DIGITAL ELECTRONICS

* Basic Properties of Switching Algebra:

Boolean variable → {0, 1}

Operators → OR, AND, NOT.

(i) $x \cdot x = x$ $\quad x + x = x$

$x + 1 = 1$ $\quad x + 0 = x$

$x \cdot 0 = 0$ $\quad x \cdot 1 = x$.

(ii) $x + \bar{x} = 1$; $x \cdot \bar{x} = 0$ (Complemn)

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

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

(+ distr. over ·) // SA not Math.

* SA → SE (helps to design circuits)

Q $x + \bar{x}yz + x\bar{z}$.

∴ 6 literals, 3 terms.

* Properties for simplifying Switching expressions:

1. $x + xy = x$; $x \cdot (x + y) = x$ → Absorption
2. $x + \bar{x}y = x + y$; $x(\bar{x} + y) = xy$.

* Consensus Theorem

imp

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

yz: redundant term.

* De Morgan's law

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

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

* Complement ≠ Dual

$f(a,b,c\cdots,0,1,\cdot,+) \rightarrow \bar{f}(\bar{a},\bar{b},\bar{c}\cdots 1,0,\cdots +,\cdot)$

$\rightarrow f_d(a,b,c\cdots 1,0,\cdots +,\cdot)$

eg: $xy \xrightarrow{\bar{f}} \bar{x}+\bar{y}$ (imp)

$xy \xrightarrow{f_d} x+y$

* Switching Functions (i/p → o/p)
- SE act as functions.
  eg: $f(a,b) = (\bar{a}+\bar{b})\cdot ba$.
- Every funcn expr. as canonical form.

(i) Canonical SOP
- MinTerm: Product term which cont. each of n var as factors (comp/uncomp) (imp)
- Sum of all minT = canonical SOP / disjunctive NF

eg: $a+bc \rightarrow$ SOP. $abc + \bar{a}bc \rightarrow$ canon SOP.

$f = \Sigma(1,2,4,6)$

(ii) Canonical POS / conjunctive NF
- MaxTerm: Sum term which contains each of n var.
  Product of all maxT = Canonical POS

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

$f = \Pi(0,3)$ (imp)

NOTE: Canonical SOP & POS are 2 representation of same function.

* $\bar{x}y + z + xyz$ (SOP) → standard circuit

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

NOTE: 1. Many expressions rep same function, but SOP/POS: unique (canonical) (imp)
2. Two funct are equivalent, if SOP/POS forms are identical.

* No. of Functions
If n boolean variables, 2^n rows
$\therefore 2^{2^n}$ Boolean functions.

* If n K-ary variables; m-ary functn?
K^n rows (inputs)
$\therefore mK^n$ m-ary functions

* If K boolean variables, How many 'm' minterm functions: ${}^{2^K}C_m$
$\because 2^K$ rows

* Neutral functions: No of minTerm = no of maxTerm. (imp)
No of neutral functions $\Rightarrow {}^{2^n}C_{2^{n-1}}$

* Function Representations

(i) Venn diagram

imp

- if n Variable → 2^n regions.
- Only boolean funct^n (Shade / no Shade)

8 reg (3 var)

(ii) Contact representation

Series → AND
Parallel → OR

AC'C X. [not allowed].

4) NAND (↑) → $A \uparrow B = \overline{A \cdot B}$

(i) $A \uparrow A \neq A$. imp
(ii) commutative, not associative.

5) NOR (↓) → $A \downarrow B = \overline{A + B}$

(i) $A \downarrow A \neq A$
(ii) commutative, not associative.

6) EXOR → $A \oplus B = A\overline{B} + \overline{A}B$

- modulo 2 sum

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

(i) $A \oplus A \neq A = 0$.
(ii) Both associative and commutative.

7) EXNOR → $A \odot B = AB + \overline{A}\overline{B}$

[equality comparator]

- 2 bit comparator
- $A \oplus B = \overline{A \odot B}$

A	B	A ⊙ B
0	0	1
0	1	0
1	0	0
1	1	1

(ii) $A \odot A \neq A = 1$
(iii) both associative & commutative.

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

imp

- XOR = 1 ; for odd no of 1s
- XNOR = 1 ; for even no. of 0s
- when even i/ps, [XOR, XNOR] are complements; when odd i/ps, (XOR = XNOR)

→ $A \oplus B = \overline{A \odot B}$ (even).
$A \oplus B \oplus C = A \odot B \odot C$. (odd)

imp

NOTE: In a EXOR / EXNOR KMap, no. of minT = no. of MaxT. → [routing funct^n]

* FUNCTIONAL COMPLETENESS : Set of oper is FC, if every SF can be expres by means of operation in it.

{+, ·, −} → FC.
{+, −} / {·, −} → FC.

NOTE: complement can be extracted from eq^n, not derived.

eg → Pg 17 (Rev Copy) [f(A, B) = $\overline{A} + B$]

imp → EX-OR / EX-NOR → NFC

imp

{⊕, 1} → NOT; {⊙, 0} → NOT.

* SELF DUAL FUNCTION → f = fd

A BF is Self dual if:

(i) It is neutral function (imp)

(iii) F doesn't contain 2 mutually exclusive terms. [(0,7), (1,6), (2,5), (3,4)]

No of Self dual function = $2^{2^{n-1}}$ → 2 choices in each pair.

imp → - Self dual functns are closed under complementation. (NF, SD)

* ELECTRONIC GATES

- 2 Volts {H, L}

H → 1, L → 0 } PLS; H → 0, L → 1 } NLS.

- AND & OR, PLS and NLS are duals of each other.

② MINIMIZATION OF BF

* Minimal exp: Min no of terms in SOP exp provided there is no other such exp with same no of terms / fewer literals.

NOTE: Irredundant exp: can't reduce further

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

* K-MAP

- n var KMap: 2^n cells

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

→ Cyclic code

→ decimal values.

→ Subcube (2^m cell) [wy].

Size of SC = (n − m) literals

imp

NOTE: **No. of Product terms = No of SC**

No of literals / term = Size of SC (n − m)

SOP

* Covering functions

imp

g → f. [g implies f]

∴ f covers g. ; when g = 1, f also 1.

→ f contains all min Term in g.

No. of covering functions = $2^{2^n - k}$

eg: f(ab + c) = ab + c

(f) (f1) (f2)

∴ f1 → f, f2 → f

- Every SC is an implicant of F.

I → SC
PI → cant entirely be part of another PI
EPI → if it covers atleast 1 mT of f which isnt covered by any other PI

eg → Pg 81 [How to find EPI/PI]

* Cyclic K Map → no EPI (Vimp)

NOTE: If all 1s are covered by the EPIs then its an unique minimal exp.

* Dont cares : some i/p never occur
- Unspecified function: same funct are not defined for same i/p comb^n
  If K DC combinations: f $\xrightarrow{\text{repres.}}$ class of 2^K functions.

eg → Pg 93

eg → Pg 97 [PI chart example] (imp) → 2 minimal exp

* If a function has minT & mutually exclusive term, it cant be made self dual.
  If 3 Var → Sum in Pair: 7
  If 4 Var → Sum in Pair: 15

eg → Pg 105 [Q2]

eg → Pg 106 [Imp]
PI − EPI = RPI
(0,15), (1,14), (2,13) ...

* Variable Entered K-Map
- introduce a var in K-Map to reduce its size.

eg → Pg 108X, 110
[minimization of VKM]

* without DCS
SOP, POS → f

with DCS (imp)
SOP → f_1
POS → f_2.

NOTE: If minimal SOP independent of w, y ; POS is also indep. of w & y.

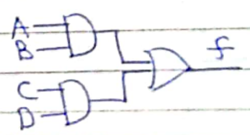
eg → Pg 119 [Vimp Q] (Vimp)

③ DESIGN & SYNTHESIS OF COMBINATIONAL CIRCUITS

- [SA → SE → Design of digital circuits (using Gates)] (SF)

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

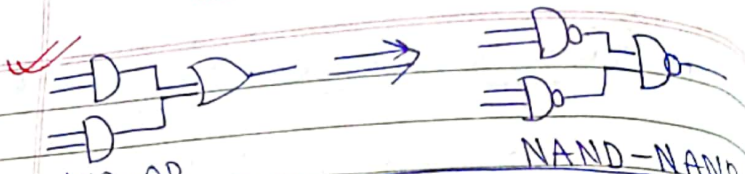
(imp)



(AND-OR realization)

* We need to minimize no. of Gates
- heat prod ↓
- cost/time ↓
- Space ↓

imp → $Q = D$

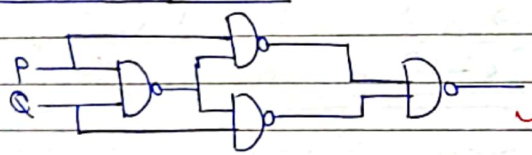


imp →
- SOP → AND-OR → NAND-NAND
- POS → OR-AND → NOR-NOR

eg → Pg 137 [Count no. of NAND Gates]
eg → Pg 139 (P+QR) [OR-AND realz]

* NAND & NOR as Universal Gates
→ functionally complete.

EXOR using NAND



imp →

	NAND	NOR
EXOR	4	5
EXNOR	5	4

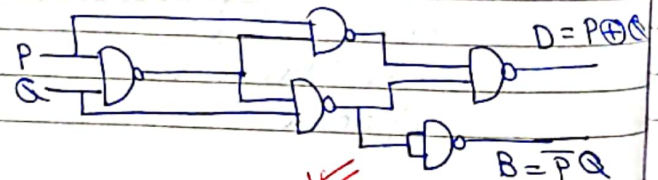
Vimp

* Half Subtractor

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

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

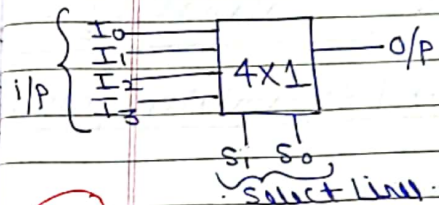
imp →



1) * MULTIPLEXER [Electronic Switch]

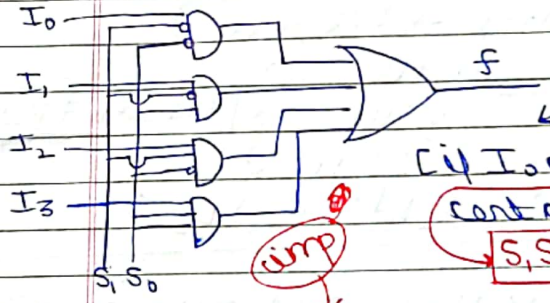
- connect ant out of 2^n i/p to o/p.
- functionally complete.

$2^n \times 1$ Mux
i/p → 2^n lines, bits → n, No of SL : n



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

imp → $f = \overline{S_1}\overline{S_0}I_0 + \overline{S_1}S_0I_1 + S_1\overline{S_0}I_2 + S_1S_0I_3$
[Characteristic equation]



4X1 Mux
[2^n AND gates]

[if I_0 comes out, $I_{1,2,3}$ can't come out]
$S_1S_0 = 00$

imp

Q Implement EXOR using mux.

$EXOR = \underbrace{\overline{A}B}_{01} + \underbrace{A\overline{B}}_{10}$

$I_1 = I_2 = 0$
$I_0 = I_3 = 1$

eg → Pg 151 [From char. eqn] → Truth Table.

$2^n \times 1$ mux can implement n var. functn without Gates, and also > n var. function (with extra Gates)

imp

enable

$Y = E(\bar{S}_1\bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1\bar{S}_0 I_2 + S_1 S_0 I_3)$

if E = 0 ; no response

E = 1 ; AND Gates react & give O/P

eg → Pg 158 [i/p of mux depends on Select lines] (imp)

eg → Pg 160 [Cascading multiplexer]

Expansion of Mux [Scalibility of mux]

Q 8x1 from 2x1 mux.

2x1 mux → 2 lines

∴ 1 mux → 2 lines

1 line → 1/2 mux

8 lines — 4 Mux

4 lines — 2 Mux

2 lines — 1 Mux

3 levels / 7 Mux

eg → Pg 36 [Rev Copy] (Diag)

mx1 mux using nx1 mux (imp)

$K = \lceil \log_n m \rceil$: (no. of levels)

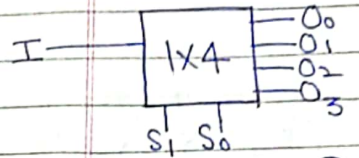
$D = \sum_{K=1}^{\log_n m} \left(\frac{m}{n^K}\right)$ (no. of devices)

Q 32x1 from 4x1 (half patented of mux used) (imp)

$K = \lceil \log_4 32 \rceil = 3$

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

2) DEMULTIPLEXER [One to Many]

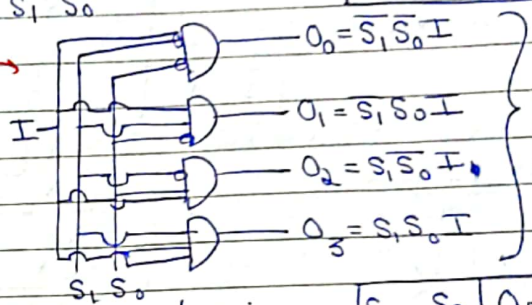


1 i/p — 1x2^n

2^n o/ps ;

where n = No of SL.

(imp)



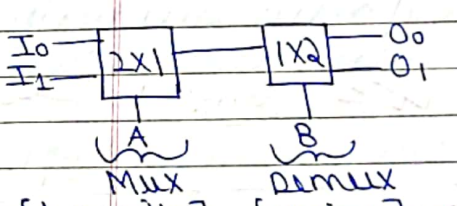
2^n AND Gates

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

(imp) SL selects o/p line

remove OR gate from mux. All i/p combine to a single i/p.

→ used in construction of Switches.



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

cross

→ No 2 o/p lines will show I at the same time in the case of demux.

(imp) Mux → Selects i/p line

Demux → Selects o/p line

- active low decoder with NAND = active high decoder with OR = SOP
- active low decoder with AND = POS

(imp)

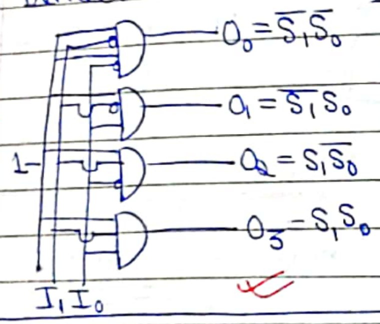
3) * DECODER [$n \times 2^n$ Decoder]

- No select lines.

I_1	I_0	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- Demux → Decoder (I = 1)

2^n AND Gates



[∵ 2 select lines]
all min Terms with 2 Var.
[active high decoder].
(imp)

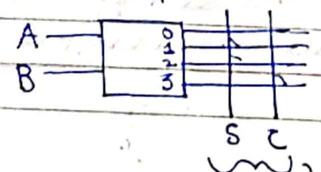
eg → Pg 175 [Implement any function in canonical SOP using OR]

eg → Pg 175 [In case of active low decoder use NAND Gate outside to realise func]

eg → Pg 177, Pg 179 (imp)
[active high & low decoders] (Imp)

* Decoder converts 1 code to another.
[BCD → excess 3 etc]

* ROM Implementation using Decoder



if n var : $n \times 2^n$ decoder

ROM = decoder + Matrix

if : m functn of n variables → - $n \times 2^n$ decoder.
- Horiz. lines = 2^n.
- Vertical lines = m
- No of links = $2^n \times m$
- $2^n + m$ connections
(imp)

⇒ If we implement functions using mux + decoder; links stay constant but connect^n s ↓ (imp)

eg → Pg 187 & Pg 45 (Rev Copy)

* Enable i/p are used for expansion of decoders

eg → Pg 201 [Enable + i/p] (imp)

* Construct 3X8 using 1X2 decoder or 2X4 using 1X2.

eg → Pg 203 [Decoder decides which o/p line is to be selected].

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

$K \geq \log_n m$; $n_D = \sum_{K=1}^{\log_n m} \left(\frac{m}{n^K}\right)$

eg: 6X64 using 4X16.
$K = \lceil \log_{16} 64 \rceil = \lceil 3/2 \rceil = 2$ levels

(imp)

$n_D = \frac{64}{16} + \frac{64}{16^2} \Rightarrow 5$ devices.
→ $\frac{m}{n^K}$

eg → Pg 210 [Address ranges; expansion of decoder]

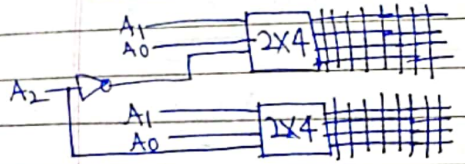
* ROM [Read only Memory]

imp

4W x 8bits → Size of ROM

2^n

(i) Increase no. of words (Address ranges)



(ii) Increase no. of wordsize (bits)
- just remove the NOT Gate.

eg → Pg 216 [Address range]

4) * ENCODERS [2^n x 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

imp

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

→ Convert 1 No System to another Number System.

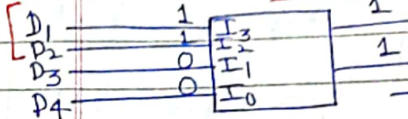
→ Non Priority encoders doesn't support Simultaneous i/p activation.

imp

* Priority Encoders → simultaneous i/p activation.

- used in interrupt servicing.

imp

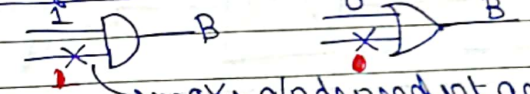


$\therefore I_3 > I_2 > I_1 > I_0$ [Priority]

* HAZARDS → Temporary (through delays)
→ Permanent → SC, OC

* To test whether load is working: Single fault analysis.

Gate



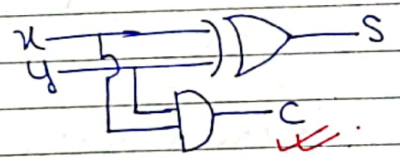
S-a-0 / S-a-1

eg → Pg 226

6 Addrs

(1) * HALF ADDER : 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



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

(2) * 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

$S = x \oplus y \oplus z$

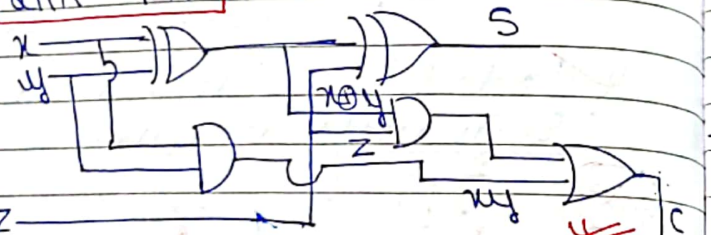
$C = xy + yz + zx$

or

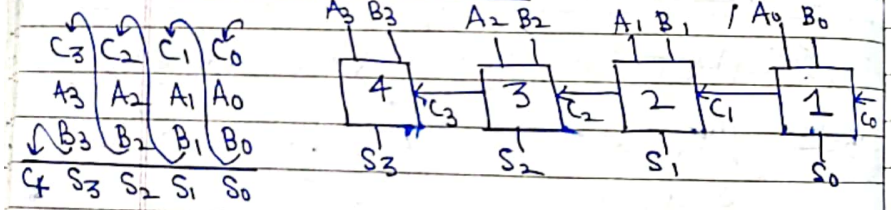
$(x \oplus y)z + xy$

imp

* 2HA = FA



(3) * RIPPLE CARRY ADDER → Time = $\Theta(n)$



→ Carry ripples through the FAs

Drawback: Takes lot of time

If n bits → n FAs ∴ $T = n \times T_{FA}$

Vimp

(4) * Carry Lookahead Adder

$T \geq \Theta(\log_K n)$; n = size of i/p.

K = FAN in of AND Gate

imp

Generate funct^n — Propagation funct^n

$G_i = A_i B_i$; $P_i = A_i \oplus B_i$ imp

$C_1 = C_0 P_0 + G_0$

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

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

$C_4 = C_0 P_0 P_1 P_2 P_3 + G_0 P_1 P_2 P_3 + G_1 P_2 P_3 + G_2 P_3 + G_3$ imp

Carry terms depend entirely on i/ps, not on other carry.

Drawback: complex & costly but faster.

- No rippling effect.

eg → Pg 57 CP in diag

$\Theta = \log_K n$

(5) * Hybrid Adder

- Mix of both (CLA & RCA)

$C_3\ C_2\ C_1\ C_0$
$A_4\ A_3\ A_2\ A_1$
$B_4\ B_3\ B_2\ B_1$
$C_4\ S_4\ S_3\ S_2\ S_1$

$A_8\ A_7\ A_6\ A_5$
$B_8\ B_7\ B_6\ B_5$
$S_8\ S_7\ S_6\ S_5$

⇒ 8 bit Hybrid Adder = 2 4 bit CLA Adder

(6) * Serial Adder [made of D flipflops]

- Using 1 FA, we add 2 nos using Shift registers. imp

Slow compared to all adders but cheapest.

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

$S = x \oplus y \oplus z$

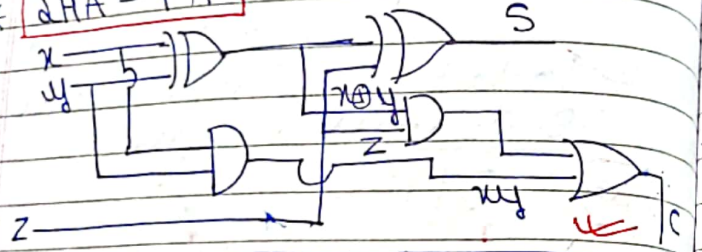
$C = xy + yz + zx$

or

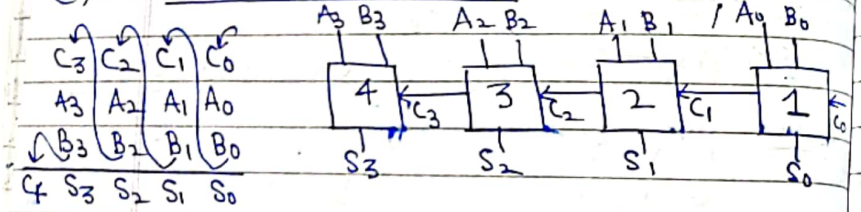
$(x \oplus y)z + xy$

imp

* 2HA = FA



(3) * RIPPLE CARRY ADDER → Time = Θ(n)



→ Carry ripples through the FAs

Drawback: Takes lot of time

If n bits → n FAs ∴ $T = n \times T_{FA}$

Vimp

(4) * Carry Lookahead Adder

imp

$T \geq \Theta(\log_K n)$; n = size of i/p.

K = FAN in of AND Gate

Generator func^n → $G_i = A_i B_i$; Propagator func^n → $P_i = A_i \oplus B_i$ imp

$\therefore$

$C_1 = C_0 P_0 + G_0$

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

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

$C_4 = C_0 P_0 P_1 P_2 P_3 + G_0 P_1 P_2 P_3 + G_1 P_2 P_3 + G_2 P_3 + G_3$ imp

Carry terms depend entirely on i/p's, not on other carry.

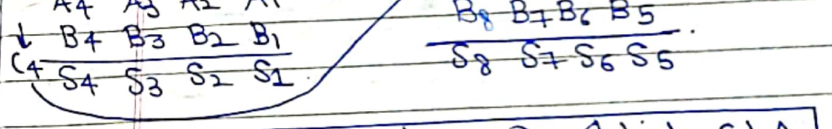
Drawback: complex & costly but faster.

eg → Pg 57 [Pin diag]

$\Theta = \log_K n$

(5) * Hybrid Adder

- Mix of both (CLA & RCA).



⇒ 8 bit Hybrid Adder = 2 4 bit CLA Adder

(6) * Serial Adder [mode of 0 flipflops]

- Using 1 FA, we add 2 nos using Shift registers. imp

Slow compared to all adders but cheapest.

* COMPARATORS

(ii) 2 bit comparator : compare 2 bit unsigned no.s $[A\{A_2A_1\}, B\{B_2B_1\}$

2 bit comparator = $2^{2\cdot2}$ values

$x_1 = A_1 \odot B_1$, $x_2 = A_2 \odot B_2$

3 cases : equal, greater, less.

$[A_3A_2A_1]$ $[B_3B_2B_1]$

(iii) 3 bit comparator : $2^{2\cdot3} = 2^6 = 64$ Values

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

→ $A = B$: $x_1 x_2 x_3 = 1$ $\{A_3A_2A_1\}\{B_3B_2B_1\}$

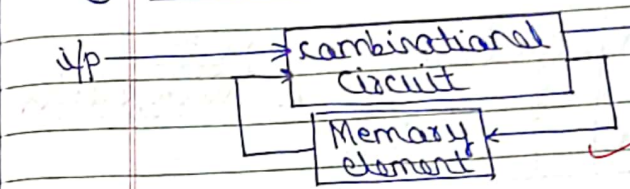
→ $A > B$: $A_3\overline{B_3} + x_3A_2\overline{B_2} + x_3x_2A_1\overline{B_1}$
$= 1$

→ $A < B$: $\overline{A_3}B_3 + x_3\overline{A_2}B_2 + x_3x_2\overline{A_1}B_1 = 1$.

∴ For n bit comparator : (2n bits)
∴ Total combinations = 2^{2n}.

- $A = B$: 2^n combinations (imp)
- $A < B$: $\frac{2^{2n} - 2^n}{2}$ combinations
- $A > B$: $\frac{2^{2n} - 2^n}{2}$ combinations

(4) SEQUENTIAL CIRCUITS



→ o/p depends on i/p & previous state also.
→ involves feedback.

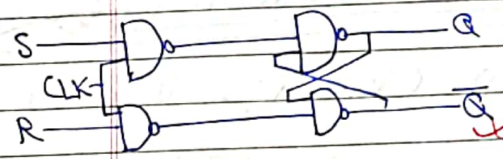
Smallest Memory element → flipflop (1 bit)

bistable multivibrator [2 states ∴ multi State Vibrator]

* latch used to build a FF.

(1) SR-FlipFlop

$Q_{n+1} = f(S, R, Q_n)$
↳ next State.



imp

Characteristic Table

S	R	Q_n	Q_{n+1}	
0	0	0	0	latch
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	∅	Invalid
1	1	1	∅	

$Q_{n+1} = S + \overline{R}Q_n$

Excitation Table

Q_n	Q_{n+1}	S	R
0	0	0	∅
0	1	1	0
1	0	0	1
1	1	∅	0

{1000}

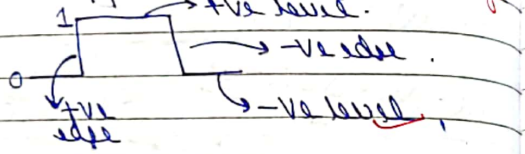
Date Page 22

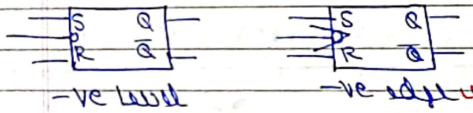
Function Table → from Char Table

S	R	Q_{n+1}
0	0	Q_n (Latch)
0	1	0 (Reset)
1	0	1 (Set)
1	1	ϕ (Invalid)

VEM
when, var : Q_n

↳ +ve edge triggered FF activates/reacts to i/p only during +ve edge (level).





-ve level -ve edge

eg → Pg 267 | Pg 269 [Clock diagrams] imp

(2) JK FlipFlop (derived for J=K=1)

Char Table

J	K	Q_n	Q_{n+1}	
0	0	0	0	} latch/Hold
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	1	} toggle/Race Around condn.
1	1	1	0	

Funct Table

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

$\therefore$ $Q_{n+1} = J\bar{Q}_n + KQ_n$ imp

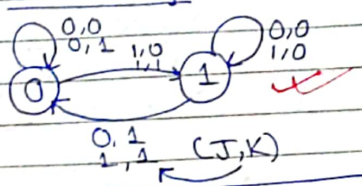
Date Page 23

classmate

• Excitation Table

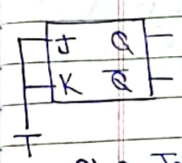
Q_n	Q_{n+1}	J	K
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

• State dig [From excitn Table]



(3) T-FlipFlop

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



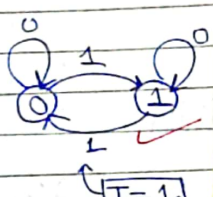
T	Q_{n+1}
0	Q_n → Latch
1	$\bar{Q}_n$ → Complement

Char Table

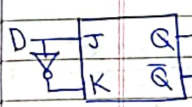
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excit Table

$Q_n \to Q_{n+1}$	T
0 → 0	0
0 → 1	1
1 → 0	1
1 → 1	0



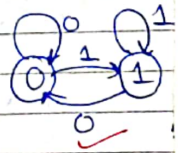
(4) D-FlipFlop



D	Q_{n+1}
0	0 (Reset)
1	1 (Set)

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

$Q_n \to Q_{n+1}$	D
0 → 0	0
0 → 1	1
1 → 0	0
1 → 1	1



$\therefore$ $Q_{n+1} = D$ imp

[edge 1 delay]

→ used to correct uneven prop delay.

→ Always derive Functⁿ table from characteristic eqⁿ. (imp)
State diag from excitation table.

eg → Pg 278 [from Funct Table also]

* FlipFlop Conversion

- Get the char table of the target FF.
- Replace next State using excitation of the given FF.

eg → Pg 284 (imp)

- Functⁿ Table from State diag.

eg → Pg 290 [XY using JK flipflop] (↑ target)

(5) Number SYSTEM

- How we represent numbers.
  - → Base n = n digits
  - → All digits must be less than base.
  - → Every digit : single literal

* Conversion to base 10:

- $(abc)_x = (ax^2 + bx + c)_{10}$
- $(1010)_2 \Rightarrow (10)_{10}$ } Shortcut

* Conversion from base 10

2	11	
2	5	1
2	2	1
2	1	0
	0	1

$\Rightarrow (1011)_2$

if, $(1056)_{16} \longrightarrow (\ )_2$

$\therefore 16^1 = 2^4$

[⇒ every digit in B-16 is rep by 4 digits in Base-2.]

eg → Pg 305 [Convert to big Base, then 2]

* Max Value of Binary no. when converted to Base 10:

$111 \longrightarrow 2^3 - 1 = 7$

(imp) $\therefore$ $\boxed{MaxV = b^n - 1}$, b = base, n = no of bits.
↳ in decimal

eg → Pg 310

(imp) if, 10 digit → Base 4, how many digits rep in base 2.

$\therefore 2^n - 1 \geq 4^{10} - 1 \quad \therefore n \geq 10 \lg_2 4$

eg → Pg 316 $\boxed{(123)_5 = (x8)_y}$

- Base can never be negative

* Complementary Number System

- We can do subt. with additⁿ circuit.
  - → diminished radix compliment
    $(b-1s)$ comp $= \bar{x} = (b^n - 1) - x$
  - → radix complement (imp)
    bs comp $= \bar{x} = b^n - x$.

radix compliment

eg → Pg 327 [2s complement]

$y - x = y + \bar{x}$ → found using complement of y No. System.

In: (b−1)s complement → 2 rep of 0
bs complement → 1 rep of 0 (imp)

* Subt in (b−1)s complement

- if carry: discard it & add 1.
  no carry: again find (b−1)s comp of result & negate it.

eg → Pg 341 [Q 2] (make it same no of bits) [carry found]

* Subtraction in bs complement ($x = b^n - x$)

- if carry: ignore it.
  no carry: take bs complement & negate it. (imp)

eg → Pg 346 [10s complement]

∴ 2s complement > 1s complement [popular]

* Binary numbers
  → unsigned no.s
  → Signed no. → Signed Mag / 1s comp / 2s comp. (Vimp)

Signed nos.

	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

∵ 1 → signed (signed represent)

* In Signed Magnitude: 2 rep of 0. (000, 100)
* In 1s comp: 2 rep of 0. ∵ [000 → 0, 111 → 0]
* 2s complement: range is more & only 1 comb for 0.

eg → Pg 360 [−7 → 0111 → 1001 (2s comp)]

* Code → Weighted (SM, 2S) / → Unweighted (1s) (imp)

if 2s: (Weighted)
1 1 1 = −4 + 2 + 1 = −1

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 $2^{n-1}-1$

if n = 4 → SM: [−7, 7]
1s: [−7, 7]
2s: [−8, 7] (more R) (imp)

Sign Bit Extension

⇒ In Signed Magnitude, if we want to extend, we take the Signed Bit & put it in MSB.

eg: 1 1 1 0
↓ ↓ ↓
10000110 [8 bit register]

⇒ In 1s-comp, take SB & copy it first 4 bits. + 1001 = −6.

11111001

⇒ In 2s comp, if we have a chain of bits in MSB, we can del all 1s, till we get one 1.

eg: 111101 ⟶ 101 ⇒ −3

Overflow (imp)

- if nbit + nbit ⟶ max(n+1 bit) (carry)

(i) In unsigned, carry indicates overflow

(ii) In signed bit, overflow is indicated
- 2 +ve nos add ⟶ result: −ve.
- 2 −ve nos add ⟶ result: +ve.

Cin | A3 A2 A1 A0
B3 B2 B1 B0
out

if, Cin ≠ Cout (overflow)
Cin = Cout (no overflow) (imp)

Binary Codes

- Numeric ⟶ BCD
- Alphanumeric
  - ASCII
  - EBCDIC

- Weighted rep ⟶ formula
- unweighted ⟶ no formula

BCD (imp)

- Weighted (8421, 3321, 2421)
- Non-Weighted (Gray / Excess 3)

eg ⟶ Pg 14 [3 codes] (8421, excess 3, 3321) [Rev copy] (imp)

- 3321 ⟶ Self Complimenting code

NOTE: If a code has to be Self complim., Sum == 9, converse not true.

eg ⟶ Pg 4 & Pg 5 [on BCD, excess 3, 2241] (imp)

(1) BCD Addition: Add 6 for correction when: (i) invalid [when > 9] (ii) carry is generated.

eg ⟶ Pg 8

(2) Excess 3 Addition:
- If carry: add (+3)
- If no carry: subtract (−3).

eg ⟶ Pg 9 [Q2] (imp)

Gray Code (unweighted BCD)

- reflexive
- unit distance code
- Cyclic code [used in KMap]
- Non-weighted code

eg → Pg 17 [Rev Copy]

∴ Gray code takes less bits than BCD

* Binary to Gray & Vice-Versa

(i) Binary → Gray (BG)

1100 = 12 (b4 b3 b2 b1)

1010 (Gray) (g4 g3 g2 g1)

∴

g4 = b4
g3 = b4 ⊕ b3
g2 = b3 ⊕ b2
g1 = b2 ⊕ b1

(ii) Gray → Binary (GB)

1010 (G4 G3 G2 G1)

1100 (B4 B3 B2 B1)

B4 = G4
B3 = G4 ⊕ G3
B2 = G4 ⊕ G3 ⊕ G2
B1 = G4 ⊕ G3 ⊕ G2 ⊕ G1

imp → Test (circuit)

* Error Detection

- To detect 1 bit error, the dist. b/w every 2 valid codes must atleast be 2.

eg → Pg 19 [1 bit error]

if t bit error ⇒ the distance should be atleast (t+1) detection

* Error Correction

Vimp [Hamming dist.]

if t bit error ⇒ dist should be atleast (2t+1) correction

* Floating Point Conversion

(i) Binary → decimal

110.101 → 6.625 [$6 + 5/2^3$]

($2^2\ 2^1\ 2^0 . 2^{-1}\ 2^{-2}\ 2^{-3}$)

(ii) Fraction → Binary (Gate) [always done]

eg → Pg 34 (0.625)

* Floating Point Representation

- For some FP, binary no. always possible but for every BN, FP ~~not~~ possible always. (imp)

for every FP → binary not possible.

→ Normalization: Just 1 representation ∀ every no.

imp →

bias = 2^{n-1} ; n = exponent bits

∴ comparing 2s complement : unne

explicit Normalisation

eg: 0.101×2^{-2} → M = 1010, E = −2 + 16 = 14, S = 0

imp: E = 5, S = 1, M = 4

bias = $2^4 - 1 = 15$

put in bias

* Implicit Normalisn: Move RP to the right of first 1.

$(-1)^S \times 1.M \times 2^{E-bias}$ (to get decim)

* Explicit Normalisn: Move RP to the left of first 1.

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

eg → Pg 44 [Largest decimal no. given bits of SEM] imp

[0.11111111] → $(1 - 1/2^8)$

eg → Pg 47 [if nos round to 0, usually Packed]

imp → Mantissa → Normalised SM form

ii. | S | E | M | (1 | K | n−1−K) ∴ Bias = $2^{K-1} - 1$

* IEEE STANDARDS

1985 Standard → Single Precision (32), Double Precision (64)

(1) Single Precision (32 bit)

S	E	M
1	8	23

(32)

Excess 127

*

E(8)	M(23)	Value
000...0	00...0	±0
111...1	00...0	±∞
1 ≤ E < 254	xx...x	Implicit NF
E = 0	M ≠ 0	Fractional
E = 255	M ≠ 0	NAN

imp:

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

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

F. Nor F = $(-1)^S \times 0.M \times 2^{-126}$

(2) Double Precision (64)

S	E	M
1	11	52

− excess 1023.

E = 0, M = 0, ± 0

E = 2047, M = 0, ± ∞

E ≠ 0, M ≠ 0, INF

E = 0, M ≠ 0, Fractional

E = 2047, M ≠ 0, NAN

∴ INF = $(-1)^S \times 1.M \times 2^{E-1023}$

FNF = $(-1)^S \times 0.M \times 2^{-1022}$

eg → Pg 56 (Represent 117) imp

* If we multiply n bits, Space $O(T) = O(2^n)$

Booth Alg reduces it.

$O(n^{log_2 3})$ → (Time complexity)

* BOOTHE'S ALGORITHM

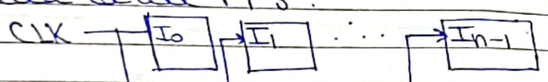
→ decreases no of additions. [Smallest & Largest +ve no.]

eg → Pg 64/65 (imp)

(4) * SEQUENTIAL CIRCUITS

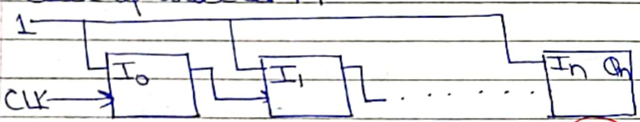
* Counters → Sync. / → Async

(i) Synchronous Counter : common clock applied to all FFs.



(imp) $T_{PD\text{-}Syn} = T_{FF} + T_{comb}$ (prop-delay)

(ii) Asynchronous Counter :- o/p of 1 FF → clock of another FF.



$T_{PD\text{-}asyn} = N \times T_{FF} + T_{comb}$ (imp)

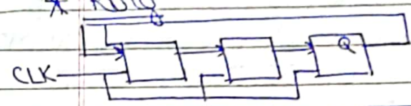
∴ Synchronous is faster than Asynch.

(1) * SHIFT Counter (Synchronous) — D flipflop used (imp)

→ Ring $[I_0 = Q_{n-1}]$

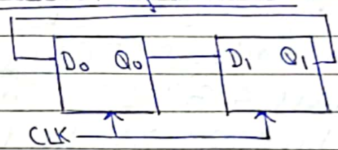
→ Johnson $[I_0 = \bar{Q}_{n-1}]$

* Ring



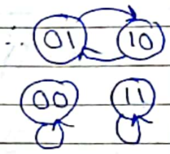
Johnson

(i) Mod 2 Ring counter (2 counting states)



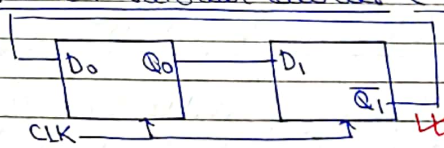
$D_1 = Q_0$
$D_0 = Q_1$.

Q_1	Q_0	Q_{1N}	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



(imp) mod N Ring counter = N D-Flipflops

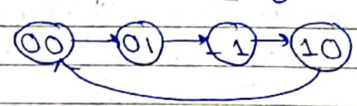
* Mod 4 Johnson Counter (4 count states)



$D_0 = \bar{Q}_1$
$D_1 = Q_0$.

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

∴



(imp) mod N Johnson counter = N/2 D-Flipflops

With 4 FFs, we can create mod 4 ring counter, but mod 8 Johnson Counter

eg → Pg 78-79 [Excitation column] [Design Synchronous Count using T flipflops] (imp)

if n States → $\log_2 n$ FFs required

eg → Pg 83 [Mod 4 G] (imp)
(T FF for MSB, XY for LSB).

* If init. State is 001 ?
after 25 CC → ? , & Sequence Generator:

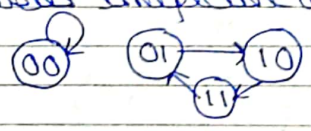
eg → Pg 33 (Rev Copy)

$T_{PD} = T_{FF} + T_{comb}$

eg → Pg 88 [Deriving clock frequency for Synchronous circuits]

* Self Starting & free running

Self Starting: possible to enter count States irrespective of initial State. (imp)



×

Free running: if it contains all possible States in counting loop.

NOTE:

→ Mod 4 Johnson Counter [2 D flipflops]

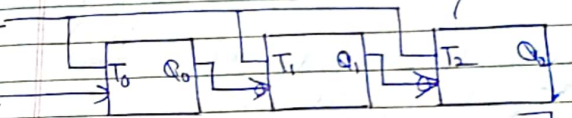
Synchronous Counter → T flipflop
Asynchronous Counter → T flipflop
Shift register → D flipflop (imp)

② ASYNCHRONOUS COUNTER

- O/p of 1 FF → clock to another FF
- T flipflop used.

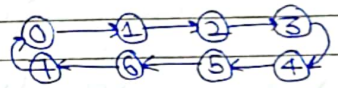
(i) Mod 8 up counter

→ 8 counting States [Free running].



(imp)

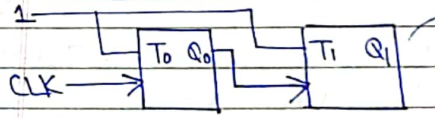
$Q_{0N} = \overline{Q_0}$, every clock
$Q_{1N} = \overline{Q_1}$, when $Q_0 : 1 \to 0$.
$Q_{2N} = \overline{Q_2}$, when $Q_1 : 1 \to 0$.



∴ 8 counting States.

∴ Mod n up counter = $\log_2 n$ FFs.
(n States)

(iii) Mod 4 down counter (imp)



$Q_{0N} = \overline{Q_0}$, for every clock
$Q_{1N} = \overline{Q_1}$, when $Q_0 : 0 \to 1$

[free running] 

∴ 4 counting States.

* Applications of FF

- Shift register [Serial adder]
- Counter [Count no of clock

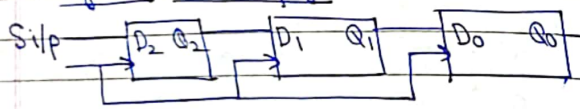
eg → Pg 104 [Mod 8 Random counter designing]

* Shift register : used as sequential memory accumulators [CO]

(iii) 3 bit Shift register → 3 D flipflops (imp)

enter 101, read 101.

eg → Pg 37 [Rev Copy]



NOTE: If n bit Shift register (SISO)
Entry → n clocks
Read → n−1 clocks (imp)
∴ 2n−1 clock Pulses

eg → Pg 107 [Shift register] (imp) [Procedure]

eg → Pg 117 [To construct bit Sequence, follow this format]

Q_1	Q_0	Q_{1N}	Q_{0N}
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

(imp)

$Q_{1N} = Q_0$
$Q_{0N} = \overline{Q_0}$;
J=K=1
$= \overline{Q_0}$,
J=K=0

Construct state diagram out of this.

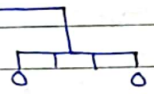
* Gate 2016 Q. [4 FFs required]
If sequence is: 0 → 1 → 0 → 2 → 0 → 3
Clock: 0 → 1, 0 → 2, 0 → 3 ∴ 3 outgoing edges.
∴ $\lceil \log_2 3 \rceil = 2$
∴ Add 2 more bits
∴ 4 bits total (imp)

* Binary Adder/Subtractor

eg → Pg 39 if x=0, A+B (Gate)
x=1, A−B

* BCD Adder (imp)

$f = S_4S_3 + S_4S_2 + C_4$



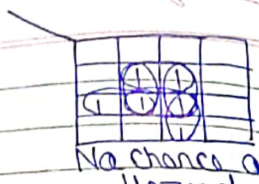
if $\begin{bmatrix} C_4 = 1 \text{ or} \\ S_4S_3S_2S_1 > 9 \end{bmatrix}$ ⇒ 1 (invalid).
∴ correction needed ∴ add 6 (0110).

$A_4A_3A_2A_1$ $B_4B_3B_2B_1$
Total combination = 16 × 16
Valid combination = 10 × 10

* PREVIOUS YEARS

eg → Pg 41 $f_3 = \Sigma(1,4,5)$
And → Intersection
OR → Union


Chance of Static Hazard


No chance of Static Hazard.

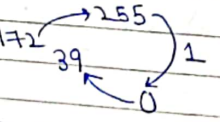
$T = \log_2 n$

* In a n bit Carry look ahead Adder,
  No. of AND gates = $n(n+1)/2$
  No. of OR Gates = n.

* EXOR → NFC $\{\oplus, 1\} \rightarrow$ NOT
  $\{\odot, 0\} \rightarrow$ NOT.
  ∴ EXOR, NOT → NFC
  2x1 mux → FC → PFC
  EXOR, AND → FC. NOT with 1.

* PLA: used to implement combinational circuit.
  RAM: Sequential circuit.

* 1 clock needed to change contents of a 8 bit upcounter from 172 to 39.
  ∴ (255 − 172) + 1 + 39 = 123.



* 00/11 lead to oscillation in RS FF.

* No of D flipflops to design a mod n up counter = $\log_2 n$

* 8087 arithmetic coprocessor is used for floating point arithmetic.

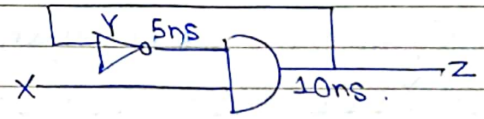
* ROM is used to store multiplication of 2 8bit unsigned no:s
  max space for result = 2n = 16.
  ∴ $2^{16} \times 16$ decoder.
  ∴ 64K X 16.

* Worst case of an implement of Booth is when pairs of 01s / 10s occur very frequently in multiplier.

* In a cyclic Prime Implicant K Map.
  No. of minimal exp = 2 [no comm PI]

eg → Pg 45 (Rev Copy).
3 2x1 mux required to implement a Half Subtractor.

eg → Pg 46 [7 NAND Gates req. to implement 4X1 mux]



(Doubt)

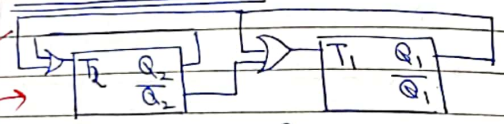
eg → Pg 49 : FA → 3, AND → 6.

TEST-1 (2019)

* Minim decimal equivalent of $(1AC)_x$
$\Rightarrow x^2+10x+12 \quad x > C = 12$.
$\Rightarrow 13^2+130+12 = 311$

* Q on clock diagram & time delay.
Ref → Q14 [Imp] Gate

TEST-2 (2019)

* 

imp
$Q_{2N} = T_2 = Q_2 + Q_1$
$Q_{1N} = T_1 = \overline{Q_2} + Q_1$
} Now construct Table & draw State diag.

* (−29) in recorded form (Booths Algo)
↓ 2s comp
Gate
$111000\underline{1}\,\underline{1}\,\underline{0}$
| 00−100+10−1 |

00/11	: 0
01	: +1
10	: −1

* $P = (A72E5)_{16}$: 2s complement form

$[1010\ 0111\ 0010\ 1110\ 0101]_2$

16XP : 4bit left shift

$\therefore [0111\ 0010\ 1110\ 0101\ 0000]_2$
↳ 2s complement.

imp
0111 0010 1110 0101 0000

0111 0010 1110 0100 1111
↳ 1s complement form

* The min. value of propagation delay in each FF, when a 10 bit ripple counter skip a count which is clocked at 10 MHz is ? imp

$T = n \times t_{FF}$

$\Rightarrow \dfrac{1}{10\times10^6} = 10 \times t_{FF}$ $\therefore$ $t_{FF} = 10ns$

[mod value of counter = 2]

* If a counter repeats (its initial state) after 2 Clock Pulses. (No of counts = 2)

Output freq = $\dfrac{\text{Input Frequency}}{2}$

$\therefore$ As, Clock Pulse ↑, Fout ↓ imp

* Diff. b/w 201 & next larger double Precision no.

$201 \rightarrow (1.1001001)_2 \times 2^7$ imp

0	7+1023	1001001
S	E(11)	M(52)

Next

0	7+1023	1001001 1
S	E(11)	M (52)

Diff = $(1.1001001 \ldots 1 \times 2^7) - (1.1001001 \times 2^7)$
2^{-52}
$\Rightarrow 2^7 [1.1001001..1 - 1.1001001]$
$\Rightarrow 2^7 \times 2^{-52} = 2^{-45}$

eg → Q11 [Binary to Gray Code converter using 8 bit Shift register]

* $Y = Q_2 \oplus Q_1 \oplus Q_0$ (imp)

CLK → FF_0. [Synchronous counter]

$1 = Q_0$ — CLK of FF_1.

CLK

$1 = Q_0$, $1 = Q_1$ — CLK of FF_2.

CLK

carry if $(Q_0 = Q_1 = 1)$

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

- $Q_{0N} = \overline{Q_0}$ (Always clock)
- $Q_{1N} = \overline{Q_1}$ $(Q_0 = 1)$
- $Q_{2N} = \overline{Q_2}$ $(Q_0 = Q_1 = 1)$

eg → Q12

* Lockout State: Counter can't enter into the counting states (even after n no. of clock pulses.

* MULTI TEST

* $(135)_6$ (Base 6). (imp) Find 6's complement. [mistake]

⇒
```
  555
 -135
 ----
  420   ← 5's comp.
 + 1
 ----
  421   ← 6's comp
```

* Half Subtractor is used to cant. Full Subtract or. 2HS = FS

Diff = $x \oplus y \oplus z$

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

Full Adder. Sum = $x \oplus y \oplus z$

Carry = $x(y \oplus z) + yz$

eg → Q22 [Coll-Dold Grp] Implementation of Synchronous counter using D flipflops & Mux

* FlipFlop with Preset & clear

$\overline{Pr}$	$\overline{Clr}$	CLK	J K	Q	$\overline{Q}$
0	0	X	X X	1	1
0	1	X	X X	1	0
1	0	X	X X	0	1
1	1	↓	0 0	Q_0	
1	1	↓	0 1	0	

*

$\overline{Pr}$	$\overline{Clr}$	CLK	J K	Q	$\overline{Q}$
0	1	X	X X	1	0
1	0	X	X X	0	1
0	0	X	X X	X	X
1	1	↑	0 0	Q	$\overline{Q}$

V imp

* Asynchronous counter with just clear.

eg → 010

A B C	A_N B_N C_N	$\overline{Clr} = \overline{BC}$
000	1 0 0	1
100	0 1 0	1
010	1 1 0	1
110	0 0 1	1
001	1 0 1	1
101	0 1 1	1
011	0 0 0	0

$A_N = \overline{A}$

$B_N = \overline{B}$, A: 1→0

$C_N = \overline{C}$; B: 1→0

clr = 1

∴ Mod 6 Counter.

* In Boothes Algo:

$(-121) \times (-113)$

imp → multiplicand, multiplier

Recoded Multiplier (Boothes)

⇒ recoded multiplier ?

⇒ 2s comp of 113 i.e −113 = 10001111

∴ 100011110

−100+1000−1

∴ [100 + 1000 − 1] · Ans

Same type

* Max no. of boolean exp that can be formed for the function:

$f(\overline{x}, y, \overline{z}) = f(x, y, z)$ → $2^{2^{n-1}}$

i.e. $\begin{matrix} x & y & z \\ 0 & 0 & 0 \end{matrix} = \begin{matrix} \overline{x} & y & \overline{z} \\ 1 & 0 & 1 \end{matrix}$ ∴ (0,5);

→ [Self dual like] (har pair ke $\frac{\text{ek}}{2}$ value)

Similarly, [P/A for each pair].

(0,5), (1,4), (2,7), (3,6).

are mutually exclusive.

∴ Choose 1 in each Pair. ∴ $2^4 = 16$ ways.

∴ if for $\Sigma(1,2,3 \ldots 7)$, $= 2^8 = 256$. [P/A for each elem]

total functions

Ripple carry Adder

* 16 bit ripple carry Adder using 16 FA.

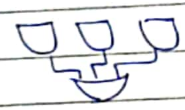
XOR = 20ns, AND = 15ns, OR = 10ns.

worst case delay?

⇒ [15 × Carry + Sum]

imp

Carry = AB + BC + CA



∴ AND + OR = 25.

Sum = 20.

∴ 15×25 + 20 = 395 ns

* 32 bit Ripple carry Adder, 32 FA.

Carry = 6ns, Sum = 14 ns.

worst case delay =

31×6 + 14 = 200ns

[$T_{pd} = (n-1)\,Carry + Sum$]