12/7/22

Digital logic

Number system ***

$N = \sum a_i x^i$

$(340)_{10} = 3\times10^2 + 4\times10^1 + 0\times10^0$ [for decimal, $x=10$, a = Digits]

$(a_2 a_1 a_0)_{10}$

$(1010)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 8+2 = (10)_{10}$

$(b_3 b_2 b_1 b_0)_2$

Bit (Binary digits)

$x = 2$ [x = Radix / Base]

1) $(0.43)_{10} = 4\times2^{-1} + 3\times2^{-2}$ ||

$(a_{-1} a_{-2})_{10} = (2.75)_{10}$

2) $(0.11)_{10} = 1\times2^{-1} + 1\times2^{-2}$

$(a_{-1} a_{-2})_{10} = \frac{1}{2} + \frac{1}{4}$

$= 0.5 + 0.25 = (0.75)_{10}$

[Binary → Decimal: $N_2 = ()_{10}$]

3) $(0.101)_2 = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$(b_{-1} b_{-2} b_{-3})_2 = (0.625)_{10}$

$b_{-1} = 1$

$b_{-2} = 0$

$b_{-3} = 1$

Octal system ($x=8$)

Digits → 0 – 7

1) $(127.4)_8 = 1\times8^2 + 2\times8^1 + 7\times8^0 + 4\times8^{-1} = (87.5)_{10}$

$(a_2 a_1 a_0 . a_{-1})_8$

	Digits
Decimal →	0 – 9
Binary →	0 – 1
Hexadecimal ($x=16$) →	(0–9, A, B, C, D, E, F)

A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

[$N_8 = ()_{10}$]

$(C)_{16} = (12 \times 16^1 + 1 \times 16^0)_{10}$

$(d_1 d_0)_{16} = (19 3)_{10}$

[Hexadecimal → decimal
$N_{16} = ()_{10}$]

N_{10}
$N_2 = ()_{10}$
$N_8 = ()_{10}$
$N_{16} = ()_{10}$

[0xabcd ≠ abcd H
= N_{16} ↳ Hex. rep.]

① i) $(1\,1\,0\,1\,0\,1)_2$ (5 4 3 2 1 0)

$= 2^5 + 2^4 + 2^2 + 2^0$

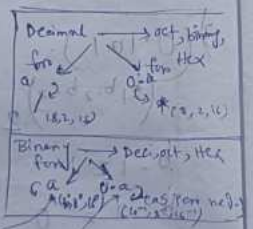
$= (53)_{10}$

Binary ⟷ Decimal

ii) $(53)_{10} = (110101)_2$

$(110101)_2$

2 | 53
2 | 26 — 1
2 | 13 — 0
2 | 6 — 1
2 | 3 — 0
1 — 1



② i) $(0.101)_2$ (−1 −2 −3)

$= 2^{-1} + 2^{-3}$

$= (0.625)_{10}$

ii) $(0.625)_{10} = (0.101)_2$

carry

0.625 × 2 = 0.25 — 1
0.25 × 2 = 0.5 — 0
0.5 × 2 = 0 — 1 ↓

③ $(53.625)_{10} = (110101.101)_2$

Decimal ⟷ octal

$N_{10} = ()_8$

① ii) $(153)_{10} = (231)_8$

8 | 153
8 | 19 — 1
8 | 2 — 3

$(231)_8$

$(0.513)_{10} = (0.40651)_8$

carry

0.513 × 8 = 0.104 — 4
0.104 × 8 = 0.832 — 0
0.832 × 8 = 0.656 — 6
0.656 × 8 = 0.248 — 5
0.248 × 8 = 0.984 — 1
~~0.984 × 8 = 0.872 — 7~~
0.872 × 8

[Have to take 5 digits after Point]

Hexadecimal ~~xxxxx~~ $N_{10} = ()_{16}$

Decimal ↑↓

for Integaro Part → ÷ 16
for Fractional Part → × 16

Intermediate state

octal ⟷ Hexadecimal [$N_{16} = (\ )_8$

Binary ⟷ Hexadecimal $N_{16} = (\ )_2$

Here we use the intermediate state as binary

4 bit binary (8 4 2 1)	Decimal value	octal
0 0 0 0	1	0 0
0 0 0 1	2	0 1
0 0 1 0	3	0 2
0 0 1 1	4	0 3
0 1 0 0	5	0 4
0 1 0 1	6	0 5
0 1 1 0	7	0 6
0 1 1 1	8	0 7
1 0 0 0	9	1 0
1 0 0 1	10	1 1
1 0 1 0	11	1 2
1 0 1 1	12	1 3
1 1 0 0	13	1 4
1 1 0 1	14	1 5
1 1 1 0	15	1 6
1 1 1 1	16	1 7

$(15)_{10}$ $\boxed{(15)_{10} = (17)_8}$ [In decimal the b value of 15 is equal to the value of 17 in octal (w. Binary)]

Hexadecimal
0
1
2
3
4
5
6
7
8
9
10 → A
11 → B
12 → C
13 → D
14 → E
15 → F

① $(157)_8 = (001101111)_2$ [3 bit]
octal → Binary

② $(A2C)_{16} = (101000101100)_2$ [4 bit]
Hexadecimal → binary

③ $(0.125)_8 = (.001010101)_2$

④ $(0.D2)_{16} = (.110100100)_2$

⑤ $(0.111100)_2 = (0.74)_8$ [Binary → octal]

⑥ ~~(530101~~ $(0.11110010)_2 = (0.F2)_{16}$ [Hexadecimal ← Binary]

• B ⟷ 0 / 1
3 bit group, 4 bit group

~~(157)$_8$ = (00~~

① $(157)_8 = (001101111)_2$ Binary
↓ octal
$= (061)_{16}$ Hexadecimal

② $(6B.F2)_{16} = (01101011.11110010)_2$ (imginary 0 / imginary 0)
Before the dp / After the decimal point
$= (153.744)_8$

③ $(53.74)_8 = (101011.111100)_2$
$= (2B.F)_{16}$

12 → 8 + 4 = 12 → 1100
14 → 8 + 4 + 2 = 14 → 1110
13 → 8 + 4 + 1 = 13 → 1101
} using weightage find the binary

• 1's Complement

MSB LSB
0000 1010 (4 bit) —1's→ (1111 0101)$_{1's}$ [0 → 1, 1 → 0]

If intermediate form → decimal

$()_8 \longrightarrow ()_{16}$

$()_8 \longrightarrow ()_{10} \longrightarrow ()_{16}$

Ex: $(231)_8 \rightarrow$ # $231_{2\ 1\ 0}$ $= 2\times10^2 + 3\times10 + 1\times10^0$

$= 200 + 30 + 1$

$= 2\times8^2 + 3\times8^1 + 1\times8^0$

$= 128 + 24 + 1$

$= (153)_{10}$

$(153)_{10} \longrightarrow$ 16 | 153 | 9 | $= (99)_{16}$

$()_{16} \longrightarrow ()_{10} \longrightarrow ()_8$

$(99)_{16} = 9\times16^1 + 16\times9$

$= (153)_{10}$

$(153)_{10} =$ 8 | 153 | 19 — 1 | 2 — 3 $= (231)_8$

2's Complement ⟶ 1's + 1

(1)

i) 1111 0101 ⟶ 1's
+1 ⟶ 1

$(11110110)_{2's}$

ii) (0000 1010) } 1's + 1 = 2's
copy (direct process)
↓ 2's
$(1111 0110)_{2's}$

[LSB = Least Significant Bit
MSB = Most Significant Bit]

[Copy the first 1 until it appears copy the 0's also]

(2) 0000 1000
↓ 2's
$(1111 1000)_{2's}$

other process: 1's ⟶ 1111 0111
+ 1
$(1111 1000)_{2's}$

[Copy the 0s until the first 1 appears]

(3) (0000 0001)
↓ 2's
$(1111 1111)_{2's}$

other process: 1's ⟶ 1111 1110
+1
$(1111 1111)_{1's}$

Uses

Signed numbers ⟶ signed magnitude
⟶ signed 1's complement
⟶ signed 2's complement

sign bit = MSB

MSB { 0 = +(ve)
1 = −(ve) } How to identify the sign of a no.

MSB → Sign bit | Rest part → magnitude

(−1) = 1 | 000001 (s. | mag)
(+1) = 0 | 000 001 (s. | mag)
} Signed magnitude → 8 bit format

$(-1)_{1's}$ = 0000 0001 ⟶ 8 bit format
⟶ $=(1111\ 1110)_{1's}$

$(+1)_{1's}$ = 0000 0001

(1111 1110) Signed 1's
↓ 1's
(0000 0001) magnitude

(+1) = (0000 0001)
(−1) = (1111 1110)
} Signed 1's

To get the mag we have to take 1's complement

(+1) = 0000 0001
(−1) = 1111 1111
} Signed 2's

(1111 1111) signed 2's
↓ 2's
(0000 0001) magnitude

To get the mag we have to take 2's complement

(1111 1000) signed 1's

↓1's

- 0000 0111 = -7

magnitude

(11111011) Signed 2's

↓2's

(-00000100) = (-5)

magnitude

(-12) = ( ) Signed 1's

(-14) = ( ) Signed 2's

① 000 1010 = -12

↓ 1's

1111 0101

-000 0010 ← -12

↓ 1's

(1111 0101) signed 1's

Ans - 1111 0011

+12 → 0001010

↓ 2's

1110 110

-12

② -000 0100 = -14

↓ 2's

(1111 1011) signed 2's

+14 → 0001100

↓ 2's

1110 1100 → -14

Ans - 1111 0010

Not those

- Binary coded Decimal (BCD)

$(185)_{10}$ = (0001 1000 0101) BCD [0 - 9]

~~128~~ = ~~(0001 0010 1000)~~

0 - 9 = (1011 1001)$_2$

256, 128, 64, 32, 16, 8, 4, 2, 1

By weightage binary no

- self complementing code

Excess 3 Code

0 → 0011	[Not + 3]	
1 → 0100	[0 → 0+3	
5 → 1000	1 → 1+3	
6 → 1001	2 → 2+3	
7 → 1010	9 → 9+3]	
8 → 1011		
9 → 1100		

BCD 8421	2421
0000	0000
0001	0001
0101	0110
0110	1100
0111	1101
1000	1110
1001	1111

84-2-1 (weightage)

0 → 0000

1 → 0111

5 → 1011

6 → 1010

7 → 1001

8 → 1000

9 → 1111

For (185)$_{10}$ = (1011001)$_2$

for reverse check the upper bound should be above one and those up per bounds which are carrying '1' they will add one to another

(Reverse check by using weightage 256, 128, 64, 32, 16, 8, 4, 2, 1)

- *** Self Complementing code
- Excess 3 Code

0.13 0.0213

2 4 2

-12 = (?) 1's

① 12 = 0 0 0 0 1 1 0 0

128 64 32 16 8 4 2 1

↓ 1's complement

~12 1 1 1 1 0 0 1 1

-12 → (1111 0011)1's

② -14 = (?)2's

14 = 0 0 0 0 1 1 1 0

128 64 32 16 8 4 2 1

↓ 2's complement

~14 = 1 1 1 1 0 0 1 0 -14 = (11110010)2's

(Reverse check using weight

ASCII character code – 7 bit format

→ 0100011

0 → (011) 0000 → 0

(011) 0001 → 1

(011) 10001 → 17

No. system binary

(x AND y) = x.y

(x OR y) = x+y

(NOT (x)) = $\bar{y}$ = y' = y'

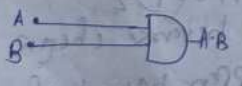
} Binary operation x, y binary variable

x	y	x.y
0	0	0
0	1	0
1	0	0
1	1	1

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

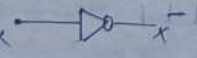
y	$\bar{y}$
0	1
1	0

• AND gate

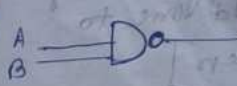


• OR gate

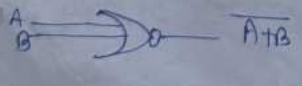
• NOT gate



• NAND gate



• NOR gate



• XOR (exclusive OR) ⊕

• XNOR (exclusive NOR) ⊙

XOR →

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

[even no. of 1 = 0
odd no. of 1 = 1]

XNOR →

x	y	x⊙y
0	0	1
0	1	0
1	0	0
1	1	1

[even = 1
odd = 0]

Ans

1 bit = 2^1 = 2, [0, 1]

2 bit = 2^2 = 4

3 bit = 2^3 = 8

4 bit = 2^4 = 16

10 bit = 2^{10} = 1024 = 1K

20 bit = 2^{20} = 1048576 = 1 mega

30 bit = 2^{30} = 1 G

1 GB = 2^{30} Byte

= $2^{30} \times 8$ bit

1 GB = 2^{33} bit

4 bit = unsigned → 0-15 [0000 to 1111]
(All are +ve no.)

* Signed magnitude – 0,000 1,111 → -7
1,000 0,111 → +7
↓
(+7 to -7)

* Signed 1's Comp form – 0000 (+0)
1111 (1's C) → 0000 (-0)
(+7 to -7)
[16 combination including +0 and -0]

1's C 1000
0111 (-7)

* Signed 2's Comp form –
[in this we don't have -0] [only 0 is included]
[7 to -8]

0000
1's C 1111 +1
2's C 1000
1000 → (-8)

* Formula for signed 2's comp – (-2^{n-1}) to $(2^{n-1}-1)$

* Boolean Algebra [Binary variable and operation, logic operations]

x, y, 1, 0, AND (·), OR (+), NOT ()'/($\bar{}$)

* AND gate (Binary multiplication)

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

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

Identity laws
0 · x = 0
1 · x = x
x · x = x
x · $\bar{x}$ = 0

* OR gate (Binary addition)

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

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

Identity laws
0 + x = x
1 + x = 1
x + x = x
x + $\bar{x}$ = 1

NOT gate

$$\overline{\overline{x}} = x$$

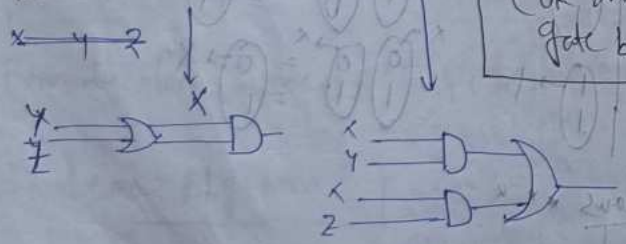
$\overline{\overline{0}} = 0$	
$\overline{\overline{1}} = 1$	

$x + (y + z) = (x + y) + z$ → Associative Law (OR gate supports)

$x \cdot (y \cdot z) = (x \cdot y) \cdot z$ → Commutative Law [AND gate supports]

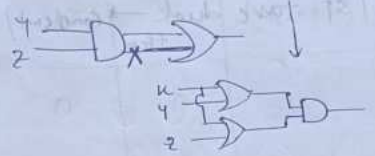
$$\begin{cases} x + y = y + x \\ x \cdot y = y \cdot x \end{cases}$$

$x \cdot (y + z) = x \cdot y + y \cdot z$ → Distributive Law (OR and AND gate both supports)



Excecutive law

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



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

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

$= x(1 + z + y) + y \cdot z$

$= x + y \cdot z$

$[1 + z + y = 1$ ∵ Boolean algebra$]$

$x + (y \cdot z) = (x + y) \cdot (x + z)$ → Executive Law (AND & OR gate both supports)

Duality

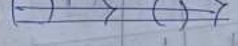
$x \cdot (y + z) = x \cdot y + x \cdot z$

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

AND → OR, OR → AND → Duality

Operator's Precedance

AND > OR



$(.) > ( )' > (\cdot) > (+)$

1st Braket > NOT > AND > OR

Absorption

$x + (x \cdot y) = x \cdot (1 + y) = x$

$x \cdot (x + y) = x + (1 \cdot y) = x$

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

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

~~$x(x + y)$~~ $= x + x \cdot y$; $= x(1 + y) = x$

De morgan's theorem / [Rule of complement]

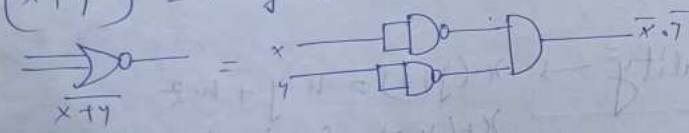
[1st take dual → then complement]

$$\overline{(x+y)} = \bar{x}\cdot\bar{y}$$
$$\overline{(x\cdot y)} = \bar{x}+\bar{y}$$

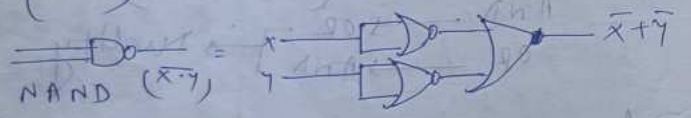
Dual ⟶ (+) ⟷ (·)

Complement → 0 ⟷ 1 of a function

$\overline{(x+y)} = \bar{x}\cdot\bar{y}$ ↓



(NOR)



$\overline{(x\cdot y)} = \bar{x}+\bar{y}$ ↓

Prove $\overline{(x+y)} = \bar{x}\cdot\bar{y}$ (Truth table)

x	y	(x+y)	$\overline{(x+y)}$	$\bar{x}$	$\bar{y}$	$\bar{x}\cdot\bar{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

we can see $\overline{(x+y)} = \bar{x}\cdot\bar{y}$

Prove $\overline{(x\cdot y)} = \bar{x}+\bar{y}$ (Truth table)

x	y	(x·y)	$\overline{(x\cdot y)}$	$\bar{x}$	$\bar{y}$	$\bar{x}+\bar{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

we can see $\overline{(x\cdot y)} = \bar{x}+\bar{y}$

D·m theorem by function

$\overline{(x+y)} = \bar{x}\cdot\bar{y}$

F_1 ↓ F_2 ↓

$F_1 = \overline{x+y}$, $F_2 = \bar{x}\cdot\bar{y}$

Eg

(1) $F = x + \bar{y}\cdot z$ → method of evaluation ← [shortcut] (type-1)

[x + (ȳ · z)] As per Precedence

x	y	z	$\bar{y}$	$\bar{y}\cdot z$	$x+\bar{y}z$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

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

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

$= \bar{x}yz + x$

② $F_2 = \bar{x}\cdot\bar{y}\cdot z + \bar{x}\cdot y\cdot z + x\cdot\bar{y}$

$= \bar{y}(\bar{x}+x)$

x	y	z	$\bar{x}\bar{y}z + \bar{x}yz + x\bar{y}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

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

$= x\bar{y}(z+\bar{z})$ ← [shorthand calculation]

$= x\bar{y}\cdot 1$

$= x\bar{y}$

other process

$F_2 = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}$

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

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

→ Reduction (literal)

Here 8 literal reduced in 4

[(Production) Here 3 Production reduced in 2]

Reduction → Literal

→ Production

① $x(\bar{x}+y) = xy$

② $x + \bar{x}y = x+y$

$\Rightarrow (x+\bar{x})\cdot(x+y) = x+y$ $[\because x+yz = (x+y)(x+z)]$

(1)

③ $(x+y)(x+y') = x + xy' + xy$

$= x\cdot 1 + xy$

$= x$

④ $xy + x'z + yz = x(xy + x'z + yz)$

$xy + yz$

$y(x+z)$

$= xy + x'z + yz(x+\bar{x})$

$= xy + x'z + xyz + x'yz$

$= x'z(1+y) + xy(1+z)$

$= x'z + xy$

⑤ $(x+y)\cdot(x'+z)\cdot(y+z) = (x'+z)\cdot(x+y)$

$\Rightarrow [xz + x'y + yz](y+z)$

$= xyz + x'y + yz + xz + x'yz + yz$

$= xyz + yz + xz + x'y$

$= yz(1+x) + xz + x'y$

$= yz + xz + x'y$ (LHS)

$(x'+z)(x+y)$ (RHS)

$= xz + yx' + yz$

LHS = RHS (proved)

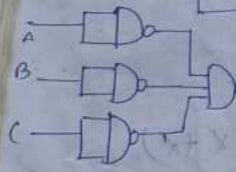
* Complement of a function

$(A+B+C)' = (A+x)'$ [Let $B+C = x$]

$= A' \cdot x'$

$= A' \cdot (B+C)'$

$\boxed{(A+B+C)' = A' \cdot (B' \cdot C')}$

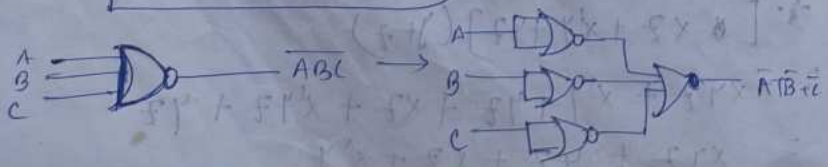


$\boxed{(A+B+C+D)' = A'B'C'D'}$

$\overline{A\,BC} = \bar{A} + B\bar{x}$ [Let, $BC = x$]

$= \bar{A} + \overline{BC}$

$\boxed{\overline{ABC} = \bar{A} + \bar{B} + \bar{C}}$



$F_1 = x'yz' + xy'z'$

$F_1' = (x'yz' + xy'z')'$

$= x'(yz+y)$

$= [x'z(x+y')]' = \overline{x'y'z'} + \overline{xy'z}$ [A·B]

$= (x+\bar{y}+z)\cdot(x+y+\bar{z}')$

$= (x'z)'$

$= xz'$

$A' = (x'yz')' = (x'\,\textcircled{B})' = x+y'+z$

Similarly

$B' = (x'y'z)' = x+y+z'$

$F_1 = [(x'yz') \cdot (xy'z)]$

↓ in one step

$= [(x+z)\,[(x+y'+z)\cdot(x+y+z')]$

$F_2 = x(y'z' + yz)$

$= xy'z' + xyz$

$F_2' = [\overline{(xy'z')(xyz)}]$

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

$= x'+yz'+y'z$

[$x'y' + x'z' + xy + yz' + xz + y'z$]

$= x'(1+y'+z') + yz' + y'z$

$= x'(1+1) + yz' + y'z = x' + yz' + y'z$

others

Process $F_2 = x(y'z' + yz)$

$F_2' = [x(y'z' + yz)]'$

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

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

$= x' + yz' + zy'$

in one step, $[x(y'z' + yz)]' = x' + (y+z) \cdot (y'+z')$

$= x' + yz' + zy'$

$(x+y+z) \cdot (x'+y'+z')$

$x \cdot 0 = 0$) Dual
$\bar{x} + 1 = 1$
↓ identity

Sm — min term

x	y	z		
0	0	0	$x'y'z'$	m_0 [min term Zero] (Binary Combination is 0)
0	0	1	$x'y'z$	m_1 [min term One] (< < is 1)
0	1	0	$x'yz'$	m_2 [i = 0, 1, 2, — 7]
0	1	1	$x'yz$	m_3
1	0	0	$xy'z'$	m_4 ↓ for here
1	0	1	$xy'z$	m_5
1	1	0	xyz'	m_6
1	1	1	xyz	m_7

Base = 1
Based(') = 0

- Fun can be represented in sum of min term $F = \sum m_i = \sum_m (0,2,4,6)$ → (Any combination)
  ↓ this min terms are in the F (function)

Im

- max term = complement of min term

Base = 0
based = 1

max term

$M_0 = \bar{m}_0 = (x+y+z)$	$M_4 = \bar{m}_4 = x'+y+z$
$M_1 = \bar{m}_1 = (x+y+z')$	$M_5 = \bar{m}_5 = x'+y+z'$
$M_2 = \bar{m}_2 = x'+y'+z$	$M_6 = \bar{m}_6 = x'+y'+z$
$M_3 = \bar{m}_3 = x'+y'+z'$	$M_7 = \bar{m}_7 = x'+y'+z'$

- max Prime = If Prime then 1
- $G = \prod_M (1,3,5,7)$
- Fun can be represented in Product of max term.

x	y	z	F	G
0	0	0	1	1
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

Som ⟷ POM
F ⟷ G

$F = G$, when $F = \sum_m (0,2,4,6)$, $G = \prod_M (1,3,5,7)$
$F = \bar{G}$, when $F = \sum_m (0,2,4,6)$, $G = \prod_M (0,2,4,6)$

If Same index
$G = \prod_M (0,2,4,6)$
$F = \sum_m (0,2,4,6)$
then, $F = \bar{G}$

- $F = \sum_m (1,2,3,4)$
  $\bar{F} = \prod_M (1,2,3,4)$
  If we keep $\prod_M$ and exclude $(1,2,3,4)$ then
  $\bar{F} = F = \prod_M (0,5,6,7)$
  → How to Convert Som ⟷ POM

max term.

x	y	z	F	G
0	0	0	1	0 1
			0	0

m_0 [min term ze
m_1 [min term 0
m_2 [i = 0, 1,
m_3 for
m_4
m_5
m_6
m_7

nted in sw

$i] = \sum_m (0, 2$

this
and

plement of

$m_4 =$
$m_5 = \bar{x}$ = x'+y+z'

$F = \bar{F}$, when $F = \sum_m (1,2,3,4)$
$\bar{F} = \prod_m (1,2,3,4)$

$F = F[\bar{F}]$, when, $F = \sum_m (1,2,3,4)$
$F = \prod_m (0,5,6,7)$

$F = \sum_m (1,2,3,4)$

$\bar{F} = \prod_m (1,2,3,4)$

$\therefore \bar{F} = \prod_m (1,2,4,5)$

$\bar{\bar{F}}(F) = \prod_m (0,5,6,7)$

∴ If, $\bar{F}$ given, $F = ?$ then,

① ② ③ ④

② find $F = F'$, $F' = \prod_m (1,2,3,4)$

③ find $F = F$, $F = \prod_m (0,5,6,7)$

$\bar{F} = \prod_m (1,2,3,4)$

$F = \sum_m (1,2,3,4)$ [Result in S]

$\bar{\bar{F}} = F = \prod_m (0,5,6,7)$ [Result in P]

(m) others Power

ex

, 4, 6)

, 4, 6)

, 3, 4) then

$\bar{\bar{F}} = F = \prod_m (0,5,6,7)$

$G = \prod m(1,3,5)$

$\overline{G} = \sum_m (1,3,5)$

$\overline{\overline{G}} = \sum_m (0,2,4,6,7)$

Pom ⟶ Som

[SOP → Som]

① $F = A + \bar{B}C$ [Sum of Product term] [SOP] / [Som]

$F = \sum_m (1,4,5,6,7) \rightarrow$ Som $= m_1 + m_4 + m_5 + m_6 + m_7$

$F = \prod_m (0,2,3) \longrightarrow$ POM $= M_0 M_2 M_3$

$F(A,B,C) = A(B+\bar{B})(C+\bar{C}) + \bar{B}C(A+\bar{A})$

$= A[BC + B\bar{C} + \bar{B}C + \bar{B}\bar{C}] + \bar{B}C[A+\bar{A}]$

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

$= m_7 + m_6 + m_5 + m_4 + m_1$

[SOP → Pom]

② $F = xy + x'z$ [Sum of max terms] [Pos] ~~[Som]~~ [Pom]

$F(x,y,z) = xy(z+z') + x'z(y+y')$

$= xyz + xyz' + x'yz + x'y'z$

[Pos → Pom] $= (xy+x')(xy+z)$ [Applying executive law]

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

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

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

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

$= m_0 M_4 M_5 M_2$

2/8/22

Sum of minterm (Som)

$f(x,y) = \sum m(1,2,3)$: OR gate $= m_1 + m_2 + m_3$

x	y	F(x,y) = OR
$m_0 = (x'y')$	0 0	0
$m_1 = (x'y)$	0 1	1 [Present = 1]
$m_2 = (xy')$	1 0	1
$m_3 = (xy)$	1 1	1

Product of max term (Pom)

$F(x,y) = \prod_M (0,3) = M_0 \cdot M_3$

$F = \overline{\overline{F}} = \sum_m (1,2)$

x	y	F(x,y)
(x+y)	0 0	0 [Present = 0]
(x+y')	0 1	1
(x'+y)	1 0	1
(x'+y')	1 1	0 [XOR gate]

- Canonical form ⇌ Som / Pom
- SOP (Sum of Product) / POS (Product of sum) → standard form

$F = (x'+y)(x+z)(y+z)$

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

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

$(x+y+z)(x'+y+z)$

$F = \Pi m(1,3,5)$

Since the procedure to get SOP → SOM and POS → POM is one different. So, if the m is SOP → POM, the intermidiate step will be SOM

Alternatively -
if POS → SOM, the intermidiate step will be POM

(a) If you want to directly get SOP → POM, then convert the given function into that form by which you can directly get POM, with the help of exclusive law.

Similarly, If you want to directly get POS → SOM, then convert the given func into that form by which you can directly get SOM, with the help of reverse of exclusive law.

[S.f. $(xy + x')(xy + z) = xy + x'z$]

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

$(x+y+z')$

$P = xy + x'z$

$= (x \cdot y + x \cdot 1)(x' \cdot y + z)$

$= (x+1)(y+1)(x+z)(y+z)$

$= 1 \cdot (x'+y)(x+z)(y+z)$

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

$(x'+y+z)(x'+y+z')(x+y+z)$

$(x+y'+z)(x'+y+z)$

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

$(x+y+z)(x+y'+z)$

$(x+y+z)(x'+y+z)$

$= M_6 \; M_7 \; M_0 \; M_4 \; M_2$

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

$= M_4 \; M_5 \; M_0 \; M_2$

using 2 variable how many func we can represent –

(In form of SOM)

	x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}
m_0 $x'y'$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
m_1 $x'y$	0	1	0	0	0	0	1	1	1	1	0	0	0	0
m_2 xy'	1	0	0	0	1	1	0	0	1	1	0	0	1	1
m_3 xy	1	1	0	1	0	1	0	1	0	1	0	1	0	1

0 ≡ (Ground) 1 ≡

m_0 → 2 [absent / present → 2 combination]

m_1 → 2

m_2 → 2

m_3 → 2

$2^4 = 16$

$F_0 = 0$ = ground = GND

$F_1 = x \cdot y$ = AND

$F_2 = x \cdot y' = x > y$

$F_3 = xy' + xy = x$

$F_4 = x'y = x < y$

$F_5 = x'y + xy = y$

$F_6 = x'y + xy'$ = xor gate

$F_7 = x'y + (xy' + xy) = x'y + x = x + y$ = OR gate

$F_8 = x'y' = \overline{x+y}$ = NOR gate

$F_9 = x'y' + xy$ = x NOR gate

$F_{10} = x'y' + x'y = y'$ = NOT gate

$F_{11} = x'y' + xy' + xy = y' + xy = (x+y)(y'+y) = \overline{y \cdot x'} = \overline{x'} = x$

if y, then x (inhibite) If y is true then x is true [i.e. $y=1, x$]

F_{12}	F_{13}	F_{14}	F_{15}
1	1	1	1
1	1	1	1
0	0	1	1
0	1	0	1

} 7 basic gates are present

Supply (all one) / VDD = Vcc

} greater than / less than → special → 4 special are present

$F_{12} = x'y' + x'y$

$= x'$ = NOT gate

$F_{13} = x'y' + x'y + xy$

$= x' + xy$

$= x' + y$

$= \overline{x' \cdot y'} = \overline{y} = y$

If x is true (that means $x=1$) then y is true (Implication)

$F_{14} = x'y' + x'y + xy'$

$= x' + xy'$

$= x' + y'$

$= \overline{x \cdot y}$ = NAND gate

$F_{15} = 1$ = supply

$[x'y' + xy' + x'y + xy]$

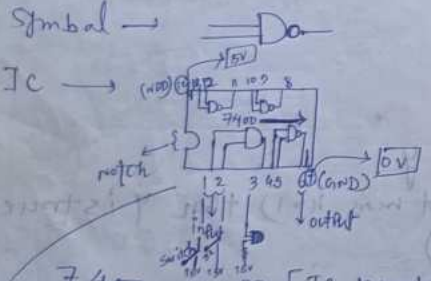
$= y' + y = 1$

1) NAND } universal gate — 7400
2) NOR } universal gate — 7402
3) NOT — 7404
4) AND } Basic gate — 7408
5) OR } Basic gate — 7432
6) XOR — 7486
7) X NOR — 747266

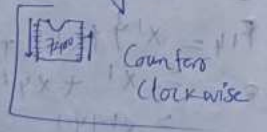
(1) NAND gate (what is NAND gate)
(i) Symbol (ii) IC (what)
(iii) Boolean (what) expression (what)
(iv) Truth table (0,1) (what)
(v) verification table (on, off) (what)
→ what to do in lab report

All NAND gate has the direction →

Symbol →
IC →



[GND have to connect the point to the grounded]
[VDD = supply]



74 — — 00 [IC numbers]
→ NAND IC (7400)

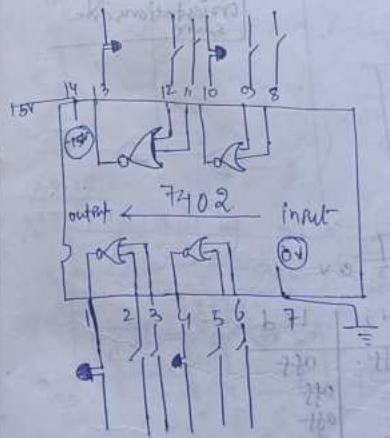
SW1	SW2	Led
off	off	on
off	on	on
on	off	on
on	on	off

→ verification table

T.T

SW1	SW2	led
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate



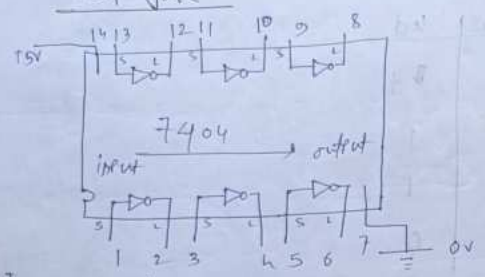
V.T

SW1	SW2	Led
off	off	on
off	on	off
on	off	off
on	on	off

II

SW1	SW2	Led
0	0	1
0	1	0
1	0	0
1	1	0

NOT gate



T·T

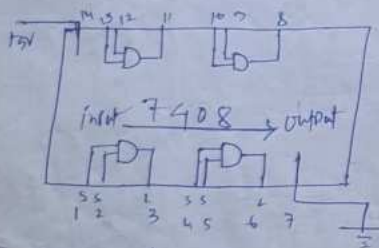
x	x'
0	1
1	0

V·T

SW	led
on	off
off	on

Except NOR gate every orientations are same

AND gate



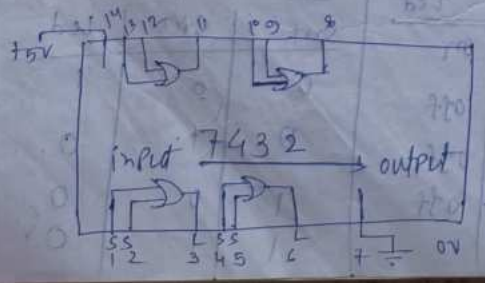
TT

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

VT

SW1	SW2	Led
off	off	off
off	on	off
on	off	off
on	on	on

OR gate



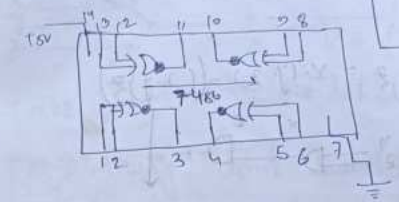
TT

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

VT

SW1	SW2	led
off	off	off
off	on	on

XOR



on	off	on
on	on	on

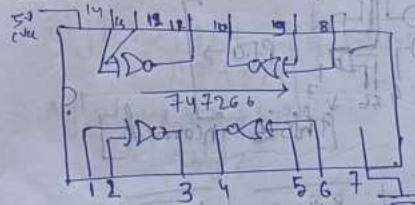
T·T

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

V·T

SW1	SW2	led
off	off	off
off	on	on
on	off	on
on	on	off

XNOR



T·T

x	y	x⊙y
0	0	1
0	1	0
1	0	0
1	1	1

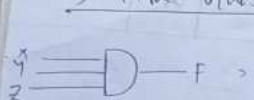
V·T

SW1	SW2	led
off	off	on
off	on	off
on	off	off
on	on	on

16/8/22

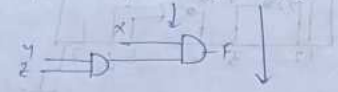
class

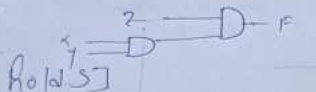
3-input gates



(AND gate)

$F = x \cdot y \cdot z = x \cdot (y \cdot z) = (x \cdot y) z$

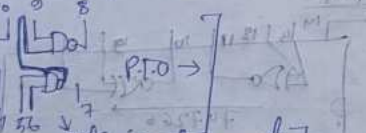




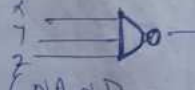
[Associativity Holds]

(OR gate)

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

[3 input ic - 

[This is incorrect]

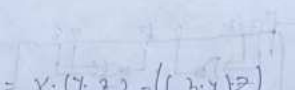


(NAND gate)

$= \overline{x \cdot y \cdot z} \neq \overline{x \cdot y \cdot z} \neq \overline{x \cdot y \cdot z}$

$(\bar{x}+\bar{y}+\bar{z}) \neq (xy+\bar{z}) \neq (\bar{x}+yz)$

[Associativity donot Hold in this case]



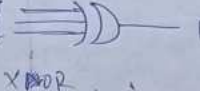
$= \overline{x+y+z} \neq \overline{x+y+z} \neq \overline{x+y+z}$

$(\bar{x} \cdot \bar{y} \cdot \bar{z}) \neq (x+y) \cdot \bar{z} \neq \bar{x} \cdot (y+z)$

(NOR gate)

[Associativity donot holds]

[Not gate don't have any multiple input]



(XOR gate)

$F = x \oplus y \oplus z = (x \oplus (y \oplus z)) = ((x \oplus y) \oplus z)$

[Associativity holds]

$x \oplus (y'z + yz')$

(m) $x(yz + y'z') + x'(y'z + yz')$ ← $= x(y'z + yz')' + x'(y'z + yz')$

$= x(\overline{y'z} \cdot \overline{yz'}) + x'y'z + x'yz'$

$= x(y+z) \cdot (y'+z') + x'y'z + xyz'$

$= xyz + xy'z' + xy'z' + x'y'z + x'yz'$

$= xy'z' + x'yz' + xyz + x'y'z$

$(xy' + x'y) \oplus z$

$= (xy' + x'y) z' + \overline{(xy' + x'y)} z$

$= xyz' + x'yz' + (xy + x'y')z$

$= xy'z' + x'yz' + xyz + x'y'z$

$x \oplus y \oplus z = xyz + x'y'z + x'yz' + xy'z'$

[Between the previous two Process, can follow one Process to get $x \oplus y \oplus z$]

$x \oplus y = xy + x'y'$

$\overline{x \oplus y} = x'y + xy' = x \odot y$

complement of XOR = XNOR

complement of XNOR = XOR

$F = x \oplus y \oplus z$

$= x \oplus (y \oplus z)$

$= (x \oplus y) \oplus z$

(XNOR gate)

[Associativity holds]

$x \oplus (y \oplus z)$

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

$= x(yz + y'z') + x'\overline{(yz + y'z')}$

$= xyz + xy'z' + x'[(yz)' \cdot (y'z')'] + x'[yz + y'z']$

$= xyz + xy'z' + x'[(y' + z') \cdot (y + z)] + x'yz + x'y'z'$

$= xyz + xy'z' + x'[y'z + yz'] + x'yz + x'y'z'$

$= xyz + xy'z' + x'y'z + x'yz' + x'yz + x'y'z'$

$(x \oplus y) \oplus z$

$= [xy + x'y'] \oplus z$

$= (xy + x'y')z + \overline{(xy + x'y')} \cdot z'$

$= xyz + x'y'z + [(xy)' \cdot (x'y')'] \cdot z'$

$= xyz + x'y'z + [(x' + y') \cdot (x + y)] z'$

$= xyz + x'y'z + [x'y + xy']z'$

$= xyz + x'y'z + x'yz' + xy'z'$

$= xyz + x'y'z + xy'z' + x'yz'$

$= x \oplus y \oplus z$ [Proved]

$1 \cdot 1 \cdot 1 = 1$ [Identity of AND gate]

$0 + 0 + 0 = 0$ [Identity of OR gate]

$\overline{1 \cdot 1 \cdot 1} = \overline{1} = 0$ [Identity of NAND gate]

$\overline{0 + 0 + 0} = \overline{0} = 1$ [Identity of NOR gate]

x	y	z	XOR	XNOR
0	0	0	0	1
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	0

[Even = 0, Odd = 1]

XOR = XNOR

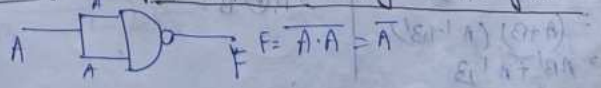
XNOR = XOR

Universal gate

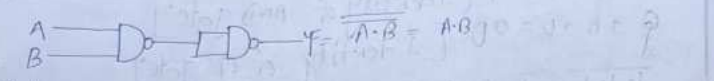
① NAND → AND, OR, NOT, XOR, XNOR } implements

② NOR → AND, OR, NOT, XOR, XNOR } implements

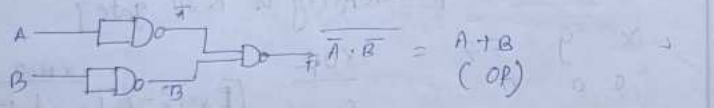
NOT gate implementation using NAND gate –



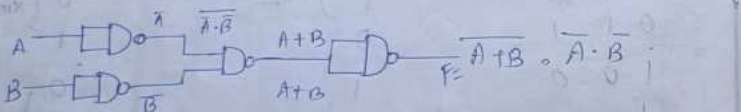
NAND → AND



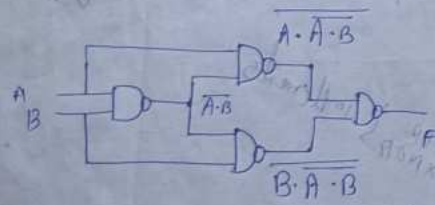
NAND → OR



NAND → NOR



NAND → XOR



$F = \left\{\left(A \cdot \overline{A \cdot B}\right) \cdot \left(B \cdot \overline{A \cdot B}\right)\right\}$

$= \overline{A(\overline{A+B})} \cdot \overline{(B \cdot (A+\overline{B}))}$

$= \overline{A\overline{B}} \cdot \overline{\overline{A}B} \Rightarrow \overline{x} \cdot \overline{y} = \overline{(x+y)}$

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

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

$= A \oplus B$

$= \overline{(A'+B) \cdot (A+B')}$

$= \overline{A'B' + AB}$

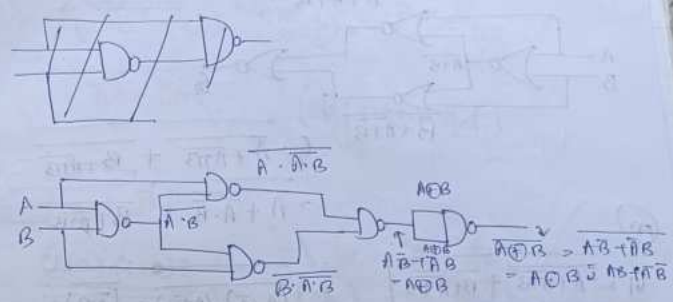
$= \overline{A'B'} \cdot \overline{AB}$

$= (A+B)(A'+B')$

$= AB' + A'B$

$= A \oplus B$

NAND → XNOR



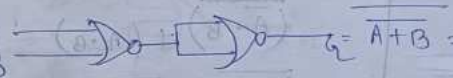
$Y = \overline{A\overline{B} + \overline{A}B}$

$(\overline{A\overline{B}}) \cdot (\overline{\overline{A}B})$

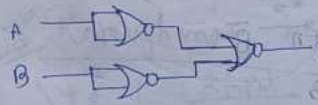
$(\overline{A} + B)(A + \overline{B})$

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

NOR → OR

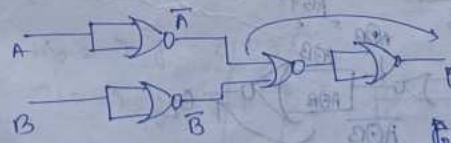


NOR → AND



$= A \cdot B$

NOR → NAND



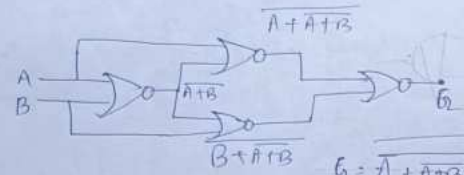
$\overline{A+B} = \overline{A} \cdot \overline{B}$

$= A \cdot B$

or

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

NOR → XNOR



$$G_2 = \overline{A+\overline{A+B}} + \overline{B+\overline{A+B}}$$

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

(m)

$$G_2 = \overline{A+\overline{A+B}} + \overline{B+\overline{A+B}}$$

$$= \overline{\overline{A}(A+B)} + \overline{B(A+B)}$$

$$= \overline{(\overline{A}B + A\overline{B})}$$

$$= A \odot B$$

(m)

$$G_2 = \overline{A+\overline{A+B}} + \overline{B+\overline{A+B}}$$

$$= \overline{A+(\overline{A}\cdot\overline{B})} + \overline{B+(\overline{A}+B)}$$

$$= \overline{(A+B)} + \overline{(\overline{A}+B)}$$

$$[\because \overline{x+y} = (\overline{A+B})(\overline{A}+B) = A\overline{B}+\overline{A}B = A \odot B]$$

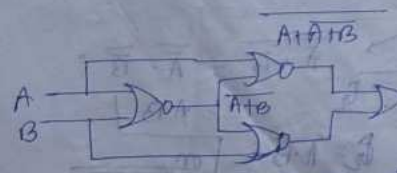
$$= \overline{(A+\overline{B})\cdot(\overline{A}+B)} + \overline{(A\cdot\overline{B})}$$

$$= (A+\overline{B})\cdot(\overline{A}+B) + (\overline{A}+B)$$

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

$$= A \odot B$$

NOR → XNOR



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

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

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

$$= A \oplus B$$

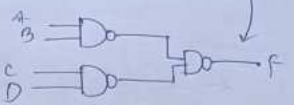
2 level function using NAND gate

$$F = AB + CD$$

$$= \overline{(\overline{AB} + \overline{CD})} \quad \text{(SOP)}$$

$$= \overline{\overline{AB}\cdot\overline{CD}}$$

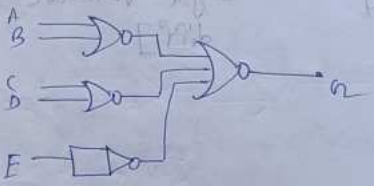
(By NAND gat)



$$G_2 = \overline{(A+B)(C+D)E} \quad \text{(POS)}$$

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

(By NOR gate)



Karnaugh map

2 input

x \ y	$\bar{y}$ (0)	y (1)
$\bar{x}$ (0)	m_0	m_1
x (1)	m_2	m_3

→ K map (truth table)

x \ y	0	1
0	m_0	m_1
1	m_2	m_3

$m_0 = \bar{x}\bar{y} = 0\cdot 0 = 0$

$m_1 = \bar{x}y = 0\cdot 1 = 0$

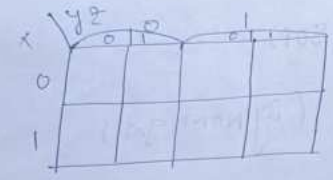
$m_2 = x\bar{y} = 1\cdot 0 = 0$

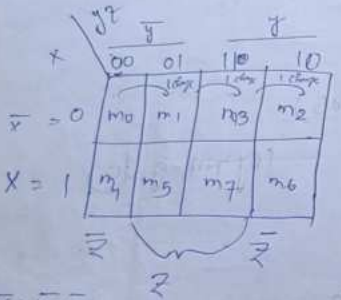
$m_3 = xy = 1\cdot 1 = 1$

→ verify the position

3 input ***

msd = x





[All units contain single variable diff]

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

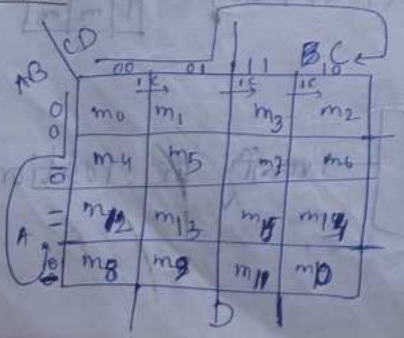
$m_3 = \bar{x}\,yz$

$m_2 = \bar{x}\,y\bar{z}$

$m_4 = x\bar{y}\bar{z}$

$m_6 = xy\bar{z}$

4 input ***



A → MSD

D_2 LSD

row = AB

Column = CD

[All units contains single variable diff]

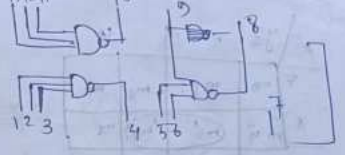
$m_{15} = AB\cdot C\cdot D$

$m_8 = A\bar{B}\bar{C}\bar{D}$

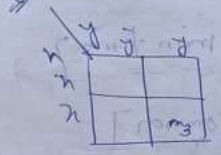
$m_{11} = A\bar{B}CD$

$m_{12} = AB\bar{C}\bar{D}$

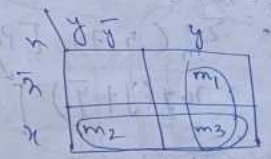
[3 input IC →



23/8/22



$F = m_3 = x\cdot y$ (AND)

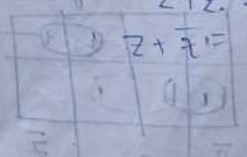


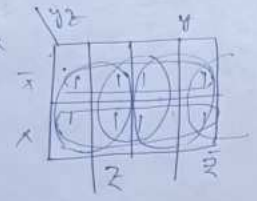
$F(x,y) = \sum_m (1,2,3)$

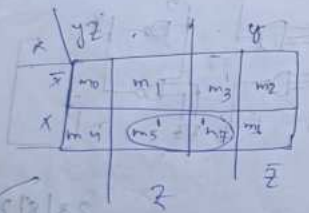
$\bar{x}y + x\bar{y} + (xy + xy)$

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

or gate (y+x)







$F_1 = \sum_m (5,7)$ [Product of min terms]

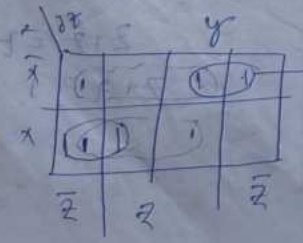
$= xz(y + \bar{y})$ [those are common]

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

$= xz(y + \bar{y})$

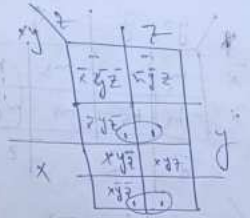
(Collection of adjusent min terms)

$F(x,y,z) = \sum_m (2,3,4,5)$



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

$= x \oplus y$



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

(XOR gate)

$F_2 = \sum_m (0,2)$

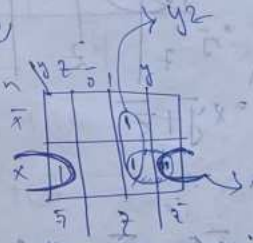
$= \bar{x}\bar{z}$ (common)

$F_3 = \sum_m (4,6)$

$= x\bar{z}$ (common)

$F_4 = \sum_m (3,4,6,7)$

$= yz + x\bar{z}$



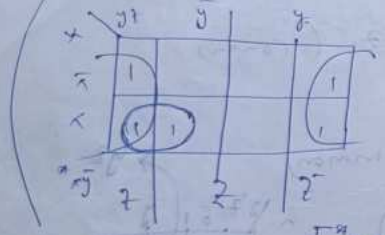
Non overlapping minimum collection we will find the bigger non overlapping collection.

$F(x,y,z) = \sum_m (0,2,4,6)$

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

[y, ȳ / z, z̄ / x, x̄ both present = minimized]

$F = \sum_m (0, 2, 4, 5, 6)$



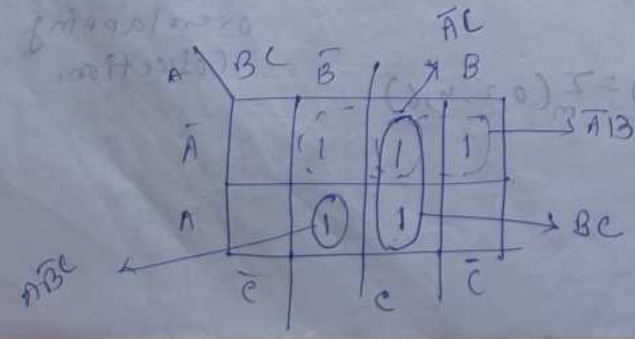
$= x\bar{y} + \bar{z}$

["z, z̄ both present that's why minimized]

i) $F = \bar{A}C + \bar{A}B + A\bar{B}C + BC$

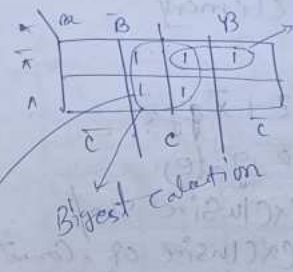
What is sum of the ~~fune~~ min term, minimal SOP?

$F(A,B,C) = \sum_m (?)$



$F(A,B,C) = \sum_m (1,2,3,5,7)$

minimal Sum of Product (non overlaping pijule calection)



[(1 1 1), ~~(1 0 1)~~ NO
(1 0 1) → yes]

[overlapping ambality's fmula $2^n = 1, 2, 4, \&16 \ldots$]

[1 common → 3
2 " → 2
" → 1]

minimal SOP = $\bar{A}B + C$

$F = \bar{A}\bar{B}C + \bar{A}B + A\bar{B}C + BC$

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

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

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

$= \bar{A}B + C(\bar{A} + 1)$

$= \bar{A}B + C$

(m)

$\bar{A}C + \bar{A}B + A\bar{B}C + BC$

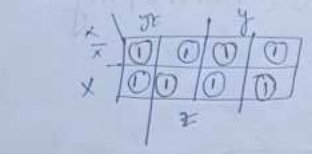
$= \bar{A}C + \bar{A}B + A\bar{B}C + (A + \bar{A})BC$

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

$= \bar{A}B + AC + \bar{A}C$

$= \bar{A}B + C$

$F(x,y,z) = \sum_m (1,2,4,7)$



[Can't called diagonal element]

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

$= x \oplus y \oplus z$ (XOR gate)

[exclusive sum/ exclusive OR, cannot take anything common, that means no overlapping]

(Denoted by black)

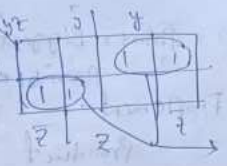
$F(x,y,z) = \sum_m (0,3,5,6)$

$= x \odot y \odot z$

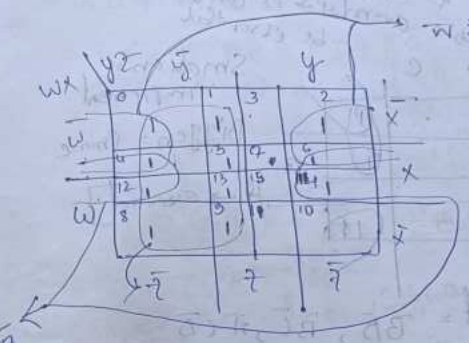
(XNOR gate)

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

$= x \odot y \odot z$



this two are diagonal. So, it will be XOR gate



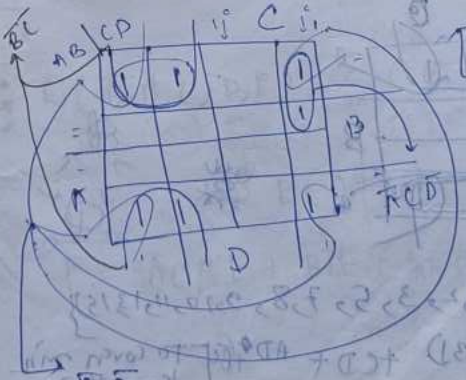
[W is MSD bit always keep here]

[W, X minimized]

$F(w,x,y,z) = \sum_m (0,1,2,4,5,6,8,9,12,13,14)$

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

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



[can take common elements]

minimal SOP

$B\bar{D} + \bar{B}\bar{C} + \bar{A}C\bar{D} +$

Essential Prime implement

Prime = Biggest possible Implement = Product term

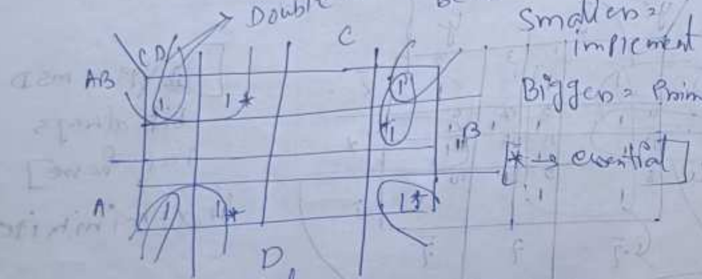
- F = EPI's + PI's$_1$
  = EPI's + PI's$_2$

Double overlap → it can't be essential

Smaller = implement

Bigger = Prime

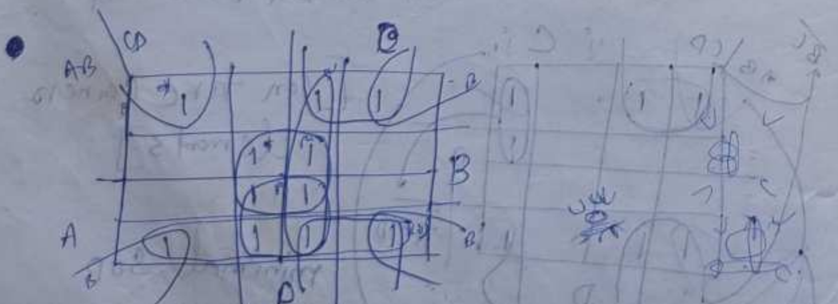
[* → essential]



3 Prime implement = $\bar{B}\bar{D}$, $\bar{B}\bar{C}$, $\bar{A}C\bar{D}$

Essential = A Prime implement that covers a specific minterm

(Function) $F = \bar{B}\bar{D} + \bar{B}\bar{C} + \bar{A}C\bar{D}$ (All are essential Prime implement)

- 

$F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$

$= \bar{B}\bar{D} + BD + CD + AD + \bar{B}C$

$= \bar{B}\bar{D} + BD + CD + A\bar{B}$

[To cover minterm these boundary are needed]

PI's → $\bar{B}\bar{D}$, BD, CD, $\bar{B}C$, AD, $A\bar{B}$

EPI's → $\bar{B}\bar{D}$, BD

$= \bar{B}D + BD + \bar{B}C + A\bar{B}$

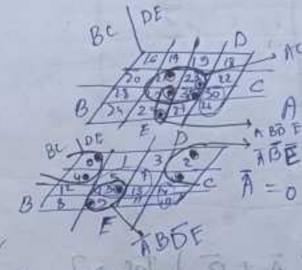
$= \bar{B}\bar{D} + BD + CD + A\bar{B}$

$= \bar{B}\bar{D} + BD + CD + AD$

[Essentials are common, remaining will be possible combination]

[$\bar{B}\bar{D}$, BD common for all EPI: BD, $\bar{B}\bar{D}$]

11111 = 32 (5 variable)



$P_1\bar{A} + P_1A = P_1(A + \bar{A})$
$= P_1$

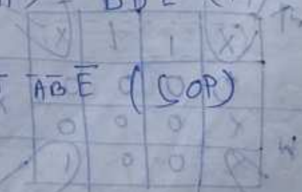
[If mirror image then can reduce that variable ($A + \bar{A}$)]

$P_2\bar{A} + P_2A = P_2(\bar{A} + A)$
$= P_2$

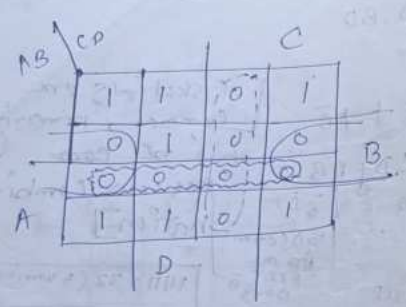
$F(A, B, C, D, E) = \sum_m(0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$

$B\bar{D}E(A + \bar{A}) = B\bar{D}E$ (mirror image)

$F = ACE + B\bar{D}E + \bar{A}\bar{B}\bar{E}$ (SOP)



$F(A,B,C,D) = \sum_m (0,1,2,3,8,9,10)$



[How to get POS]

$F = B\bar{D} + CD + AB$ (1) (2) (3)

$F = \overline{(B\bar{D} + CD + AB)}$

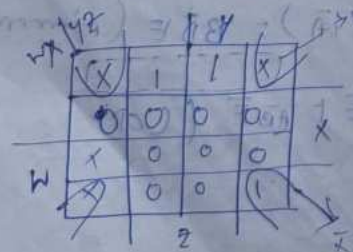
$F = (\bar{B} + D)(\bar{C} + \bar{D})(\bar{A} + \bar{B})$ [POS]

Don't care condition

(output unspecified, input absent, i.e. mean term is not present)

X → 0 [Don't specified]
→ 1 need not

$F(w,x,y,z) = \sum_m (1,3,10) + \sum_d (0,2,8,12)$



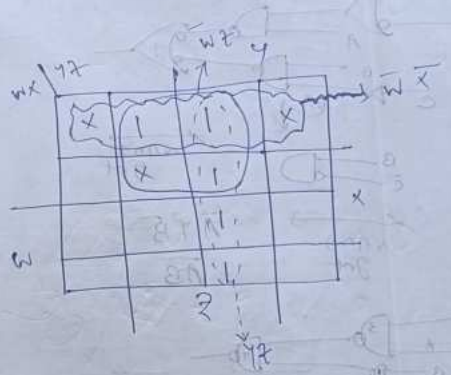
[x = Don't care]

[Consider 1 = covered
Consider 0 = Not covered]

$F = \bar{x}\bar{z} + w\bar{x}$

[Any collection should atleast contain (minterm)]

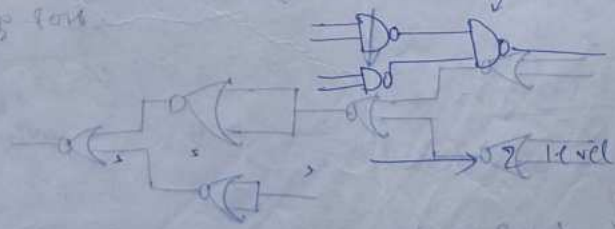
$F = \sum_m (1,3,7,11,15) + \sum_d (0,2,5)$



$F = \bar{w}z + \bar{w}\bar{x} + yz$ [yz is essential, as it covers 1 must have to take, if don't take x that doesn't matter. To take 1 must have to take that, which covers 1, i.e essential to cover]

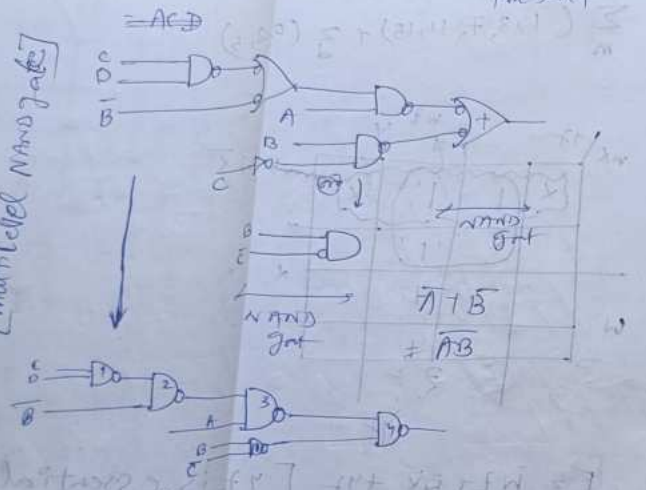
as, ⊃ and ○ covers the remaining (1,1) of yz]

Sop can be implemented in → 2 level NAND



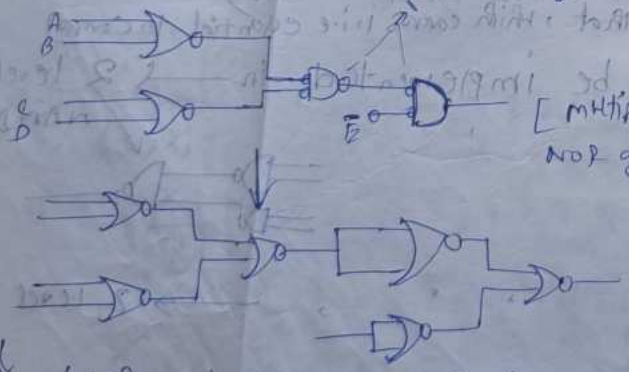
POS → 2 level NOR

$F = A(CD+B) + B\bar{C}$ → both sop and pos present



NAND implementation

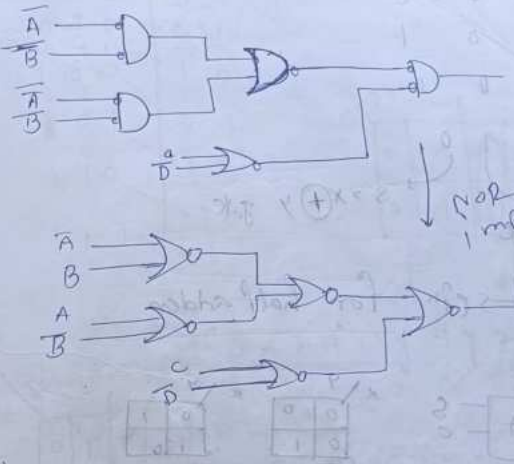
$F = (A+B)(C+D)\bar{E}$



↓

(NOR gate implementation)

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



UW

30/8/22

Boolean Algebra → Binary Arithmatic

$X+X=X$ ↑ OR

$X+Y =$ "CS" (carry bit & Sum bit)

↑ Add ↑ 2 bit

half adden

X	0	0	1	1
+Y	+0	+1	+0	+1
CS	00	01	01	10
	CS	CS	CS	CS

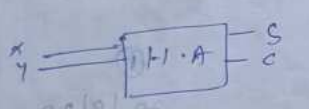
Carry ↙ ↘ Sum

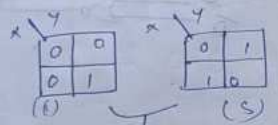
x	y	C (MSB)	S (LSB)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

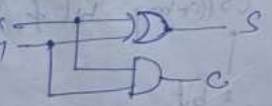
C = x·y gate

S = x ⊕ y gate

C = x·y
S = x⊕y
} eqn for half adder



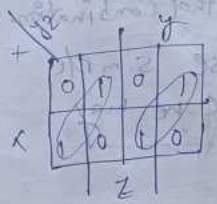




Full Adder

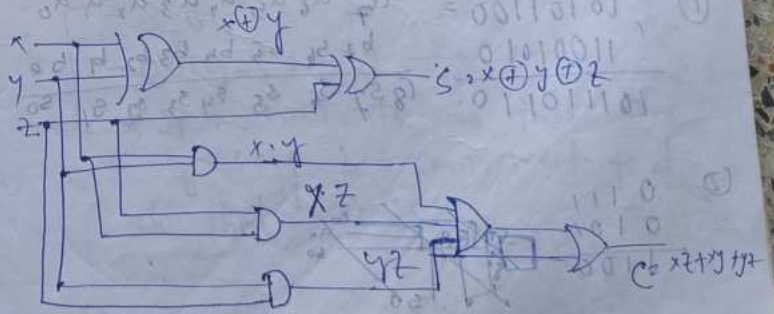
x
+ y
+ z (carry in)
CS → Sum
(carry out)

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

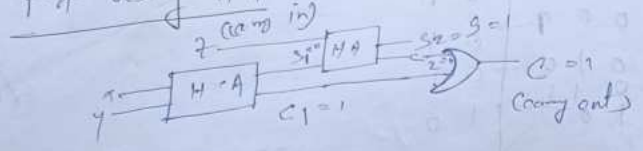


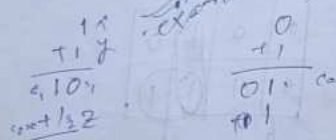
S = x ⊕ y ⊕ z

C = xz + xy + yz



F.A using H.A



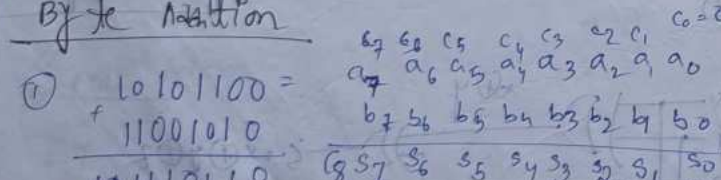


→ we are using OR gate to get that combination.

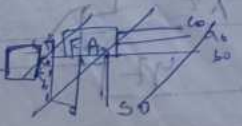
[Basically we need XOR gate but OR gate is enough as this 1,1 combination will never occure If we use XOR gate]

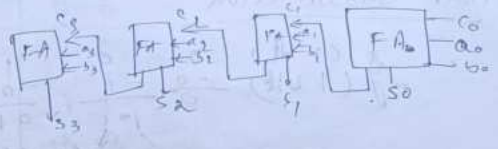
If we use simple XOR gate we may lose 10.

Byte Addition



② 0 1 1 1
0 1 0 1
1 1 0 0

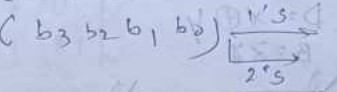


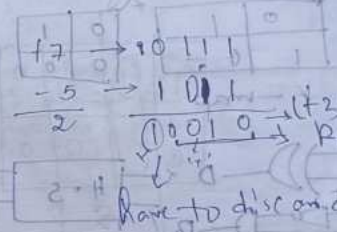


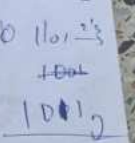
Subtraction (by using 4 bit)

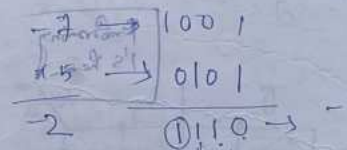
A (a3 a2 a1 a0)

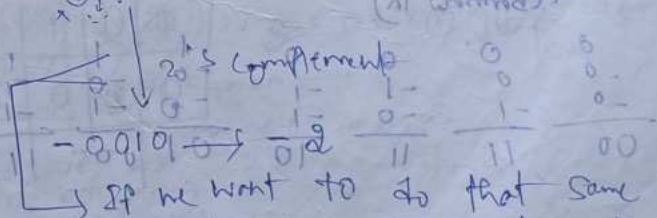
−B (b3 b2 b1 b0)

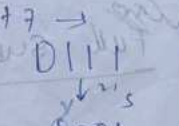












Half subtraction

x (1 bit)
−y (1 bit)

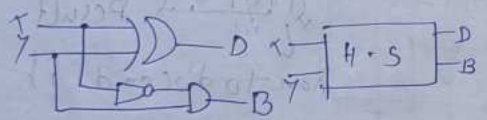
Borrow bit
Difference

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$D = x \oplus y$

$B = \bar{x}y$

K-map

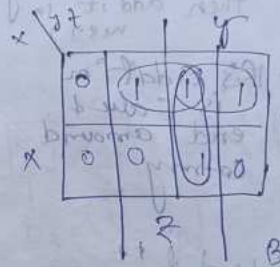


Full subtractor

x
−y
−z (Borrow in)

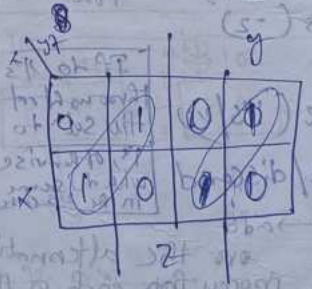
Performing 1's here

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

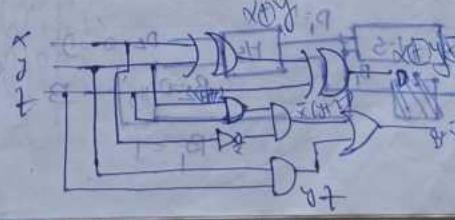


$B = \bar{x}y + \bar{x}z + yz$

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



$D = x \oplus y \oplus z$



Substraction (By 1's complement)

7 − 5

5 − 7

0 0 0 0 0 1 1 1
1 1 1 1 1 0 1 1 (2's)
1 0 0 0 0 1 1 0 0

in 1's subtraction

7 − 5 ⟶ 0111 + 1010 = (1) 0001 ⟵ (If we get a carry then add it in next)

If carry = 1 then, add 0001 + 1 = 0010 (+2) ⟸ this addition is called end around carry

5 − 7 ⟶ 0101 + 1000 = 1101 ⟵ (If we don't get a C then 1's comp.)

1101 ↓ 1's = − 0010 ⟶ (−2)

0111 ↓ 1's 1000

Carry (+) and 0 ⟶ −ve (1's / 2's)

1 ⟶ add / discard

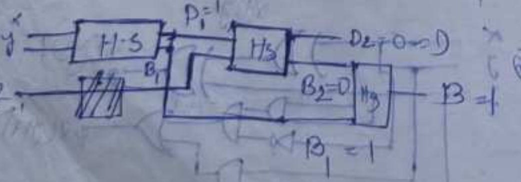
If 0, 1's through out the sol to 1's, otherwise with the same in the 2's case

2's ⟶ discard, 1's ⟶ add

FS in the form of HS — or the alternative process for each of them

x − y = B D

B D − 7 = B D



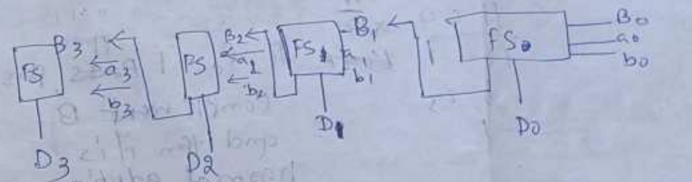
① H/W How to solve the borrow function

⇓

0 − 1: (B) 1 (D) 1

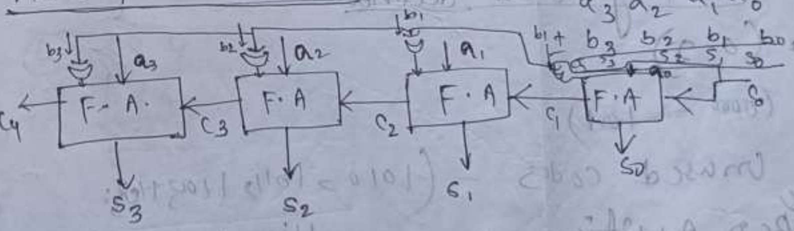
① 1 0 1 0 1 1 0 0
− 1 1 0 0 1 0 1 0
0 0 1 1 1 1 1 0

② 0 1 1 1
− 0 1 0 1
0 1 0



CW

Adder / substractor ⟶ (A ± B) 6/9/22



C₀	Bᵢ	XOR
0	0	0
0	1	1
1	0	1
1	1	0

complement bitwise (B)
↓
1's complement + 1

$C_0 = 0$ addition (Pass B)

$C_0 = 1$ 1's subtraction

↓

Bitwise Complement (B)

↓

1's Complement + 1

2 → 0 0 1 0 → A

+3 → 0 0 1 1 → B

0 1 0 1

If $C_0 = 0$ Pass B and it will be simple addition

2 → 0 0 1 0 → A

−3 → 1 1 0 0 → B

1 1 1 1

↓ 1's

−1

If $C_0 = 1$ Pass complement B and then it's normal addition

Binary coded decimal

(0 - 9)

(0000 = 1001)

6 unused codes → (1010, 1011, 1100, 1101, 1110, 1111)

BCD Addition

(A)

4 → 0100

+5 → 0101

9 → 1001 ⇒ (9)

↳ No error code & proper code

(B)

4 → 0100

+8 → 1000

12 → 1100 (Binary 12)

(0001 0010)

0110

0010

Error code, unused code

[have to add 6]

this is unused code so we have to convert the unused code so we have to convert it

In All the cases we have to add 6 (0110)]

(C)

8 → 1000

+9 → 1001

17 10001 (in BCD → 11, but we need 17 so add 6)

10111 (23 in BCD)

0110

10001 (17 in BCD)

(C) in Carry (K)

(carry generation (carry is = 1)

(D)

184

576

760

0000 1000 0100

+ 0101 0111 0110

0111 0111 1010

+0110

+0001 0001 +0110

0111 0001 0000

7 1 6 0

if (code > 9)

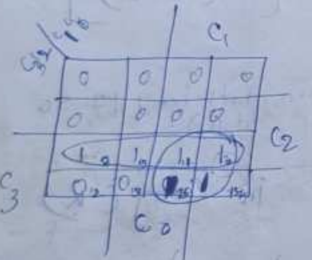
add 6

detecting greater than 10 — detect greater than 12

BCD ⟶ $C_3C_1 + C_3C_2$ + K(carry)

(10, 11, 14, 15) (12, 13, 14, 15)

greater than 9 — add K if carry is generated



⟶ (K-map for unused codes)

⟶ K-map of greater than 9

(detecting the unused codes)

$(C_3C_1 + C_3C_2) + K$

↓ unused code — ↓ carry ①

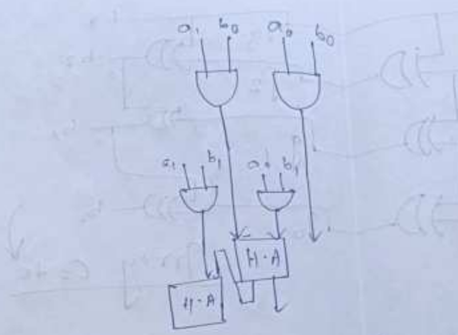
*** partial product by 1-bit

A × B

a_1 a_0 (multiplicant) — a_1 a_0 → (multiplicant)

b_1 b_0

	b_1	b_0	
C	a_1b_0	a_0b_0	PP_0
	a_1b_1	a_0b_1	PP_1
a_1b_1	a_0b_1	x	
	S	a_0b_0	



PP_0 ⟵ 4-bit partial product (more than 2 bit)

SL (PP1) — PS_0

SL (PP2) — PS_1

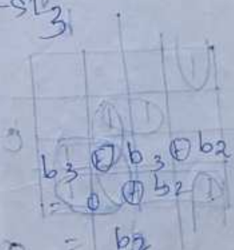
SL (PP3) — PS_2

	a_1b_0	a_0b_0	
a_1b_1	a_0b_1	⊗	⟵ SL 1
⊗	⊗	⊗	⟵ SL 2
⊗	⊗	⊗	⟵ SL 3

** Gray Code ⟷ Binary

Binary	Gray Code
0) 0000	0000
1) 0001	0001
2) 0010	0011
3) 0011	0010
4) 0100	0110
5) 0101	0111
6) 0110	0101
7) 0111	0100

$b_3\,b_2\,b_1\,b_0$ — $g_3\,g_2\,g_1\,g_0$

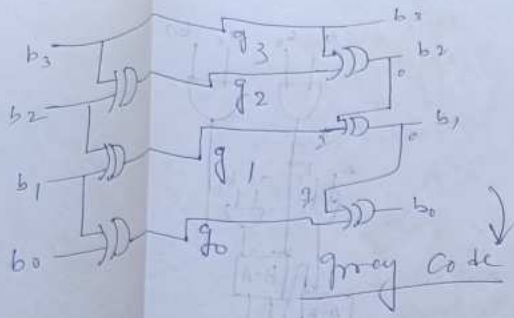


$b_3 \oplus b_3 \oplus b_2$

$b_3 \oplus 0 \oplus b_2$

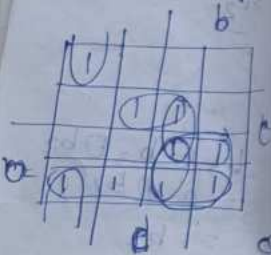
$= b_2$

Gray	Binary
$g_3 = b_3$	$b_3 = g_3$
$g_2 = b_3 \oplus b_2$	$b_2 = b_3 \oplus g_2$
$g_1 = b_2 \oplus b_1$	$b_1 = b_2 \oplus g_1$
$g_0 = b_1 \oplus b_0$	$b_0 = b_1 \oplus g_0$



Quine mcCluskey om (Programming method to solve K-map sequentially)

$F(a,b,c,d) = \sum(0,5,7,8,9,10,11,14,15)$ using Sm method.



① Gate testing
② Universal gate
③ FM
④ F.S

Sm

1110
1011
0111
0000
1000
0101
1001
1010

Group				
0	0	0000	(0,8)	-000
1	8	1000	(8,9)	100-
			(8,10)	10-0
2	5	0101	(5,7)	01-1
	9	1001	(9,11)	10-1
	10	1010	(10,14)	1-10
			(10,11)	101-
3	7	0111	(7,15)	-111
	11	1011	(11,15)	-111
	14	1110	(14,15)	111-
4	15	1111		

		$b\bar{c}\bar{d}$		
0	(0,8)	-000	(8,9,10,11)	10--($a\bar{b}$)
			(8,10,9,11)	10--
1	(8,9)	100-	(10,11,14,15)	1--1--(ac)
	(8,10)	10-0	(10,14,11,15)	1--1--
2	(5,7)	01-1	($a\bar{b}d$)	
	(9,11)	10-1		
	(10,11)	101-		
	(10,14)	1-10		
3	(7,15)	-111	(bcd)	
	(11,15)	1-11		
	(14,15)	111-		

PI	0	5	7	8	9	10	11	14	15
*ac							①		①
*$a\bar{b}$				①		①			
bcd			①				①		
*$\bar{a}bd$		①			①				
*$\bar{b}\bar{c}\bar{d}$	①								

EPI

(As No product term can cover 0, so it is essential prime implicant)

the function is - $ac + a\bar{b} + \bar{a}bd + \bar{b}\bar{c}\bar{d}$ ②

(EPI) ✓

13/9/22

magnitude comparator

x_i y_i	$x_i = y_i$	$x_i > y_i$	$x_i < y_i$
0 0	1	0	0
0 1	0	0	1
1 0	0	1	0
1 1	1	0	0

[At a time only one output (E/G/L) is active, not an one more than one can be activated at a time]

[$E_i = x_i \odot y_i$ (E = equality function) [XNOR gate]
$G_i = x_i \bar{y}_i$ (G = greater than)
$L_i = \bar{x}_i y_i$ (L = less than)

1 bit comparators

2 bit comparators

$x_1 x_0$ $y_1 y_0$ F G L

$F = (x_1 = y_1)$ and $(x_0 = y_0) = \boxed{F_1 \cdot F_0}$

$G = (x_1 > y_1)$ or $(x_1 = y_1) \cdot (x_0 > y_0)$

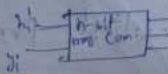
$= \boxed{x_1 \bar{y}_1 + E_1 \cdot (x_0 \bar{y}_0)} = \boxed{G_1 + E_1 G_0}$

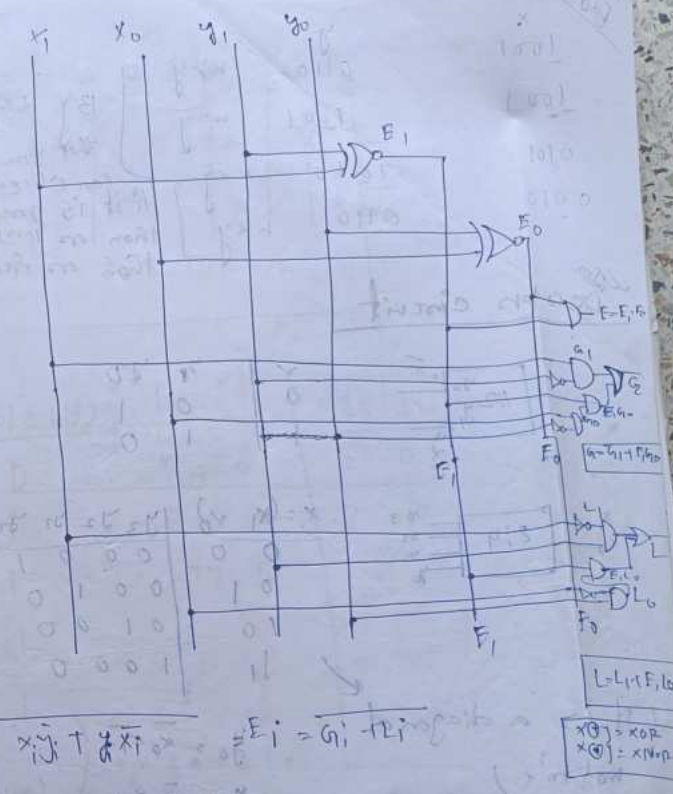
$\boxed{E_i = x_i \odot y_i}$

$L = (x_1 < y_1)$ or $(x_1 = y_1) \cdot (x_0 < y_0)$

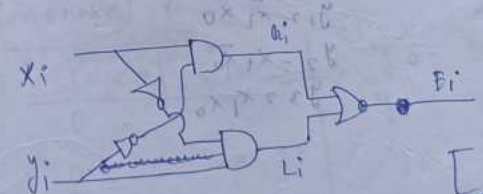
$= \boxed{\bar{x}_1 y_1 + E_1 \cdot (\bar{x}_0 y_0)} = \boxed{L_1 + E_1 L_0}$

Block diagram





$x_i \bar{y}_i + \bar{x}_i y_i = \overline{E_i} = \overline{G_i + L_i}$



$E = E_3 \cdot E_2 \cdot E_1 \cdot E_0$

$G = G_3 + E_3 \cdot G_2 + E_3 \cdot E_2 \cdot G_1 + E_3 \cdot E_2 \cdot E_1 \cdot G_0$

$L = L_3 + E_3 \cdot L_2 + E_3 \cdot E_2 \cdot L_1 + E_3 \cdot E_2 \cdot E_1 \cdot L_0$

[If all bits are equal or not you can decide from the last bit]

tion is - $ac + a\bar{b} + \bar{a}bd + \bar{b}\bar{c}\bar{d}$ ②

(EPI) ✓

tude comparator

$x_i = y_i$	$x_i > y_i$	$x_i < y_i$
1	0	0
0	0	1
0	1	0
1	0	0

⊙ y_i (E = equality fun

y_i (G = greater than)

(L = less than)

comparators

perators

$y_1 y_0$ E G

) and $(x_0 = y_0) =$

y_1) or $(x_1 = y_1)$

$+ E_1 \cdot (x_0 \bar{y}_0)$ = G

⊙ y_i

$x_1 < y_1$) or $(x_1 = y_1)$

$\bar{x}_1 y_1 + E_1 \cdot (\bar{x}_0 y_0)$

gram

n-bit mag. Comp.

13

Pg (62)

• Circuit diagram of 2-bit :-

x_1 x_0 y_1 y_0

E_1

E_0

$E = E_1 \cdot E_0$

G_1

G

L

E_0

$E = E_1 \cdot E_0$

G_1

G

$E_1 G_0$

F_0

$G = G_1 + E_1 G_0$

L

$E_1 L_0$

L_0

F_0

$L = L_1 + E_1 L_0$

$x(\odot)$ = XOR

$x(\bar{\odot})$ = XNOR

bits one

not you can

decide

from

the last

bit)

$E_1 \cdot G_0$

$E_1 \cdot L_0$

Comparing Example

x	y	
1001	0110	x>y
1001	1001	x=y
0101	1010	x<y
0010	0110	x<y

By doing that you can check if it is greater than or less than or equal

Decoder circuit

1:2 decoder

x	y_1	y_0
0	0	1
1	1	0

2:4 decoder

$x = (x_1\ x_0)$	y_3	y_2	y_1	y_0
0 0	0	0	0	1
0 1	0	0	1	0
1 0	0	1	0	0
1 1	1	0	0	0

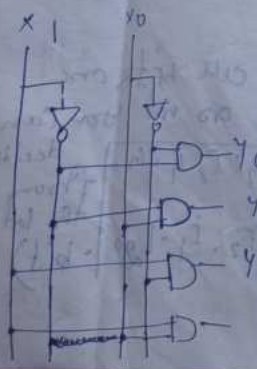
(It is a diagonal matrix)

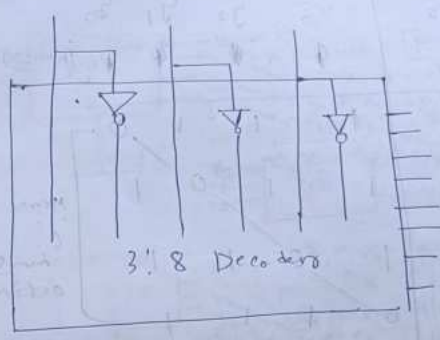
$y_0 = \bar{x}_0 \bar{x}_1$

$y_1 = \bar{x}_1 x_0$

$y_2 = x_1 \bar{x}_0$

$y_3 = x_1 x_0$





(Enable bit)

E	x_1	x_0	y_3	y_2	y_1	y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

(High enable)

$\bar{E}$ (for enable)

E	x_1	x_0	y_3	y_2	y_1	y_0
0	x	x	0	0	0	0

② $y_0 = \overline{E \bar{x}_1 \bar{x}_0}$

$y_1 = \overline{E \bar{x}_1 x_0}$

$y_2 = \overline{E x_1 \bar{x}_0}$

$y_3 = \overline{E x_1 x_0}$

} Enable decoders

① $y_0 = E \bar{x}_1 \bar{x}_0$

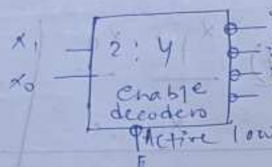
$y_1 = E \bar{x}_1 x_0$

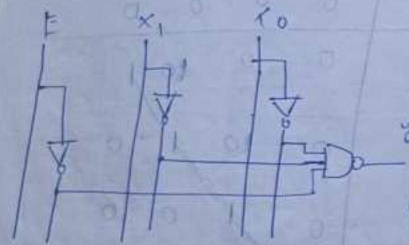
$y_2 = E x_1 \bar{x}_0$

$y_3 = E x_1 x_0$

E	x_1	x_0	y_3	y_2	y_1	y_0
1	x	x	ϕ	ϕ	ϕ	ϕ (inactive)
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Means '0' means active





$y_0 = \overline{E \bar{x}_1 \bar{x}_0} = E + x_1 + x_0 = 0$

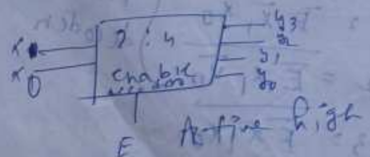
$y_1 = \overline{E \bar{x}_1 x_0} = E + x_1 + \bar{x}_0 = 0$

$y_2 = \overline{E x_1 \bar{x}_0}$

$y_3 = \overline{E x_1 x_0} = \overline{1 \cdot 1 \cdot 1}$ → (0 done low)

my need 3 input AND gates

For active high enable



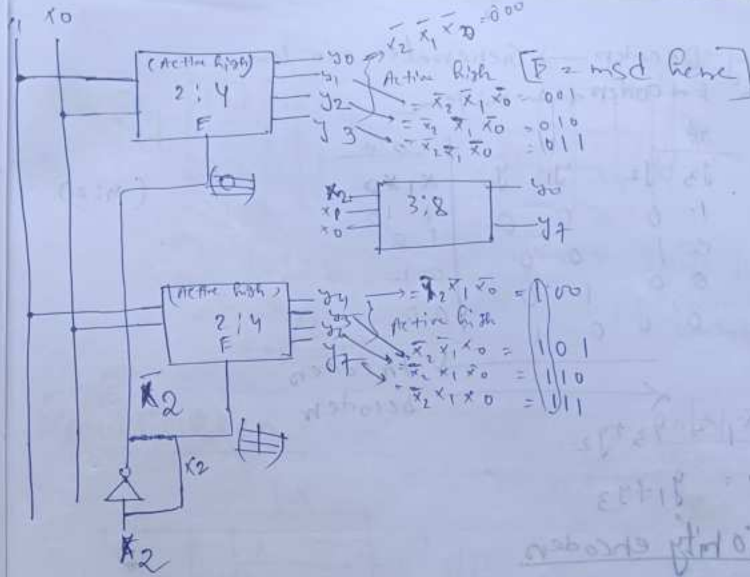
$Y_0 = E \bar{x}_1 \bar{x}_0$

$y_1 = E \bar{x}_1 x_0$

$y_2 = E x_1 \bar{x}_0$

$y_3 = E x_1 x_0 = 1$

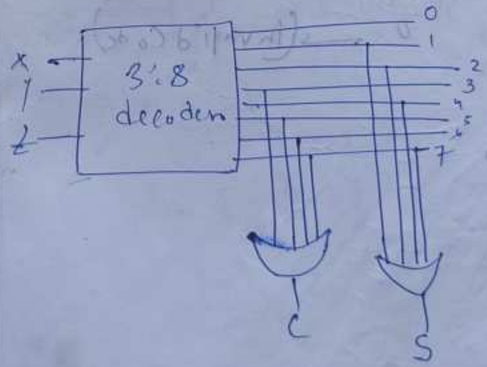
need – us
{ 3 input NAND gate }



$S = \Sigma_m(1, 2)$ [Half adder V S]

Full adder $S = \Sigma_m (1, 2, 4, 7)$

$C = \Sigma_m (3, 5, 6, 7)$



C | S o/p
0
0
0
1
0
1
1
1

Binary → Decoder → Generate min term
Encoder ← min term

I/P				O/P	
y_3	y_2	y_1	y_0	x_1	x_0
1	0	0	0	1	1
				1	0
0	1	0	0	0	1
0	0	1	0	0	0
0	0	0	1		

(4:2)

→ Encoder
← Decoder

$x_1 = y_3 + y_2$

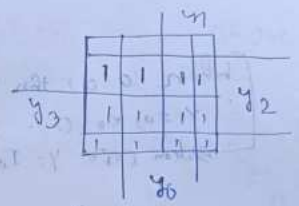
$x_0 = y_1 + y_3$

Priority encoder

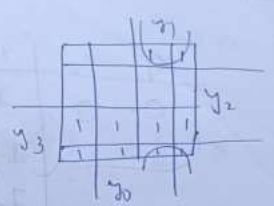
O/P				ap		V
y_3	y_2	y_1	y_0	x_1	x_0	
1	x	x	x	1	1	1
0	1	x	x	1	0	1
0	0	1	x	0	1	1
0	0	0	1	0	0	1
0	0	0	0	x	x	0 → (invalid Code)

(4:2)

$V = y_3 + y_2 + y_1 + y_0$
(validity bit)



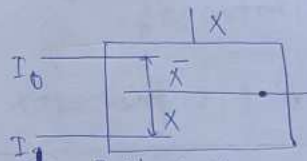
$x_1 = y_2 + y_3$
(Priority encoder)



$x_0 = y_3 + \bar{y}_2 \cdot y_1$
(Priority encoder)

multiplexer

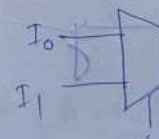
20/9/22



$Y = I_0 \bar{X}$
$Y = I_1 X$

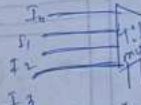
when $X = 0$ I_0 will be connected to Y
" $X = 1$ I_1 " " Y

Symbol -



2ⁿ-line I/P n-bit

1-line O/P



1-line O/P
4-line I/P
2-bit

In 4 : 2
$\Rightarrow \log_2 4 = 2$

[The base of ln = 2 it is normal log]

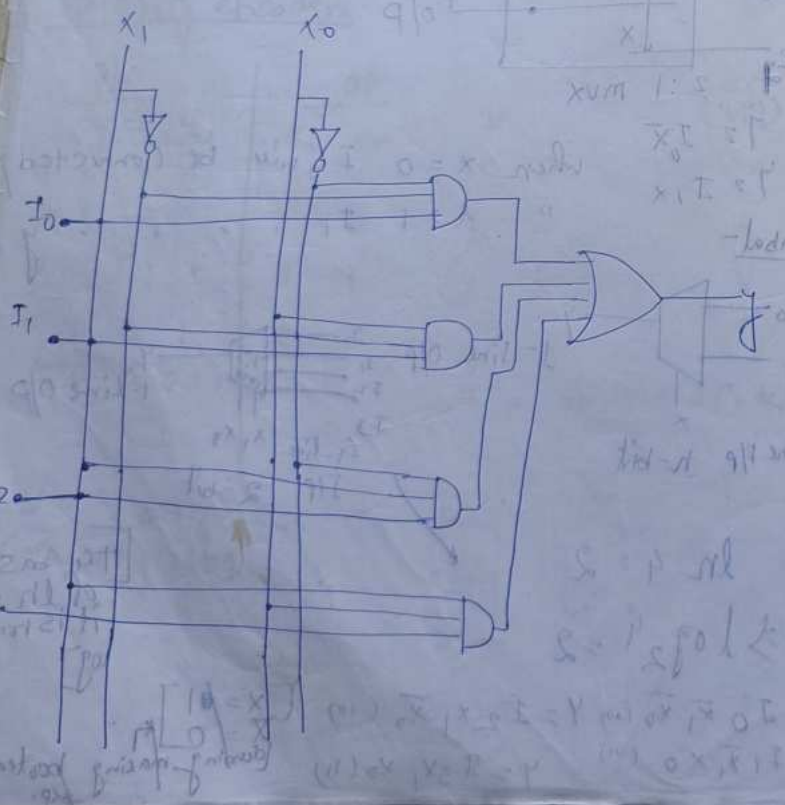
$Y = I_0 \bar{x}_1 \bar{x}_0$ (00) $Y = I_2 x_1 \bar{x}_0$ (10)
$Y = I_1 \bar{x}_1 x_0$ (01) $Y = I_3 x_1 x_0$ (11)

[$x = 1$, $\bar{x} = 0$ during making boolean exp.]

x_1	x_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

When '10' 0, then $y = I_0$
$\therefore x_1 = 0, x_0 = 0$.
Boolean exp = $y = I_0 \bar{x}_1 \bar{x}_0$

Implementation

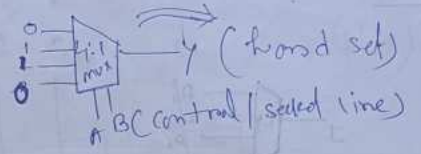


either show these (previous) 4 terms or.
that is, one eqn -

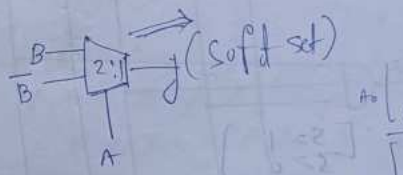
$$y = I_0 \bar{x}_1 \bar{x}_0 + I_1 \bar{x}_1 x_0 + I_2 x_1 \bar{x}_0 + I_3 x_1 x_0$$

How to implement function using mux

Implement a XOR gate using mux -



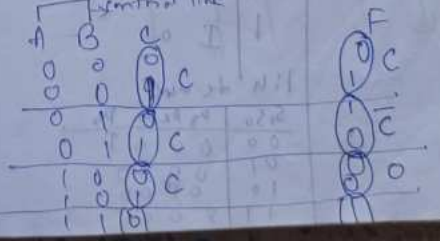
XOR using 2:1 mux

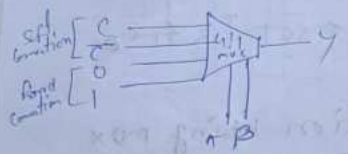


A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

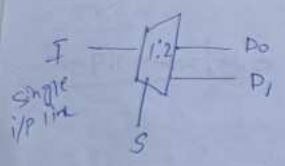
(1) $F(A, B, C) = \sum_m (2, 6, 7)$ – Implement this F using mux.
3 input bit & min terms
So, we have to use 4:1 mux

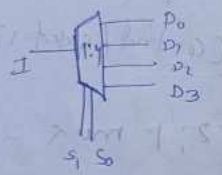
MSD = control line

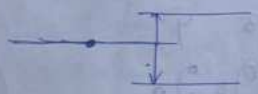




De - multiplexer







$D_0 = I\,\bar{S}$ $\quad [S=1,\ S=0]$

$D_1 = I\,S$ $\quad (1:2)$

$D_0 = I\,\bar{S_1}\,\bar{S_0}$

$D_1 = I\,\bar{S_1}\,S_0$ $\quad (1:4)$

$D_2 = I\,S_1\,\bar{S_0}$

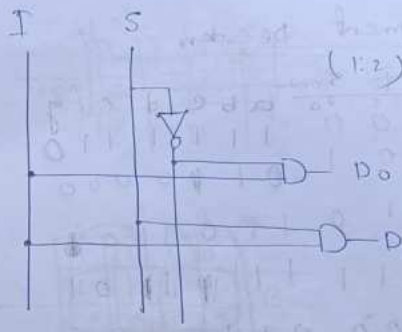
$D_3 = I\,S_1\,S_0$

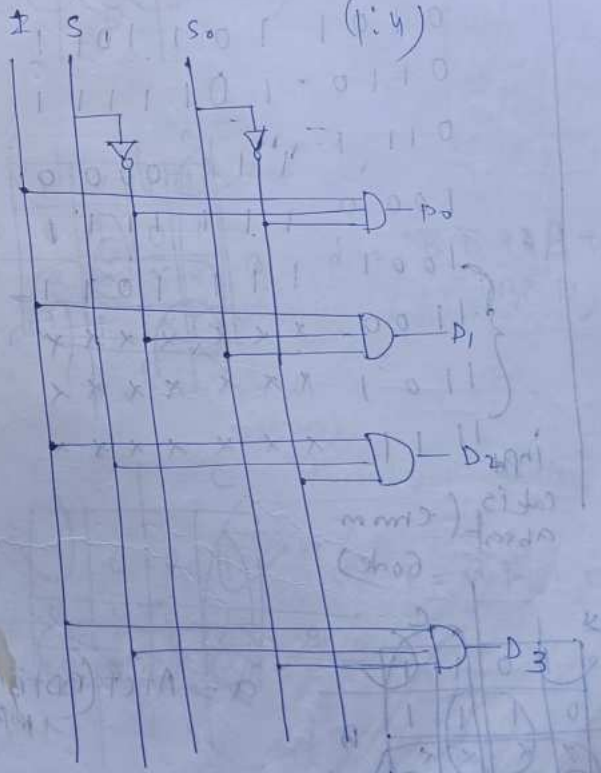
1:2 de-mux

S	D_1	D_0
0	0	I
1	I	0

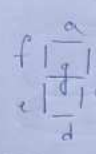
1:4 de mux

$S_1 S_0$	D_3	D_2	D_1	D_0
0 0	0	0	0	I
0 1	0	0	I	0
1 0	0	I	0	0
1 1	I	0	0	0





BCD to 7 segment Decoder

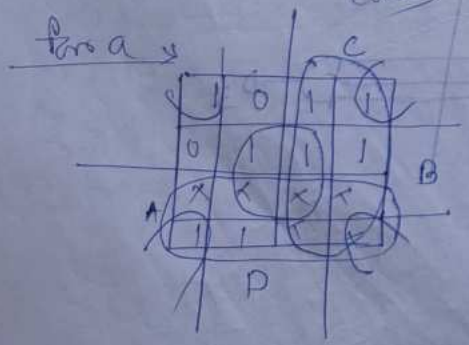


How to show

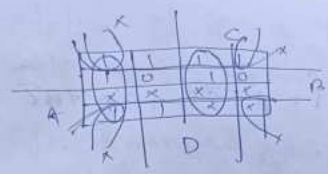
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

Input code is absent (error code)

for a →

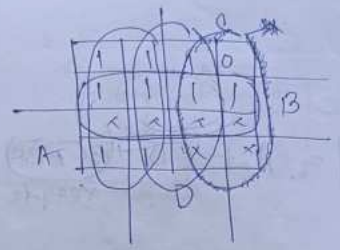


$a = A + C + (BD + B\bar{D})$ → NOR

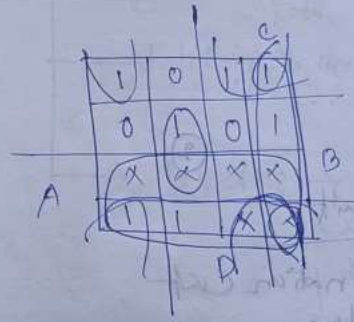


$b = \bar{B} + \bar{C}\bar{D} + CD + \cancel{B\bar{D}}$

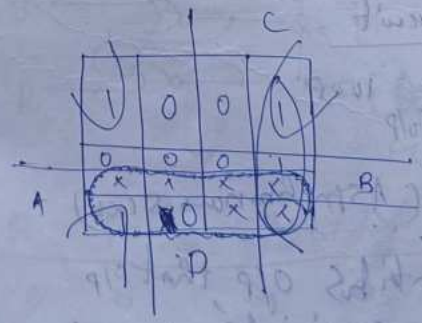
× max 2 gate



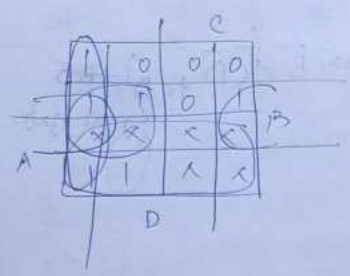
$c = \bar{C} + D + B$



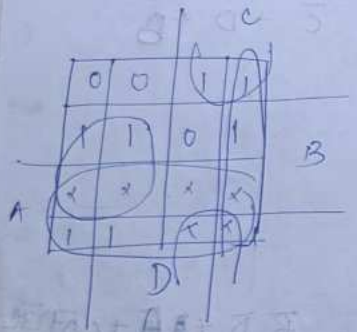
$d = \bar{B}\bar{D} + \cancel{B}\cancel{A} + C\bar{B} + \bar{B}\bar{C} + B\bar{C}D$



$e = \bar{B}\bar{D} + C\bar{D} + \cancel{A}$



$f = A + C\bar{D} + B\bar{D} + B\bar{C}$

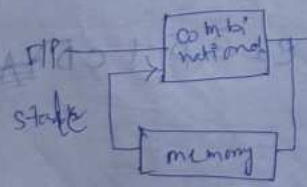


$J = A + \bar{C}\bar{D} + (B\bar{C} + \bar{B}C)$
XOR gate

Combination circuit

Combination CKt
(No loop)

Sequential circuit



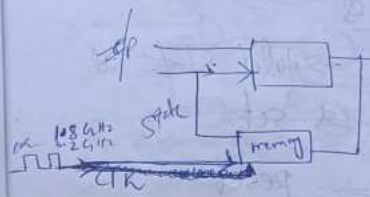
(Asynchronous CKt)

memory contains O/P that O/P is needed by combinational ckt.

O/P = function (I/P, State)

present state = It is a function of previous state and input

synchronous / Asynchronous
Clock no clock



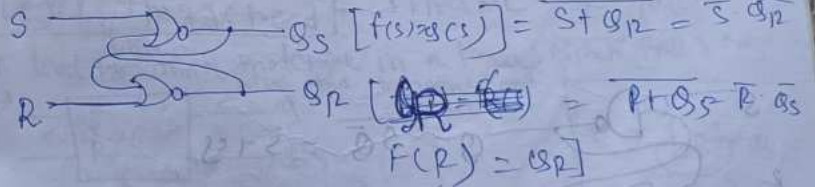
(synchronous sequential CKt)

memory

latch is

latch → Asynchronous sequential CKt and it can store data

flip-flop → synchronous sequential CKt and it can store data



$Q_S = \overline{S + Q_R} = \bar{S}\,Q_R$

$Q_R = \overline{R + Q_S} = \bar{R}\,\bar{Q_S}$

S	R	$Q_S^{t+\delta}$	$Q_R^{t+\delta}$
0	0	Q_R	$\bar{Q_S}$
0	1	$\bar{Q_R}=1$	0
1	0	0	$\bar{Q_S}=1$
1	1	0	0 → not allowed

By rewriting the eqn = $Q[Q_S] = \overline{S+\bar{Q}} = \bar{S}\,Q$

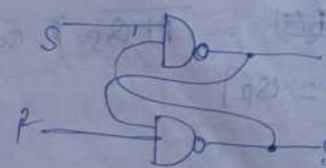
$\bar{Q}[Q_R] = \overline{R+Q} = \bar{R}\,\bar{Q}$

S	R	Q	$\bar{Q}$	
0	0	$\bar{Q} = Q$	$\bar{Q}$	→ Hold ~~state~~ set
0	1	1	0	→ set
1	0	0	1	→ Reset
1	1	0	0	→ not allowed

NOR based SR Latch :-

NOR SR the relation between S and Q is complement so we can write

NOR $\bar{S}\bar{R}$



$Q = \overline{S\bar{Q}} = \bar{S} + Q$

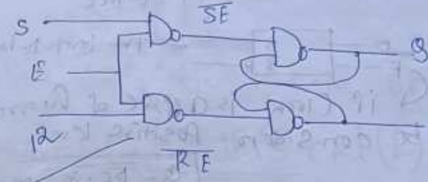
$\bar{Q} = \overline{RQ} = \bar{R} + \bar{Q}$

S	R	Q	$\bar{Q}$	
0	0	1	1	NA
0	1	1	0	set
1	0	0	1	Reset
1	1	Q	$\bar{Q}$	Hold state

NAND based $\bar{S}\bar{R}$ latch

27/9/22

Enabled SR Latch



$Q = \overline{\overline{SE}\cdot\bar{Q}} = SE + Q$

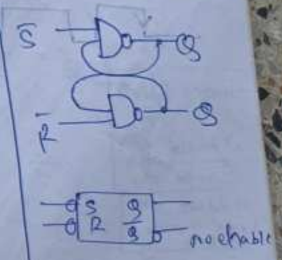
$\bar{Q} = \overline{\overline{RE}\cdot Q} = RE + \bar{Q}$

(function table)

E	S	R	Q	$\bar{Q}$	
0	x	x	Q	$\bar{Q}$	→ hold
1	0	0	Q	$\bar{Q}$	→ hold
1	0	1	0	1	→ Reset
1	1	0	1	0	→ set
1	1	1	1	1	→ N.A.

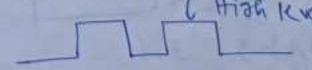
Level triggered → Enable SR latch

definition: A level to which material in a tank either falls or rises and triggers some action.

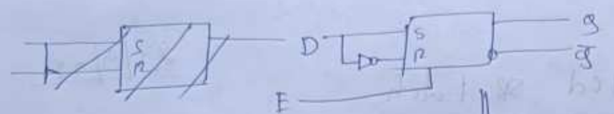


Without flop it's a latch.
when clock is applied it will be level sensitive flip flop.

clock

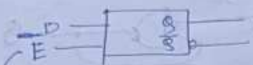


D latch (Don't have N.A condition)
(Drawback of SR latch)



↓ Replace

Single input ← enable latch



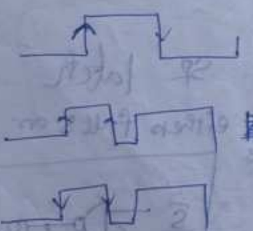
← +ve level D-latch

↳ if clock is applied here it will be considered positive level sensitive latch.

$Q = \overline{\overline{D}E \cdot \overline{Q}} = DE + Q$

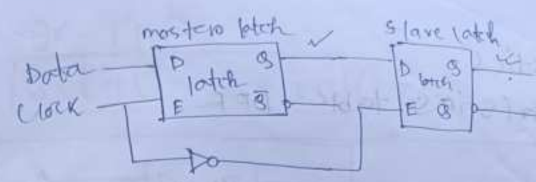
$\overline{Q} = \overline{\overline{D}E \cdot Q} = \overline{D}E + \overline{Q}$

E	D	Q	Q̄ (function table)
0	X	Q	Q̄ → hold
1	0	0	1 → reset
1	1	1	0 → set



↑ → data is accepted this time positive edge sensitive

↓ → negative edge sensitive

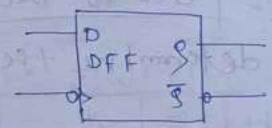


master-slave

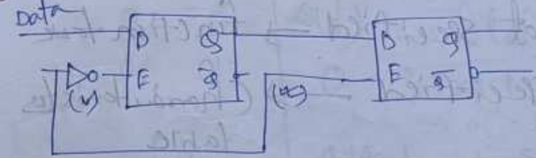
output will get this time

-ve edge sensitive DFF

Symbol of this ckt (1)



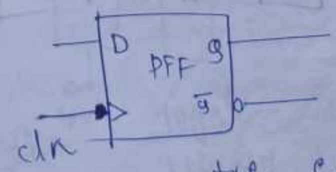
- ve edge sensitive DFF



master-slave

data will be aviable during this time

Symbol of this ckt (1)



+ve edge sensitive

only arrow - +ve
bubble, arrow → -ve
No bubble arrow → level sensitive

Characteristics

Characteristics table DFF

D	Q(t+1)	
0	0	reset
1	1	set

Q(t) → Previous step

Q(t+1) → next step

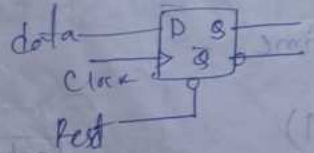
Characteristics eqn - $Q(t+1) = D(t) = D$

only present input determine the next step

time is not specified → function table

time is specified → characteristics table

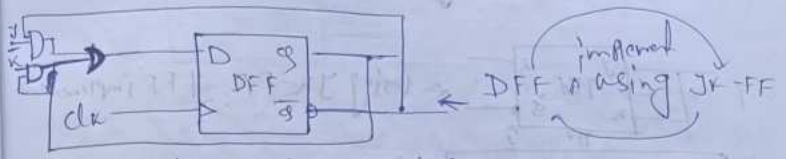
One reset can be added in diagram



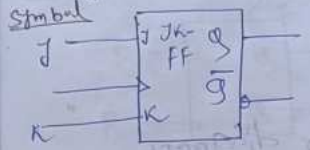
It can be set by data also [ex function table]

JK - FF

$Q(t+1) = J\bar{Q} + \bar{K}Q$ (characteristics eqn)



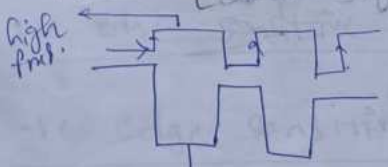
+ve edge sensitive DFF



Char. table of JK - FF

J	K	Q(t+1)	
0	0	Q(t)	→ hold (Previous step)
0	1	0	→ reset
1	0	Q+Q = 1	→ set
1	1	$\bar{Q}(t)$	→ complementary (next state)

The benifit of complementary function.

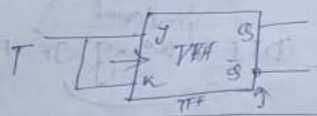


toggle - frequency division

toggle - specifically to change between two states of an electronic device by means of a simple hardware or software control.

(such as button, switch etc)

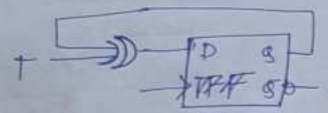
TFF (T-FF has state and complement state only)



= using JK-FF, TFF implemented

$Q(t+1) = T\bar{Q} + \bar{T}Q$ (char. eqn)

$Q(t+1) = T \oplus Q$



Another diagram using XOR

using DFF implemented TFF

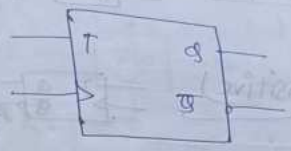
Char. table of TFF

T	Q(t+1)
0	Q(t), hold
1	$\bar{Q}(t)$, complement / toggle mode

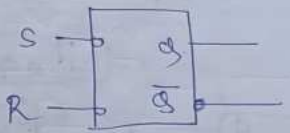
When T = 1 then toggle mode

Here also,

Symbol of TFF



$\bar{S}\bar{R}$ latch

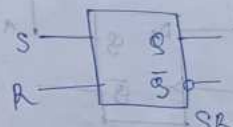


$\bar{S}\bar{R}$ latch

(low logic)

(input low)

SR latch

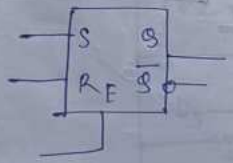


SR latch

(high logic)

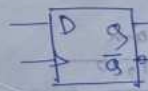
(input high)

Enable SR latch



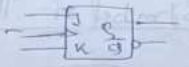
A Clock is present in flip flop. (Synchronous)

+ve edge sensitive DFF

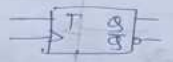


-ve edge sensitive DFF

JK-FF (+ve edge Sensitive)



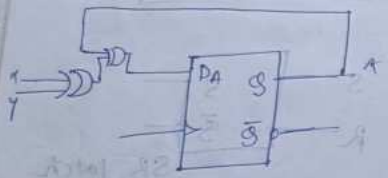
TFF (+ve edge sensitive)



Q How to implement this FSM using TFF

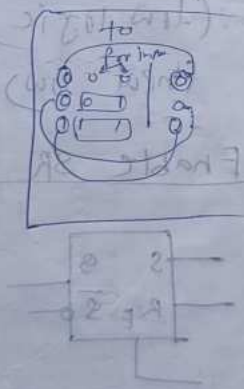
① $A(t+1) = A \oplus x \oplus y = D_A$ [A = State, x, y = Input]

[next state eqn] [D_A - Assigning this in DFF]

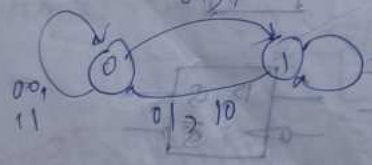


State table:

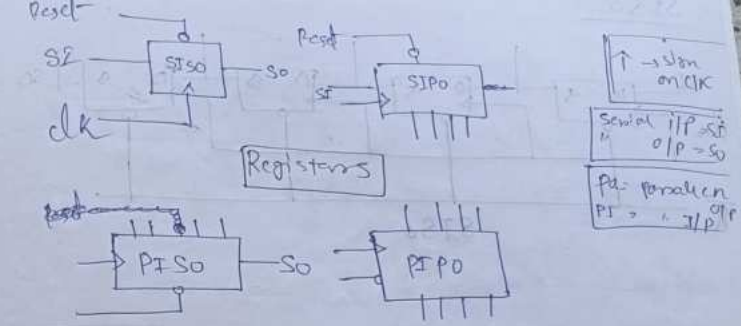
A(t)	x	y	A(t+1)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



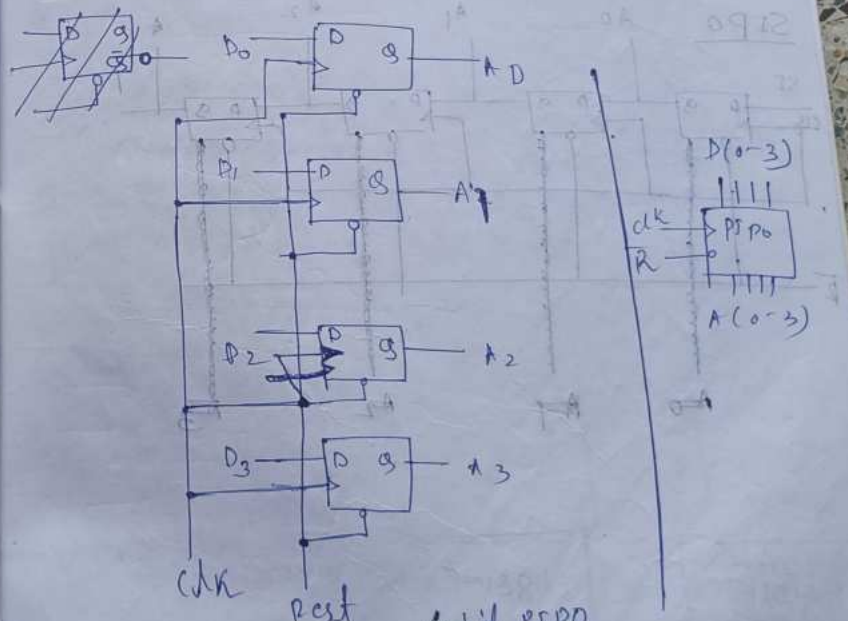
State diagram



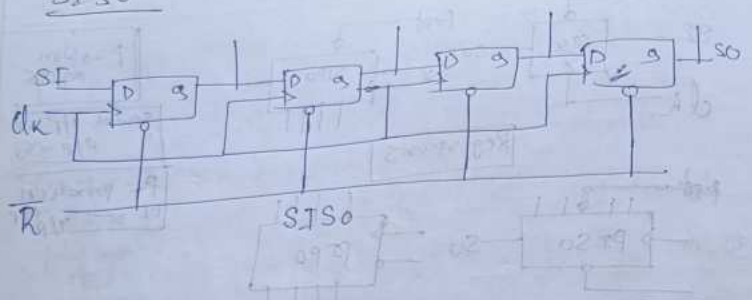
4-bit register



PIPO

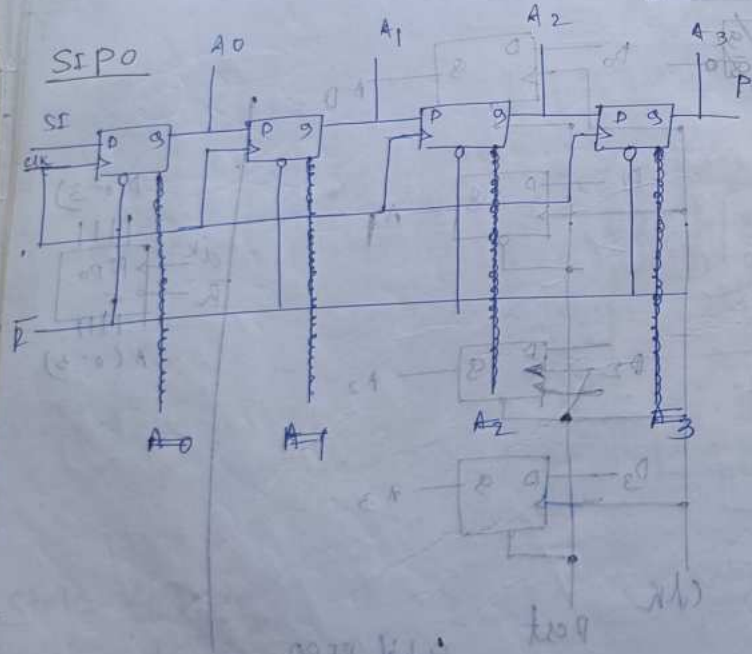


SISO



PISO → loads like universal register

SIPO



Universal register

2 control pin to control the operation

- ✓ 3:8 decoder → IC → 74138
- 4 bit comparator → " → 7485
- 8:1 mux → " → 7415
- ✓ DFF → " → 7474
- ✓ 4:1 → " → 74153
- ✓ JKFF → " → 7476

✓ → in syllabus

Implementation of FSM to table, state diagrams, circuit / machine 11/10/22

State eqn

$A(t+1) = A(t)\,x(t) + B(t)\,x(t)$ [2-bit state $A(t+1)$]

$B(t+1) = A'(t)\,x(t)$ [4 steps]

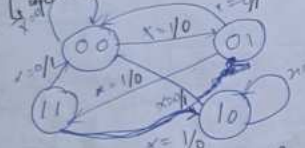
[next step depends upon input] [$x(t)$ = input]

$A(t+1) = Ax + Bx$ } next step

$B(t+1) = A'x$ } $(A+B)x$

State diagram

4 - steps (2-bit state)



$y(t+1) = A(t)x'(t) + B(t)\,x'(t)$ output

$y(t+1) = Ax' + Bx' = (A+B)x'$

[* Black pointed = output]

Depending upon the Present State. What is next step and o/p

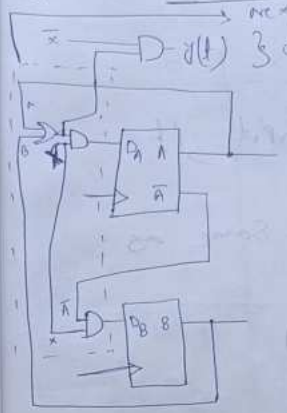
A	B	x	A(t+1)	B(t+1)	y(t+1)
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Present State

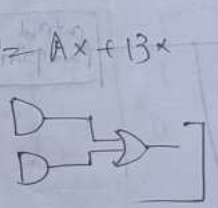
State table

State machine

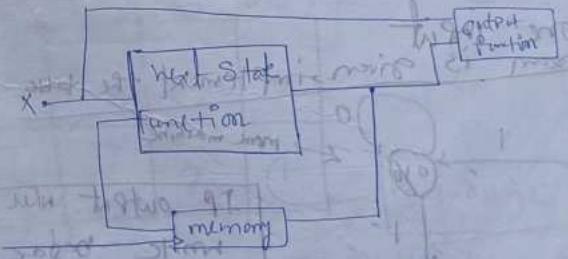
next State function & output function



If $A(t+1) = Ax + Bx$ then



Mealy machine

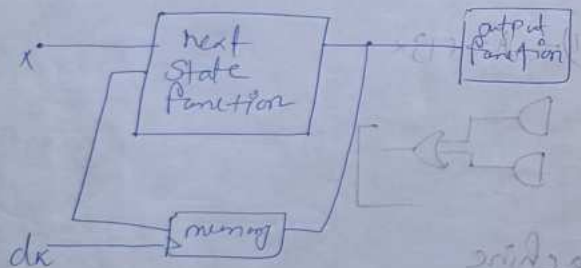


Here, output function is Asynchronous. And output function depends upon the input & state

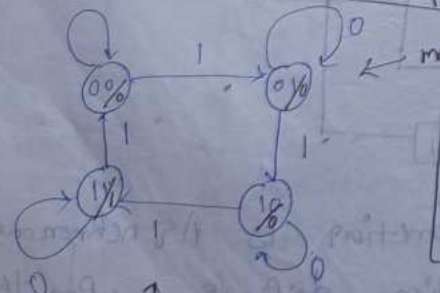
Moore machine

It doesn't depend upon input, it depends upon state only.
the eqns for this are same as previous except,

$Y(t+1) = A\bar{B} + B\bar{B} = (A+B)$



Ex: For one input
State diagram is given, implement the table



If output will write before the input ["0, 0"] then mealy machine

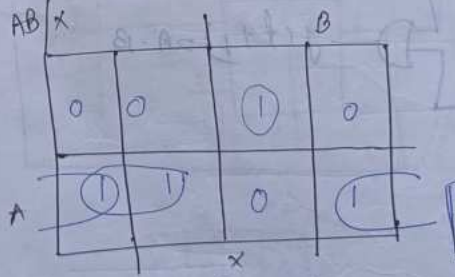
Output don't depend on input. Block indicates the present state.

State table

Present State A	B	x	A(t+1) T_A	B(t+1) T_B	Y(t+1)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	0	1	1

Excitation table of TFF

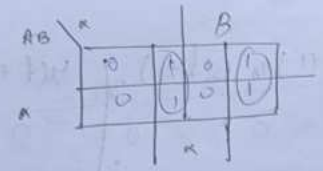
T_A	A(t+1)	T_A
0 → 0 transition	0	0
0	1	1
1	0	1
1	1	0



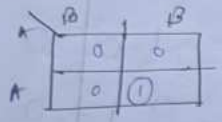
$D_A = A\bar{B} + A\bar{x} + AB\bar{x} = A \odot B\bar{x}$

using DFF the table will be as same as that upperse one.

D_A will be implemented by the circuit diagram (from mealy machine)



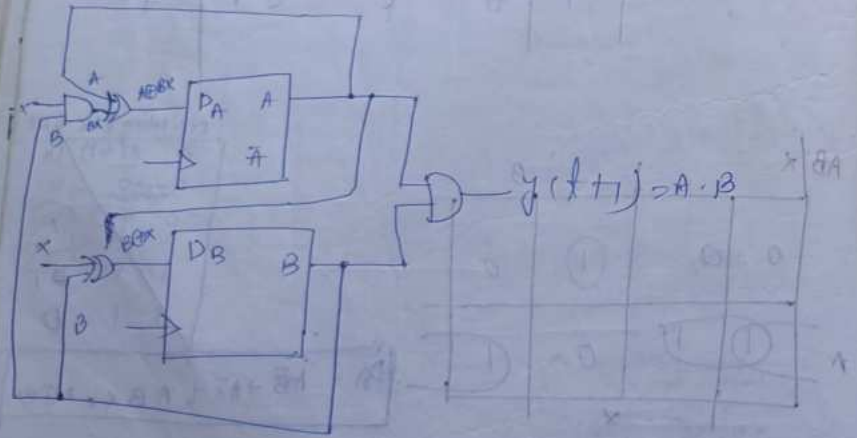
$D_B = \bar{B}x + B\bar{x}$
$= B \oplus x$



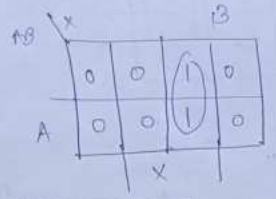
$y(t+1) = A \cdot B$

Here we can see output is not dependent on x. So, it's more mealine.

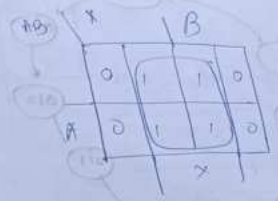
Circuit diagram –



K-maps for (T_A, T_B)



$T_A = Bx$

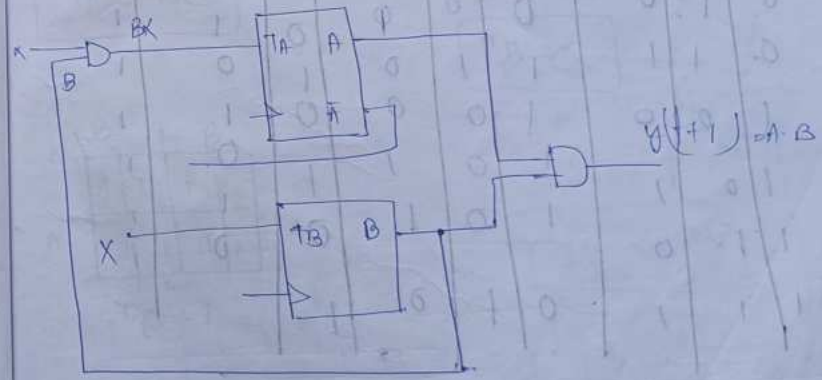


$T_B = x$

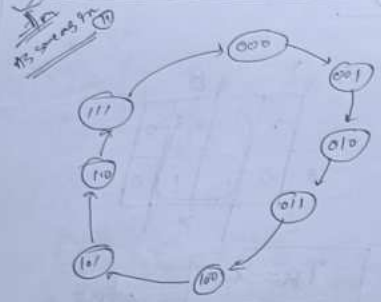
output = mealine

$y(t+1) = A \cdot B$

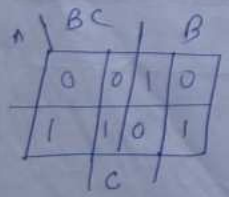
Circuit diagram for (T_A, T_B)

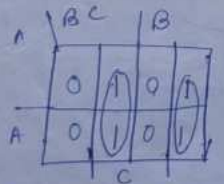


Comparative DFF as well as DFF



A	B	C	A(t+1)	T_A	B(t+1)	T_B	C(t+1)	T_C
0	0	0	0	0	0	0	1	1
0	0	1	0	0	1	1	0	1
0	1	0	0	0	Φ	0	1	1
0	1	1	1	1	0	1	0	1
1	0	0	1	0	0	0	1	1
1	0	1	1	0	1	1	0	1
1	1	0	1	0	0	0	1	1
1	1	1	0	1	1	0	0	1



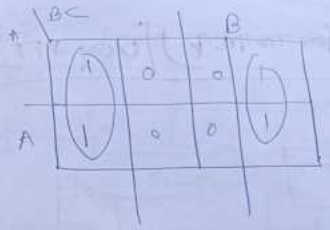


$D_B = \bar{B}C + B\bar{C}$

$D_B = B \oplus C$

$D_A = AB + A\bar{C} + \bar{A}BC$

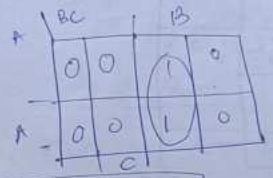
$D_A = A \oplus BC$



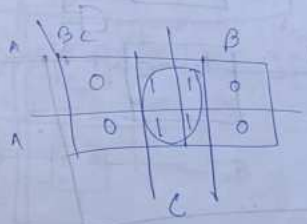
$D_C = B\bar{C} + \bar{B}C$

$D_C = \bar{C}$

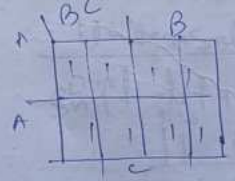
for (T_A, T_B, T_C)



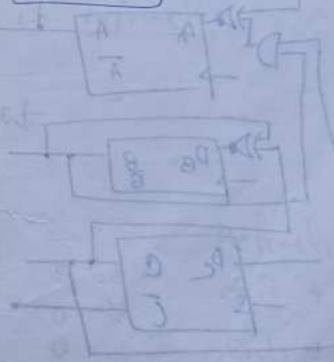
$T_A = BC$



$T_B = C$

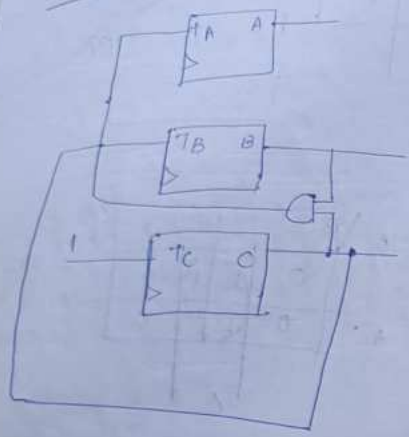


$T_C = 1$

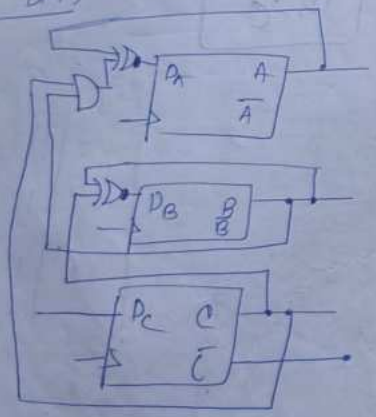


ckt diagram (for TA TB TC) [using DFF]

TFF

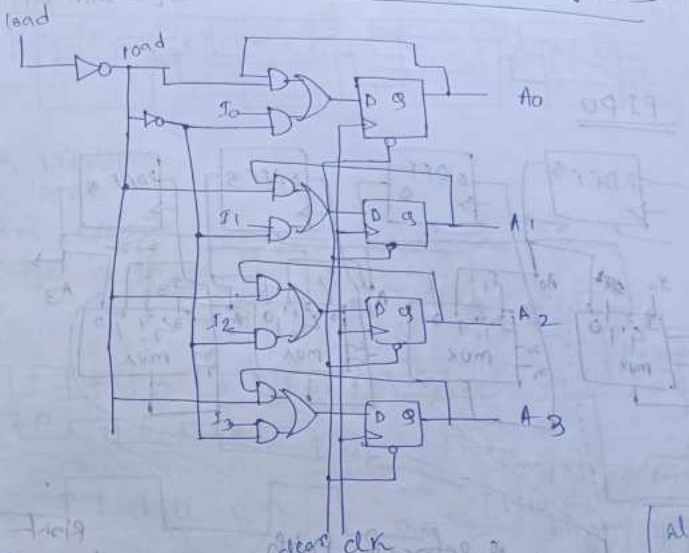


DFF



18/10/22

Parallel load and refresh (register)

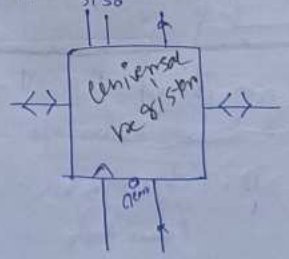


A1=Ī

The opposite of load is refresh

Universal register

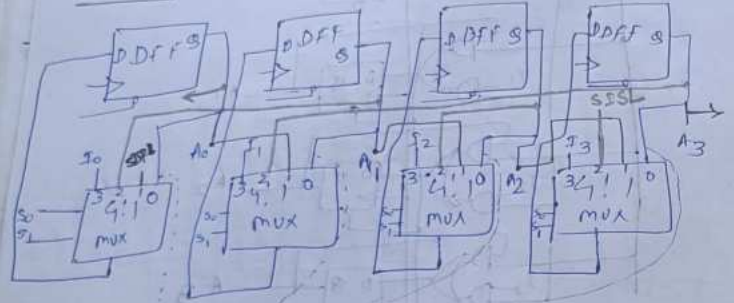
shift left + shift right



S_1	S_0	
0	0	Refresh
0	1	shift + Right
1	0	shift + left
1	1	PIPO

S1 = S
S0 = Shift Right
S1 = Shift Left

PIPO

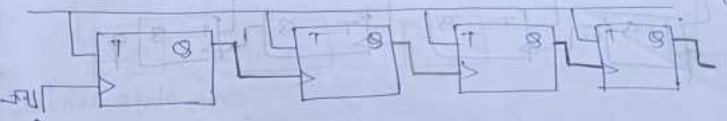


Refresh Path

Right shift path

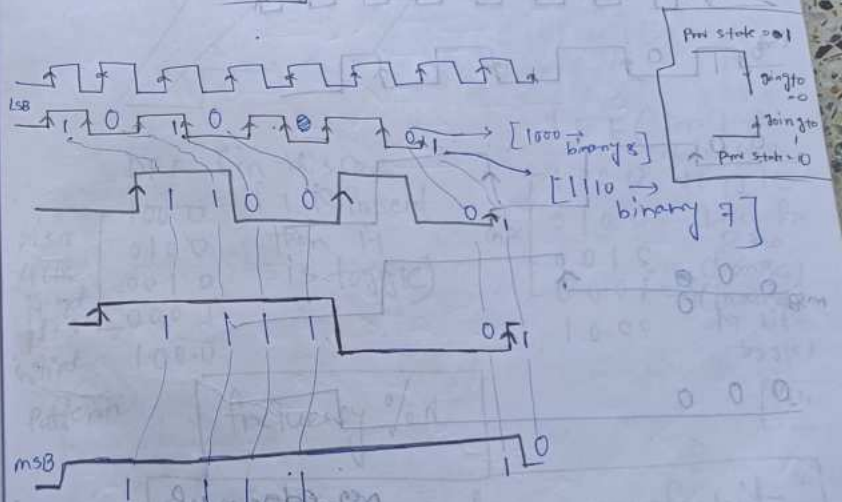
Left shift path

Binary Ripple counter

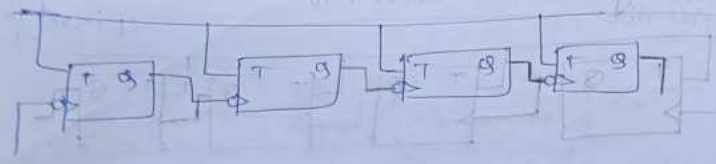


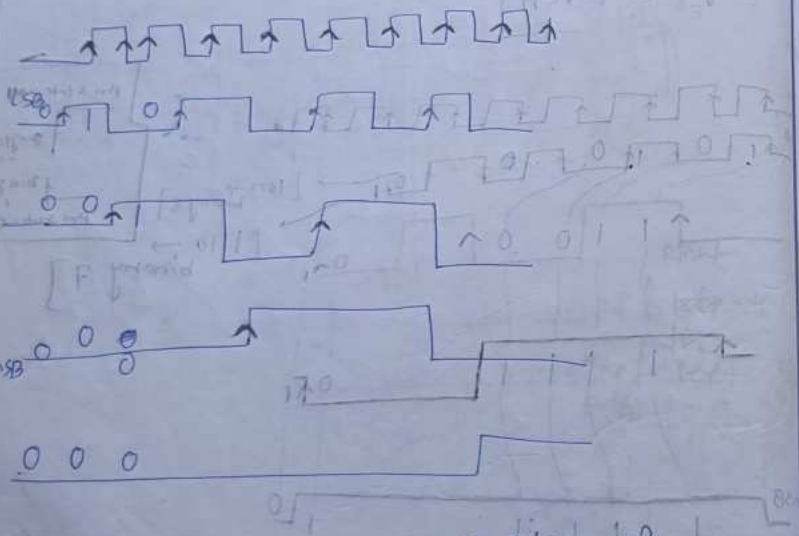
7 is set to -1

This flipflop is toggle when positive edge is taken



If there is a clear/preset

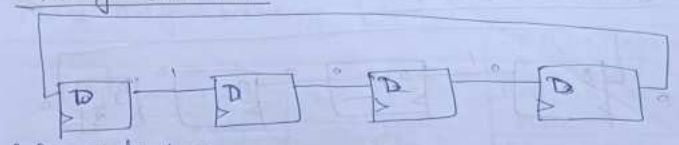




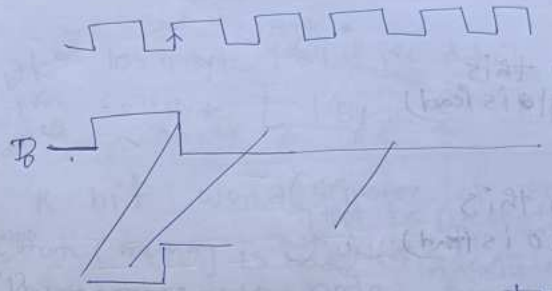
neg edge = up counter

pos edge = down counter

Ring counter



Initial state - 1000



DFF (in this case if 1 is present then it is toggle)

After 4 clk it get it's initial pattern

1000
0100
0010
0001
1000

TFF (For 1 the data will toggle and for 0 no change)

4 case

1000
0100
0010
0001 → (inverted = 1st bit toggle)
1000

frequency % N

Frequency = Per column no. of bit

Jon shon Counter

DFF

1000 (for this
1100 ← 0,1,0 is fead)
1110
shifted 1111
0111 ← (for this
0011   1,0 is fead)
0001
0000

(3)

Read only memory [Rom]

10/10/22

EEPROM = Electrically Erasable programming Rom

EPROM also exist.

1 K B → Byte

n bit is 1K address ← 1024 Byte [Bit of the data]

$= 2^{10}$ Byte [1 Byte = 8 bit]

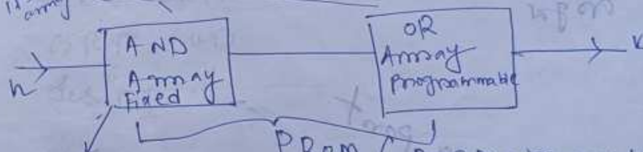
K bit word (sequence of 1 and 0 = word) that represents a

Meaning of Rom (memory is read data in a sequence)

Programmeable node

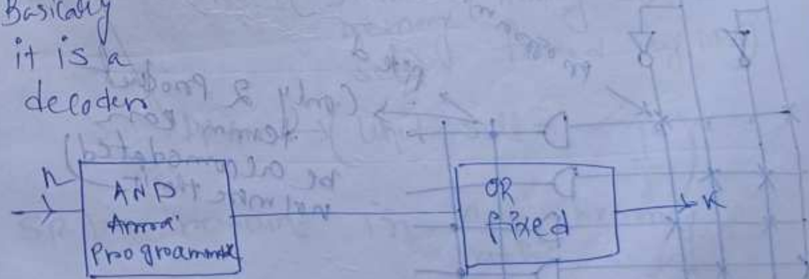
Basically its sum term ← ↓ It has 2 Ports

Meaning of [Ram: you can read memory in any sequence]



PROM (Programmable Rom)

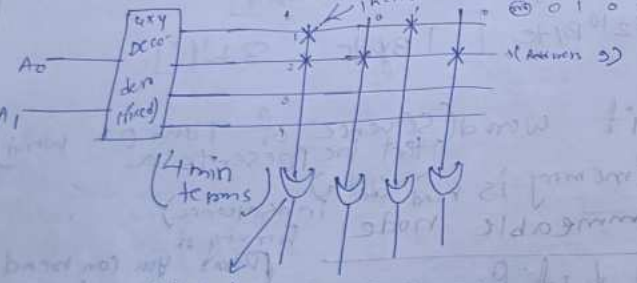
Basically it is a decoder



PAL (Programmable array logic)

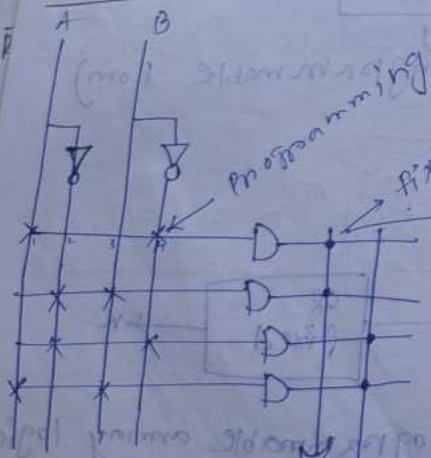
PLA (Programmable logic array)

PROM



(Input prog)

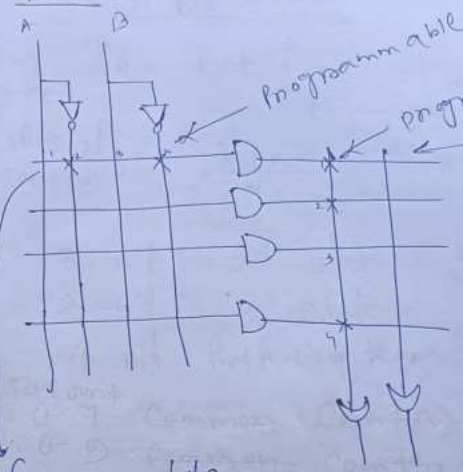
PAL



Sum of 2 Product term $AB\bar{T}AB$

Sum of 2 Product term

PLA



(accomodate Product term as per our desire)

RAM → SRAM (Static → [memory still is memory cell] formed by DFF)

DRAM → (Unit cell — ⊣⊢)

SRAM remain in ~~cash~~ memory cache

1/11/22

N bit

Ring counter $f_R = \frac{f_{clk}}{N}$ (N) states

Johnson counter $f_J = \frac{f_{clk}}{2N}$ (2N) states

3-bit = 2^3 state = half counter set up

2-bit = 2^2 state = half counter set up

4-bit half Counter = $2^4 = 16$ State

To count
:. 0-7 Common Counter = 3-bit
:. 0-9 Common Counter = 4-bit

We can Present symbol using 8 bit = 2^8 bit

32 bit = 16 × 4 bit

0-5 bit = 3-bit

Design a BCD Counter using DFF to represent 0-9 symbol

In 4 using 4-bit.

1 MB = 2^{20} × 8 bit
2^{10} = K
2^{20} = M
2^{30} = G

Im
BCD to counters using DFF (0-9 Counter)

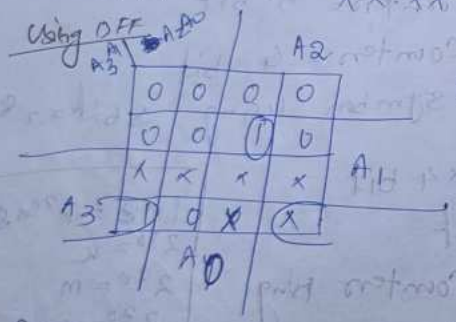
Prev State (A) $A_3 A_2 A_1 A_0$	Next State $D_3 D_2 D_1 D_0$ $A_3 A_2 A_1 A_0$	A(t+1)
0000	0001	
0001	0010	
0010	0011	
0011	0100	
0100	0101	
0101	0110	
0110	0111	
0111	1000	
1000	1001	
1001	0000	

Comp =
0 → to 1
0 → to 0
0 → to 0
1 → to 0

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

0 1 1 1
1 0 0 0
1 0 0 1
0 0 0 0
x x x x undefined state
x x x x Don't Care State
x x x x
x x x x
x x x x

Using DFF

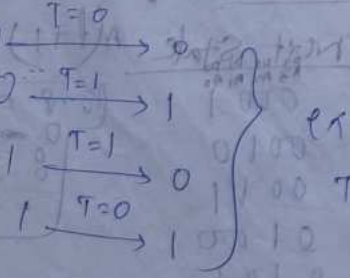


$D_3 = A_3\bar{A}_2\bar{A}_1\bar{A}_0 + A_2A_1A_0$

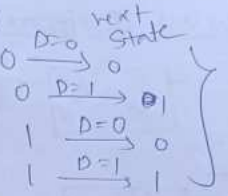
D=0 Q=0
D=1 Q=1

C, DFF is transparent

How to make 0 from 0 is called excitation table



excitation table of TFF

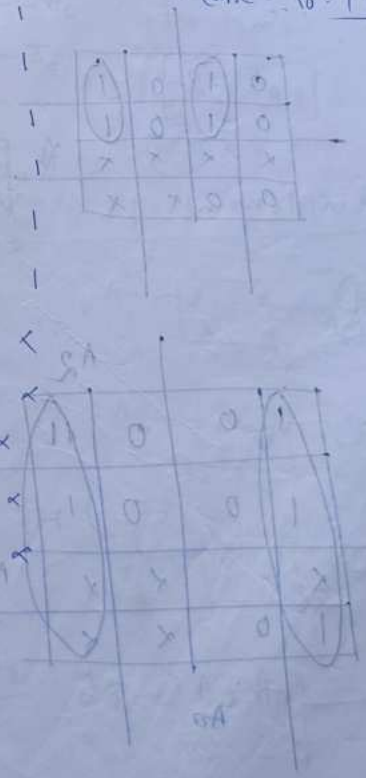


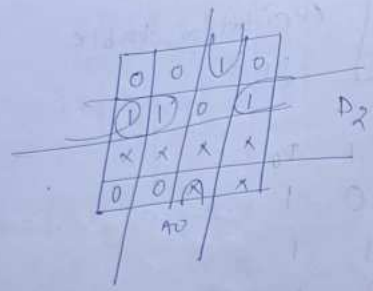
excitation table of DFF

In case of TFF to be fixed to store the same table

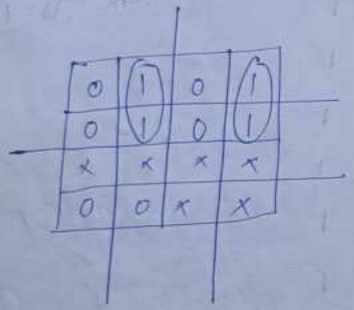
T_3	T_2	T_1	T_0
0	0	0	1
0	0	1	1
0	0	0	1
0	0	1	1
0	0	0	1
0	0	1	1
0	0	0	1
1	0	1	1
0	0	0	1
1	0	0	1
x	x	x	x
x	x	x	x
x	x	x	x
x	x	x	x
x	x	x	x

All 1 and don't care = $T_0 = 1$

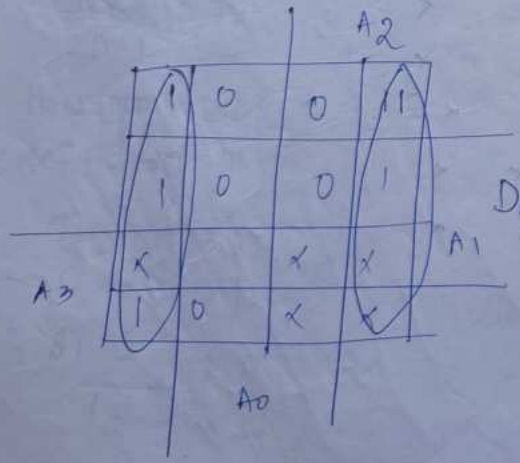




$D_2 = (\quad) + (\quad)$

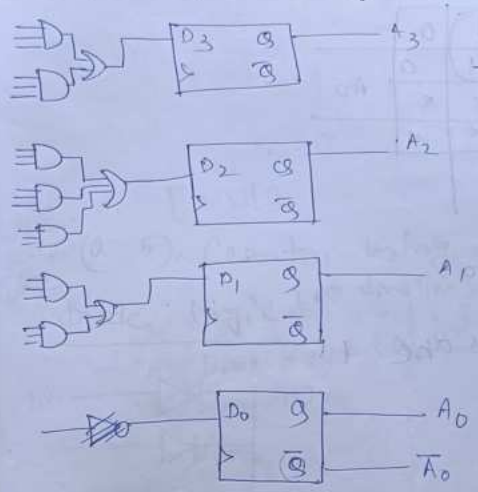


$D_1 = (\quad) + (\quad)$



$D_0 = (A_2\bar{A}_0) + (\bar{A}_0\bar{A}_2) = \bar{A}_0$

Implement D using A-



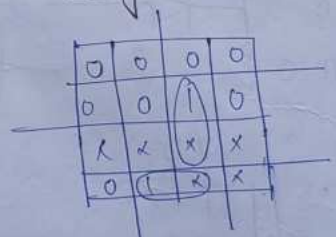
Complexity

:7 AND gates (3 i/p AND gates), 3 OR gates (2, 3 - i/p OR gates), 4 DFF

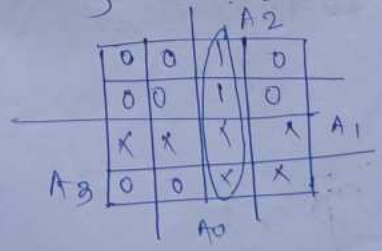
$[\bar{Q} = \bar{A}_0]$

~~Implement~~ ~~using~~

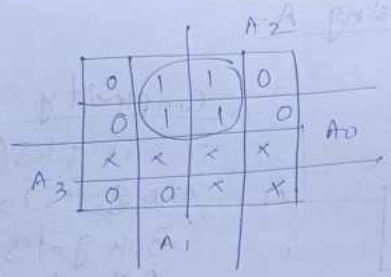
~~is~~ using TFF



$T_3 = (\quad) + (\quad)$



$T_2 = A_2 A_0$



$T_1 = A_3 A_0$

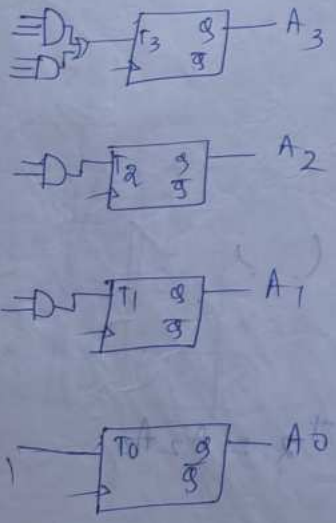
$T_0 = 1$ As all are 1.

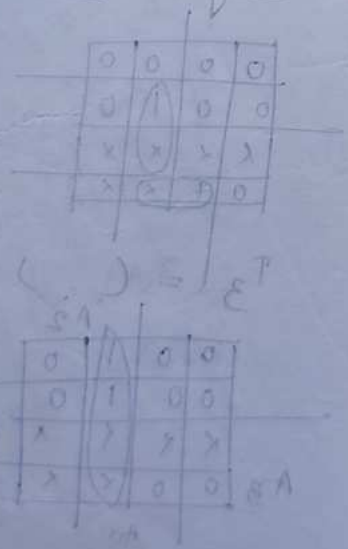
Complexity

4 AND gate

1 OR gate

Implement T using A



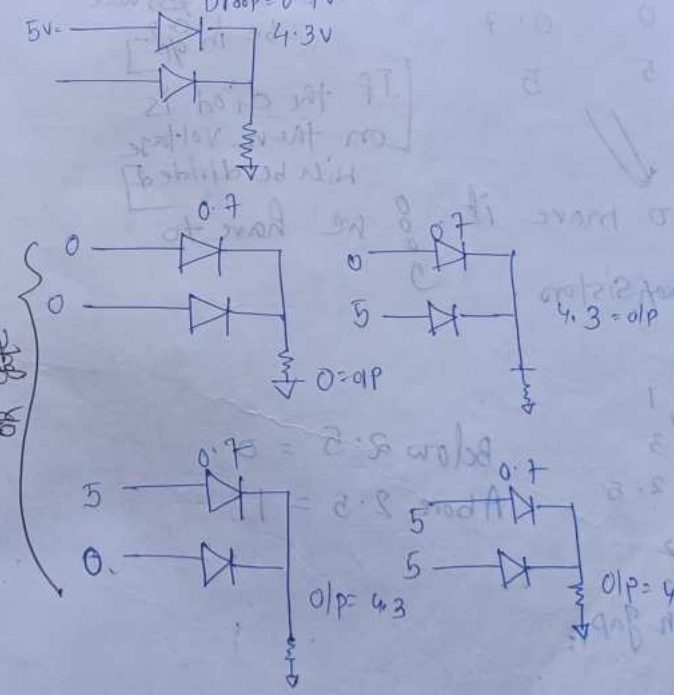


HW [3-bit]

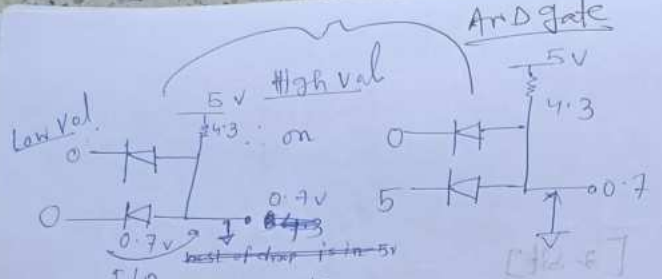
Assignment (0 - 5) Counter using DFF / TFF

Diode logic (no deadline, But ASAP)

Drop = 0.7V = Cut in voltage



AND gate

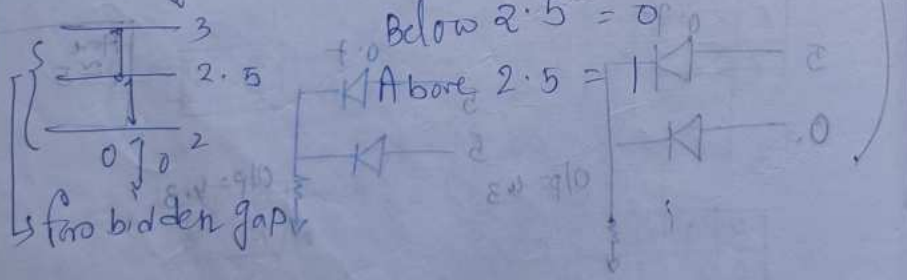


I/p		o/p
0 (Gnd)	0 (Gnd)	0.7
0	5	0.7
5	0	0.7
5	5	5

[If any path is closed on then ~~5~~ 0.7 will be in o/p]

[If the diod is on then voltage will be divided]

to make it 0 0 we have to use transistors.



P2 Prev years Qn

1) How do you compare the following numbers?

a) $(1.10)_2$,
b) $(1.10)_5$,
c) $(1.10)_8$,
d) $(1.10)_{10}$
e) $(1.10)_{16}$.

Comparisons between the following numbers

The 1st one is in binary
The 2nd one " "
" 3rd " " " octal
" 4th " " decimal
" 5th " " hexadecimal

a) $(1.10)_2 = 2^0 + 2^{-1} = 1 + 0.5 = (1.5)_{10}$

b) $(1.10)_5 = 5^0 + 5^{-1} = 1 + 0.2 = (1.2)_{10}$

c) $(1.10)_8 = 8^0 + 8^{-1} = 1 + 0.125 = (1.125)_{10}$

d) $(1.10)_{10} = 1 + 10^{-1} = (1.10)_{10}$ (it is in decimal already)

e) $(1.10)_{16} = 16^0 + 16^{-1} = 1 + 0.0625 = (1.0625)_{10}$

2) Represent the decimal number $(-25)_{10}$ into
a) 8-bits signed magnitude form.
b) 8-bits signed 1's complement form
c) 8-bits signed 2's complement form.

25 =

0	0	0	1	1	0	0	1
128	64	32	16	8	4	2	1

a) $-25 = 0\,0,0011001$

b) $(-25)_{1's} = 11100110$

c) $(-25)_{2's} = 11100111$

3) Perform the addition in BCD code of 2 numbers 875 and 653.

~~875~~

875
+653
1528

1000 0111 0101
+ 0110 0101 0011
1110 1100 1000
+0110 0110
0001
1010 1001 0010 1000
5 2 8

[If you want to remove a term then multiply on side the $(a+\bar{a})$ or $(a\cdot\bar{a})$ respectively.]

4) Proove the equality, $a\cdot b + \bar{a}\cdot c = (a+c)(\bar{a}+b)$

LHS

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

RHS

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

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

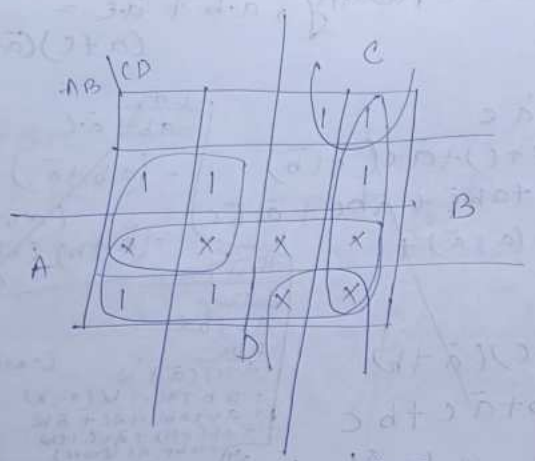
Other process
LHS
$ab + \bar{a}c$
RHS
$(a+c)(\bar{a}+b)$ [$\because a+\bar{a}=1$]
$= ab + a\bar{b}c + bc(a+\bar{a})$
$= ab + abc + \bar{a}c + \bar{a}bc$
$= ab(1+c) + \bar{a}c(1+b)$
$= ab + \bar{a}c$ (Proved)

This is the same as above

a	b	c	$a\cdot b$	$\bar{a}$	$\bar{a}\cdot c$	$a+c$	$\bar{a}+b$	P.S	x+y
0	0	0	0	1	0	0	1	0	0
0	0	1	0	1	1	1	1	1	1
0	1	0	0	1	0	0	1	0	0
0	1	1	0	1	1	1	1	1	1
1	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	1	0	0	0
1	1	0	1	0	0	1	1	1	1
1	1	1	1	0	0	1	1	1	1

(Proved)

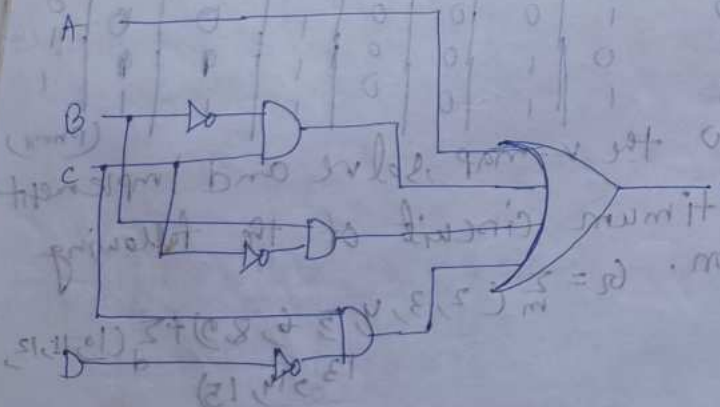
5) Draw the K-map, solve and implement a optimum circuit of the following function, $G = \sum_m(2,3,4,5,6,8,9) + \sum_d(10,11,12,13,14,15)$



$G = A + \bar{B}C + C\bar{D} + B\bar{C}$

$= A + \bar{B}C + B\bar{C} + C\bar{D}$

the final form this is $G = A + \bar{B}C + B\bar{C} + C\bar{D}$



5) Perform the subtraction (P-Q) in 2's complement form, where P = 11100101 and Q = 11101101.

11100101 → P = 229

11101101 → Q = 237

237 = 11101101

↓ 2's

−237 = 00010011

now,

11100101

00010011

11111000

↓ 2's

−00001000 (−8)

∴ P − Q

11100101

11101101

111010010

∴ the result of P − Q in 2's complement

237 = 11101101

↓ 2's

−237 = 00010011

∴ now,

11100101

00010011

11111000

↓ 2's

−00001000 (−8)

1. the result is = −00001000

magnitude = −00001000 = (−8)

7. Design a full-subtractor using two 4:1 multiplexers.

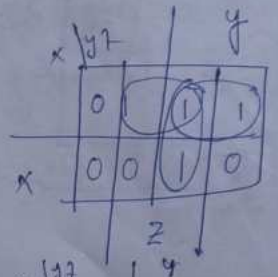
full subtractor

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z (Bin)	0	1	0	1	0	1	0	1
B D	00	11	11	10	01	00	00	11

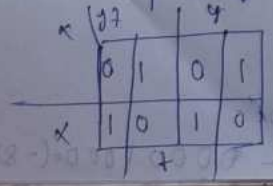
B = Borrow, D = difference

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-maps

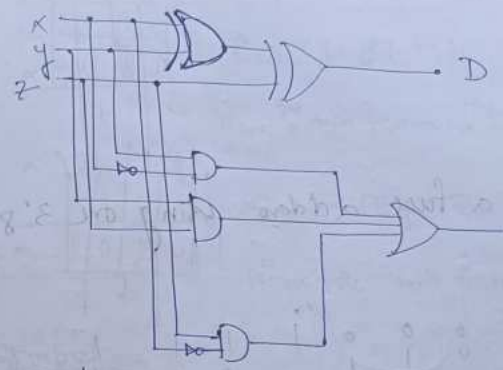


$B = \bar{x}z + \bar{y}z + \bar{x}y$

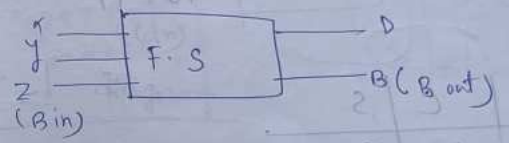


$D = x \oplus y \oplus z$

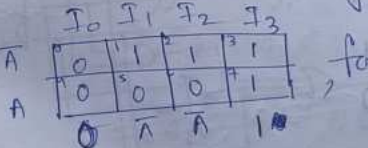
ckt diagram

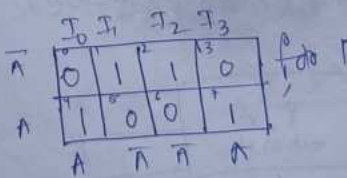


Symbol



full subtractor using 2, 4:1 multiplexers

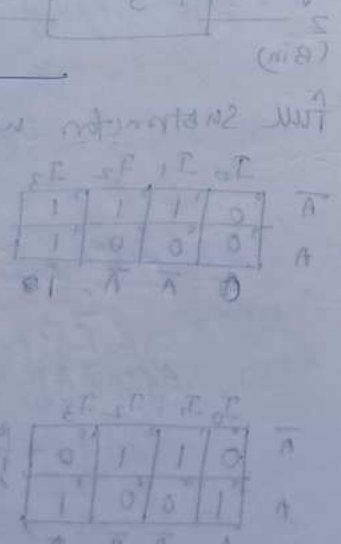




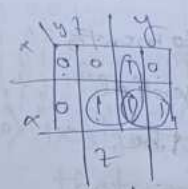
8] Design a full adder using one 3:8 decoder.

full adder

x	y	z (carry in)	C (carry)	S (sum)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

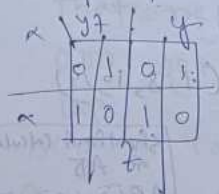


V - maps



$C = xy + yz + xz$

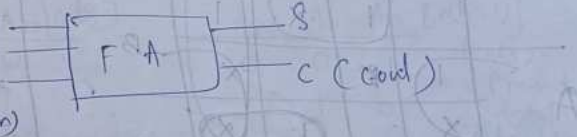
Here the min terms are $\sum_m (3, 5, 6, 7)$



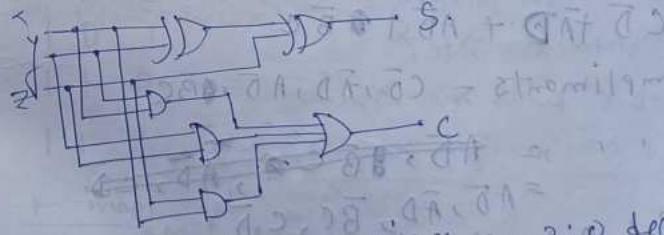
$S = x \oplus y \oplus z$

Here the min terms are $\sum_m (1, 2, 4, 7)$

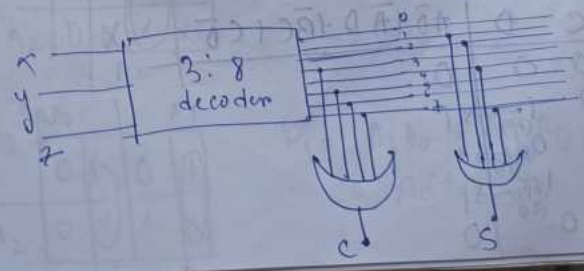
Symbol



Ckt diagram

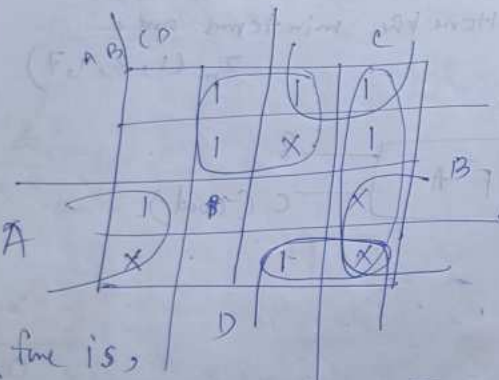


a full adder using one 3:8 decoder



9) Solve the prime implicements using Kmap and draw essential Prime implicants table of the following.

$G = \sum_m (1,2,3,5,6,11,12) + \sum_d (7,8,10,14)$



Shorthand calculation of $A\bar{D}$

$A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + ABC\bar{D}$

$\rightarrow A\bar{B}\bar{D}(\bar{C}+C) + AB\bar{D}(\bar{C}+C)$

$\rightarrow A\bar{B}\bar{D} + AB\bar{D}$

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

$= A\bar{D}$

the fine is,

$G = C\bar{D} + \bar{A}D + A\bar{D} + \bar{B}C$

Prime implimants - $C\bar{D}, \bar{A}D, A\bar{D}, \bar{B}C$

Essential " = ~~$A\bar{D}, \bar{B}C, A\bar{D}, C\bar{D}$~~

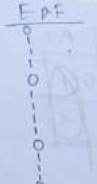
$= A\bar{D}, \bar{A}D, \bar{B}C, C\bar{D}$

∴ Essential Prime implement = $A\bar{D} + \bar{A}D + \bar{B}C + C\bar{D}$

" " table -

A	B	C	D	$A\bar{D}+\bar{A}D+\bar{B}C+C\bar{D}$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1

table is



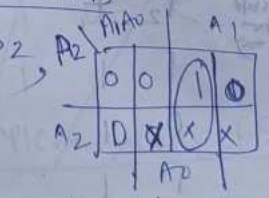
10) Design a up counter using DFF that repeats the sequence 000, 001, 010, 011, 100 in infinite loop.

BCD up counters using DFF to represent 0-4 symbol using 3-bit -

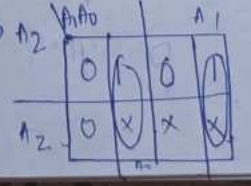
A_2	A_1	A_0 [A(t)]
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

D_2	D_1	D_0	[A(t+1)]
A_2	A_1	A_0	
0	0	1	[A(t) = Present State, A(t+1) = next State]
0	1	0	
0	1	1	
1	0	0	
0	0	0	
x	x	x	
x	x	x	
x	x	x	

K-maps

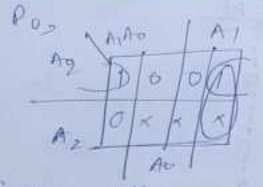


$D_2 = A_1A_0$



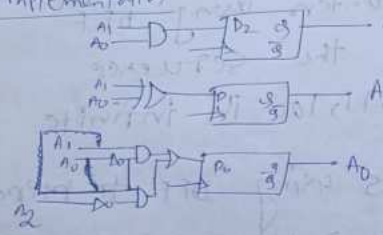
$D_1 = A_1\bar{A}_0 + \bar{A}_1A_0$

$= A_1 \oplus A_0$



$D_0 = A_1\bar{A_0} + \bar{A_2}\bar{A_0}$

implementation

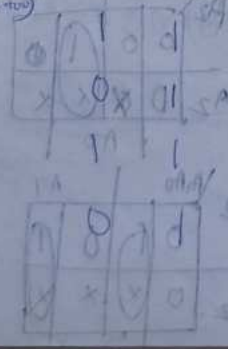


Complexity - 4 AND, 1 OR, 1 XOR, 3 DFF

11) Design a state machine using TFF that implements a 2-bit down-counter when input x=1, otherwise holds its current state.

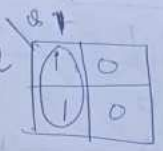
State table with excitation table -

present state		next state		FLIP FLOP	
Q_2	Q_1	Q_2'	Q_1' (from next state state it from 3rd row)	T_2	T_1
0	0	1	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	1	1



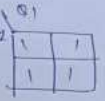
K-map

T_2,



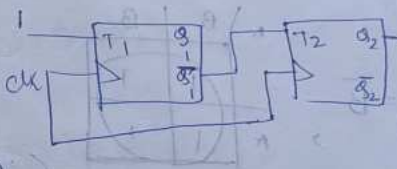
$T_2 = \bar{Q_1}$

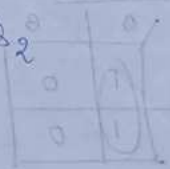
T_1,



$T_1 = 1$

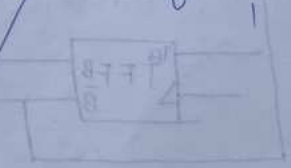
implementation





A_2	A_1	$[A(t)]$ present state	$[A(t+1)]$ next state	T_2	T_1
0	0	1	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	1	1

implementation of state machine



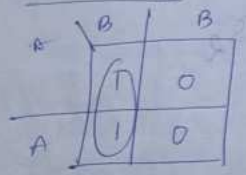
P.T.O

Present state		next state			
A	B	A(t+1)	B(t+1)	T_A	T_B
0	0	1	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	0	1

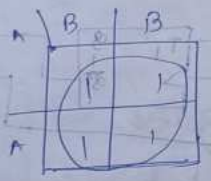
Kmaps as same as previous

Implementation of state diagram

K – maps

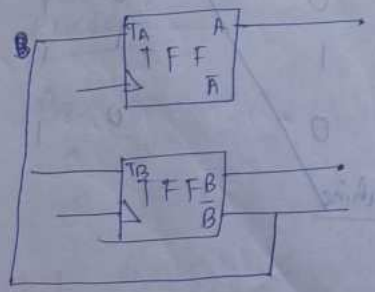


$T_A = \overline{B}$



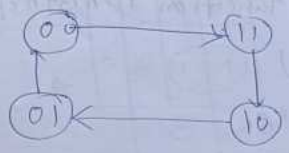
$T_B = 1$

Implementation of state ~~diagm~~ machine



Formal

State diagram



DL pyq year 7th (2021)

Q minize the Boolean function $F = AB + A'C + BC$

$F = AB + A'C + BC(A + A')$

$\rightarrow AB + A'C + ABC + A'BC$

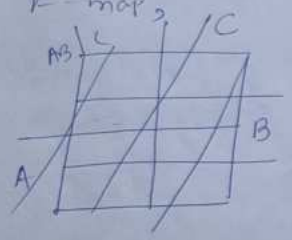
$\rightarrow AB + ABC + A'C + A'BC$

$= AB(1 + C) + A'C(1 + B)$

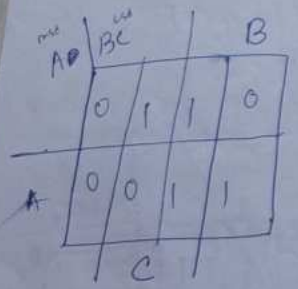
$\rightarrow AB + A'C$

the func is $= AB + A'C$

K-map,



for SOP $= F = AB + A'C + BC$



the Som (Sum of minterms)

$= \sum_m (1, 3, 6, 7)$

other Process, $F(A, B, C) = AB(C + C') + A'C(B + B') + BC(A + A')$

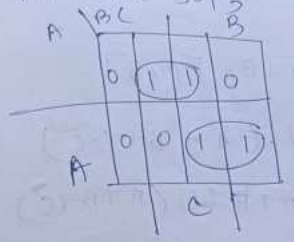
$\rightarrow ABC + AB C' + A'BC + A'B'C + ABC + A'BC$

$= ABC + A'BC + AB C' + A'B'C$

$= m_7 + m_3 + m_6 + m_1$

$\therefore F = \sum_m (1, 3, 6, 7)$

minimal SOP,



$F = A'C + AB$

Other Process,

$F = AB + A'C + BC(A + A')$

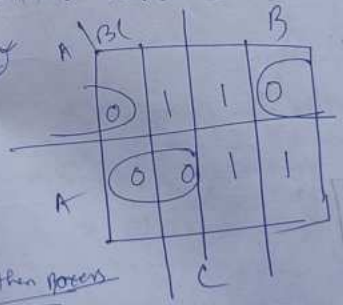
$= AB + A'C$

$F = AB + A'C + BC$

$F = \sum_m (1, 3, 6, 7)$ [Som]

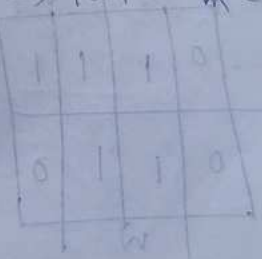
$= \prod_m (0, 2, 4, 5)$ [Pom]

Other Process,



$\leftarrow F = AB + A'C + BC = AB + A'C$

$\rightarrow$ Pom $= \prod_m (0, 2, 4, 5)$



Other Process

$\bar{F} = \bar{A}\bar{C} + A\bar{B}$

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

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

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

the POS is $= (A + C)(\bar{A} + B)$

$F = \prod_m (0, \ldots)$

the other process to get ⊗ form –

$F(A,B,C) = (A+C)(\bar{A}+B)$ [POS]

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

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

$(\bar{A}+B+\bar{C})$

$= M_0 \cdot M_2 \cdot M_5 \cdot M_4$

$\therefore F = \prod_m (0,2,4,5)$

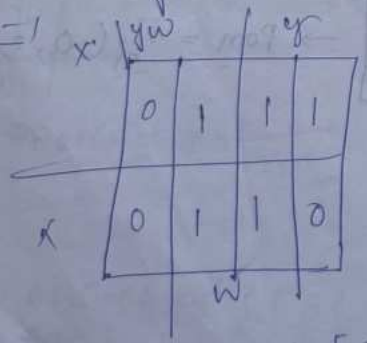
2) Express the Boolean function $G = AB + A'C$ as product of maxterms.

done (a)

3) Express the Boolean function $G = W + X'y$ as sum of minterms.

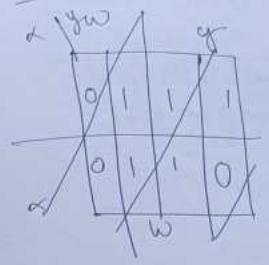
$G = W + X'y$

Process – I



$\therefore G = \sum_m (1,2,3,5,7)$ [Som]

Process – II



$G = W + X'y$ (Sop)

$= W(X'+X)(y'+Xy) + X'y(W+W')$

$= W[X'y' + X'y + Xy' + Xy] + X'yW + X'yW'$

$= X'y'W + X'yW + Xy'W + XyW + X'yW + X'yW'$

$= X'y'W + X'yW + XyW + X'yW' + XyW$

$= m_1 + m_3 + m_5 + m_2 + m_7$

$\therefore G = \sum_m (1,2,3,5,7)$ [Som]

4) Perform subtraction (A – B) of two unsigned binary numbers A = 010010 and B = 010100 using 2's complement method.

A = 010010

B = 010100

↓ 2's complement

B is = 101100

12
-14
-2

∴ A − B is

```
  010010
  101100
  ------
  111110
```

↓ 2's complement

−000010 (−2)

∴ the result is = −000010

5] Convert the Hexadecimal number $(306.E)_{16}$ into Octal number (Base 13).

$(306.E)_{16} = (0011000000110.1110)_2$

$= (1406.70)_8$

6] Add 2 decimal no. 182 and 598 using BCD addition method.

```
 182
+598
 780
```

182 → BCD → 000110000010

598 → BCD → +010110011000

```
  011100011010
+ 000101100000
  ------------
  011100100000
  (7  2  0)
```

```
  0111 0010 0000
+      0110
  ---------------
  0111 1000 0000
  (7   8    0)
```

7. Convert the decimal number $(153.513)_{10}$ into octal number (Base 8).

$(153.513)_{10}$

```
8 | 153
8 |  19 - 1
    2  - 3
```

$\therefore (153)_{10} = (231)_8$

Comma

0.513 × 8 = 0.104 — 4

0.104 × 8 = 0.832 — 0

0.832 × 8 = 0.656 — 6

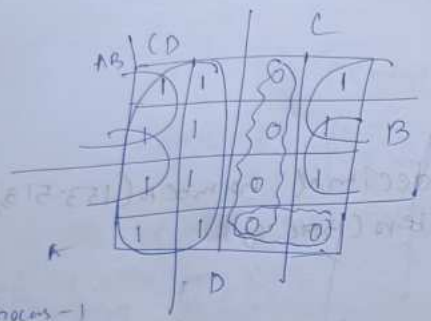
0.656 × 8 = 0.248 — 5

0.248 × 8 = 0.984 — 1

$\therefore (0.513)_{10} = (0.40651)_8$

$\therefore (153.513)_{10} = (231.40651)_8$

8) Find the complement of a Boolean function $F(A,B,C,D) = \sum_m (0,1,2,4,5,6,8,9,12,13,14)$ using K-map method.



Process-1

$F = C' + A'D' + BD'$

If the function is $F = \bar{C} + \bar{A}\bar{D} + B\bar{D}$

$= \bar{A}\bar{D} + B\bar{D} + \bar{C}$

$\bar{F} = \bar{C} + \bar{A}\bar{D} + B\bar{D}$

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

Process-II

$\bar{F} = C\bar{D} + AC\bar{B}$

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

$= A\bar{B}C + ADC + \bar{B}CD + CD$

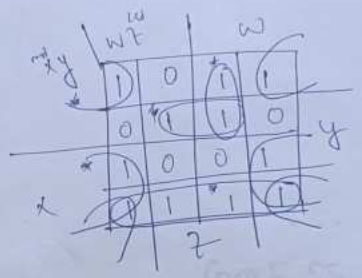
$= A\bar{B}C + CD(1+A) + \bar{B}CD$

$A\bar{B}C + CD(1+A+\bar{B})$ $= A\bar{B}C + CD + \bar{B}CD$

$= A\bar{B}C + CD[\because 1+A+\bar{B}=1]$ $= A\bar{B}C + CD(1+\bar{B}) = A\bar{B}C + CD$

9) Find the essential prime implicants of the Boolean function $F(w,x,y,z) = \sum_m (0,2,3,5,7,8,9,10,11,13,15)$

K-map for the function,



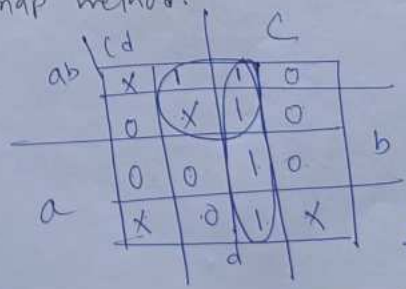
$F = x\bar{y} + x\bar{z} + \bar{y}\bar{z} + \bar{x}yz + \bar{x}wz$

P.I. are $= x\bar{y}, x\bar{z}, \bar{y}\bar{z}, \bar{x}yz, \bar{x}wz$

E.P.I $= \bar{y}\bar{z}, x\bar{z}, x\bar{y}, \bar{x}yz, \bar{x}wz$

the func for E.P.I $= x\bar{y} + x\bar{z} + \bar{y}\bar{z} + \bar{x}yz + \bar{x}wz$

10. minimize the Boolean function $F(a,b,c,d) = \sum_m (1,3,7,11,15) + \sum_d (0,5,8,10)$ using K-map method.



$F = Cd + \bar{a}d$

the function is $F = Cd + \bar{a}d$

After minimizing the function is $F = d(\bar{a}+C)$