adding 2 4 bit nos (eg)





$\therefore$ II waits for C_1 $\therefore$ Starvation.

→ Carry ripples through the FAs.

Drawback: Takes a lot of Time

n bits → nFA ∴ $T = n \times T_{FA}$

* Carry Look Ahead Adder

$C_1 = C_0(A \oplus B_0) + A_0 B_0$ ...

$G_i = A_i B_i$ → Generator funct^n

$P_i = A_i \oplus B_i$ → Propagation funct^n

∴ $C_1 = C_0 P_0 + G_0$

$C_2 = C_1 P_1 + G_1$

$C_3 = C_2 P_2 + G_2$

$C_4 = C_3 P_3 + G_3$

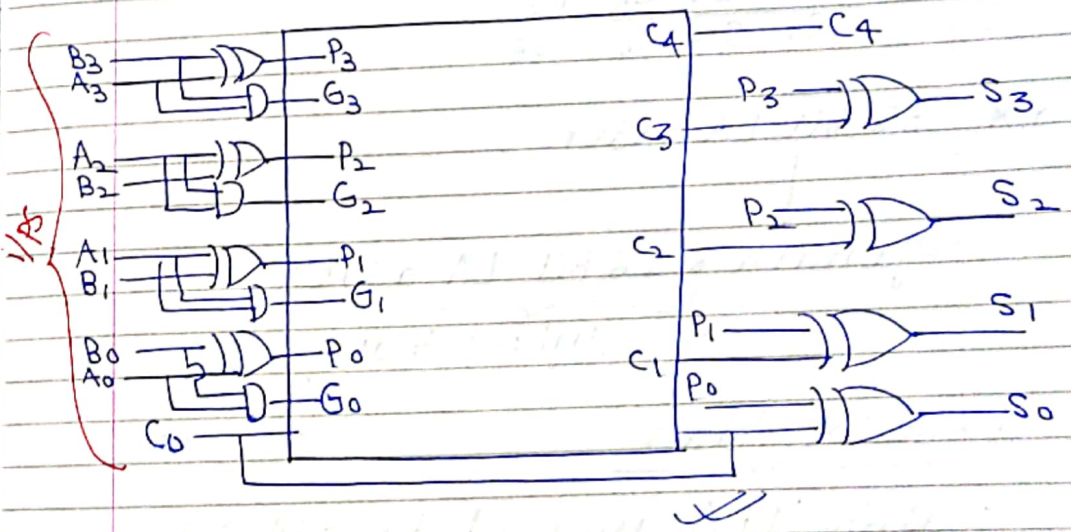
∴
$$C_1 = C_0 P_0 + G_0$$
$$C_2 = C_0 P_0 P_1 + G_0 P_1 + G_1$$
$$C_3 = C_0 P_0 P_1 P_2 + G_0 P_1 P_2 + G_1 P_2 + G_2$$
$$C_4 = C_0 P_0 P_1 P_2 P_3 + G_0 P_1 P_2 P_3 + G_1 P_2 P_3 + G_2 P_3 + G_3$$

∴ Carry Terms depend entirely on the i/ps, not on other adders.

$C_3 \Rightarrow C_0$ either propag. through S-0,1,2 or generated in S-0 & prop. through S1-S2 or generated in S-1 & prop through S2 or generated in S-2.

Drawback
- complex
- Costly.

(But faster)



– No Rippling Effect.

* Hybrid Adder

- Mix of both (CLA & RCA).

c_3 c_2 c_1 c_0
c_4 A_4 A_3 A_2 A_1
B_4 B_3 B_2 B_1
S_4 S_3 S_2 S_1

↓ ↓ ↓ ↓
Pipeline

A_9 A_8 A_7 A_6 A_5
B_9 B_8 B_7 B_6 B_5
S_9 S_8 S_7 S_6 S_5

** 32 bit LA adder

↓

go with 16 bit LA adder.

Hybrid Adder.

* Serial Adder

- using 1FA, we need to add 4 no.
- use Shift register.

* See Pg 443 (How it works)

- Very slow compared to all other adder
- Cheapest

* 8 Bit comparator

- compare unsigned no.'s

2 bit comparator = $2^4 = 16$ values
3 bit comp. = $2^6 = 64$ combinations
4 bit comp = $2^8 = 256$ comb.

$x_2 = A_2 \odot B_2$

$x_1 = A_1 \odot B_1$ $\{A_2 A_1\}$ $\{B_2 B_1\}$

$\therefore$ Also 1 $[A = B]$

$$\boxed{(A_1 \odot B_1)(A_2 \odot B_2) = 1}$$

$[A_2A_1]$ $[B_2B_1]$

Case 2 ($A > B$)

$$A_2\overline{B_2} + (A_2 \odot B_2)(A_1\overline{B_1}) = 1$$

Case 3 ($A < B$)

$$\overline{A_2}B_2 + (A_2 \odot B_2)(\overline{A_1}B_1) = 1$$

* 3, 4 bit comparator.

$x_1 = A_1 \odot B_1$,
$x_2 = A_2 \odot B_2$
$x_3 = A_3 \odot B_3$.

- $A = B$ → 3 bit comp.

$x_1x_2x_3 = 1$

- $A > B$ $A_3A_2A_1$ $B_3B_2B_1$

$$A_3\overline{B_3} + x_3A_2\overline{B_2} + x_3x_2A_1\overline{B_1} = 1$$

- $A < B$

$$\overline{A_3}B_3 + x_3\overline{A_2}B_2 + x_3x_2\overline{A_1}B_1 = 1.$$

* Analysing all the cases of comparators

- For n bit comparison: (2n bits).

∴ 2^{2n} → total comb^n

- $A = B$: 2^n combinations.
- $A < B$: $\frac{2^{2n} - 2^n}{2}$ comb.
- $A > B$: $\frac{2^{2n} - 2^n}{2}$ comb.

* Time complexity of Ripple Carry Adder:

$$\theta(nT_{FA}) = \theta(n)$$

* Time complexity of Carry Lookahead Adder:

$$\theta(\log_K n)$$

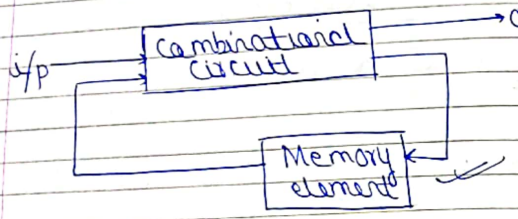
where, n: size of i/p (no. of bits)
K: FAN-in of AND Gates.

or, $$T = \theta(\log_a n + \log_b n)$$

where, a: FAN-in of OR Gate.
b: FAN-in of AND Gate.

* SEQUENTIAL CIRCUITS.

- Combinⁿ Circuits dont involve feedback.



- o/p depends on prev. states also along with i/p.
- [o/p depends on ext i/p & on present contents of memory elements].

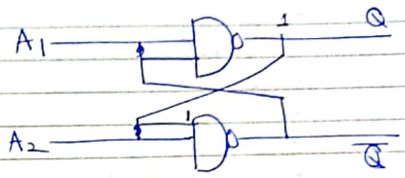
VI

- Flip-Flop: building block of memory [1 bit].

BB of RAM → FlipFlop

- FlipFlop: 2 States (0, 1).
  - ↳ bistable multivibrator.
  - Smallest memory elements.

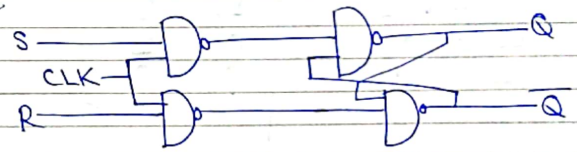
* Latch



- If we latch 0 ($A_1 = 0$), $Q = 1$, $\bar{Q} = 0$.
  [This bit won't change until we change it].

- Black.
- Latch used to build FF

* SR - FlipFlop.



Q_n: prev. state
Q_{n+1}: next state.

$$Q_{n+1} = F(S, R, Q_n)$$

i/p, present state

Char Table

S	R	Qn	Qn+1	
0	0	0	0	Latch mode
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	ϕ	Invalid
1	1	1	ϕ	

$Q_{n+1} = S + \bar{R} Q_n$ ← Char Eqn.

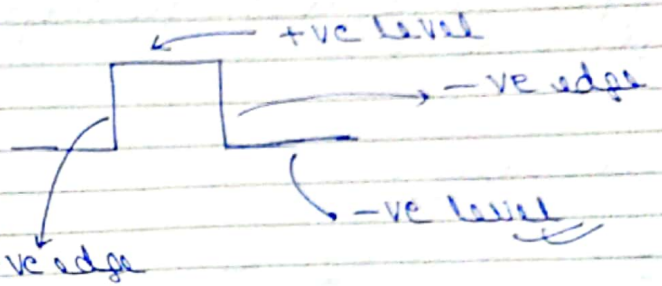
Excitation Table

Qn	Qn+1	S	R
0	0	0	ϕ {0,1}
0	1	1	0
1	0	0	1
1	1	ϕ	0

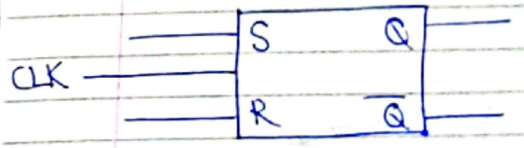
Function Table

S	R	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	ϕ

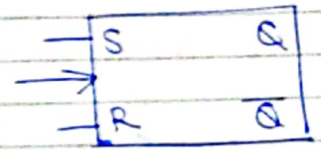
* FF activates/reacts to i/p during +ve level.
  → +ve level triggered.



- +ve level

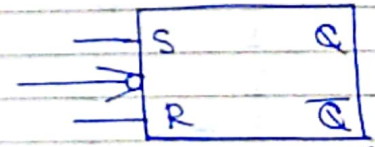


+ve edge



- -ve level

-ve edge



Refer Pg 267 / Pg 269 for eg: → transparent mode time ↓

→ Edge triggered: Disturbance won't affect much.

* JK Flip Flop

- JK flip flop same as SR flip flop.
- Use for J = K = 1.

Function Table:

J	K	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Q_n}$ [Complement]

· Char. Table

J	K	Qn	Qn+1	
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	1	} Toggle
1	1	1	0	}

$$Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$$

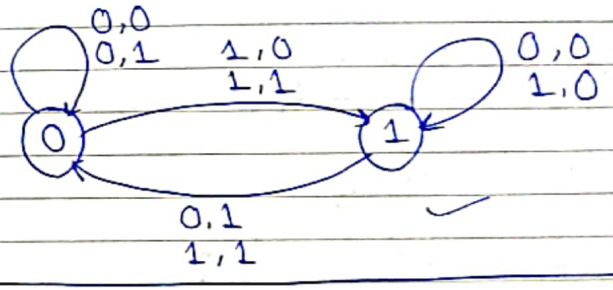
· Excitation Table

Qn	Qn+1	J	K
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

0 1
1 1

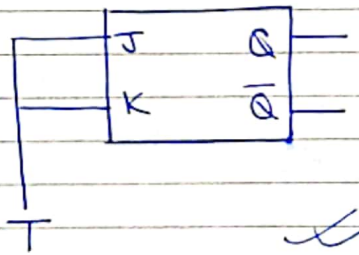
* State diag (from excit° table)

(0,0)
J K



* T-Flip Flop

- latching & Complementation (Toggle).



T	Qn+1
0	Qn → Latch
1	$\overline{Q_n}$ → Toggle

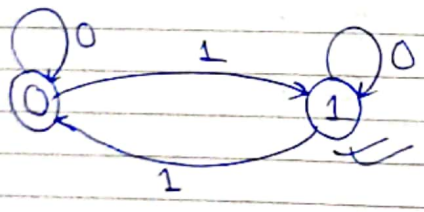
Char Table

T	Qn	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

$$Q_{n+1} = T \oplus Q_n$$

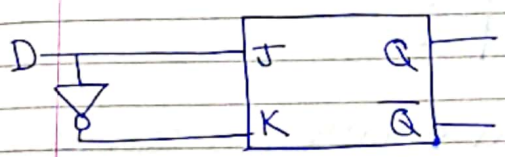
* Excitation Table

Qn → Qn+1	T
0 → 0	0
0 → 1	1
1 → 0	1
1 → 1	0



T = 0/1

* D-FlipFlop (Delay)



JK → 01/10 (2 pairs).

D	Qn+1
0	0 (R)
1	1 (S)

D	Qn	Qn+1
0	0	0
0	1	0
1	0	1
1	1	1

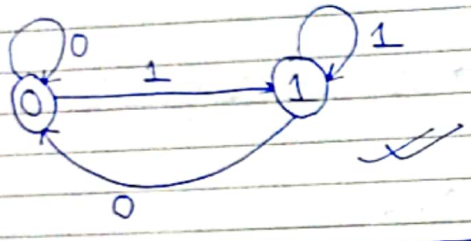
$$Q_{n+1} = D$$

- No feedbacking.
- Faster than propagation delay.

Excitation table

Qn → Qn+1	D
0 → 0	0
0 → 1	1
1 → 0	0
1 → 1	1

- D used in various appln's like counters.



- Always derive Function Table first from Char Eqn.
- Write State diag. from Excitn Table.
- (See Eg. 278).

* Intro to FF conversion

- Get the Char. Table of the target FF.
- Replace the next state using excitn of the given FF.
- example → Pg 284

- From char. Eqn → Funct. Table.

1 example: Pg 290

* Counters

- need to count clock cycles.
- accurate timing & control signals
- T & D flipflop mostly used.

* NUMBER SYSTEM

- how we represent/count no.s.
- Base n = n digits
- Every digit must be represented with a single literal.
- All digits must be less than the base.

eg: $(0126)_6$ X.

* Conversion to base 10.

- $(abc)_x = (ax^2 + bx + c)_{10}$
- $(123)_7 = 1 \cdot 7^2 + 2 \cdot 7^1 + 3 \cdot 7^0 = (66)_{10}$

Q $(1010)_2 \Rightarrow (10)_{10}$

1 —
2 ——
5 ———
10 ————

↖ Easy Method.

$1 \times 2 + 0 = 2$
$2 \times 2 + 1 = 5$
$5 \times 2 + 0 = 10$ ✓

* Conversion from Base 10

- $(11)_{10} = (1011)_2$

2 | 11		
2 | 5	1	↑
2 | 2	1	↑
2 | 1	0	↑
0	1	↑

$\Rightarrow (1056)_{16} \longrightarrow (\quad)_2$

$16^1 = 2^4$.

$\Rightarrow$ Every digit in base 16, has to be represented by 4 digits in Base 2.

→ Working Start from LSB.

1 eg → Pg 305

NUMBER SYSTEM

- $33 = 27^1$

- Every 3 digit in Base 3 has to be repr. in 1 digit in Base 27

* From Decimal to Binary:

→ Convert to big base & then base 2.

⇒ Max value of B-2, when converted to Base 10

$1\ 1 = 3 \quad 2^2 - 1$

$1\ 1\ 1 = 7 \quad 2^3 - 1$

$\therefore \boxed{b^n - 1}$

where, b = base
n = no. of bits

Example — Pg 310

$2^n - 1 \geq 4^{10}$

⇒ 10 digit — Base 4

(to be represented in)

? — Base 2

$$n \geq 10 \log_2 4$$

- Refer 1 example → Pg 316

→ Always check condn's first

- Base/Radix of any no. can't be negative.

* Complementary Number System

- We can do subtraction also with the addition circuit.

→ 2 complementary no. System

→ diminished radix compliment $[b^n - 1] - x$

→ radix compliment $[b^n - x]$

eg: Base 2

→ 1's comp. $[2^n - 1] - x$

→ 2's comp $[2^n - x]$

⇒ {radix comp. > popular diminished radix comp}

⇒ Base 2

$2^4 - 1 = (1111)_2$

- See 1 eg → Pg 327

→ $y - x = y + \overline{x}$

→ complement (DR/R).

VII

b-1's comp. ⇒ $\overline{x} = (b^n - 1) - x$

b's comp. ⇒ $\overline{x} = b^n - x$

* In diminished Radix compliment (b-1s) complement

→ 2 representations of 0.

eg: 0000···0 → 0
1111···1 → 0
} Base 2

disadvantage

eg: 000···0 → 0
999···9 → 0
} Base 10

* In radix compliment, only one representation of 0.

(b-1)s → 2 patterns of 0
bs → 1 pattern of 0

* Subtraction in diminished Radix (b-1)s compliment

$x + \bar{y}$

$\Rightarrow x + (b^n - 1 - y)$

$\Rightarrow b^n - 1 + (x - y)$

get rid of this: b^n → end around carry

→ (b-1)s complement.

Case 1: $(x-y) \geq 1$:
we get carry: discard carry and add 1

Case 2: $(x-y) < 1$
find (b-1)s comp. of the result and negate it (signed repr.).

$\Rightarrow x + \bar{y}$
$\Rightarrow x + b^n - 1 - y$
$\Rightarrow b^n - 1 + (x-y)$ [Carry not visib.]
$\Rightarrow b^n - 1 - (b^n - 1^0 + x - y)$ ∵ $x - y < 0$
$\Rightarrow -(x-y)$

See eg: Pg 341 (same no. of bits).

* Subtraction in radix compliment (b)s compliment

→ bs complement

Case 1: $(x-y) \geq 0$
- we get carry; ignore it.

Case 2: $(x-y) < 0$
take bs complement & negate it.

See ig: Pg 346.

$x + \overline{y}$

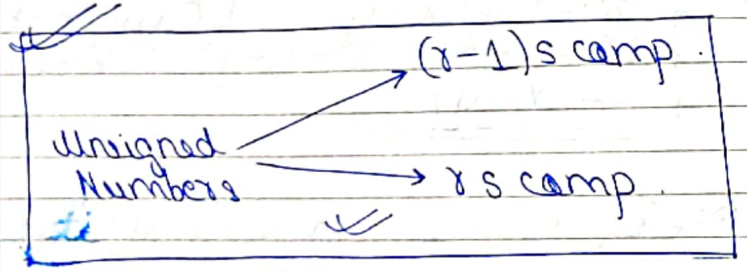
$\Rightarrow x + (b^n - y)$

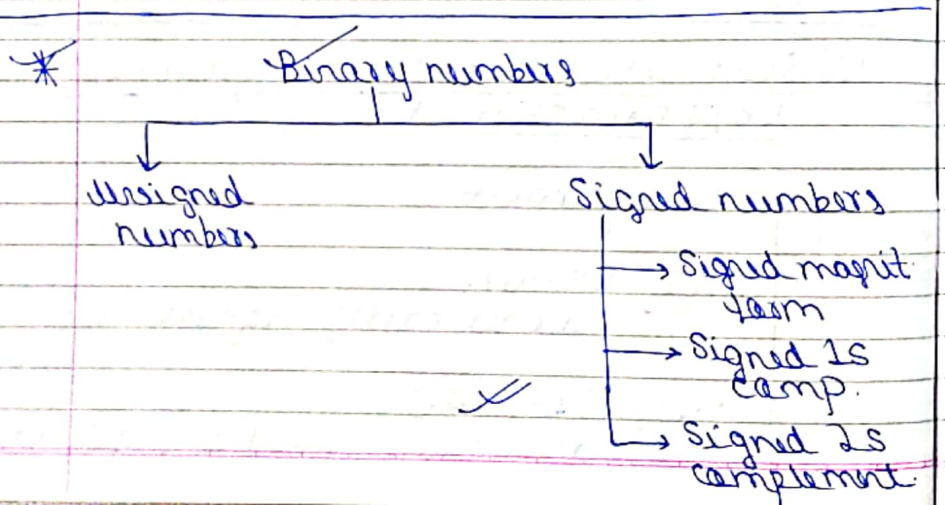
$\Rightarrow b^n + (x - y)$ if, $x - y \geq 0$.

↳ ignore this.

2s compliment > 1s compliment

- no end around carry





14. → Integer
.135 → Fraction
} Fixed point repr.

10.14 → Floating point rep

	unsigned	Signed Mag.	1s	2s
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Signed representation.

eg: 0101 → +ve
1001 → -ve
} signed.

→ Positive no. always same in all (SM, 1s, 2s).

* Signed Magnitude

Drawbacks

- 100 = -0 → 0
  000 = +0 →
  ∴ 2 repr. of 0
- Addition & Subtraction takes separate h/w & complex.

* 1s complement

- -2 ⟶ 101 (1s comp form).
- 000..0 → 0
  111..1 →
  ∴ 2 comb. in 1s comp
- generates end around carry.

* 2s complement [most popular]

- Range is more
- Only 1 comb. of 0
- Same h/w for addition & subtn & no end around carry

₹359,360 ⟶ 1 cr

- Codes → Weighted
  → Unweighted

SM: weighted
1s: non-weighted
2s: weighted

eg: 2s

$\underset{-2^2}{1}\ \underset{2^1}{1}\ \underset{2^0}{1} = -4+2+1 = -1$

eg: S ___

$(-1)^S \times (\text{Mag.}) = \text{SM}$

- Range of 2s comp. > Range of 1s / SM.

n bits

- Range of SM = $-(2^{n-1}-1)$ to $2^{n-1}-1$
- Range of 1s = $-(2^{n-1}-1)$ to $2^{n-1}-1$
- Range of 2s = -2^{n-1} to $2^{n-1}-1$

eg: n = 4

SM: [-7, 7]
1s: [-7, 7]
2s: [-8, 7] $\{-2^3$ to $2^3 - 1\}$

* Sign Bit Extension

- Unsigned

0 1 1 0

0000 0110 8 bit register

* In Signed Magnitude, when we want to extend, we take the signed bit & put it in MSB.

eg: 1 1 1 0

10000110

* 1s compliment

+6: 00000110

-6: 11111001

∴ 1001 = -6

11111001

- take the SB, & copy it in first 4 bits

* 2s comp

- If we have a chain of bits in the MSB we can delete all the 1s till we get one 1.

eg: 11101

101

∴ 1101 ⇒ -3

extension

11111101 8 bit register

* Overflow :

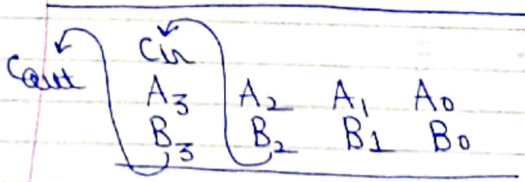
- If we add two n bits, result will be maximum (n+1) bits.
- condⁿ in which the result cant be fit into the space we are given.

→ Unsigned

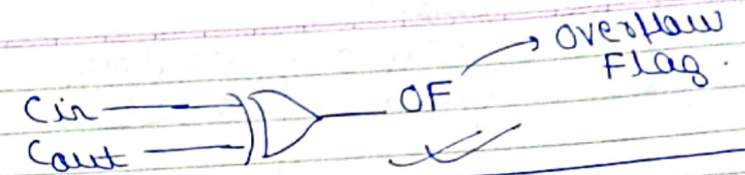
Carry indicates overflow.
[1 eg → Pg 375]

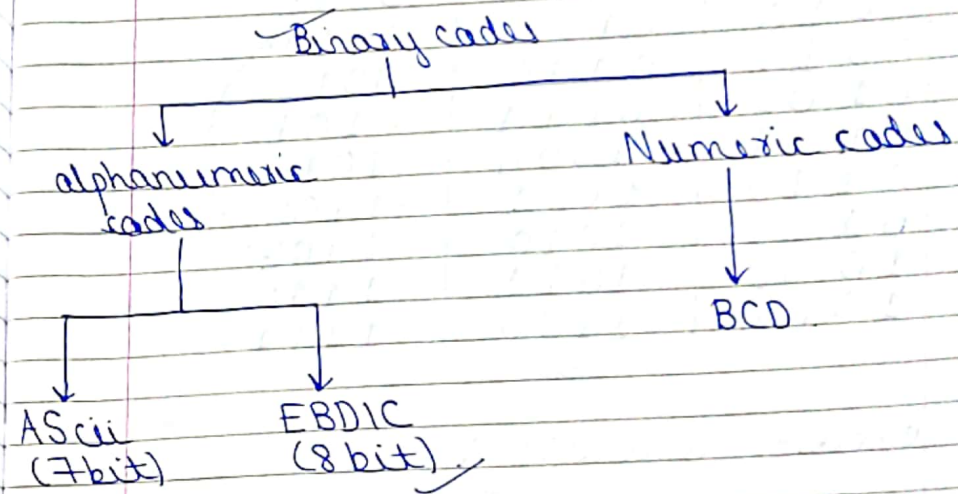
→ Signed

Overflow is indicated by 2 cases :
1. When 2 +ve no.s are added, & the result is -ve.
2. When 2 -ve no.s are added, & the result is +ve.

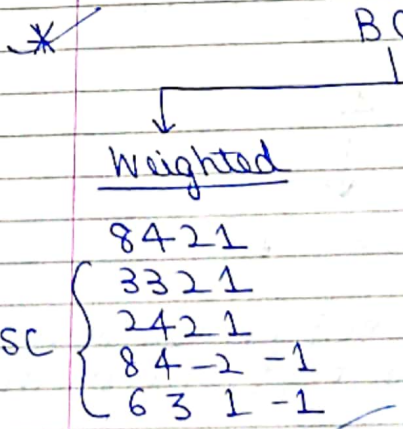


where, $C_{in} \neq C_{out}$: (Overflow)
$C_{in} = C_{out}$: (no overflow).

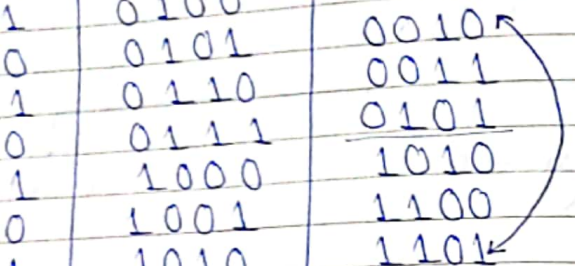




- [Weighted rep. no. → formula
  NW code → no formula]



	(sequential) 8421	(sequential) (Self comp) excess 3	(Self comp) 3321
0	0000	0011	0000
1	0001	0100	0001
2	0010	0101	0010
3	0011	0110	0011
4	0100	0111	0101
5	0101	1000	1010
6	0110	1001	1100
7	0111	1010	1101
8	1000	1011	1110
9	1001	1100	1111



- excess3 : unweighted

- 3321

3 → 0011 / 0100 / 1000

(choose smallest among them)

- 5 ? 9's comp = 4

1010 ← 0101 (1's)

[If a code has to be self complementing, the sum of W has to be 9, but reverse not always True]

⇒

Code	Self complimenting
Excess 3	Yes
8421	No
3321	Yes

⇒ if abcd is SC,
then a+b+c+d = 9
but reverse always not true.

8421 → not SC, sequential
Excess 3 → SC, sequential
3321 → SC

	Excess 3
0011 = 3 →	0
1100 = 12 →	9

- In BCD (Binary coded decimal)

if no. → n digits
∴ n x 4 = 4n digits (BCD)

- {2241 → SC code}

Eg → Pg 5

* BCD Addition:

- Add 6 for correction when:
  (i) carry
  (ii) invalid

1eg → Pg 8

* Excess 3 Addition: VI

Rule 1: If carry occurs, add correction (+3)

Rule 2: If carry doesn't occur, subtract (-3).

1eg → Pg 9

* Gray code VI

- reflexive
- unit distance code
- cyclic code
- non-weighted code

· 4bit Gray code:

0 0 0 0	: 0
0 0 0 1	: 1
0 0 1 1	: 2
0 0 1 0	: 3
0 1 1 0	: 4
0 1 1 1	: 5
0 1 0 1	: 6
0 1 0 0	: 7
1 1 0 0	: 8
1 1 0 1	: 9
1 1 1 1	: 10
1 1 1 0	: 11
1 0 1 0	: 12
1 0 1 1	: 13
1 0 0 1	: 14
1 0 0 0	: 15

d=1

∴ Gray code takes less bits than BCD.

· 00 → 01 → 11 → 10 → 00 cycle code.

$(1 - \frac{1}{2^d})$

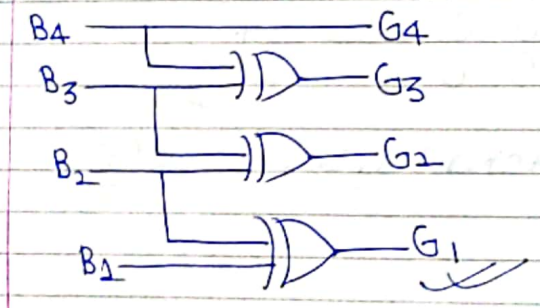
* Binary to Gray & Vice-Versa

1) Binary → Gray:

12 = 1100
↓
1010 ✓

B-No: $b_4\ b_3\ b_2\ b_1$
Gray: $g_4\ g_3\ g_2\ g_1$

$$\begin{aligned} G_4 &= B_4 \\ G_3 &= B_4 \oplus B_3 \\ G_2 &= B_3 \oplus B_2 \\ G_1 &= B_2 \oplus B_1 \end{aligned}$$ ✓



⇒ Q/ 200. Pg 15 (1 eg.)

2) Gray → Binary

1010 (Gray)
↓↑↓↑
1100 (Binary) ✓

Gray = $G_4\ G_3\ G_2\ G_1$
Binary = $B_4\ B_3\ B_2\ B_1$

$$\therefore \begin{aligned} B_4 &= G_4 \\ B_3 &= G_4 \oplus G_3 \\ B_2 &= G_4 \oplus G_3 \oplus G_2 \\ B_1 &= G_4 \oplus G_3 \oplus G_2 \oplus G_1 \end{aligned}$$ ✓

- 1 eg = Pg 16 ✓

* Error Detection:

→ If code from 1VP → 1VP, impossible to detect such error.

⇒ 1 bit error

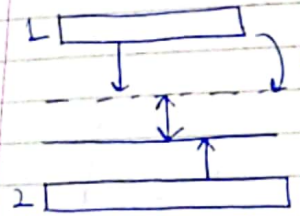
[To detect 1 bit error, the dist. b/w any 2 valid codes must be atleast 2].

→ See 1 eg: Pg 19

∴ t bit error detection → the distance should be atleast (t+1)

⇒ HD = t + 1

* Error Correction



1 bit error: closer to 1
∴ eg code = 1

∴ d ≥ 3

∴ t bit error detection ⇒ ≥ t+1

t bit error correction ⇒ ≥ 2t+1

* Hamming Code

- 1 bit error correction code.

⇒ m bits = message
p bits = parity bits (for error corr.)

∴ $2^p \geq (m+p+1)$

total cases.

eg: Send 4 bits msg:

m = 4, p = 3 $2^3 \geq 3+5$

∴ Total 7 bits received (*)

⇒ See 1 eg: Pg 28-29/30-31

* Floating Point Conversion

⇒ Binary No. → Decimal.

Q 1 1 0 . 1 0 1

2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3}

5/3 → 1

⇒ 6.625

* Fraction → Binary no.

Q 0.25

0.25
0.25
0 ← 0.5
0.5
1 ← 1.0

⇒ 0.01

[Eg - Pg 34]

- For same floating part no.'s, binary no. not possible always.
  ∵ ∞ no.'s between [0, 1]
  But, 2^n combinations out of n bits

* Floating Point representation

⇒ 4.5

| S | E | M |

Normalisation: Just 1 representation for every number.

- Put the radix point to the left of first 1.

100.1 → 0.1001×2^3

Explicit Normalisation.

1 eg → Pg 39

∵ comparing directly 2's complement is a problem

∴ Biasing: Convert -ve → +ve.

$bias = 2^{n-1} - 1$, where n = exponent bits

0.101×2^{-12} → Explicit Normalisn

M = 1010
E = -2 + 16 = 14
S = 1

} Biasing

∴ To convert back into decimal

$[(-1)^S \times 0.M \times 2^{E - bias}]$

- Implicit Normalisation is better.

Q 100.111

1.00111×2^2

[Move the RP to MSB & place it to the right of it]

Conversion

$(-1)^S \times 1.M \times 2^{E-bias}$: Implicit Norm.

(Move the RP to MSB, & place it to the right of it)

$(-1)^S \times 0.M \times 2^{E-bias}$: Explicit Norm.

(Move the RP to MSB, & place it to the left of it).

Eg = Pg 44.

- $1.11111111 \rightarrow 1 + \left[1 - \frac{1}{28}\right]$

0.11111111

E↑ range↑
M↑ accuracy↑

Eg = Pg 47.

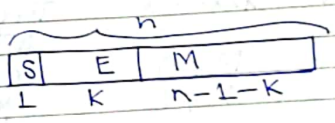
0

near 0 closed to 0 : densely packed
away from 0 : sparsely packed

- Mantissa → normalized SM form.

True Exp = Biased Exp − bias

*

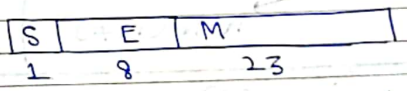


Bias = 2^{K-1}.

* IEEE Standards

- 1985 standard
  - → Single Precision : 32 bits
  - → Double Precision : 64 bits.

* Single Precision (32)



32 bits.

- Excess 127.

S(1)	E(8)	M(23)	Value
0/1	000...0	000...0	±0
0/1	111...1	000...0	±∞
0/1	1 ≤ E ≤ 254	XXX....X	Implicit NF.
0/1	E=0	M≠0	Fractional
0/1	255	M≠0	NAN.

Imp. NF = $(-1)^S \times 1.M \times 2^{E-127}$
fractional = $(-1)^S \times 0.M \times 2^{-126}$

- E=0
  - → ±0 (M=0)
  - → FN (M≠0)
- E=255
  - → ±∞ (M=0)
  - → NAN (M≠0)

** Double Precision (64).

S	E	M
1	11	52

64.

- Excess 1023

S(1)	E(11)	M(52)	Value
0/1	E=0	M=0	±0
0/1	E=2047	M=0	±∞
0/1	1 ≤ E ≤ 2046	M=XXX...X	INF
0/1	E=0	M≠0	F
0/1	E=2047	M≠0	NAN.

INF = $(-1)^S \times 1.M \times 2^{E-1023}$
F → $(-1)^S \times 0.M \times 2^{-1022}$

Ieg → Pg 56

* If we multiply n bits:
worst space for result: O(2n).

- No. of Add^n s = n(1) in multiplier.
- No. of Shifts = no. of bits in multipli.

→ Partial Sum Method.

∴ Booths alg better.

- Any sequence/block of 1s can be written as a diff b/w 2 no.s and these are powers of 2.

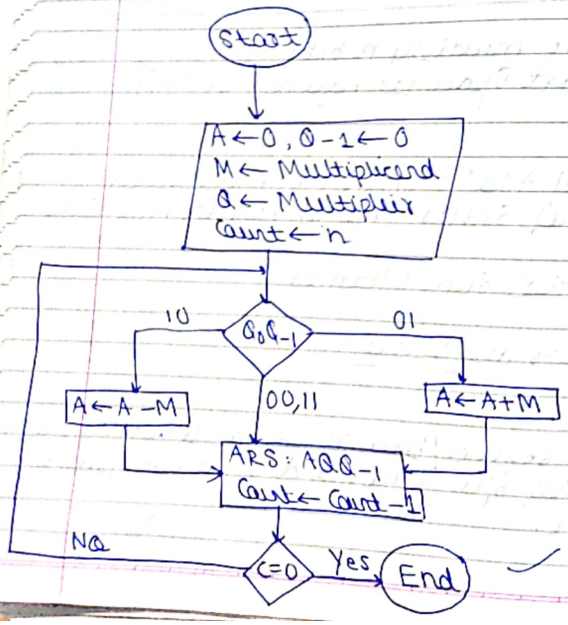
$$[0\,1\,1\,1\cdots 1\,1\cdots 0 = 2^{j+1} - 2^i]$$

(2j, 2i)

- BA decrease no. of additn's.

00 - ARS
10 - RE of 1s (A-M, ARS)
11 - ARS
01 - LE of 1s (A+M, ARS)

- No. of steps = No. of bits in multiplier

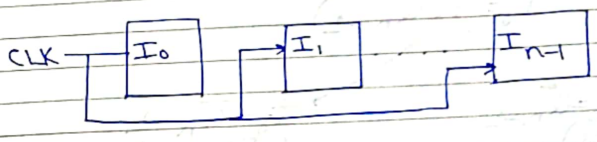


Pg 64-65 - 1 eg

* SEQUENTIAL CIRCUITS

- Synchronous

- comm. clock applied to all other FFs.



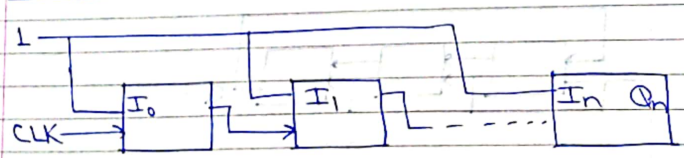
Prop. Delay

- $T_{pdSyn} = T_{FF} + T_{comb}$

$T_{CLK} \geq T_{pdSyn}$

- Faster

- Asynchronous



o/p of FF = i/p clock to next FF

$$T_{pdaSyn} = N \times T_{FF} + T_{comb}$$

* Shift Counter → Synchronous

1) $I_i = Q_{i-1}$; $I_1 = Q_0$, $I_2 = Q_1$

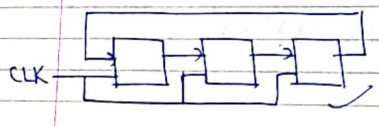
(Data is shifted from 1FF to another)

2) $I_0 = f(Q_{n-1})$

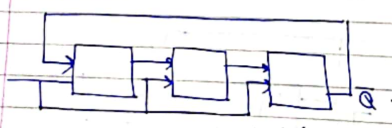
$I_0 = Q_{n-1}$ (Ring)

$I_0 = \overline{Q}_{n-1}$ (Johnson)

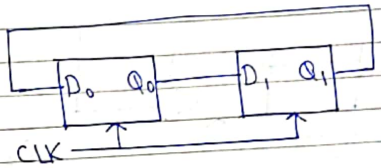
Ring



Johnson

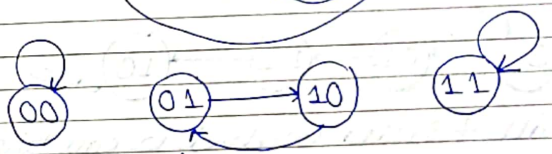


* Mod 2 Ring counters



- $D_1 = Q_0$
  $D_0 = Q_1$

Pre Q_1	Pre Q_0	Next Q_{1N}	Next Q_{0N}	D_1	D_0
0	0	0	0	0	0
0	1	1	0	1	0
1	0	0	1	0	1
1	1	1	1	1	1

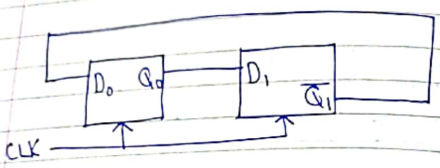


counting States

∴ Mod N Ring Counter = N D flipflops

- used to produce round robin scheduling.

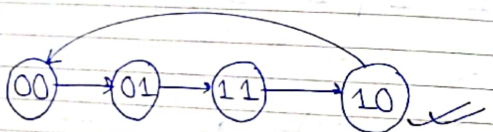
Mod 4 Johnson Counter



$D_0 = \overline{Q_1}$
$D_1 = Q_0$

Pre		Next			
Q_1	Q_0	Q_{1N}	Q_{0N}	D_1	D_0
0	0	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	1	1	0	1	0

00
01
11
10



- all 4 states involved in counting.

∴ **Mod 2N Johnson Counter = N flipflops**

- mod 2^N counter = not yet found.

Page No. 33 Date YOUVA

* Design Synchronous counter using T' flip flops.

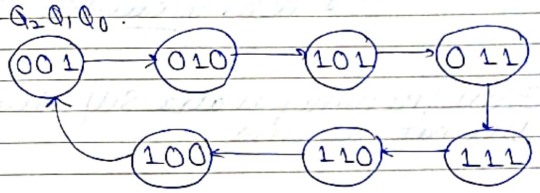
Ref: Pg 78-79

∴ If n States ⟶ $\lceil \log_2 n \rceil$ FFs are req.

- Mod 4 Gray counter is made using only D flip-flops mostly.

Ref → Pg 83

* $Q_2Q_1Q_0$.



Q if init state is 001?

after 25 CC ... ? ⇒ 7×3 + 4 = 111

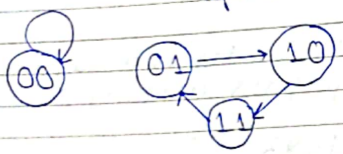
Q after 117 CC ? 117 mod 7 = 5

Tap at Q_2 ⟶ 1011100 | 1011100 | ... (Seq. Generator).

· Eg → Pg 88.

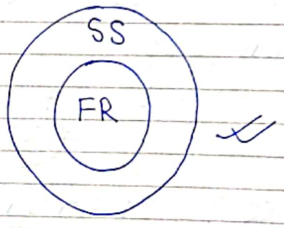
* Self Starting & Free Running

· **Self Starting**: if it is possible to enter the counting loop irrespective of the initial state.



· **Free running**: If it contains all possible states in counting loop.

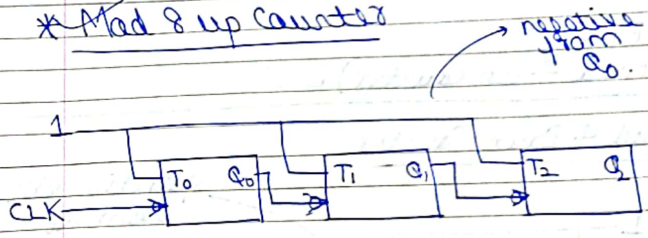
→ Every free running is also Self Starting but not vice-versa



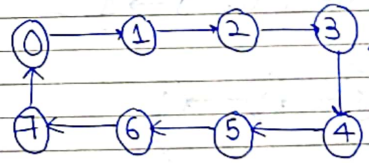
* Asynchronous Counter

- o/p of 1 FF, clock to another FF.
- T flipflop is used.

* Mod 8 up Counter

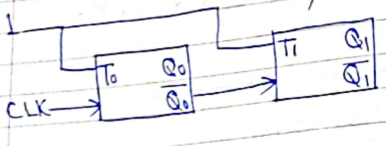


$Q_{0N} = \overline{Q_0}$, every clock
$Q_{1N} = \overline{Q_1}$, whenever $Q_0: 1 \rightarrow 0$
$Q_{2N} = \overline{Q_2}$, when $Q_1: 1 \rightarrow 0$



$\overline{Q_0}: 0 \rightarrow 1$
$\Downarrow$
$Q_0 = 1 \rightarrow 0$.

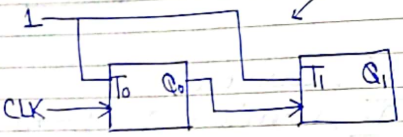
* Mod 4 up counter



positive edge from Q_0 = negative edge from Q_0.

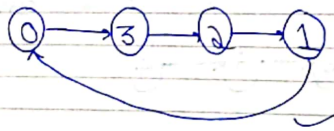
⇒ 2 FFs (mod 4 up counter).

* Mod 4 Down Counter



positive edge from Q_0 = negative edge from Q_0.

$Q_{0N} = \overline{Q_0}$; for every clock.
$Q_{1N} = Q_1$; when $Q_0 : 0 \rightarrow 1$

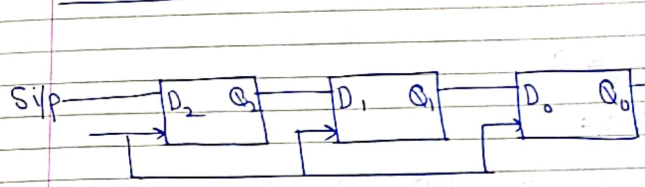


Ex → Pg 104.

* Applications of FFs

- Shift register
  - Sequential mem.
  - accumulator
- Counter
  - used to count no. of clock Pulses.

* 3 bit Shift Right register



Enter

Enter : 101

CLK	S/P	Q_2	Q_1	Q_0
0	-	0	0	0
1	1	1	0	0
2	0	0	1	0
3	1	1	0	1

Read

read : 101

CLK	Q_2	Q_1	Q_0	S O/P
0	1	0	1	1
1	0	1	0	0
2	0	0	1	1
-				

n-bit Shift register (SISO):

Entry: n clocks
Read: n clocks

∴ 2n−1 clock pulses

1 eq → Pg 107.

1 eq → Pg 117.

$Q_{IN} = Q_0$

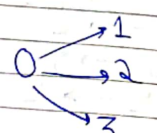
$Q_{ON} =$

Q_1	Q_0	Q_{IN}	Q_{ON}
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

If J=K=0; $Q_{ON} = Q_0$

Gate 2016

Seq: 0−1−0−2−0−3

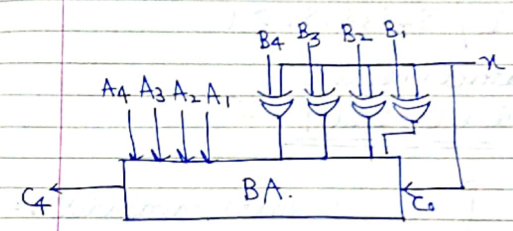


∴ 3 outgoing edges.

$\Rightarrow \lceil \log_2 3 \rceil = 2$

∴ add 2 more bits.

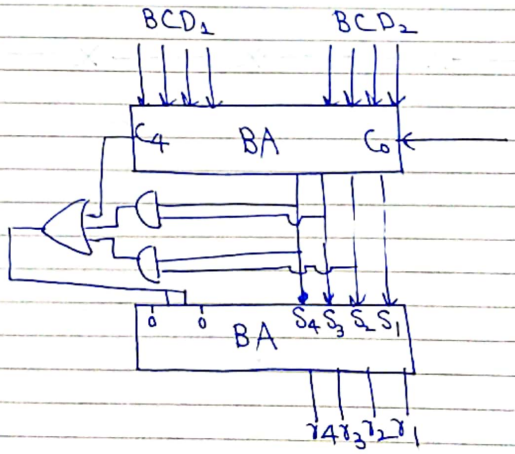
Binary Adder/Subtractor



if x=0 : A+B (Adder)

if x=1 : A−B (Subtractor).

BCD Adder

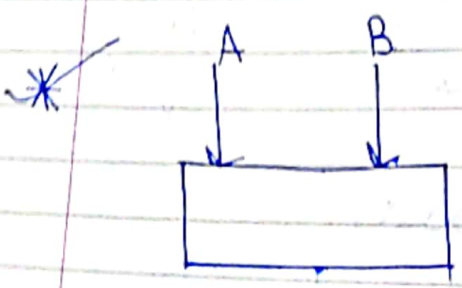


$f = S_4S_3 + S_4S_2 + C_4$

- If carry = 1 / $S_4S_3S_2S_1 > 9$.

∴ correctn needed

add 6 (0110) .



∴ Total combination

(16 x 16) .

Valid Comb = 10 x 10 .

∴ Invalid combination = 156

—— X —— X —— X ——