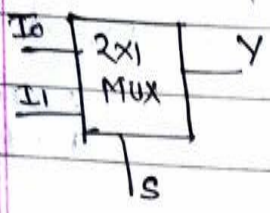
Subject DIGITAL ELECTRONICS

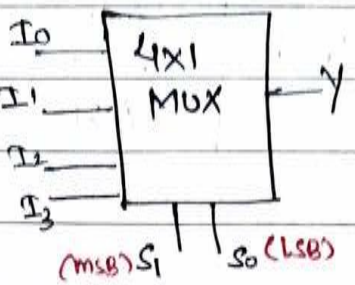
Index

Serial No	Date	Title	Page No	Teacher's Signature/Remarks
		Combinational ckts		
		Sequential ckts		
		Gates & Boolean algebra		
		Microprocessor		
		ADC & DAC		

Combinational circuits

(i) Multiplexer





S	Y
0	I_0
1	I_1

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

$Y = \bar{S} I_0 + S I_1$

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

Keypt: For 2 variables

(i) Some f^n of 2 variable can be designed using a 2x1 MUX but not all the f^n

(ii) All the f^n of 2 variables can be designed using a 2x1 MUX and a NOT gate.

(iii) All the function of 2 variable can be designed using a 4x1 MUX

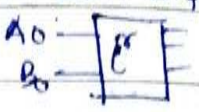
(i) Some but not all f^n of 3 var ⟶ 4x1 MUX

(ii) All $f^n(3)$ ⟶ using 4x1 MUX + NOT gate.

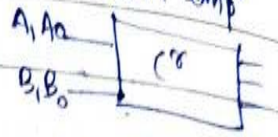
(iii) All $f^n(3)$ ⟶ using 8x1 MUX

(ii) Comparator

1 bit comp^r

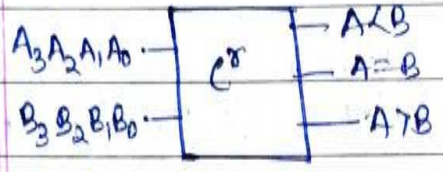


2 bit comp^r



$A = A_1A_0$ (= 10 eg)
$B = B_1B_0$

4 Bit Comparator



output expressions:

(i) $A > B$

either (a) $A_3 > B_3$ $\{A_3=1 \quad B_3=0\}$ $A_3\bar{B}_3$

OR (b) $(A_3 = B_3)$ & $(A_2 > B_2)$ $\{A_3 \odot B_3$ & $A_2\bar{B}_2\}$ $(A_3 \odot B_3)\cdot A_2\bar{B}_2$

OR (c) $(A_3 = B_3)$ & $(A_2 = B_2)$ & $(A_1 > B_1)$ $\{A_3 \odot B_3$ & $A_2 \odot B_2$ & $A_1\bar{B}_1\}$ $(A_3 \odot B_3)(A_2 \odot B_2)A_1\bar{B}_1$

OR (d) $(A_3 = B_3)$ & $(A_2 = B_2)$ & $(A_1 = B_1)$ & $(A_0 > B_0)$ $(A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)A_0\bar{B}_0$

expression:

$$Y_{(A>B)} = A_3\bar{B}_3 + (A_3 \odot B_3)A_2\bar{B}_2 + (A_3 \odot B_3)(A_2 \odot B_2)\bar{A}_1B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\bar{A}_0B_0$$

(ii) $A < B$

(a) $A_3 < B_3$ $\bar{A}_3B_3$

(b) $A_3 = B_3$ & $A_2 < B_2$ $(A_3 \odot B_3)\cdot\bar{A}_2B_2$ ... & so on

expression:

$$Y_{A<B} = \bar{A}_3B_3 + (A_3 \odot B_3)\bar{A}_2B_2 + (A_3 \odot B_3)(A_2 \odot B_2)(\bar{A}_1 \odot B_1) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\bar{A}_0B_0$$

(iii) $A = B$

$$Y_{A=B} = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

Note:

$\overbrace{A_{n-1}\ A_{n-2} \cdots A_0}^{n}$ $\overbrace{B_{n-1}\ B_{n-2} \cdots B_0}^{n}$

$\underbrace{\qquad\qquad\qquad\qquad}_{2n}$

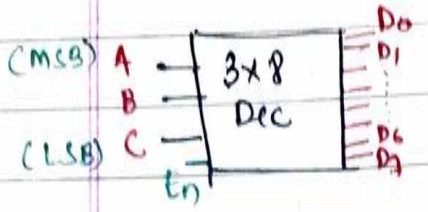
Total combination: 2^{2n}

Equal combination: 2^n

$A<B$ or $A>B$ } Unequal combination: $(2^{2n} - 2^n)$

$\because n(A<B) = n(A>B) = * \left(\frac{2^{2n} - 2^n}{2}\right)$

(iii) Decoder



$D_0 = \bar{A}\bar{B}\bar{C}$, $D_1 = \bar{A}\bar{B}C$ ...& so on.

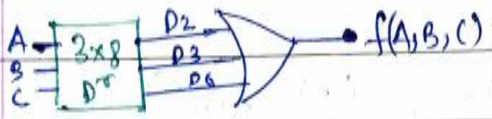
$E_n = 0 \Rightarrow D^n$ doesn't work

* eg: design $f(A,B,C) = \bar{A}B + B\bar{C}$ using 3×8 Dec^r

$f(A,B,C) = \Sigma m(01X, X10)$

$= \Sigma m(010, 011, 010, 110)$

$= \Sigma m(2,3,6)$



⊙ Decoder is universal in nature because it gives all the possible combinations / min. terms. So any f^n can be defined from Decoder.

Decoder with active low outputs

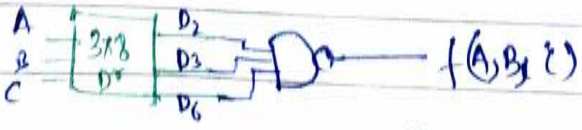
$D_0 = \overline{\bar{A}\bar{B}\bar{C}}$, $D_1 = \overline{\bar{A}\bar{B}C}$ ...& so on

* above eg:

$f(A,B,C) = \Sigma m(2,3,6) = \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$

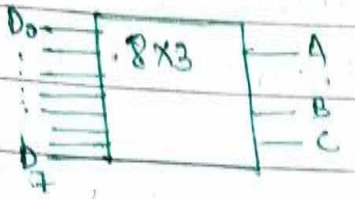
$= \overline{\overline{\bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}}}$

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



* Hence in case of D^n with Active low o/p just replace OR GATE by NAND gate.

(iv) Encoder



D0	D1	D2	D3	D4	D5	D6	D7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	1	1	1

A A = 1 when { 1XX → (1,00) (1,01) (1,10) (1,11) }
4 OR 5 OR 6 OR 7

$A = D_4 + D_5 + D_6 + D_7$ { X1X }.

$B = D_2 + D_3 + D_6 + D_7$ { XX1 }

$C = D_1 + D_3 + D_5 + D_7$

** If more than one i/p are ON at a time in Normal encoder then it will give wrong o/p

D_3	0	1	1
D_4	1	0	0
	1	1	1

⇒ o/p = 7 (wrong)

In this case we use priority encoder.

(vi) Priority encoder

If more than one input are given at a time then, only that input is taken which is highest.

i.e. If D_7 is pressed, no matter (D_0-D_6) are pressed or not

D_6 " ", no matter (D_0-D_5) are pressed/not

& so on

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
x	1	0	0	0	0	0	0	0	0	1
x	x	1	0	0	0	0	0	0	1	0
x	x	x	1	0	0	0	0	0	1	1
x	x	x	x	1	0	0	0	1	0	0
x	x	x	x	x	1	0	0	1	0	1
x	x	x	x	x	x	1	0	1	1	0
x	x	x	x	x	x	x	1	1	1	1

A=1 if (D_7 activated) OR (D_6 activated & D_7 not activated) OR [(D_6, D_7 is not activated & D_5 is activated] OR [D_4 activated & D_5, D_6, D_7 not activated]

$$A = D_7 + D_6\bar{D_7} + D_5\bar{D_6}\bar{D_7} + D_4\bar{D_5}\bar{D_6}\bar{D_7}$$

$$B = D_7 + D_6\bar{D_7} + D_3\bar{D_4}\bar{D_5}\bar{D_6}\bar{D_7} + D_2\bar{D_3}\bar{D_4}\bar{D_5}\bar{D_6}\bar{D_7}$$

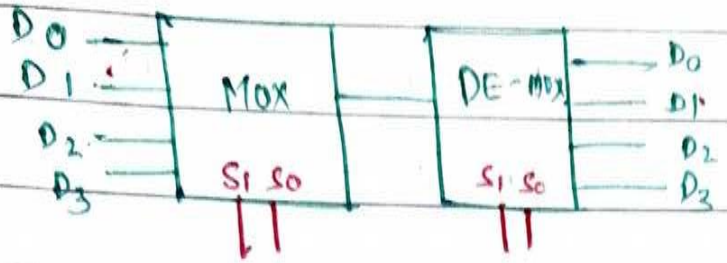
& so on.

Can be minimised using K-MAP.

My Shortcut $A = 1XX = \{(1,00), (1,01), (110), (111)\}$ — 4 OR 5 OR 6 OR 7

$$A = D_4(\bar{D_5}\bar{D_6}\bar{D_7}) + D_5(\bar{D_6}\bar{D_7}) + D_6\bar{D_7} + D_7$$

(vi) DE-MUX



① Data Selector — Data Distributor

② Parallel to series data converter — Series to parallel data converter

S_1	S_0	D_0	D_1	D_2	D_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

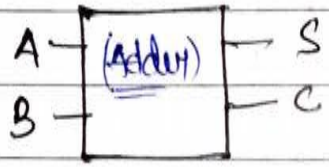
$D_0 = \overline{S_1}\,\overline{S_0}\,I$ $D_1 = \overline{S_1}\,S_0\,I$, $D_2 = S_1\,\overline{S_0}\,I$, $D_3 = S_1 S_0 I$

⑥ K-MAP uses gray code

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

} distance b/w any 2 adjacent codes is 1.

(vii) Half adder

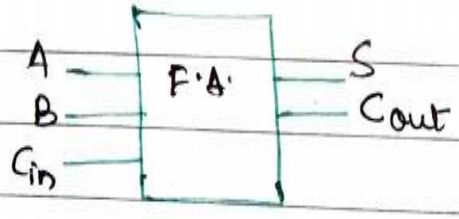


i/p		o/p	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum = $A \oplus B$

Carry = $A \cdot B$

(viii) Full adder



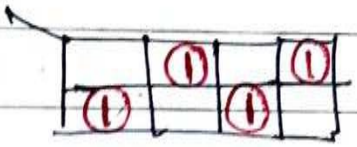
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Using K-MAP: $S = A \oplus B \oplus C$

$Cout = Cin(A \oplus B) + AB$

** Keypt:

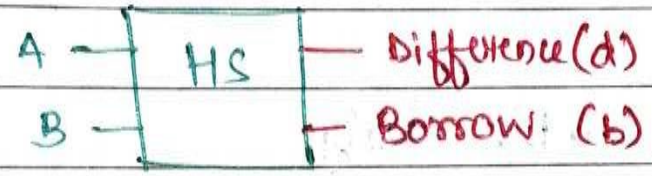
⊙ $\Sigma m(1,2,4,7) = A \oplus B \oplus C$



⊙

Half Adder/sub^r	5 NAND Gates	5 NOR Gates
Full Adder/sub^r	9 NAND Gates	12 NOR Gates
	Realisat^n using	Realisat^n using

(ix) Half Subtractor



A	B	d	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

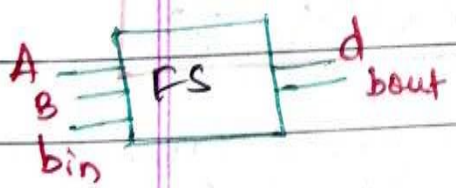
$d = A \oplus B$

$b = \bar{A}B$ {for A-B}

$= A\bar{B}$ {for B-A}

(x) Full Subtractor

{d = A - B - bin}



A	B	bin	d	bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

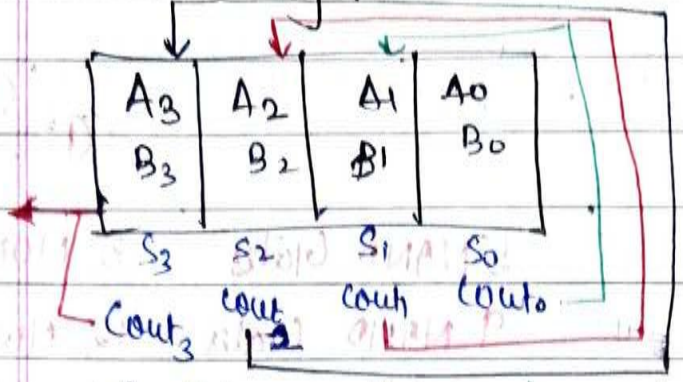
⊙ $d = A \oplus B \oplus bin$

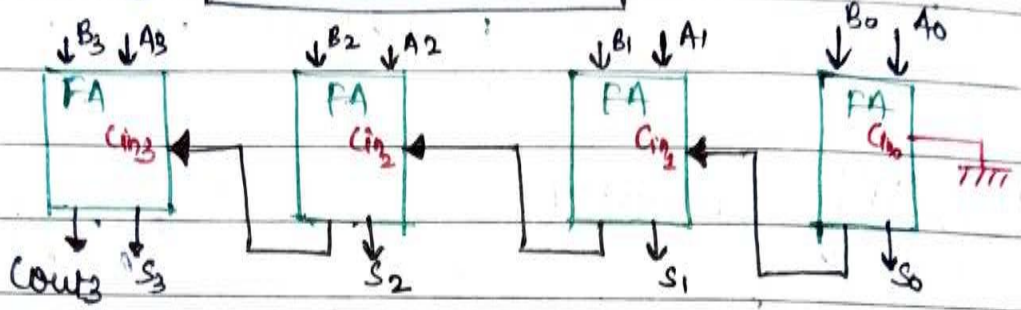
⊙ $bout = \bar{A}\,bin + \bar{A}B + B\,bin$ { for A-B-bin }

$= \bar{B}\,bin + A\bar{B} + A\,bin$ { for B-A-bin }

⊙ Imp eqn $bout = \bar{A}B + bin(\overline{A \oplus B})$ {for A-B}

(xi) Binary Parallel Adder





Adv: ⊙ Simple design

Disadv: ⊙ slow speed

⊙ High propagation delay & ↑s with no. of F.A. used

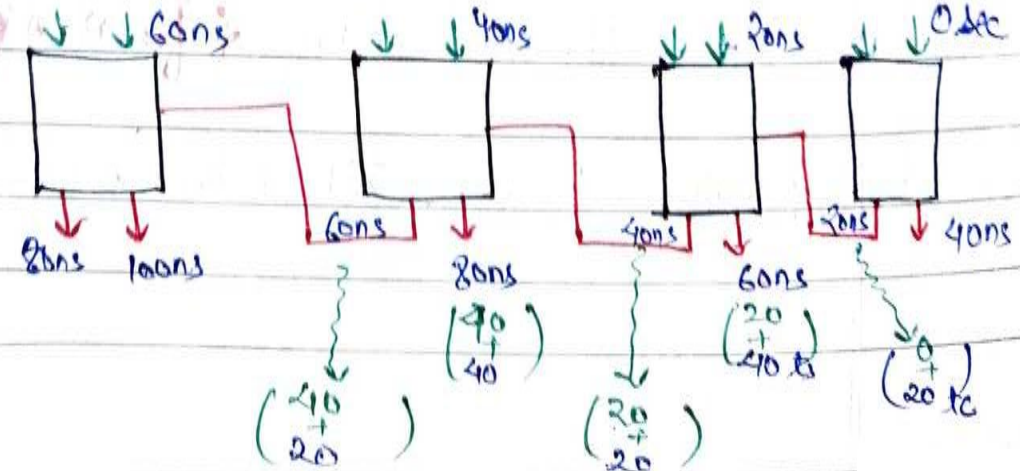
** for n-bit Addition:

* **Total time = $(n-1)t_c + \max(t_c, t_s)$**

~~$t_c + t$~~

t_c : time taken to generate carry
t_s : time taken to generate sum

** eg: if $t_c = 20ns$, $t_s = 40ns$



(xii) Look Ahead Carry Adder

FA0: $Cout_0 = AB + BCin + ACin$

$= A_0B_0 + B_0Cin_0 + A_0Cin_0 = A_0B_0 + Cin_0(A_0 + B_0)$

Carry generate $= A_0B_0$

Carry propagate $= (A_0 + B_0)$

→ $Cout_0 = Cg_0 + Cin_0(CP_0)$

→ $Cout_1 = Cg_1 + Cin_1(CP_1) = Cg_1 + Cout_0(CP_1)$

$= Cg_1 + Cg_0Cp_1 + Cin_0Cp_0Cp_1$

→ $Cout_2 = Cg_2 + Cin\,Cp_2 = Cg_2 + Cout_1\,Cp_2$

& carry on similarly

→ $Cout_3 = Cg_3 + Cin_3Cp_3 = Cg_3 + Cout_2\,Cp_3$

& carry on similarly

eg: from (Notes) 4 methods ***

Page :
Date : / /

BCD codes 8 4 [1000 0100]

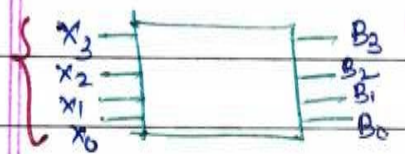
⊙ XS-3 code 3 0110

⊙ Types of BCD codes

- Weighted BCD
  - +vely weighted BCD: 8421 (By default)
    - eg: 6 → 0110
  - -vely weighted BCD: 8 4 -2 -1
    - 6 → 1010
- Non-weighted BCD
  - XS-3
  - Gray code
    - → Gray code is not BCD
    - → Gray code is non-weighted
    - → Gray code is not a non-wted BCD

Code convertors

(i) XS-3 code to BCD code



T.T.

X_3 X_2 X_1 X_0	B_3 B_2 B_1 B_0
0 0 1 1	0 0 0 0
0 1 0 0	0 0 0 1
:	:
1 1 0 0	1 0 0 1

Solve using K-MAP

Imp pnt: ⊙ $X_3X_2X_1X_0$ varies from 3 to 12, so, 0, 1, 2, 13, 14, 15 are dont care terms. Don't forget to add this in K-MAP

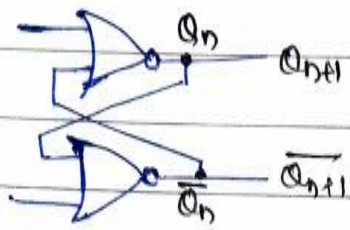
⊙ Similarly proceed for other converts just take care of dont care terms. wherever X_{S-3} is included.

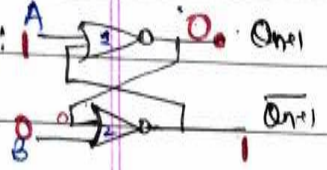
⊙ Binary to gray converter includes all no. from 0 to 15. Hence don't care not required.

Sequential circuit

(Combinational circuit) + (Memory element (Latch or FF)) = (Sequential circuit)

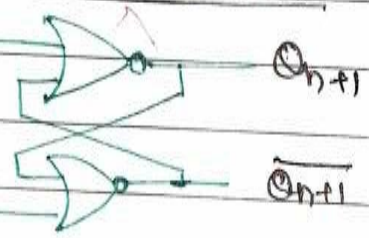
Latch (or FF)

① 

② eg: 

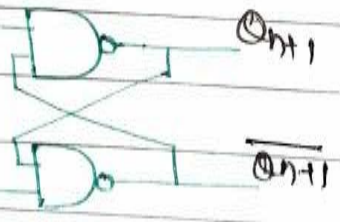
A	B	Q_{n+1}	$\overline{Q_{n+1}}$	O/P state
0	0	Q_n	$\overline{Q_n}$	HOLD
0	1	1	0	SET
1	0	0	1	RESET
1	1	0	0	INVALID

<i> S-R latch

① Reset 

Set

① Reset = 1 ⇒ $Q_{n+1} = 0$
Set = 1 ⇒ $\overline{Q_{n+1}} = 0$
① Input is treated as '1' to justify states

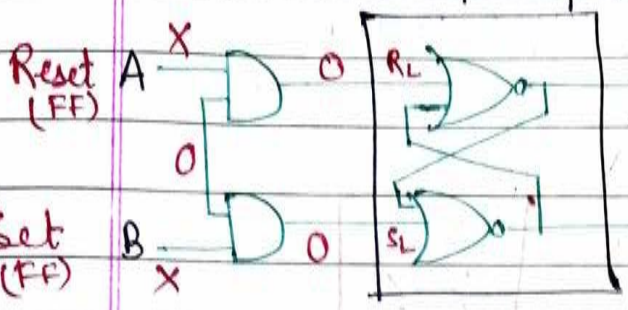
② $\overline{Set}$ 

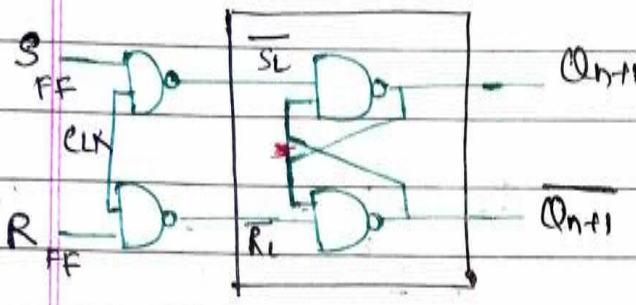
$\overline{Reset}$

② $\overline{Reset} = 0$ ⇒ $Q_{n+1} = 0$
$\overline{Set} = 0$ ⇒ $\overline{Q_{n+1}} = 0$
② Input is treated as '0' to justify states

Page :
Date : / /

(i) S-R Flip Flop





⊙ Truth table of SR Flip-Flip (NOR & NAND LATCH)

CLK	S_{FF}	R_{FF}	Q_{n+1}	$\overline{Q_{n+1}}$	State
0	X	X	Q_n	$\overline{Q_n}$	HOLD
1	0	0	Q_n	$\overline{Q_n}$	HOLD
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	X	X	INVALID

Set → $Q_{n+1} = 1$
Reset → $Q_{n+1} = 0$

(I) Characteristics table

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

	$\overline{R}\overline{Q_n}$	$\overline{R}Q_n$	RQ_n	$R\overline{Q_n}$
$\overline{S}$	0	1	0	0
S	1	1	X	X

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

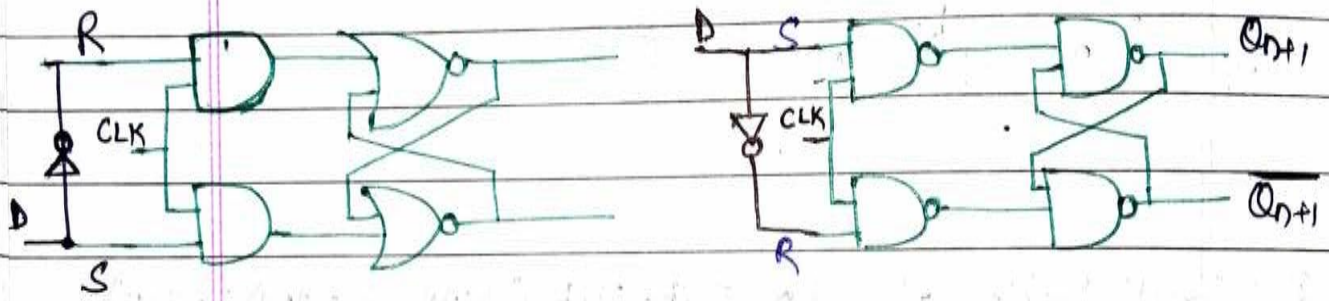
C/S eqn / Next step eqn

II Excitation table

* $Q_n = 0, Q_{n+1} = 0$ if HOLD / Reset
* $Q_n = 1, Q_{n+1} = 1$ if Set / HOLD
* $Q_n = 1, Q_{n+1} = 0$ if Reset

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

(iii) D Flip Flop



CLK	D	Q_{n+1}	$\overline{Q_{n+1}}$	State
0	X	Q_n	$\overline{Q_n}$	Hold
1	0	0	1	Reset
1	1	1	0	Set

⊙ D is always connected to Set & $\overline{D}$ to Reset

(I) Characteristic Table

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

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

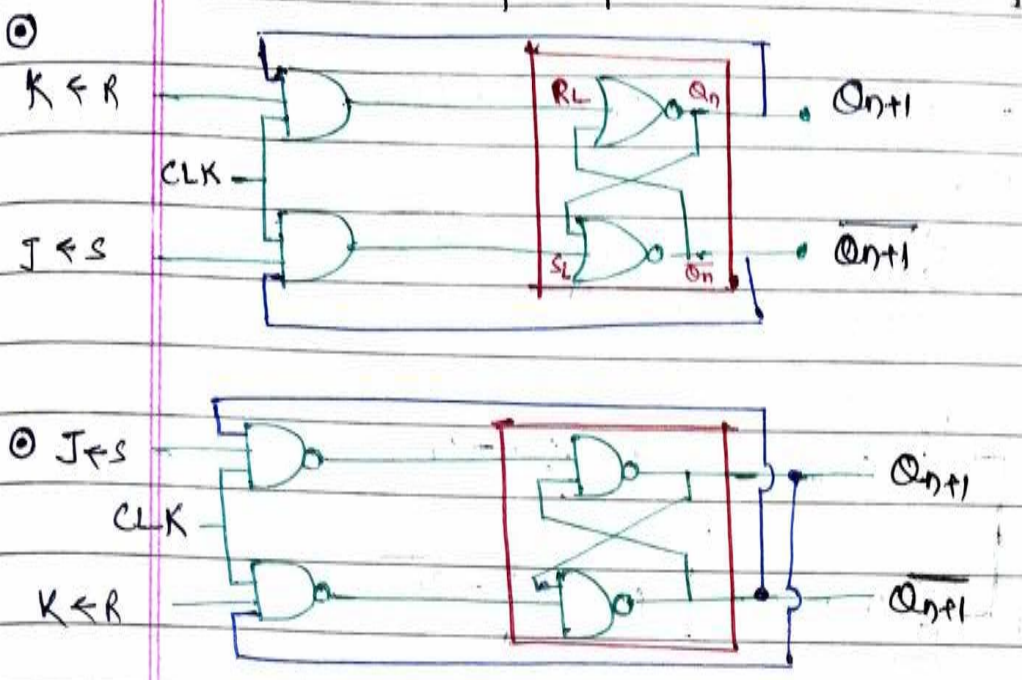
II Excitation Table

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

{ i/p → excitation, o/p → response } Hence named excitation table

Page :
Date : / /

(iii) JK Flip-flop



CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$	State
0	X	X	Q_n	$\overline{Q_n}$	HOLD
↓	0	0	Q_n	$\overline{Q_n}$	HOLD
↓	0	1	0	1	RESET
↓	1	0	1	0	SET
↓	1	1	$\overline{Q_n}$	Q_n	TOGGLE

(I) Characteristics table

	J	K	Q_n	Q_{n+1}
H	0	0	0	0
	0	0	1	1
R	0	1	0	0
	0	1	1	0
S	1	0	0	1
	1	0	1	1
T	1	1	0	1
	1	1	1	0

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

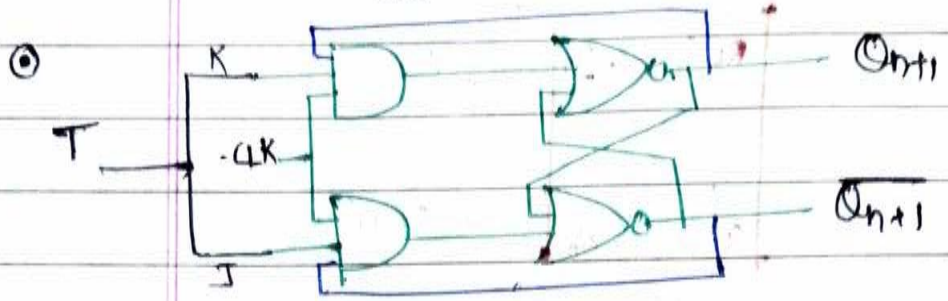
II Excitation table

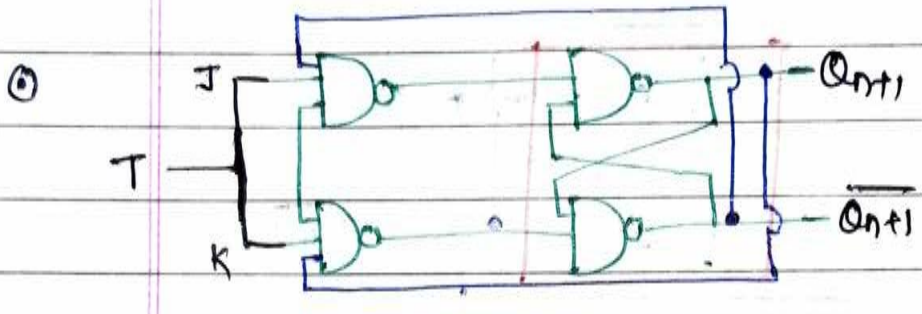
Q_n	Q_{n+1}	J	K	State
0	0	0	X	Hold Reset
0	1	1	X	Set Toggle
1	0	X	1	Reset Toggle
1	1	X	0	Hold Set

** FF me J & K ko T put karna

Page :
Date : / /

(iv) Toggle FF (T – FF)





CLK	T	J	K	Q_{n+1}	$\overline{Q_{n+1}}$	State
0	X	X	X	Q_n	$\overline{Q_n}$	Hold
1	0	0	0	Q_n	$\overline{Q_n}$	Hold
1	1	1	1	$\overline{Q_n}$	Q_n	Toggle

(I) Characteristics table.

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

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

II Excitation table.

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

Page :
Date : / /

FF conversion : {X FF to Y FF}

X → Excitation table Y → C/S table

Counters

<i> Synchronous counter :

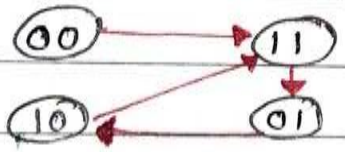
Case-(1) Next eqn is given

⊙ Compare given NSE with Flipflop NSE & get the values of variable pins (eg: J, K etc)

⊙ Draw state transition table:

Present state		Next state	
A_n	B_n	A_{n+1}	B_{n+1}
0	0	—	—
0	1	—	—
1	0	—	—
1	1	—	—

⊙ Draw state transition diagram:



Case-(2) State transition table is given

⊙ Extend the table & get values of variable pins (J_A, K_A) from Q_n Q_{n+1} & so Q_n.

Q_A	Q_B	Q_{A+1}	Q_{B+1}	J_A K_A	J_B K_B

⊙ Solve K-MAP seperately for J_A, K_A etc & get expressions.

⊙ Draw realisation ckt & Put those values of pins in NSE of FF

Page :
Date : / /

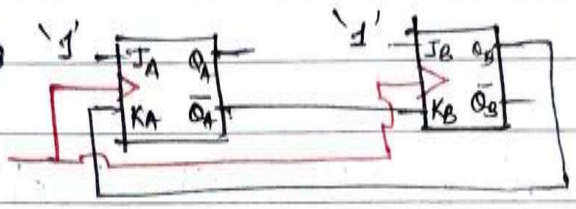
case (3) If realised ckt is given.

① A_{n+1} = ? in terms of A_n, B_n

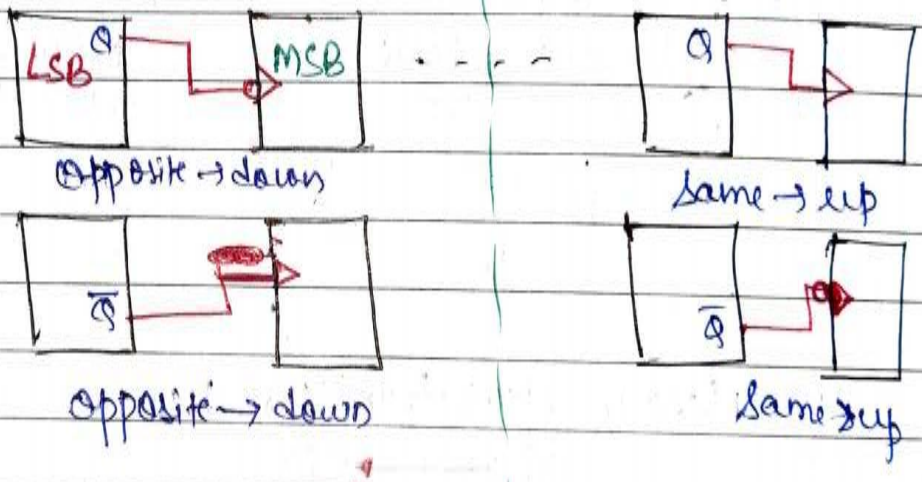
② State transition table

③ State transition diagram

eg:



(ii) Asynchronous counter



Shortcut

for → up counter

MOD	Q_C Q_B Q_A	Active low clr (NAND)	Active High clr (AND)
MOD 1	0 0 1	$\overline{Q_A}$	Q_A
MOD 2	0 1 0	$\overline{Q_B}$	Q_B
MOD 3	0 1 1	$\overline{Q_B Q_A}$	$Q_B Q_A$
MOD 4	1 0 0	$\overline{Q_C}$	Q_C
MOD 5	1 0 1	$\overline{Q_C Q_A}$	$Q_C Q_A$
MOD 6	1 1 0	$\overline{Q_C Q_B}$	$Q_C Q_B$
MOD 7	1 1 1	$\overline{Q_C Q_B Q_A}$	$Q_C Q_B Q_A$

MOD 22 → 10110

$\overline{clr} = Q_E Q_C Q_B$

$clr = Q_E Q_C Q_B$

Page :
Date : / /

Note: In down counter we use preset, so, that whenever required we can make $Q_C Q_B Q_A = 1$

for down counter: similar table but Pre → NOR & $\overline{Pre}$ OR

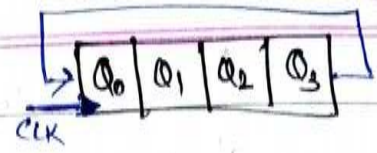
- ⊙ MOD – 7 down — 111 — (OR) $\overline{Pre} = Q_C + Q_B + Q_A$ — (NOR) $Pre = \overline{Q_C + Q_B + Q_A}$
- ⊙ MOD 7 up — 111 — (NAND) $\overline{clr} = \overline{Q_C Q_B Q_A}$ — (AND) $clr = Q_C Q_B Q_A$

- ⊙ Self starting: returning back to loop from disturbance
- ⊙ Residual clk: Max. no. of clock pulses to return back to from any unexpected state to desired count sequence.

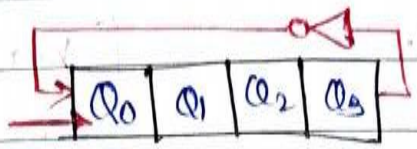
extra bits (if any) ↓

Shift Registers

<i> Ring counter



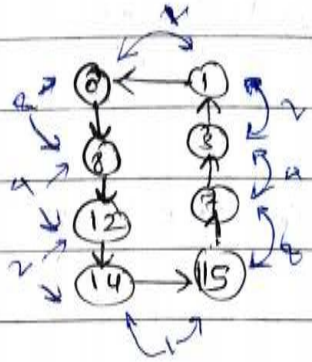
<ii> Twisted ring / Johnson's counter



↓

* used → $2n$

unused → $2^n - 2n$



Last 2 topics to be added in Notes later.

- No. of used state in Johnson counter = $2n$
- frequency of O/p = $\frac{f}{2n}$ as O/p is repeated after $(2n)$ cycle.

- for shift register table will be like

eg: 1 0 1 0

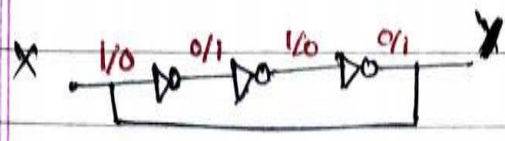
0 → 1 → 0 → 1

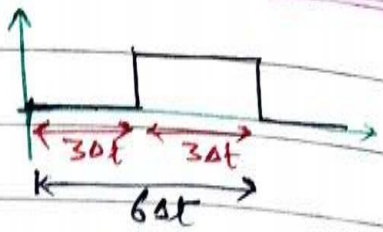
1 0 → 1 → 0

0 1 0 → 1

GATES & BOOLEAN ALGEBRA

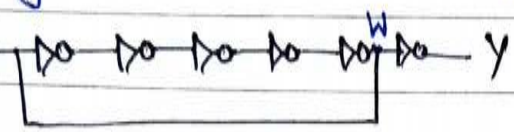
Date : / /

⊙ 

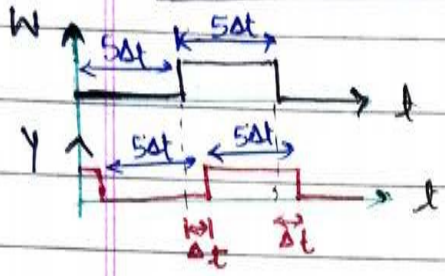


T.P. = 6Δt

Generally, T.P. = 2nΔt , $f = \frac{1}{2n\Delta t}$

⊙ 

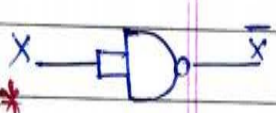
T.P. (Y) = ?

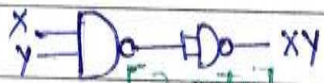


T.P. (W) = T.P. (Y) = 10Δt

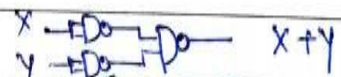
$f = \frac{1}{10\Delta t}$

⊙ NAND GATE

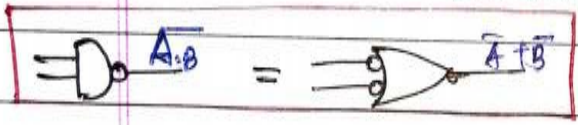




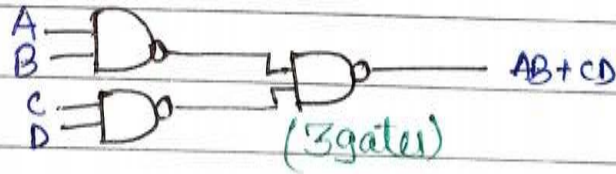
[2 gates]



[3 gates]

* 

⊙(i) AB+CD



(3 gates)

→ A+BC = A·A + BC — 3 gates

→ A+B = A·A + B·B — 3 gates

→ AB + CD + EF = x + EF ⇒ {6 gates total}

(3 gates) (3 gates)

→ *** AB + BC + CD + DA = (A+C)B + D(A+C) = (A+B)(B+D)

(3) (3) (3) (3) (3) (2)

x + y, x = A+C, x = A+C, x·y

xB + xD

9 gates {but not min} — 6 gates — 8 gates {not min}

Page :
Date : / /

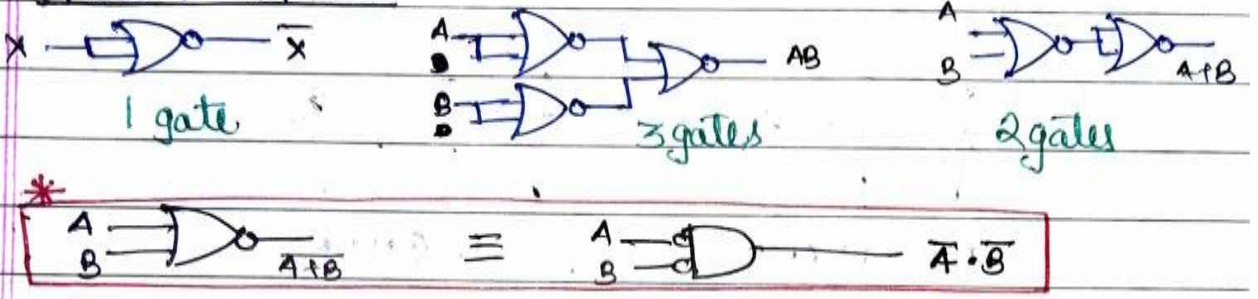
→ $ABC + DE = xC + DE$ → 5 gates

(2) (3)

** → $\bar{A} + \bar{B} + ABC = x + yC$ → 4 gates

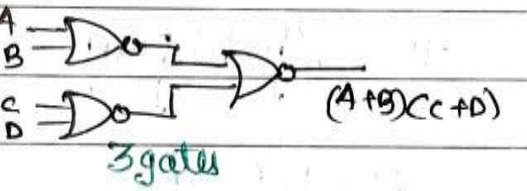
(2) x (2) y (3)

⊙ NOR GATE



*

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



→ $AB = (A+A)(B+B)$ 3 gates

→ $A(B+C) = (A+A)(B+C)$ 3 gates

→ $(A+B+C)(D+E) = (x+C)(D+E)$ → 5 gates

x (2) (3)

→ $(A+B)(C+D)(E+F) = x(E+F)$ → 6 gates

*** → $AD + BE + CD + AE + BD + CE$

(3) (3) (3)

(3) (3)

(3)

15 gates {not min}

→ or $D(A+C+B) + E(A+B+C)$

x x

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

(2)

$(D+E)(x+C)$

(3)

5 gates

or

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

(2) x x

x+C

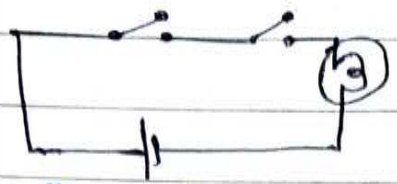
(2)

$Dy + Ey$

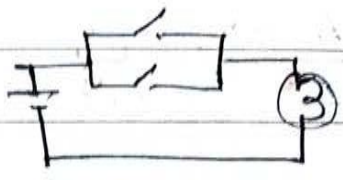
(3)

(7) {not min}

Switching circuit Representation:

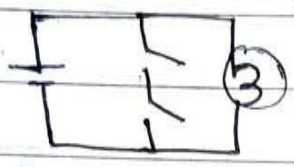
* 

(i) AND

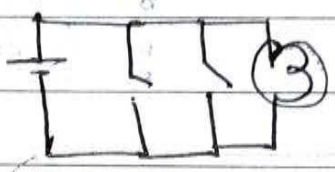


(ii) OR

(iii) NOT



(iv) NAND



(v) NOR

Special purpose gates

(i) **EX-OR** $A \oplus B = \bar{A}B + A\bar{B}$

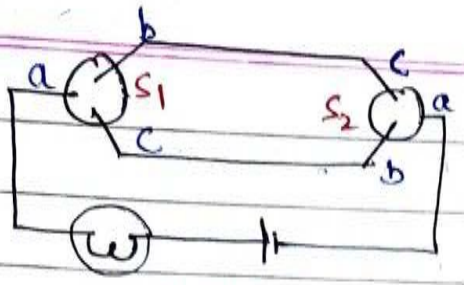
- Detects odd no. of 1's in inputs {for more than 2 i/ps}
- Inequality detector for 2 i/ps.

- $A \oplus 1 = \bar{A}$ $\quad A \oplus 0 = A$
- $A \oplus B = C \Rightarrow A \oplus C = B$ & $A = B \oplus C$
- $A \oplus B = C \Rightarrow A \oplus B \oplus C = 0$

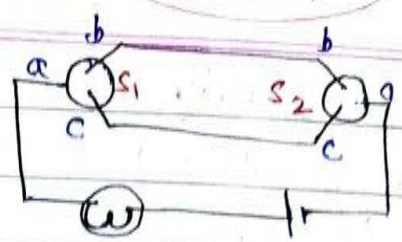
(ii) **EX-NOR** $A \odot B = AB + \bar{A}\bar{B}$

- Even no. of 1's detector & for even no. of i/ps
  Odd no. of 1's detector & for odd no. of i/ps
- Equality detector for 2 i/ps

Key point: for odd no. of I/Ps XNOR = XOR
for even no. of i/ps XNOR = $\overline{XOR}$



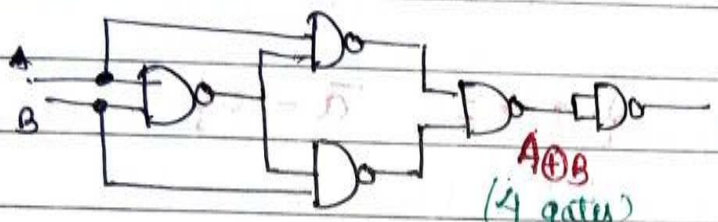
$S_1 \oplus S_2$



$S_1 \odot S_2$

Realisation:

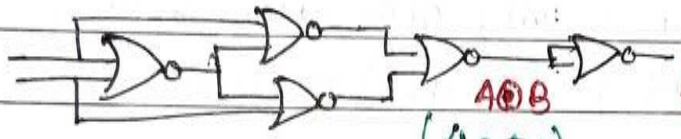
(i)



$A \oplus B$ (4 gates)

$A \odot B$ (5 gates)

(ii)



$A \odot B$ (4 gates)

$A \oplus B$ (5 gates)

Important Boolean laws:

(i) $A \odot 1 = A$ $\quad \bar{A} \odot 0 = \bar{A}$

(ii) $A \oplus 1 = \bar{A}$ $\quad A \oplus 0 = A$

(iii) $\bar{A} \oplus \bar{B} = A \oplus B = A \odot \bar{B} = \bar{A} \odot B$

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

(iv) $(A \oplus B) \oplus C = A \oplus (B \oplus C)$

$(A \odot B) \odot C = A \odot (B \odot C)$

(v) $A + BC = (A+B)(A+C)$ {Distributive law}

(vi) $A(A+B) = A$ {Absorption law}

(vii) Consensus law: It is true for the expression in the form of $AB + BC + AC$. but only one variable present in both complent & original form.

{B & $\bar{B}$ terms} $AB + \bar{B}C + AC = \bar{B}C + AB$ | $(A+B)(\bar{B}+C)(C+A) = (A+\bar{B})(B+C)$

{A & $\bar{A}$ terms} $AB + BC + \bar{A}C = AB + \bar{A}C$ | $A\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C} = A\bar{B} + \bar{A}\bar{C}$

Duality

eg: $f = AB + CD \longrightarrow f^D = (A+B)\cdot(C+D)$

$A + 1 = 1 \longrightarrow A \cdot 0 = 0$

* Do not complement variables.

$f = f_D \Rightarrow$ Self dual f^n

Expression:

(i) SOP $\{a \to 1 \quad \bar{a} \to 0\}$

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

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

$= \Sigma m\,(101, 010, 100, 110)$

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

(ii) POS $\{a \to 0 \quad \bar{a} \to 1\}$

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

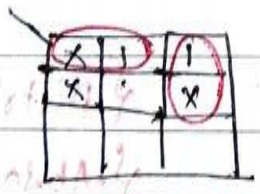
$= (\bar{a}+b+c)(a+\bar{b}+\bar{c})(\bar{a}+c+b\bar{b})$

$= (\bar{a}+b+c)(a+\bar{b}+\bar{c})(\bar{a}+b+\bar{b})(\bar{a}+\bar{b}+c)$ (same)

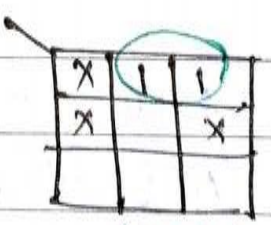
$= \pi M\,(100, 011, 110)$

$= \pi M\,(4, 3, 6)$

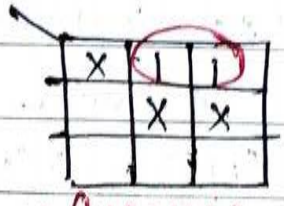
** for KMAP don't use dont care unless required.



X (WRONG) X

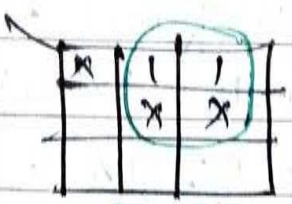


Correct



Y (WRONG) X

Could be minimised more using 'x'



Correct

MICROPROCESSOR

* Architecture

(i) ALU → +/−* operations are performed.

(ii) Registers → 6 general purpose (8 bit) B, C, D, E, H & L

(iii) Accumulator → ⊙ Part of ALU
⊙ 8 bit

(iv) Flags → ⊙ Flip-flops which are set/reset
⊙ Z, S, P, CY, AC
(Parity) (Carry) (Aux Carry)

(v) Program Counter → ⊙ 16 bit register
⊙ Points the location from where next byte is to be fetched

(vi) Stack Pointer → ⊙ 16 bit register
⊙ It points to a memory location in (R/W memory) or stack.

(vii) Instruction Register / Decoder → ⊙ Current instruction to decoder

(viii) ADDRESS BUS CONTROL BUS
DATA BUS

⊙ (O/P) ALE : Address latch enable

(I/P) READY : HIGH ⇒ Memory / Peripheral is ready to send/receive data

(I/P) HOLD : Xfer control of Address & data bus temporily.

(O/P) HLDA : Indicates CPU has received HOLD request
HIGH → HOLD request

(I/P) INTR : Interrupt request

(O/P) INTA : Interrupt Acknowledge

(I/P) TRAP : Non-maskable restart interrupt.

(I/P) RESET IN : → Resets PC, INTR, HLDA flipflops

(O/P) RESET OUT : → System reset (CPU reset)

(o/p) IO/M — whether Read/write is to memory
(i/p) SID — Serial input data line
(o/p) SOD — Serial output Data line

More examples:
i/p: X_1, X_2, ~~RD, WR~~, AD0-7
o/p: ~~AD0-7~~ A_8-A_1, RD, WR, S_0, S_1, CLK

Addressing Modes

(i) Immediate (Data is present in instr.)
(ii) Register (Data through registers)
(iii) Direct (Data from outside devices → Accumulator)
(iv) Indirect (HL pair ← address of data)

eg: MVI R, data — MVI RD, RS
Immediate — Register (destination, source)

Direct: IN 00H, OUT 01H

Indirect: MOV M, Rs; MOV RD, M; MVI M, data

Types of instruction based on size

(i) 1 byte inst. (MOV C,A) (ADD B) (CMA)
(ii) 2 byte inst. (MVI A, Data)
(iii) 3 byte inst. (JMP 2085H)

opcode → 1 byte
opcode + databyte → 2 byte
opcode + Data byte + Data byte → 3 byte
(Data byte + Data byte = Address)

Instruction Summary

Data Transfer Instructions

MOV	Copy from source to destination
MVI	Move immediate 8 bit
LDA	Load accumulator
LDAX	Load accumulator indirect
LXI	Load register pair immediate
LHLD	Load H and L registers direct
STA	Store accumulator direct
STAX	Store accumulator indirect
SHLD	Store H and L registers direct
XCHG	Exchange H and L with D and E
SPHL	Copy H and L registers to the stack pointer
XTHL	Exchange H and L with top of stack
PUSH	Push register pair onto stack
POP	Pop of stack of register pair
OUT	Output data from accumulator to a port with 8 bit address
IN	Input data to accumulator from a port with 8 bit address

Arithmetic Instructions

ADD	Add register or memory to accumulator
ADC	Add register to accumulator with carry
ADI	Add immediate to accumulator
ACI	Add immediate to accumulator with carry
DAD	Add register pair to H and L registers
SUB	Subtract register or memory from accumulator
SBB	Subtract source and borrow from accumulator
SUI	Subtract immediate from accumulator
SBI	Subtract immediate from accumulator with borrow
INR	Increment register or memory by 1
INX	Increment register pair by 1
DCR	Decrement register or memory by 1
DCX	Decrement register pair by 1
DAA	Decimal adjust accumulator

Control Instructions

NOP	No operation
HLT	Halt
DI	Disable interrupts
EI	Enable interrupts
RIM	Read interrupt mask
SIM	Set interrupt mask

Branching Instructions

JMP	Jump unconditionally
JC	Jump on carry
JNC	Jump on no carry
JP	Jump on positive
JM	Jump on minus
JZ	Jump on zero
JNZ	Jump on no zero
JPE	Jump on parity even
JPO	Jump on parity odd
CALL	Call unconditionally
CC	Call on carry
CNC	Call on no carry
CP	Call on positive
CM	Call on minus
CZ	Call on zero
CNZ	Call on no zero
CPE	Call on parity even
CPO	Call on parity odd
RET	Return unconditionally
RC	Return on carry
RNC	Return on no carry
RP	Return on positive
RM	Return on minus
RZ	Return on zero
RNZ	Return on no zero
RPE	Return on parity even
RPO	Return on parity odd
PCHL	Load program counter with HL contents
RST	Restart

Logical Instructions

CMP	Compare register or memory with accumulator
CPI	Compare immediate with accumulator
ANA	Logical AND register or memory with accumulator
ANI	Logical AND immediate with accumulator
XRA	Exclusive-OR register or memory with accumulator
XRI	Exclusive-OR immediate with accumulator
ORA	Logical OR register or memory with accumulator
ORI	Logical OR immediate with accumulator
RLC	Rotate accumulator left
RRC	Rotate accumulator right
RAL	Rotate accumulator left through carry

RAR		Rotate accumulator right through carry
CMA		Complement accumulator
CC		Complement carry
STC		Set carry

10. 8085 Microprocessor Programs

Addition of Two 8 bit Numbers

Program

	MVI	C,00	Initialize C register to 00
	LDA	4150	Load the value to accumulator
	MOV	B, A	Move the content of accumulator to B register
	LDA	4151	Load the value to accumulator
	ADD	B	Add the value of register B to A
	JNC	LOOP	Jump on no carry
	INR	C	Increment value of register C
LOOP:	STA	4152	Store the value of accumulator (sum)
	MOV	A, C	Move content of register C to accumulator
	STA	4153	Store the value of accumulator (carry)
	HLT		Halt the program

Observation

Input :	80 (4150)
	80 (4251)
Output :	00 (4152)
	01 (4153)

Subtraction of Two 8 bit Numbers

Program

	MVI	C,00	Initialize C to 00
	LDA	4150	Load the value to accumulater
	MOV	B, A	Move the content of accumulator to B register
	LDA	H, 151	Load the value to accumulator
	SUB	B	
	JNC	LOOP	Jump on no carry
	CMA		Complement accumulator contents
	INR	A	Increment value in ac
	INR	C	Increment value in re
LOOP:	STA	4152	Store the value of A r memory address
	MOV	A, C	Move contents of reg accumulator
	STA	4153	Store the value of acc memory address
	HLT		Terminate the progra

Observation

Input :	06 (4150)
	02 (4251)
Output :	04 (4152)
	01 (4143)

Multiplication of Two 8-bit Numbers

Program

	MVI	D, 00	Initialize register D to 00
	MVI	A, 00	Initialize accumulator cont to 00
	LXI,	4150	
	MOV	B, M	Get the first number in B re
	INX	H	
	MOV	C, M	Get the second number in register
LOOP :	ADD	B	Add content of A register t register B
	JNC	NEXT	Jump on no carry to next
	INR	D	Increment content of regis
NEXT :	DCR	C	Decrement content of regis
	JNZ	LOOP	Jump on no zero to addres
	STA	4152	Store the result in memory
	MOV	A, D	
	STA	4153	Store the MSB of result in memory
	HLT		Terminate the program

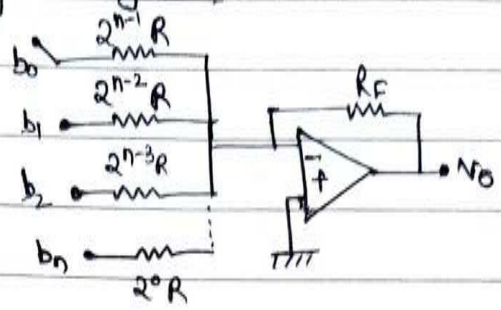
Observation

Input :	FF (4150)
	FF (4151)
Output :	01 (4152)
	FE (4153)

ADC & DAC

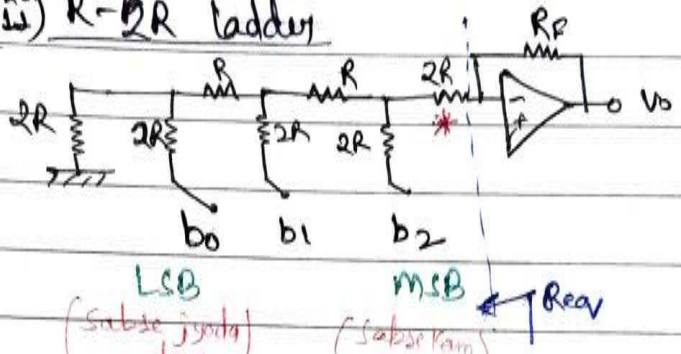
DAC

(i) Weighted register

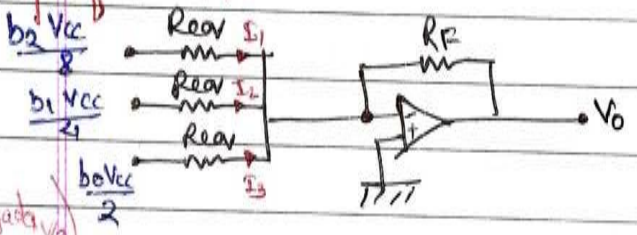


Apply Vcc for V_0

(ii) R-2R ladder



Simplified ckt.



Req = R if 2R is short
3R 2R is open

** for step size $b_1 = b_2 = b_3 = 0$, $b_0 = 1$, $V_0 = SS$

** No. of steps $= 2^n - 1$

** F.S.O. = SS × No. of steps
(full scale o/p) (step size)

** Resolution $= \frac{1}{\text{No. of step size}} = \frac{\text{Step size}}{\text{F.S.O.}}$

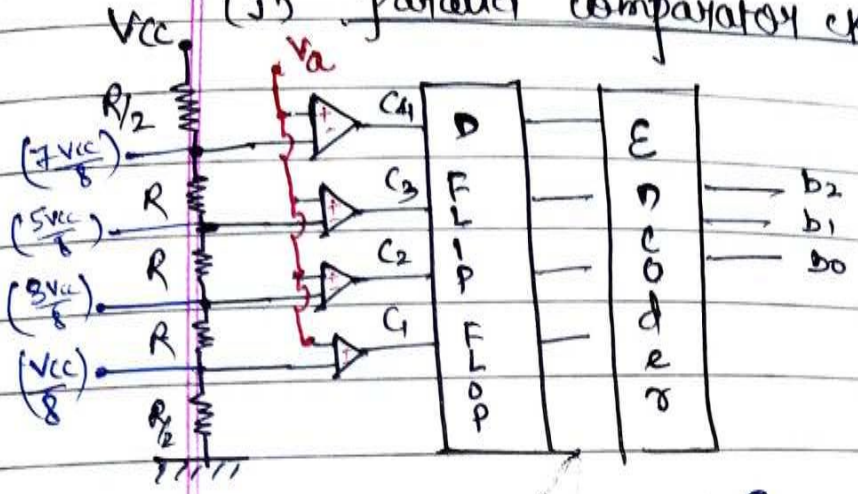
unitless

If resolution is given in V ⇒ it is step size

** Step size is defined as o/p V/g which we get with first input step (0001)

ADC

(i) Parallel comparator ckt.



	C_1	C_2	C_3	C_4
eg: $\frac{3V_{cc}}{8} < V_a < \frac{5V_{cc}}{8}$	1	1	0	0

Disadvantages:

(i) High no. of comparators are required

Adv:

(i) Fastest ADC & hence known as Flash ADC or Simultaneous comparator ADC

(ii) Conversion time = T_c [T_{CLK}]

rest ADC → later.