SDLD

C R. P Jain, Salibahanan, Morris Mano)

→ Advantage of digital signal over analog signals

→ positive logic , Negative logic

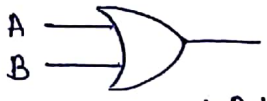
↓

higher — 0

lower — 1

Basic gates and Universal gates

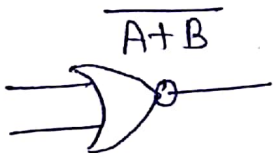
① OR



$\Rightarrow A+B$

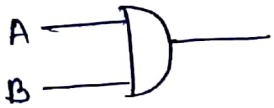
A	B	O/P
0	0	0
0	1	1
1	0	1
1	1	1

NOR



O/P
1
0
0
0

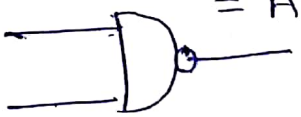
② AND



$= A \cdot B$

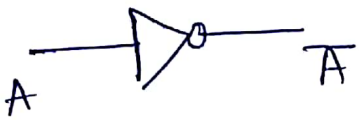
A	B	O/P
0	0	0
0	1	0
1	0	0
1	1	1

NAND



O/P
1
1
1
0

③ NOT



1	0
0	1

Ex-OR — It finds applc in various fields of digital Circuit design wrute in parity checker it gives output equal to 1

$= \bar{A}B + \bar{B}A \Rightarrow A \oplus B$

A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	0

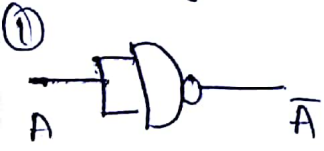
EX-NOR - Not of EX-OR it's ex-nor gate. It is also called Co-incidence gate. Output equal to 1 when all inputs are equal.

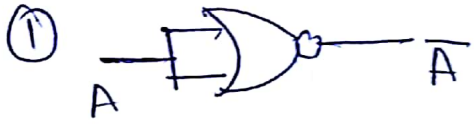
Demorgan's theorem

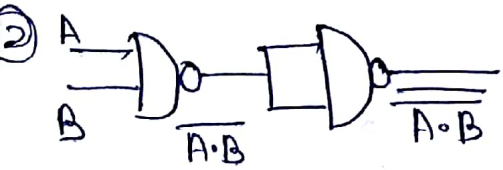
① $\overline{A+B} = \bar{A} \cdot \bar{B}$

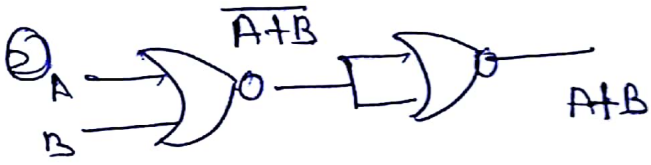
② $\overline{A \cdot B} = \bar{A} + \bar{B}$

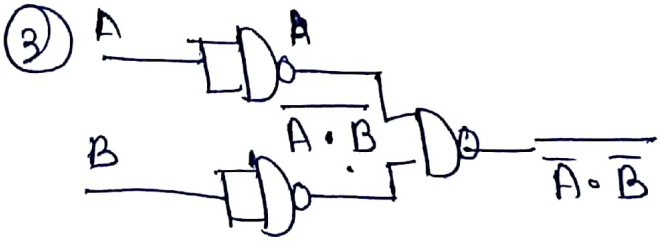
⇒ Realisation of basic gates using Universal gates and demorgans theorem.

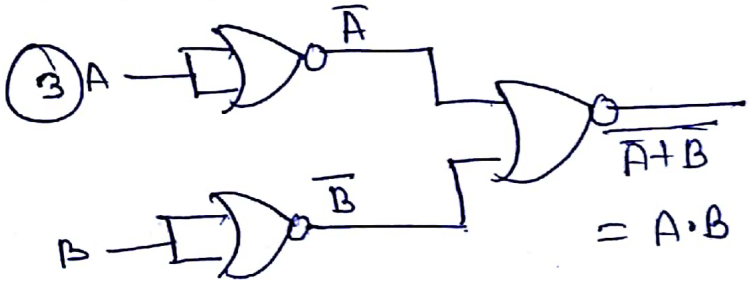
① 

① 

② 

② 

③ 

③ 

Number System

→ We will do binary, octal and hexadecimal.

→ Any number having base x and having m digits to the left and n digits to the right of decimal point can be ~~wri~~ written as general as

$$\Rightarrow a_m x^{m-1} + a_{m-1} x^{m-2} + a_{m-2} x^{m-3} - \cdots + a_1 x^0 + a_{-1} x^{-1} + a_{-2} x^{-2} + a_{-3} x^{-3} - - a_{-n} x^{-n}$$

— MSD → highest wait

a_m = MSD ⟶ (1)0 11 ($2^3\ 2^2\ 2^1\ 2^0$)

a_{-n} = LSD

Q- Convert decimal to binary

$(53 \cdot 625)_{10} \longrightarrow (?)_2$

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

$= 110101$

$\Rightarrow 110101.101$

.625 × 2 = 1.250 → 1

.250 × 2 = 500 → 0

.500 × 2 = 1000 → 1

Q $(101111 \cdot 1101)_2 \rightarrow (?)_{10}$

($2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$. $2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$)

$\Rightarrow$ 1 $1 \times 2^5 +$

$\Rightarrow 47.83$

Q Given $\left(\frac{444}{8}.456x\right)_{10} \longrightarrow (\quad)_8$

8	444	
	55	— 4
	6	— 7

674.3.

Q $(120)_8$ into $(\quad)_{10}$

Solve. $1 \times 8^2 + 2 \times 8^1 + 0 \times 8^0$

$\Rightarrow 64 + 16$

$\Rightarrow 80$

Q $(115)_{10} \longrightarrow (\quad)_{16}$

2	115	
2	27	— 1
2	13	— 1
2	6	— 1
2	3	— 0
	1	

00010111

$= 73$

Q $(237)_{10} \longrightarrow (\quad)_{16}$

16	237	
	14	— 3

$\Rightarrow$ EB

$(A3B)_{16} \longrightarrow (\ ?\ )_{10}$ P.42

Ans 2619 $= 16^2 \times A + 3 \times 16^1 + B \times 16^0$

Octal to binary

For obtaining binary equivalent replace each digit in octal number by 3 bit binary equivalent

37 6

0 011111110

binary to Octal is reverse process start making a group of LSB and replace each by decimal equivalent

$(010\ 011\ 010\ 101) \rightarrow (2325)_8$

$(2D5)_{16} \longrightarrow (?)_2$

001011010101

$(0111\ 011\ 010)_2 \longrightarrow (?)_{16}$

$\rightarrow$ 7B5

$(47)_{16} \longrightarrow (?)_8$

001000111 $\longrightarrow$ 107

$(32)_8 \longrightarrow (?)_{16}$

00011010 $\longrightarrow$ 1A

Decimal Codes

8421

Their are two decimals Code

(1) BCD (binary Coded decimal)

(2) Excess 3

(1) These are used to represent decimal no in bcd each decimal no is encoded by 4 bits

(2) Excess 3 Codes are obtained by adding 3 to bcd of any numbers

13

1

0001 0011

+3 +3

4 6

Q $(643)_{10}$ —— (Excess 3 Code)

976

8421

100101110110

---

① $(0101101)_2 \longrightarrow (?)_{10}$

$= 2^6 \times 0 + 2^5 \times 1 + 2^4 \times 0 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1$

$= 2^5 \times 1 + 2^3 \times 1 + 2^2 \times 1 + 1$

$= 32 + 8 + 4 + 1$

$= 45$

② $(6732)_8 \longrightarrow (?)_{10}$

$6 \times 8^3 + 7 \times 8^2 + 3 \times 8^1 + 2 \times 8^0$

$3072 + 448 + 24 + 2$

$= (3546)_{10}$

③ $(770)_8$

$= 7 \times 8^2 + 7 \times 8^1 + 0 \times 8^0$

$= 448 + 56 + 0$

$= (504)_{10}$

④ $(487F)_{16}$

$4 \times 16^3 + 8 \times 16^2 + 7 \times 16^1 + 15 \times 16^0$

$= 16384 + 2048 + 112 + 15$

$= 18559$

⑤ $(8F7.E2)_{16}$

$8\times16^2+F\times16^2+14\times16^1+7\times16^0+14\times16^{-1}+2\times16^{-2}$

$= 35313.234$

⑥ $(175)_{10} \rightarrow (?)_2$

2	175
2	87 — 1
2	43 — 1
2	21 — 1
2	10 — 1
2	5 — 0
2	2 — 1
	1 — 0

$\rightarrow (10101111)_2$

⑦ $(256)_{10} \rightarrow (?)_2$

2	256
2	128 — 0
2	64 — 0
2	32 — 0
2	16 — 0
2	8 — 0
2	4 — 0
2	2 — 0
	1 — 0

$\Rightarrow (100000000)_2$

⑧ $(7.6)_{10} \rightarrow (?)_2$

2	7
2	3 — 1
	1 — 1

(111.1001)

$0.6 \times 2 = 1.2$

$0.2 \times 2 = 0.4$

$0.4 \times 2 = 0.8$

$0.8 \times 2 = 1.6$

⑨ $(22.79)_{10} \rightarrow (?)_2$

2	22
2	11 — 0
2	5 — 1
2	2 — 1
	1 — 0

(10110.110)

$0.79 \times 2 = 1.58$

$0.58 \times 2 = 1.16$

$0.16 \times 2 = 0.32$

~~00101010~~

Q. $(72)_{10} \longrightarrow (?)_8$

8	72	
8	9	— 0
	1	— 1

$(110)_8$

Q. $(2197)_{10} \longrightarrow (?)_8$

8	2197	
8	274	— 5
8	34	— 2
	4	— 2

$(4225)_8$

Q. $(53.99)_{10} \longrightarrow (?)_8$

8	53	
	6	— 5

$= 65.772$

$.99 \times 8 = 7.92$

$0.92 \times 8 = 7.36$

$0.36 \times 8 = 2.88$

Q. $(802.78)_{10} \longrightarrow (?)_8$

8	802	
8	10	— 2
	8	— 2

$.78 \times 8 = 6.24$

$0.24 \times 8 = 1.92$

822.617

$0.92 \times 8 = 7.36$

Q. $(186979)_{10} \longrightarrow (?)_{16}$

16	186979

Binary Subtraction Using 1's and 2's Compliment

Q. 101010 1's compliment

010101

Subtraction is done by taking 1's Compliment of Smaller number then adding both the no.

$A - B \longrightarrow A + B'$

the Carry is generated at back it to LSB of the result.

111

↓

(1) 011

Subtraction of larger no. from Smaller no.

Takes the 1's compliment of a no. and then add the two No.

—

—

Final answer will be 1's Compliment of the result with −ve Sign. in front of it.

Q. $(1111)_2 - (1010)_2$ using 1's Compliment

Q. $(1000)_2 - (1010)_2$

1) $1111 + 0101 = +\begin{array}{r}1111\\0101\\\hline 10100\end{array} = -0010$

2) $1000 + 0101 = \begin{array}{r}1000\\0101\\\hline 1101\end{array}$

Subtraction Using 2's Compliment Method

→ 0101
1011

take the 2's Compliment of no then add the two no if end Carry inspect the result for end Carry. if Carry occurs discarded and the remaining bits will be the final answer. If their is no Carry answer will be 2's Compliment of result putting neg sign of front of it

Q= M= 1010100 M−N
N= 1000100

N= 1000100 2'sC

⇒ 0111100

```
  1010100
+ 0111100
----------
(1) 0010000
```

N−M

M = 0101100

```
  1000100
+ 0101100
----------
  1110000
```

9's and 10's C

→ 9's and 10's C method are for decimal no.
→ they change the Sub process into add process
→ 9's C of no is obtained by Sub each digit from 9 After that addition is performed the rules are same as 1's Compliment

M = 72532 M−N
N = 03250 N−M

```
  72532
  03250
---------
  69282
```

N= 99999 − 03250 = 96749

M= 99999 − 72532 = 27467

m+N = 96749 + 72532

```
(1) 69281
```

= 69282

→ 10'C

→ 10's C is obtained by leaving all least significant 0 unchanged Sub the first non zero least significant digit from 10. all the remaining higher bits from 9

→ Rules of Sub are same as 2's C

Q. M = 72532 M-N
N = 03250 N-M
using 10'C

$$N = \begin{array}{r} -99\,99\,100 \\ 03250 \\ \hline 96750 \end{array} \Rightarrow \begin{array}{r} 72532 \\ +96750 \\ \hline 69282 \end{array} = \begin{array}{r} -72532 \\ 03250 \\ \hline 69282 \end{array}$$

→ Gray Codes

→ Gray Codes are called minimum change code bcz here only 1 bit in a code group changes when we move from one step to another step.

→ they are non weighted codes.

→ 10101101
11111010

→ 11010
10111

⇒ Gray to Binary

10110101

↓

11011110

⇒ 1010111 into bin
1010111

⇒ 1100101

imp
Q = why gray codes are called reflected code?

Q = why excess 3 Codes are called self-C-codes

LAB

1. NAND NOR

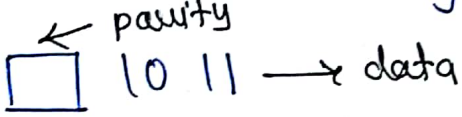
AND, OR, NOT

② $\overline{A+B} = \overline{A} \cdot \overline{B}$

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

#ASCII Code

error detecting codes to detect error we have a Concept of parity bit, a parity can be defined as extra bit which can be sent along with the data



parity can be even bit parity or odd bit parity

→ Use only for one bit error.

Standard representation of Logical function

① SOP ② POS by which we can repre-sent some logical function.

SOP = $\overline{A} \cdot \overline{B} = 0$ POS = $A+B$

In SOP form — means 0

In POS - means 1

- SOP and POS are compliment of each other.

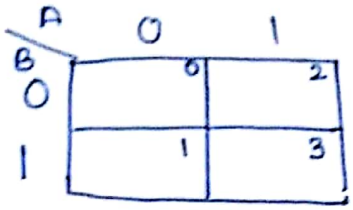
→ $\overline{A} \cdot B + \overline{C}$ → SOP

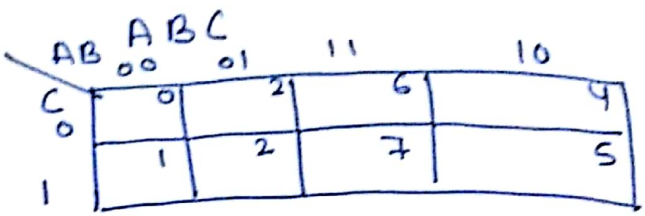
$(A+\overline{B}) \cdot C$ → POS

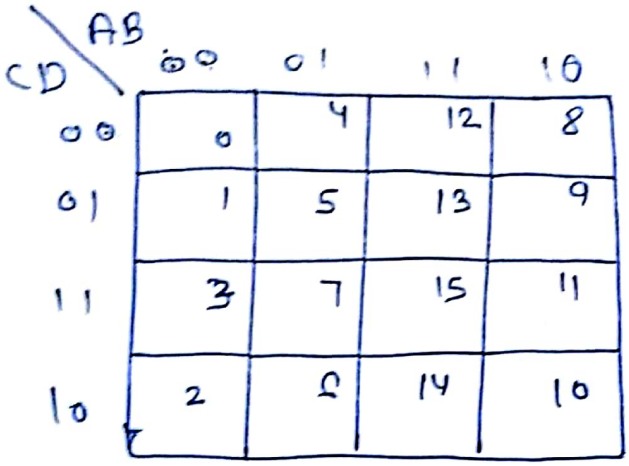
K-Map

→ It stands for Karnaugh map. It is a systematic method for simplify any boolean expression

→ if their are 3 variable = $2^3 = 8$ blocks.







A B C D
0 1 0 1

$2^2 + 2^0$

$= 4 + 1$

$= 5$

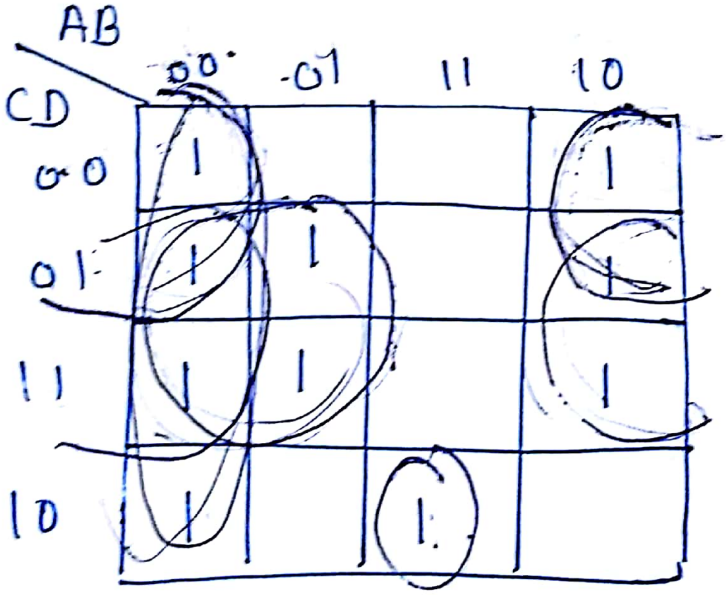
Q minimize the following $F(A, B, C, D) = \sum m(0, 1,$ → SOP C

$3, 5, 7, 8, 9, 11, 14)$

$\Pi m \rightarrow$ POS

C0

SOP



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

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

Q. for the following table Solve the K map.

A	B	C	y
0	0	0	0
0	0	1	-
0	1	0	-
0	1	1	0
1	0	0	-
1	0	1	0
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

SOP

$\Rightarrow \quad \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C}$

Standard SOP form

It means all the Variables should be present in each expression of the equation.

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

equation Convert into Standard form the expression is multiplying by OR the missing terms in C and UnC forms

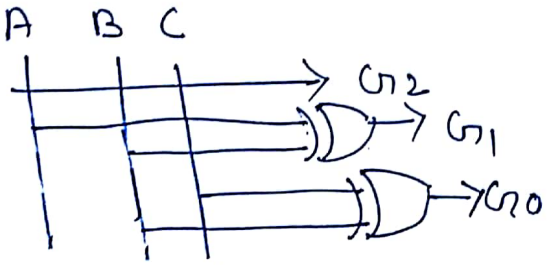
→ In S.SOP for each terms is Called min term.

$\Rightarrow \quad BC\cdot(A+\bar{A})$

Q. $F(A,B,C) = A\bar{B}\bar{C}D + \bar{A}BCD + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{D} + A\bar{C} + A\bar{B}C + \bar{B}$ Solve it by using kmap

$\Rightarrow A\bar{B}\bar{C}D + \bar{A}BCD + \bar{A}\bar{B}\bar{C}(D+\bar{D}) + \bar{A}\bar{B}\bar{D}(C+\bar{C}) + A\bar{C}(B+\bar{B})(D+\bar{D}) + A\bar{B}C(D+\bar{D}) + \bar{B}(A+\bar{A})(C+\bar{C})(D+\bar{D})$.

→ Binary to gray code (Lab)

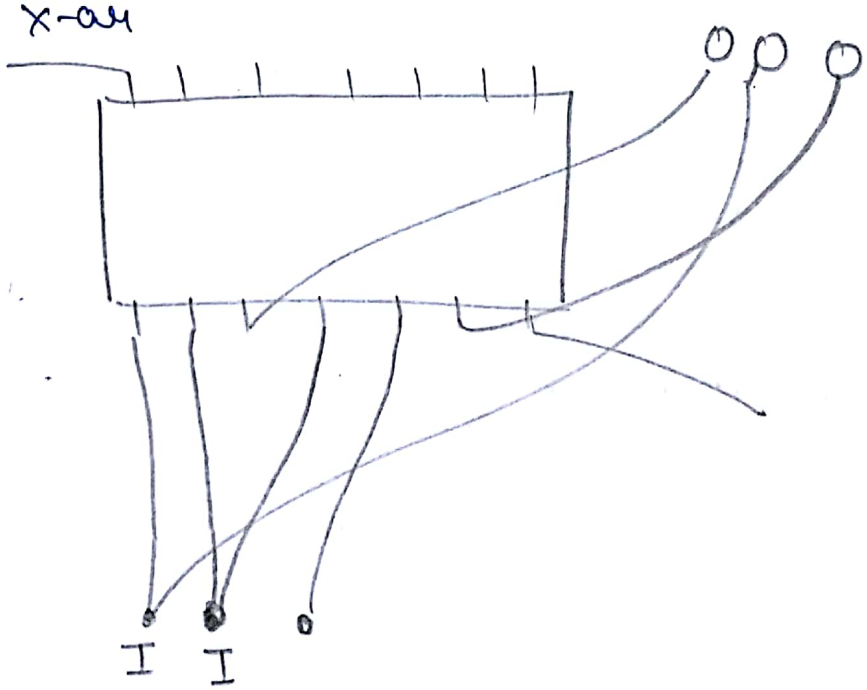


0
0 0 0 → 0
0 0 1 → 1
0 1 0

⇒ 0 —— 9 (BCD)

⇒ BCD → 0–9 binary to gray code → 0–15

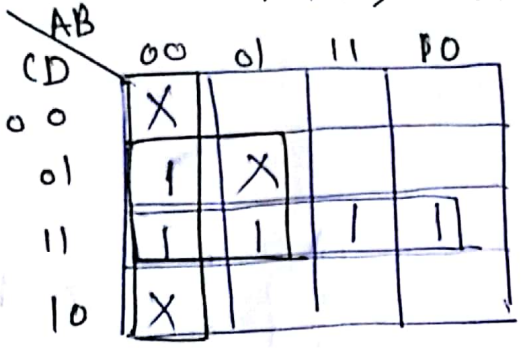
7486 → X-or



Q Dont Care Condition in K-map

→ marked as cross in K-map. they are assumed to be 0 and 1 depending on the requirement they dont effect output in either way

Q Given $f(A,B,C,D) = \sum_m (1,3,7,11,15) + d(0,2,5)$



$= CD + \bar{A}D$

Digital Arithmetic Circuit design

Their are two types digital Circuits.

1. Combinational CKt
2. Sequential CKt

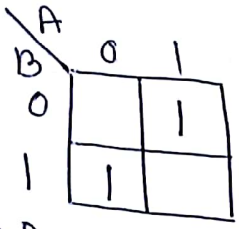
① it is those in which present output depends on present input only where as S.C. in which present output depends on present input as well as past input and output

S.CKt have a memory and Combinational CKt

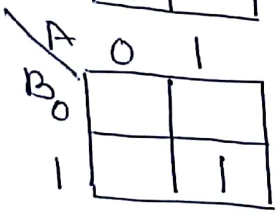
Combinational CKt

Half adder → It is a CKt which acts two bits at any given time their is no provision to add carry.

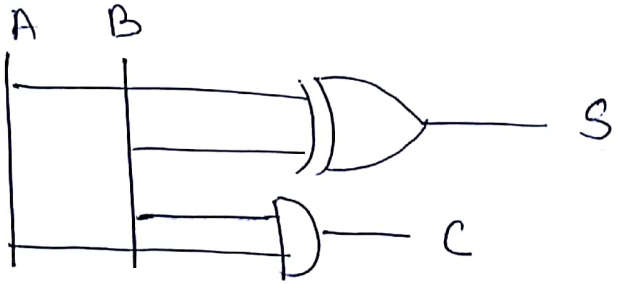
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$Sum = \bar{A}B + \bar{B}A = A \oplus B$



$C = A \cdot B$



Full adder

It is combinational CKt in which their is provision to add carry so, it is basically taking 3 input and Sum and Carry as a Output

A	B	Cn	S	C1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1

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

Sum

C \ AB	00	01	11	10
0	0	1 (2)	(6)	1 (4)
1	1 (1)	1 (3)	1 (7)	(5)

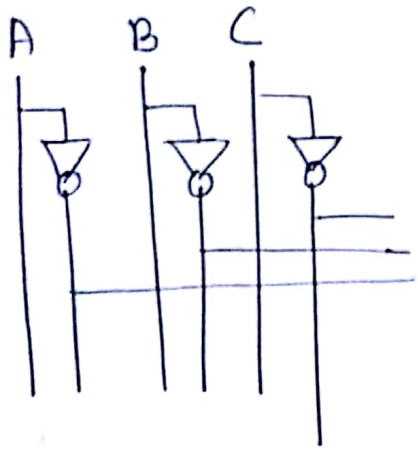
$= \bar{A}B\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$

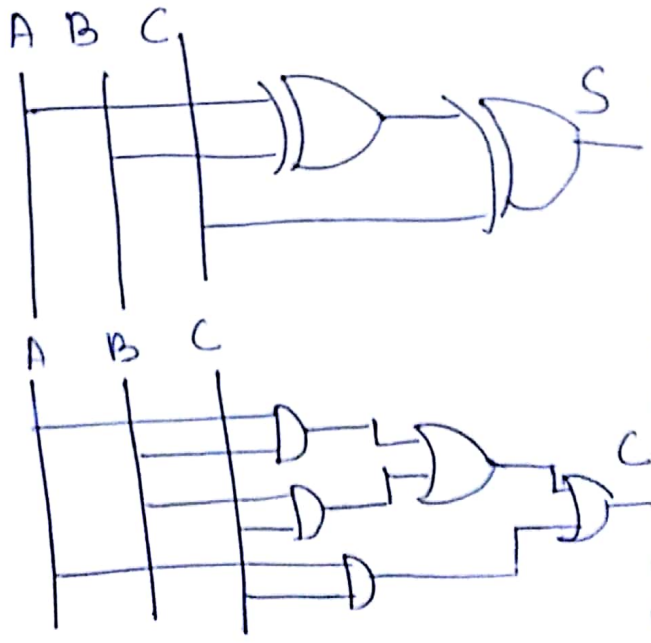
$= A \oplus B \oplus C$

Carry

C \ AB	00	01	11	10
0			1	
1		1	1	1

$= AB + BC + AC$

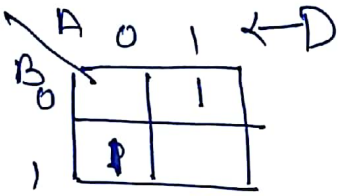




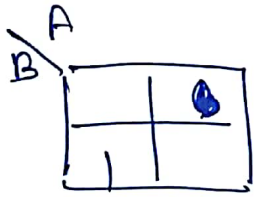
Half Subtractor

a logical CKt for subtraction of A−B where A and B are 1 bit no is referred as half subtractor. the output consist of the difference and the borrow require.

A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



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



← B

⇒ $\bar{A}B$

Full Subtractor.

Multi bit

↑Full subtractor where borrow from previous bit is also considered

A	B	C	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

D

C \ AB	00	01	11	10
0	0	1 2	6	1 4
1	1 1	3	1 7	5

$\Rightarrow \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$

$\Rightarrow A \oplus B \oplus C$

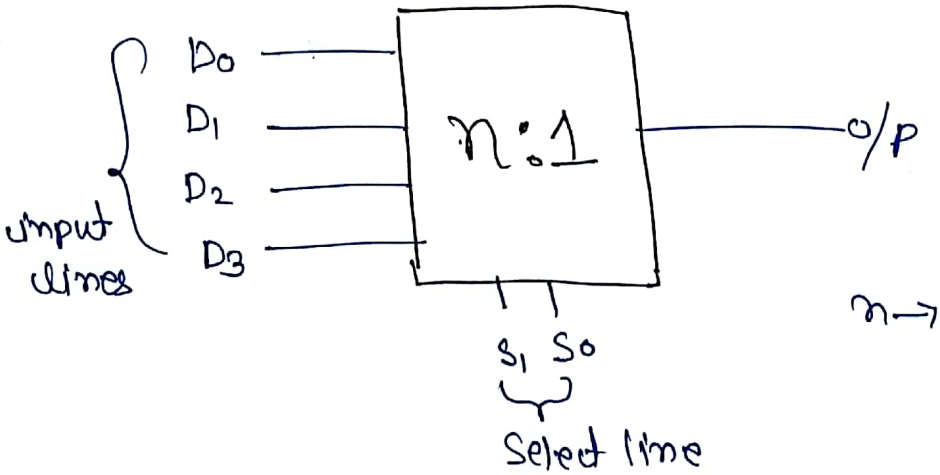
B

C \ AB	00	01	11	10
0		1		
1	1	1	1	

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

= Multiplexer

It is a special kind of Combinational CKt which has many input and one output. the input which is connected to output is selected with the help of select line



$2^n : 1$

n → no of select line

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$Y = D_0\bar{S}_1\bar{S}_0 + D_1\bar{S}_1S_0 + D_2S_1\bar{S}_0 + D_3S_1S_0$

D0

Q. im

F(

→

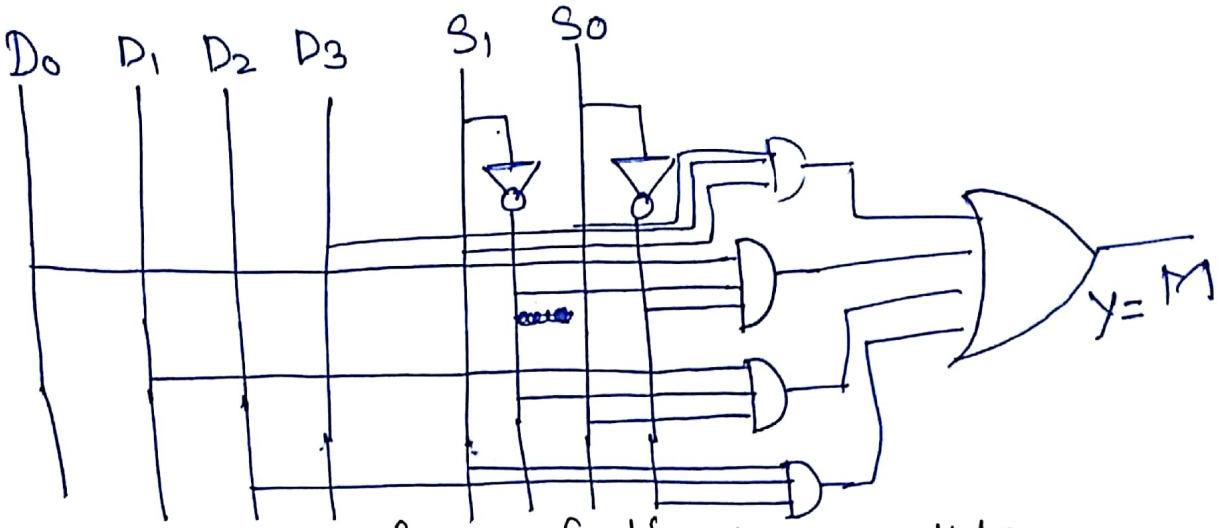
→

In

e

ft

r

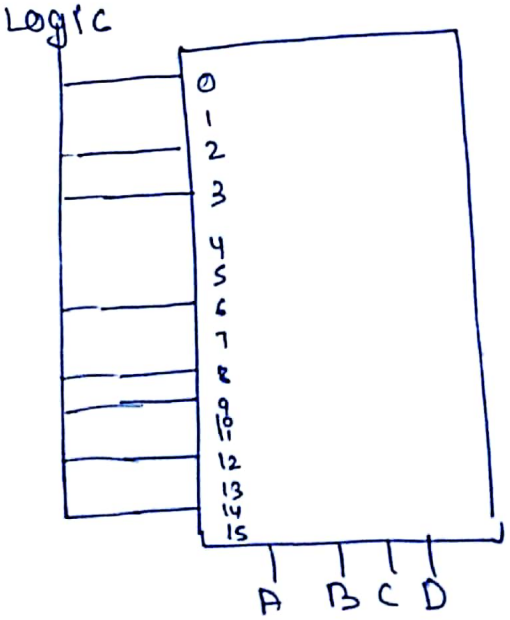


Q. implement the following funtᶜ using multiplexer

$F(A,B,C,D) = \sum (0,2,3,6,8,9,9,12,14)$

→ 4 Select line

→ 16 input line



In every multiplexer their is a pin called enable pin, which is active low and its main funtᶜ is use for cascading that means. ~~is~~ making for higher order multiplex using lower order multiplexer

Minterm – Minterms are AND terms with every variable present in either true or Complemented form

Ex- $X \cdot Y$

Maxterm – Maxterm are OR terms with every variable in true or Complemented form

Ex- $X+Y$

grey Code

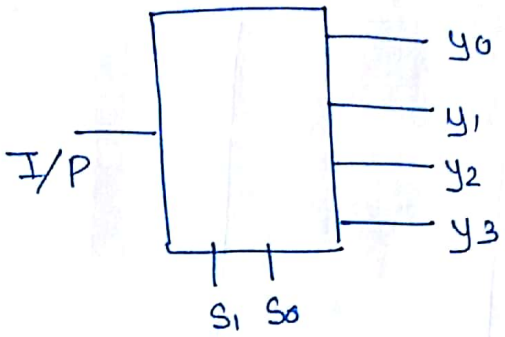
Q. grey Code is sometime refered to as reflected binary bcz the first eight value is Compare with those of the least 8 value but in reverse order

→ reflected binary Code also known as grey Code a ut is binary numeral System where two succesive value differ in only one bit

Excess 3

Q. The key feature of the excess 3-Code is that it is self complementing. this means that the 1's compiment of the excess 3 no is the excess 3 Code for the 9's C of the Corresponding decimal no. the 9's C of decimal no is found by sub each digit in no from 9.

Demultiplexer - It is a single input multiple output device input going to which o/p line is selected with the help of selected line



D	S_1	S_0	y_0	y_1	y_2	y_3
D	0	0	D	0	0	0
D	0	1	0	D	0	0
D	1	0	0	0	D	0
D	1	1	0	0	0	D

$y_0 = D\bar{S_1}\bar{S_0}$

$y_1 = D\bar{S_1}S_0$

$y_2 = DS_1\bar{S_0}$

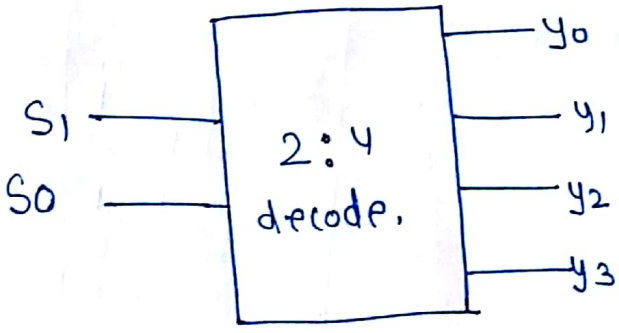
$y_3 = DS_1S_0$

→ Circuit

Decoder

It is similar to dmux accept that their is no input data. Select lines ~~becomes~~ works as a input. the relation b/w input and output is

$n : 2^n$



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

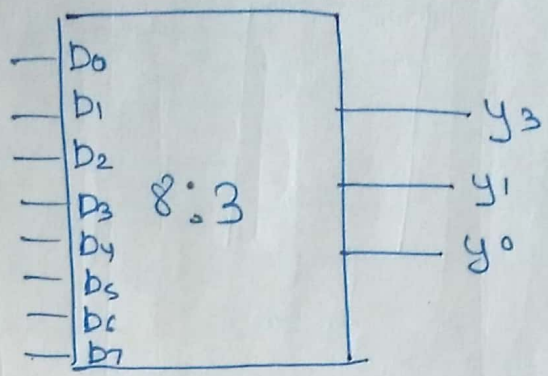
Encoder

It performs an operation opposite to decoder. relation b/w input and output $2^n : n$ at a given time only one input line is high and data is encoded or we

get binary equivalent of that no of o/p

n=3

3:8



D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

y_2	y_1	y_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

8 4 2 1

Design a block diagram of [redacted] encoder which work as decimal to BCD Converter

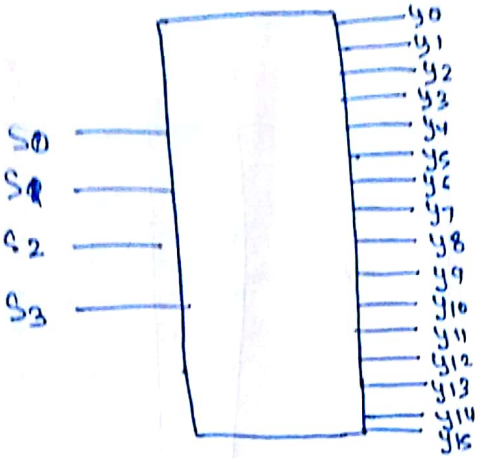
8 4 2 1

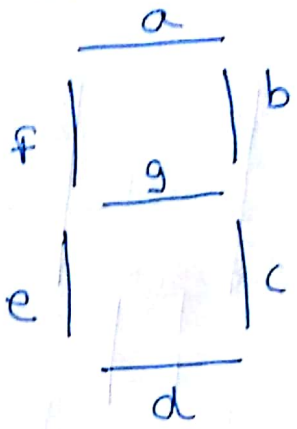
[redacted] 16:4

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1

y_3	y_2	y_1	y_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

BCD to Seven Segment decoder

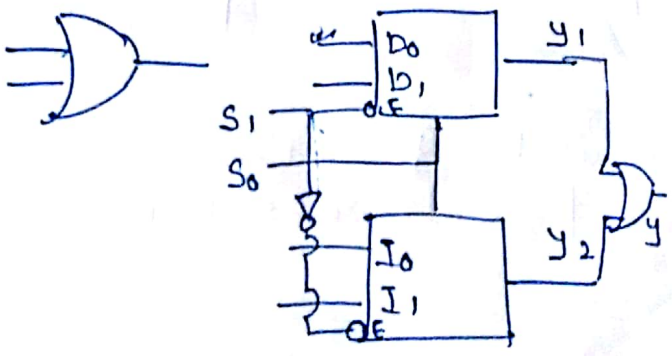




S_3	S_2	S_1	S_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

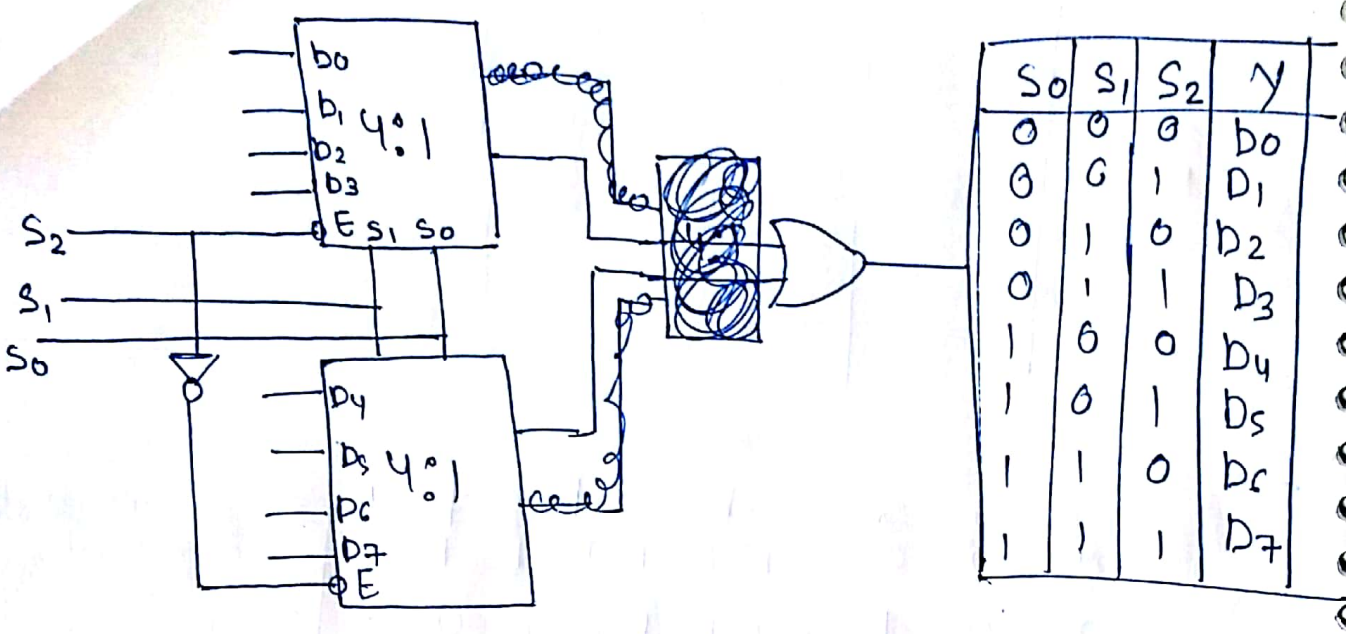
Q. design 4:1 multiplexer using 2:1 multiplexer and OR gate.

⇒



S_1	S_0	Y
0	0	D_0, I_0 I_1
0	1	D_1
1	0	D_2
1	1	D_3

design 8:1 multiplexer using 4:1 and OR gate



S_0	S_1	S_2	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

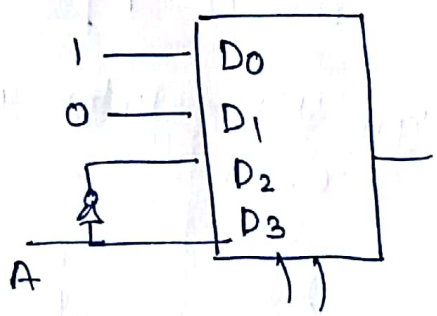
0 → working
1 → not working.

Implementation of higher order function using lower order multiplexer

⇒ given $F(A,B,C) = \sum_m (0,2,4,7)$ implement the following function using 4:1 MUX

→ MSB work as one of the data input

	(00) D_0	(01) D_1	(10) D_2	(11) D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	1	0	$\bar{A}$	A

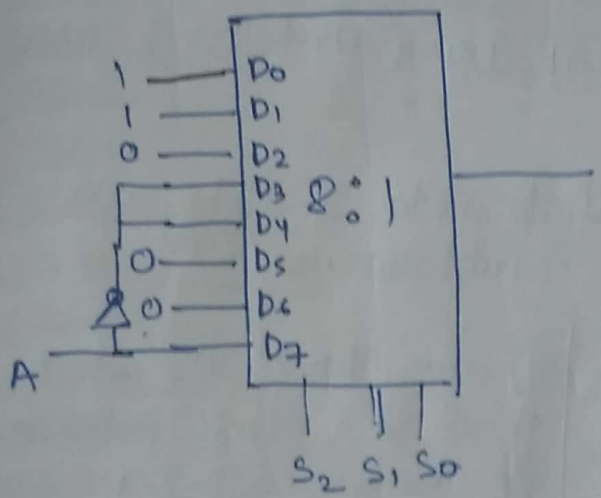


Q. implement a function $F(A,B,C,D) = \sum m(0,1,3,4,8,9,15)$ using 8:1

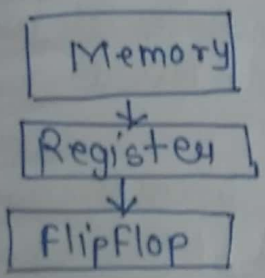
↳ work as MSB

Solve.

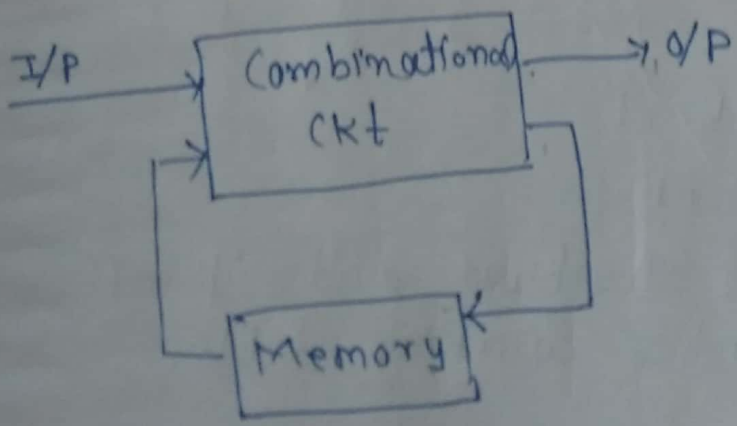
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	$\bar{A}$



Sequencial CKt



the Flipflops are basic building block of any memory at any given time a flipflop can store only one bit of data.



the contents of memory element changed every time whenever their is change of o/p of sequential ckt. So, what will be stored next to memory is called next stage of FF & the content which are stored currently is called present stage of FF.

→ So, we can say Output of Sequencial ckt is a function of time sequence of I/P & internal stage of flipflop

→ Sequential ckt are classified into 2 categories:-

1- Asynchronous (A)

(A) ckt are those in which o/p changes whenever there is a change in I/P

2. Synchronous (S)

(S) are those in which there is a clock present o/p changes with I/P only whenever we apply clock pulses.

i.e their behaviour is dependent on discrete time interval.

→ They are also called "clock sequential ckt".

→ In all practical applic they are used

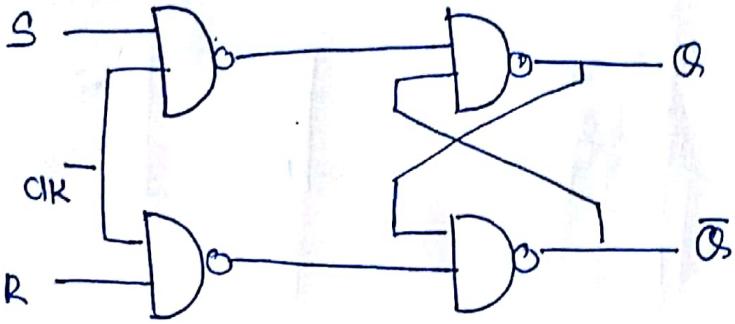
Flipflop

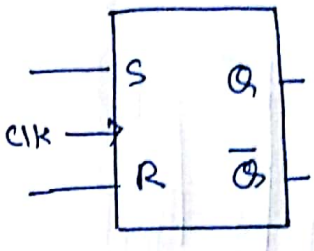
Types of FF:-

1. SR FF
2. JK FF
3. T FF
4. D FF

each ff has its internal structure made up of NAND gate & characteristics working table.

① SR FF - Made up of 4 nand gate.





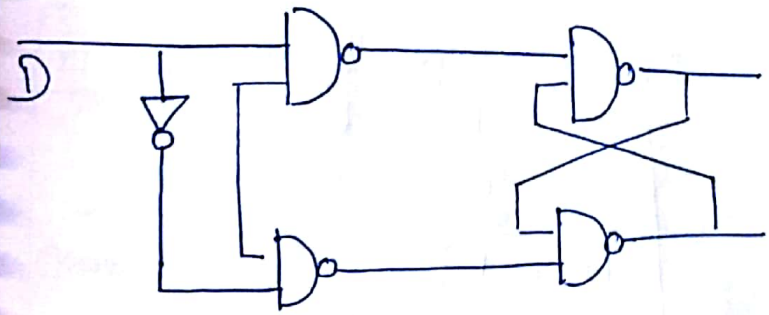
CLK	S	R	Q_{n+1}
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	?

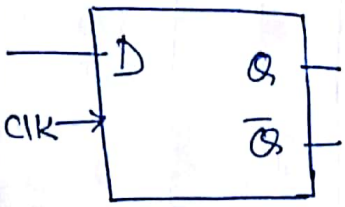
2 bit

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

3 bit.

② D FF (delay)



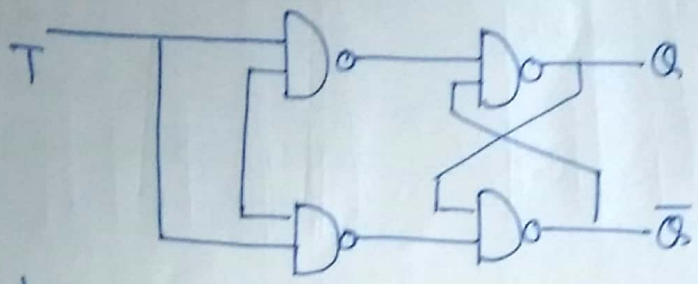


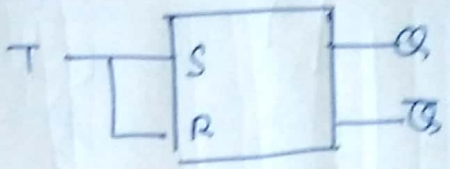
CLK	D	Q_{n+1}
1	0	0
1	1	1

whenever clock is applied, there is change in o/p & I/P

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

T FF (Toggle)





Whenever clock is applied an, I/P, T=0 then next State will be previous State only.
T=1 then next State will be bar of previous bar.

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

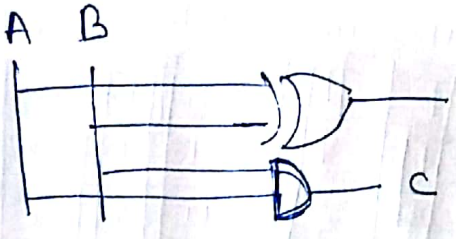
JK FF → when clock is applied

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

→

CLK	Q_n	J	K	Q_{n+1}
0	1	1	1	1
0	0	0	1	0
1	1	1	1	$\overline{Q_n}$→0
1	0	1	0	1
0	0	0	1	0
1	0	0	0	Q_n→0
0	1	0	1	1
1	1	1	1	$\overline{Q_n}$→0

lab

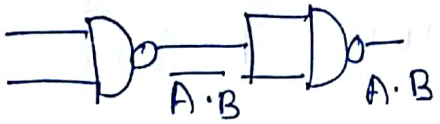


basic

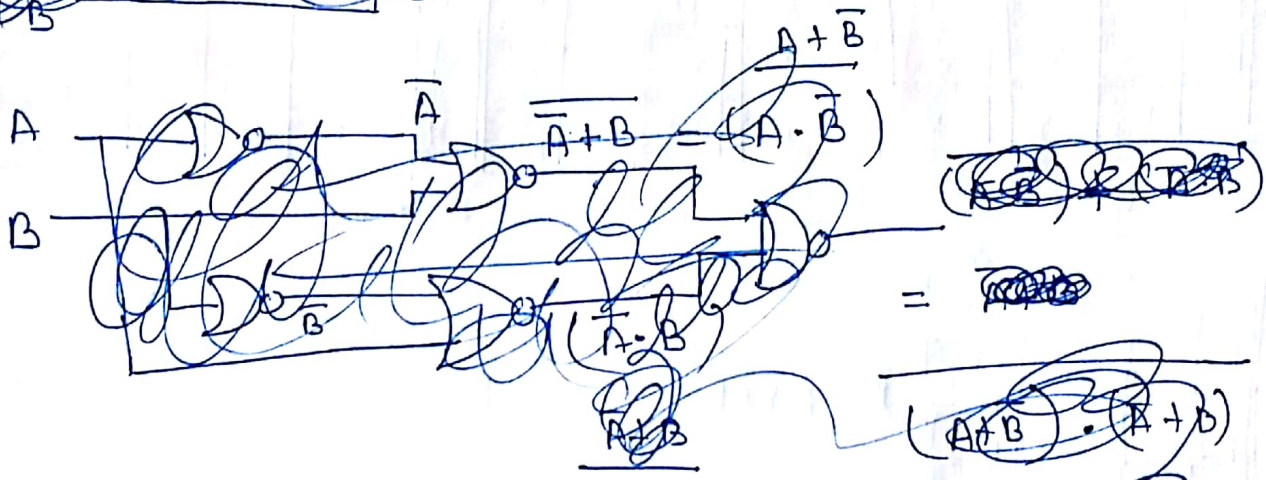
$\overline{A+B} = \overline{A} * \overline{B}$

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

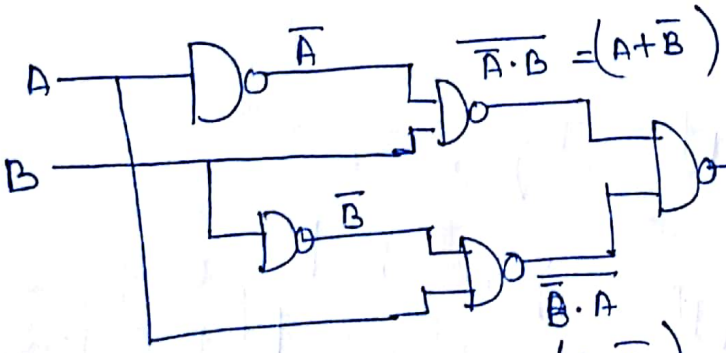
$C = A \cdot B$

① $C =$ 

②



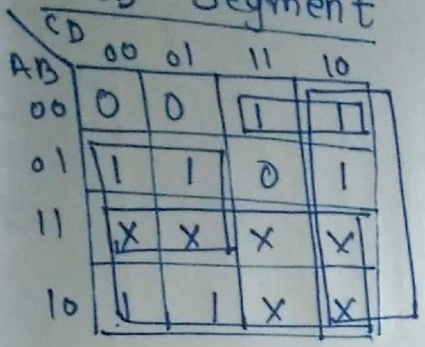
⇒



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

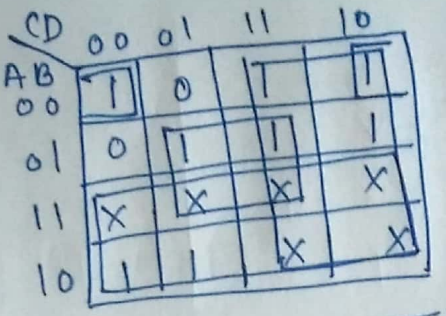
$= (\overline{A} \cdot B) + (\overline{B} \cdot A)$

BCD - Segment

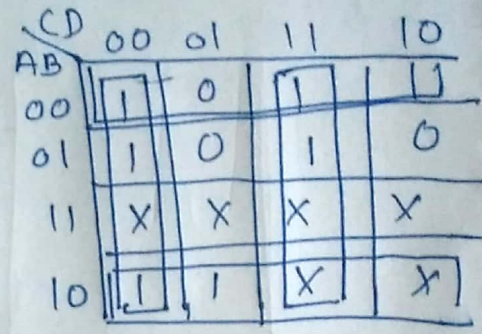


$g = \bar{B}C + C\bar{D} + B\bar{C} + B\bar{C} + A$

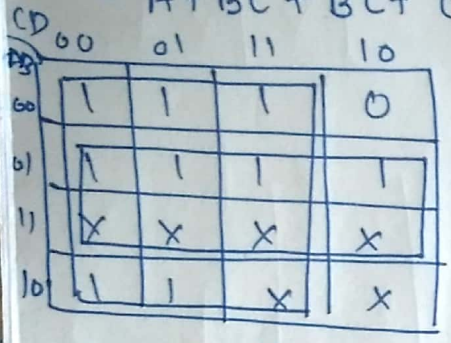
$= A + B\bar{C} + \bar{B}C + C\bar{D}$



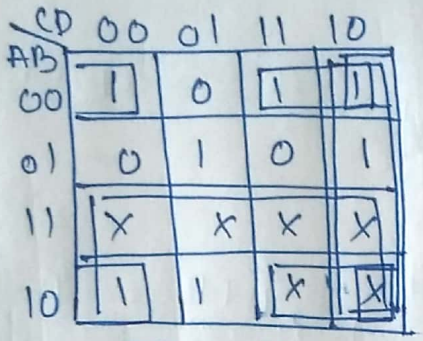
$a = A + C + BD + \bar{B}\bar{D}$



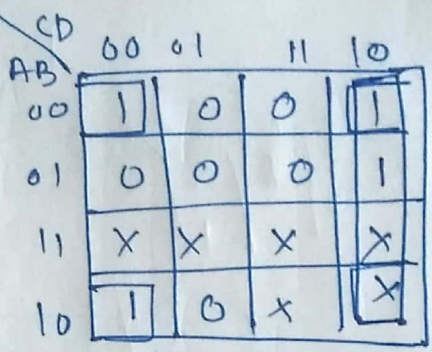
$b = \bar{B} + \bar{C}\bar{D} + CD$



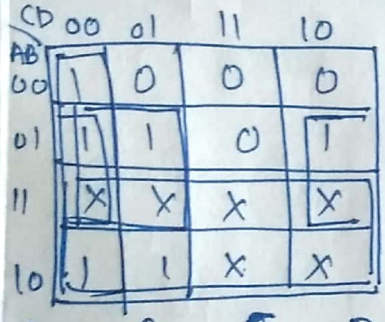
$C = B + \bar{C} + D$



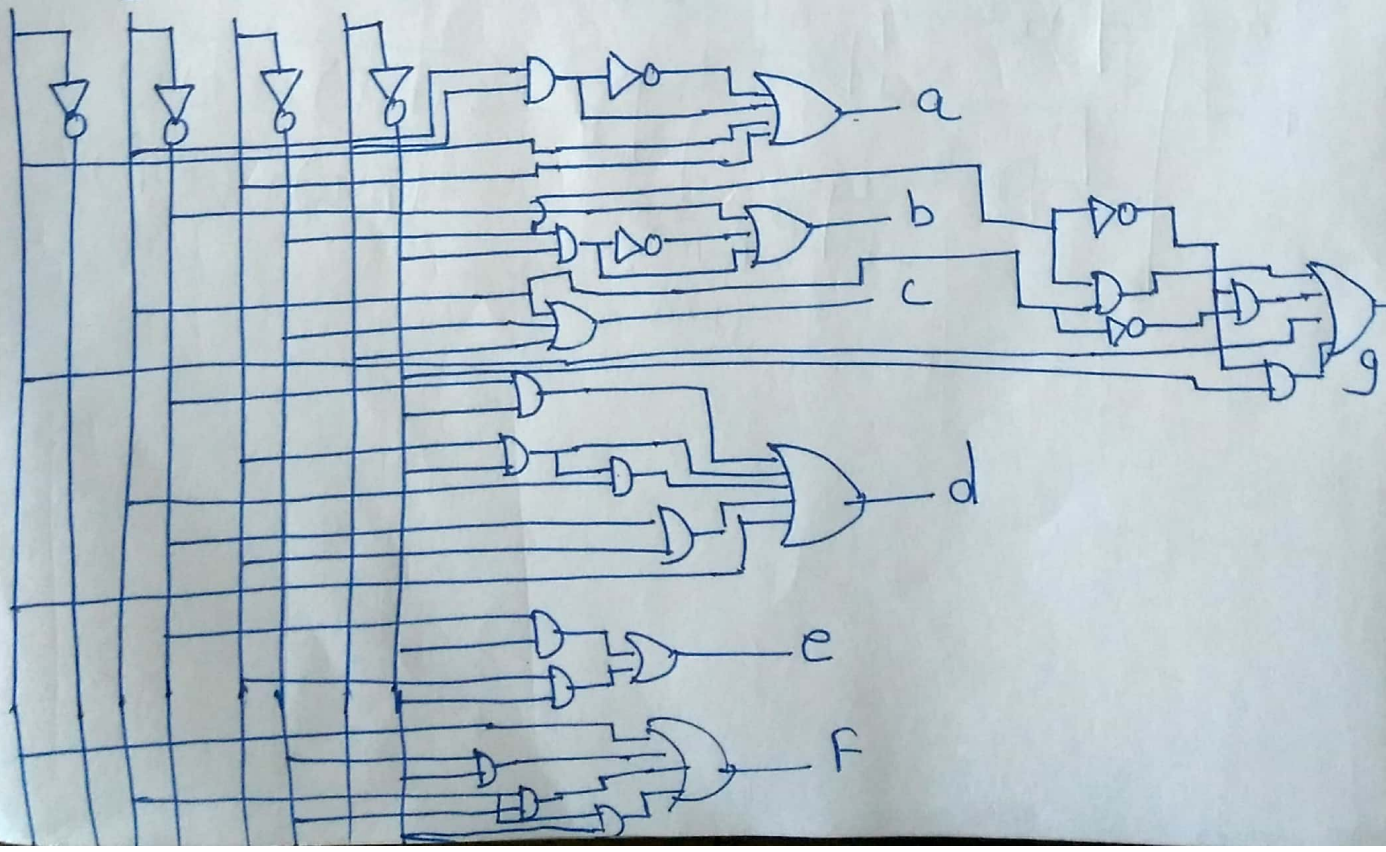
$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$

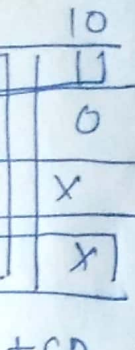


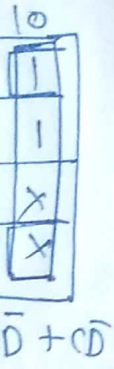
$e = \bar{B}\bar{D} + C\bar{D}$



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







→ Comparator
→ Exitation table
→ Preset & clear in FF
→ Clock
→ level trigeuing
→ Race around Condition
  1. $t_p < \Delta t$
  2. master Slave
→ Poiority Encoder
→ Conversion of ff
  D FF using SR FF
  (R) (E)

⇒ MOD 7
0 - 6

— for Counter designing
(JK and T flipflop)

Q. MOD 6

Present State			Next State.			J_2 K_2	J_1 K_1	J_0 K_0
x_2	x_1	x_0	y_2	y_1	y_0			
0	0	0	0	0	1			
0	0	1	0	1	0			
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

Q - Design MOD-3 Counter using J-K flipflop

PS		NS		J_1	K_1	J_0	K_0
x_1	x_0	y_1	y_0				
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	0	0	X	1	0	X

$2^n - 1$

q_1 \ q_0	0	1
0		1
1	X	X

$J_1 = q_0$

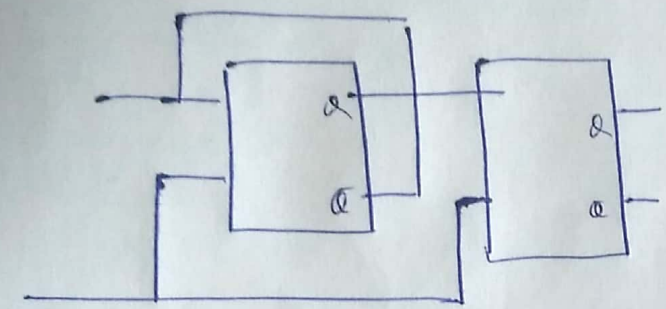
$J_0 = q_0$

$K_1 = 1$

$K_0 = 1$

$J_1 =$

x_1 \ x_0	0	1
0		
1		



Logic

Q design a Counter which Count foreign Stage.

0, 3, 4, 7

Sometime during working of the Counter it goes into same unused stage and keep Counting those un and never come on demanded.

Stage → Knockout Condition

→ overcome

① Self Starting } Counter

② Self Correcting

PS	NS
0	3
3	4
4	7
7	0

Conversions
K-map
MS FF
X-OR using NAND
Compliments (1,2)
decoder using *subtractor, Adder
TTS, CMOS
registers