Digital Logic Design

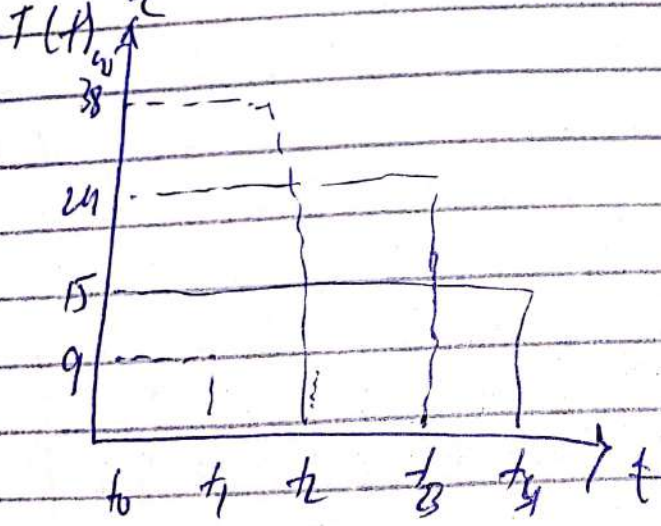
Signal: A signal is a function that represents the variation of a physical quantity with respect to any parameter.

Signal is the variation of electrical quantity (generally current or voltage) with time.

eg: If I is not differ at different times then it is a signal.

Discrete time signal: The signal which is defined for the discrete intervals of time. (it is the subset of analog signal).

Digital Signal: In digital signals, we descritize both time and magnitude.



We divide the magnitude axis into a finite number of levels and the signal can take value equal to these values only.

→ All real life signals are analog.
→ Digital signal is used in communication process to minimize the effect of noise.

noise → unwanted signal.

Digital system → sub system → Modules → Basic unit (logic gates) → circuits (transistors, resistors, capacitors)

Advantages of Digital system
(i) Noise immunity
(ii) Uses less bandwidth.
(iii) Encryption
(iv) Efficiency for longer distance transmission is high.

0 volt = 0 = OFF
5 volt = 1 = ON

Boolean Algebra
It is the set of rules used to simplify the given logic expression without changing its functionality.

eg (A) Rain →
(B) Rain forecast → OR → AND (Y) → Take umbrella
(C) Take car → NOT

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

$Y = BAC' + B'AC' + BC'$

Rules

(i) Compliment me

A compliment = A', $\bar{A}$, not A.

$$(A')' = A$$

(ii) AND.

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

$A \cdot A = A$
$A \cdot 0 = 0$
$A \cdot 1 = A$
$A \cdot A' = 0$

(iii) OR

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

$A + A = A$
$A + 0 = A$
$A + 1 = 1$
$A + A' = 1$

(iv) Distributive law

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

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

$= 1 \cdot (A + B)$

As $1 \cdot A = A$ $= A + B$

$A + A'B = A + B$

$A' + AB = A' + B$

(v) Commutative law

$A + B = B + A$

$A \cdot B = B \cdot A$

(vi) Associative law

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

Priority → (i) NOT (ii) AND (iii) OR

(vii) De Morgan's law

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

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

Example $y = BAC + B'AC' + BC'$

$y = AC'(B + B') + BC'$

$y = AC' \cdot (1) + BC'$

$A \cdot 1 = 1$ $y = AC' + BC'$

$y = (A+B)C'$

Example $AB + AB' = A(B+B') = A \cdot (1) = A$

$A + A' = 1$

$A \cdot 1 = A$

A	B	F = AB + AB'
0	0	0
0	1	0
1	0	1
1	1	1

Example $F = AB + AB'C + AB'C'$

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

$A + A'B = A + B$ $F = A[B + C + B'C']$

$F = A[B + C + C']$

$C + C' = 1$ $F = A[B + 1]$

$A \cdot 1 = A$ $\quad F = AB$.

Exple $F = (A+B+C)(A+B'+C)(A+B+C')$

$A+B = X$ $\quad F = (X+C)(A+B'+C)(X+C')$

$A \cdot B \cdot C = A \cdot C \cdot B$ $\quad F = (X+C)(X+C')(A+B'+C)$

$(A+B)(A+C) = A+BC$ $\quad F = (X+CC')(A+B'+C)$

$A \cdot A' = 0$ $\quad F = (X+0)(A+B'+C)$

$X+0 = X$ $\quad F = X(A+B'+C)$

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

$A+BC = (A+B)(A+C)$ $\quad F = A+B\cdot(B'+C)$

$F = A+B\cdot B'+B\cdot C$

$B \cdot B' = 0$ $\quad F = A+BC$

Example $\quad G = (A+B)(A+B')(A'+B)(A'+B')$

$(A+B)(A+C) = A+BC$ $\quad G = (A+B\cdot B')\cdot(A'+BB')$

$A+0 = A$ $\quad G = (A+0)(A'+0)$

$AA' = 0$ $\quad G = (A)(A')$

$G = 0$

Redundancy Theorem (Trick).

i) Three variables
ii) Each variable is repeated twice (may be A or A').
iii) One variable is complimented.
iv) Take the complemented variable.

eg $Y = AB + A'C + BC$

Apply R.T $\quad Y = A\cdot B + A'C$

Proof $\quad Y = A\cdot B + A'C + BC\cdot 1$

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

$Y = A\cdot B + A'C + ABC + A'BC$

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

$Y = AB + A'C$

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

$F = BC' + AC$

Example $F = AB' + BC + AC$

$F = AB' + BC$

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

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

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

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

Exple $F = A'B' + AC' + B'C'$

$F = A'B' + AC'$

Sum of Product Form

total no. of combination $= 2^n$ $n \rightarrow$ no. of variables

Decimal		A	B	C	F
0	m_0	0	0	0	0
1	m_1	0	0	1	0
2	m_2	0	1	0	1
3	m_3	0	1	1	0
4	m_4	1	0	0	1
5	m_5	1	0	1	1
6	m_6	1	1	0	1
7	m_7	1	1	1	1

F(

$F = A'\cdot B\cdot C' + A\cdot B'\cdot C' + A\cdot B\cdot C + ABC' + ABC$

↳ SOP form ↓ Standard or canonical form → directly from truth table

$1 \rightarrow A$
$0 \rightarrow A'$

SOP form is written only when the function is high.

(m) Minterm → ABC, AB, AC etc
(M) Maxterm → A+B+C, A+B, A+C etc

$$F(A,B,C) = m_2 + m_4 + m_5 + m_6 + m_7$$
$$= \Sigma m(2,4,5,6,7)$$

Reducing the SOP form by boolean algebra

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

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

$C + C' = 1$

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

$A + A'B = A + B$

$$F = A + BC' \rightarrow \text{Minimal SOP form.}$$

because it is minimized.

In canonical / standard SOP form each ~~the~~ minterm is having all the variables in normal or complemented form.

In Minimal SOP form each minterm does not have all the variables in normal or complimented form.

Q. For the given truth table, minimize the SOP expression

A	B	Y
0	0	0
0	1	1
1	0	0
1	1	1

$Y = A'B + AB$ → canonical / standard SOP form

$Y = (A' + A)B$

$Y = (1)B$

$Y = B$ → minimal SOP form

Q. Simplify the expression for $Y(A,B) = \Sigma m(0,2,3)$

$Y = m_0 + m_2 + m_3$

$Y = A'B' + A \cdot B' + A \cdot B$

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

$Y = (1)B' + AB$

$Y = B' + AB$

A	B	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

$A' + AB = A' + B$

$Y = B' + A$ → minimal SOP form

Product of Sum form

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

POS form is used only when the output is low.

$0 \rightarrow A$
$1 \rightarrow A'$

POS	SOP
$0 \rightarrow A$	$1 \rightarrow A$
$1 \rightarrow A'$	$0 \rightarrow A'$

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

Maxterms — Canonical POS form

To SOP

$Y' = A'B'C' + A'B'C + A'BC$

Complement of both sides.

$(Y')' = [A'B'C' + A'B'C + A'BC]'$

$Y = (A'B'C')' \cdot (A'B'C)' \cdot (A'BC)'$

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

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

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

Reducing by boolean algebra

$Y = (A+B+C).(A+B+C').(A+B'+C')$

$(A+B)(A+C) = A+BC$

$A+B = X$ ⇒ $(X+C)(X+C') = X + C.C'$

$Y = (X + C.C')(A+B'+C')$

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

Distributive $Y = A + B.(B'+C')$

Distributive $Y = A + BB' + BC'$

$BB' = 0$ $Y = A + BC'$

$Y = (A+B)(A+C')$ → minimal POS form

Q. For the given truth table, minimize the POS expression.

Maxterm	A	B	Y
M_0	0	0	1
M_1	0	1	0
M_2	1	0	1
M_3	1	1	0

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

$(A+B)(A+C) = A+BC$

$Y = B' + A.A'$

$Y = B'$ $AA' = 0$

In POS form $0 \rightarrow A$ $1 \rightarrow A'$

Y is written when it is low.

Q. $Y = \pi(M_1, M_3) = \pi M(1, 3)$

$Y = \Sigma m(0, 2)$

SOP → AND OR

POS → OR > AND

Q.

	A	B	C	Y
M_0	0	0	0	1
M_1	0	0	1	0
M_2	0	1	0	1
M_3	0	1	1	1
M_4	1	0	0	0
M_5	1	0	1	0
M_6	1	1	0	1
M_7	1	1	1	1

minterm → $Y(A,B,C) = \Sigma m(0,2,3,6,7)$

Maxterm → $Y(A,B,C) = \pi M(1,4,5)$

SOP form 0 → A' 1 → A

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

↳ canonical SOP form minterms

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

$C'+C = 1$

$Y = A'B'C' + A'B + AB$

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

$Y = A'B'C' + B$

$A + A'B = A + B$

$Y = A'C' + B$ → minimal SOP form

POS form 0 → A 1 → A'

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

↳ canonical POS form

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

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

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

$Y = B + A'$

$Y = B + (AA') + A'C'$

$Y = B + A'C'$

Distributive law ~~$Y = AB + BC'$~~

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

Minimal To Canonical Form Conversion (SOP)

$Y = A + B'C$

Step 1 → 3 variables A, B, C

Step 2 → Variables absent in each minterm

Step 3 → Involve the absent variables using $A + A' = 1$

$Y = A + B'C$

$Y = A \cdot 1 \cdot 1 + B'C \cdot 1$

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

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

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

$A + A = A$

$Y = ABC + ABC' + AB'C + AB'C' + A'B'C$ Ans

POS

i) No. of variables ii) Absent variables

iii) Involve / restore the absent variables.

eg $(A + B)(A' + C)$

$(A + B + 0)(A' + C + 0)$

$0 = C \cdot C'$ $\quad 0 = B \cdot B'$

$F = (A+B+C')(A'+C)$ → minimal POS

$F = (A+B+C')(A'+C+0)$

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

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

$X+Y\cdot Y' = (X+Y)\cdot(X+Y')$

$F = (A+B+C')(X+B)(X+B')$

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

↳ canonical POS

POS → OR 0 SOP → AND 1

Example $F = A+B'C$

$F = A\cdot 1\cdot 1 + B'\cdot C\cdot 1$

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

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

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

$A + A = A$

$F = ABC + ABC' + AB'C + AB'C' + A'B'C$.

↳ canonical Ans

For n variables → 2^n minterms or maxterms

Q. For $n=4$, what is the total no. of logical expressions.

For $n=2$ (A, B)

1	A	AB'	$AB = BA$
0	A'	$A'B$	$A+B = B+A$

$A'B + AB'$	B	$A + B'$	$A'B'$
$AB + A'B'$	B'	$A' + B$	$A' + B'$

total = 16 logical expressions for 2 variables

$16 = 2^4 \Rightarrow 16 = 2^{2^2}$ (2 → no of variables)

Total number of logical expressions for n variables are $\boxed{2^{2^n}}$

Compliment Meaning And Examples

$U = \{a, e, i, o, u\}$ $A = \{a, e, i\}$

A' or $\bar{A} = U - A \Rightarrow A' = \{o, u\}$

$U = A + A' = \{a, e, i\} + \{o, u\} = \{a, e, i, o, u\}$

In Binary

$A \rightarrow 0$ or $A \rightarrow 1$ $\quad U = \{0, 1\}$

$A = \{0\}$ $\quad A' = U - A = \{0, 1\} - \{0\} = \{1\}$

$A = \{1\}$ $\quad A' = \{0, 1\} - \{1\} = \{0\}$

$0 \leftrightarrow 1$

AND $\leftrightarrow$ OR

$\cdot \leftrightarrow +$

$0 \leftrightarrow 1$

eg $\bar{F} = \overline{(A'B + B'A)}$

let $A'B = L$ $\quad B'A = M$

$\bar{F} = \overline{(L+M)} = \bar{L} \cdot \bar{M}$

a) $\bar{F} = (\overline{A'B})(\overline{B'A})$

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

b) $\bar{F} = A \cdot B + A \cdot \bar{A'} + B'B + B'A'$

$A \cdot A' = 0$

$\bar{F} = A \cdot B + A'B'$ Ans.

Q. Consider a logic circuit with three inputs A, B, C. output is 1 for the following conditions

i) B and C are true. $B \cdot C$

ii) A and C are false. $A' \cdot C'$

iii) A, B and C are true. $A \cdot B \cdot C$

iv) A, B and C are false. $A' \cdot B' \cdot C'$

Minimize the function Y.

$Y = B \cdot C + A' \cdot C' + A \cdot B \cdot C + A' \cdot B' \cdot C'$

$Y = (1 + A) BC + (1 + B') A'C'$

$A + 1 = 1 \qquad A \cdot 1 = A$

$Y = BC + A'C'$.

Number	Base / Radix	
Binary	2	0, 1
octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	
Dvodecimal	12	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B
Hexadecimal	16	0, ... 9, A, B, C, D, E, F

left most bit → most significant bit
Right most bit → least significant bit.

10101 = 21

MSB → LSB

10100 = 20 → 00101 = 5

Bit is the smallest unit of data.
1 Nibble = 4 bits — BCD and hexadecimals
1 Bytes = 8 bits
1 word = 16 bits
1 double word = 32 words

Decimal to Binary Conversion

Divisor	Dividend	Reads
2	13	1
2	6	0
2	3	1
2	1	1
2	0	0
	0	0

Then Solution 1101

$(25.626)_{10} \rightarrow (\ )_2$

2	25	1 → LSB
2	12	0
2	6	0
2	3	1
2	1	1 → MSB
	0	

$(25)_{10} \rightarrow (11001)_2$

0.625 x 2 = 1.25

Consider the after part ↓

$0.25 \times 2 = 0.50$
$0.50 \times 2 = 1.00$
$0.00 \times 2 = 0.00$ ↓

$(25.626)_{10} = (11001.10100)_2$ → don't write

Integer ÷ 2 ↑ Decimal × 2 ↓

Decimal to Octal

Divide integer by 8 and multiply fractional part by 8.

eg $(112)_{10} \rightarrow (\ )_8$

8	112	0 → LSB
8	14	6
8	1	1 → MSB
	0	

$(160)_8 = (112)_{10}$

eg $(25.625)_{10} \rightarrow (31.50)_8$

8	25	1 → LSB
8	3	3 → MSB
	0	

$(25)_{10} \rightarrow (31)_8$

$(0.625)_{10} = (50)_8$

$0.625 \times 8 = 5.00$
$0.000 \times 8 = 0.0000$

Decimal to hexadecimal

Divide integer part by 16.
Multiply fractional part by 16.

eg $(254)_{10} \rightarrow (FE)_{16}$

16	254	14 → E → LSD
16	15	15 → F
	0	→ MSD

$16 \times 15 = 240$

eg $(25.625)_{10} \rightarrow (19.A)_{16}$

16	25	9
16	1	1
	0	

$(25)_{10} \rightarrow (19)_{16}$

1 → MSB 9 → LSB

A

$0.625 \times 16 = 10.000$

$0.00 \times 16 = 0.000$

$(0.625)_{10} \rightarrow (A0)$

If r → base
from decimal to r base conversion
multiply fractional part by r
Divide by r

Base 4 27.4

4	27
4	6
4	1
	0

3
2
1

$0.4 \times 4 = 1.6$
$0.6 \times 4 = 2.4$
$0.4 \times 4 = 1.6$
$0.6 \times 4 = 2.4$

123.1212

Octal to Binary

0 → 000
1 → 001
2 → 010
3 → 011
4 → 100
5 → 101
6 → 110
7 → 111

$(37.45)_8$
$= 011111.100101$

$(22.07)_8 =$
$(010010.000111)_2$

Binary to Octal

Make group of 3.
Integer part → me ~~left~~ right to left
fractional part → me left to right.

Eg $(010110.110)_2 \rightarrow (26.6)_8$

010 = 2 110 = 6

Binary to Hexadecimal

Make a group of 4.

Hexadecimal ↔ Binary

Octal

Binary Division

```
      016  → Ans
   8 ) 128
       0↓
       12
        8↓
       48
       48
        0
```

```
      000111  → Ans
  110 ) 101010
       -0↓
        10
         0↓
        101
        -0↓
        1010
       - 110↓
        1001
         110↓
        00110
         110
           0
```

Complement → r's complement [Radix comp]
→ (r-1)'s complement [Diminished Radix comp]

Everything other than the given number is the complement of the number.

eg complement of $(7)_{10} = 10 - 7 = 3$

complement of $(9)_{10}$ is 1.

Generalizing N → given number
n → total no. of digits in the given number.
r → base

r's comp $r^n - N$ eg $10^1 - 7 = 3$

eg let N = 5690 10's complement = ?
r = 10 N = 5690 n = 4

$10^4 - 5690 = 4310$

eg * N = 1101 2's complement = ?

r = 2 n = 4

$r^n - N \Rightarrow 2^4 - 1101 = 16 - 1101$

(16 → Decimal, 1101 → binary)

$\Rightarrow 10000 - 1101 = \underline{00011}$

(r-1)'s complement

→ In decimal system r = 10

⇒ r's complement = 10's complement

(r-1)'s complement = 9's complement.

→ for ~~decimal~~ binary system

r = 2

2's complement 1's complement.

→ for octal number system

8's complement 7's complement

r's complement $= r^n - N$

(r-1)'s complement $= r^n - N - 1$

(r-1)'s comp = r's compl − 1

(r-1)'s comp + 1 = r's compl.

In (r-1)'s compl → no borrow operation is involved

In r's compl → 1 borrow is involved.

eg 7's complment of octal no 5674 = ?

$r = 8 \quad n = 4 \quad N = 5674$

$= 8^4 - 5674 - 1$

$= (4096)_{10} - 5674 - 1$

$= (10000)_8 - 5674 - 1$

Remember is 8 in case of octal number system

10000

−1

7777

7's = 7777 − 5674 = 2103

8's cplmt of 5674 = 2103 + 1

= 2104

eg 1's complment of 1101

1111

1101

0010 → 1's

+1

0011 → 2's

for 2's cpl

Q. 1's compliment of $(1010)_2$ = ?

$(r-1)$'s compl = $r^n - N - 1$

$= (r^n - 1) - N$

let $n = 4$

$= (r^4 - 1) - N$

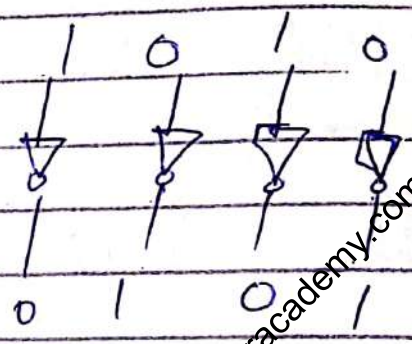
if $r = 10 \Rightarrow$ = 9999

if $r = 8 \Rightarrow$ = 7777

if $r = 16 \Rightarrow$ = FFFF

if $r = 2 \Rightarrow$ = 1111

Here
$$\begin{array}{r} 1111 \\ -1010 \\ \hline 0101 \end{array}$$ Ans



Q. 2's compliment of $(10111010)_2$

2's comp = 1's compl + 1

= 01000101 + 1

= 01000110 → 2's compli

Shortcut for 2's compliment

Step ① Write down the given number

Step ② Starting from LSB, copy all the zeros till the first 1.

Step ③ Copy the first 1.

Step ④ Compliment the remaining bits.

First 1 from LSB

eg ① 10111000 → LSB

②③ 1000

④ 01001000 → 2's compliment

eg 101100

010100 → 2's compliment

Binary Subtraction Using 1's Complement

Step ① Convert the number to be subtracted to its 1's compliment form.

Step ② Perform the addition.

Step ③ If the final carry is 1, then add it to the result obtained in step ②.
If the final carry is 0, result obtained in step ② is negative and is in the 1's compliment form.

End around carry

$$A - B = A + (-B)$$

Example $(1100)_2 - (0101)_2$

A = 1100 B = 0101

B' = 1010

A + (-B) = 1100
+ 1010
10110

final carry = 1
→ so add 1

10110
+ 1
0111

Example A = 0101 B = 1100

-B = 1's compl of B = 0011

A + (-B) = 0101
+ 0011
1000

final carry = 0 → so result is negative and 1's compl

so ans is 0111 → Answer

Binary subtraction using 2's complement

step ① Find 2's compl of the number to be subtracted.

step ② Perform the addition.

step ③ If final carry is generated then the result is positive and in its true form. If final carry is not produced, then the result is negative and is in 2's complement form.

We neglect the final carry in 2's complement.

Example 0100 = B A = 1021

A - B = ?

1's compl of B = 1011
2's compl = 1011
+ 1
1100

A + (2's cp B) = 1100
1001
10101

so A - B = 0101

Range of 2's compl

$[-2^{n-1}$ to $+(2^{n-1}-1)]$

If 4 bits

-8 to $+7$

Eg: A = 0110 B = 1011

1's comp of B = 0100

+1

0101 → 2's compl

A + (2's comp of B) 0110

0101

1011 final ans

So the B is added in 2's comp

1's comp of 1011 = 0100

+ 1

0101 → 2's compl

Ans ← A − B

Data Representation Using Signed Magnitude

Data Representation

Magnitude

Signed

⊕ and ⊖

Unsigned

only +ve

Using mode

Complement

1's compl

⊖ and ⊖

2's compl

⊕ and ⊖

In all four representations the number is represented is same way.

(a) Unsigned mag

+6 → 110

-6 → can't represent.

(b) Signed Mag

+6 → 0110 0 → sign bit

If sign bit = 0 → +ve
" = 1 → -ve

eg. -6 = 1110

eg +13 = 01101
-13 = 11101

Range of signed magnitude

$$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)$$

n → no. of variables

If n = 4 ⇒ Range -7 to +7

(c) 1's Complement Representation

To represent a negative number first write the +ve number and then take its 1's complement.

eg for -6.

1st +6 = 0110
-6 = 1001

eg for -9
+9 = 1001
-9 = 0110

eg
+0 = 0000
-0 = 1111

Range of 1's compliment

$-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

(d) Representation Using 2's compliment

Write the +ve and then take its 2's compl

eg -6

first +6 = 0110
1's compl 1001
+1
1010 → 2's compl

-6 = 1010

Only one zero in 2's compliment

Range

-2^{n-1} to $+2^{n-1}-1$

eg for n = 4 -8 to +7

-ve number of the → take its 2's compl.

Binary Coded Decimal

In this code, each decimal digit is represented by a 4 bit binary number.

$r = 10$ 0 to $(r-1)$ 0 to 9.

143 → decimal number not a decimal digit

Positional weights are 8-4-2-1

Decimal	BCD (↓ 8 4 2 1)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	XXXX
11	XXXX
12	XXXX
13	XXXX
14	XXXX
15	XXXX
~~16~~	~~XXXX~~

$2^4 = 16$

10 out of 16

Decimal numbers not digits.

X → don't care

eg for $(17)_{10}$ → BCD = ?

1 → 0001 7 → 0111

17 → 0001 0111

eg 10

1 → 0001 0 → 0000

10 → 0001 0000

eg $(156)_{10}$ → BCD = 0001 0101 0110

1 → 0001 5 → 0101 6 → 0110

BCD to Decimal

let $(1001\ 0100)_{BCD}$ → $(94)_{10}$ = ?

Start from right end make a group of 4.

0001 0100

1 4

BCD up of decimal numbers beyond 9 is called Packed BCD.

Comparison B/w Binary And BCD

	Binary	BCD
$(10)_{10}$	1010	0001 0000
$(12)_{10}$	1100	0001 0010

arithmetic operations no. of bits.

BCD is less efficient than Binary.

2421 code

Also a BCD code with positional weights 2-4-2-1.

Preferred.

Decimal Digit	Set I 2421 lower significance	Set II 2421 self complementing property
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
→ 5	0101	1011
6	0110	1100
7	0111	1101
8	1110	1110
9	1111	1111

Self complementing code is also called reflecting code.

9 = ($\bar{0}$) 8 = ($\bar{1}$) 7 = ($\bar{2}$)

Excess 3 Code

⇒ Add 3 to BCD

Decimal → BCD 8421 —Add 0011→ Excess 3.

eg 5 → 0101 → 0101 + 0011 = 1000 → (8)

1000 is excess 3 code for 5.

XS-3 code is unweighted
& is a 4 bit code.

Decimal	BCD	XS-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

XS-3 ~~for~~ works

eg 24
2 → 0010 4 → 0100

24 → ~~0100~~ 0011

0010 0011
0011 0100
0101 0111 → XS 3 for 24

Excess 3 code is the only unweighted code which is self complementing.

If sum of weights = 9 → self complementing

eg 2421 ⇒ 2+4+2+1 = 9 ✓
8421 → 8+4+2+1 = 15 ≠ 9 ✗

XS 3 code Addition

$(27)_{10} + (39)_{10}$

$(27)_{10} \rightarrow 0010\ 0111 \rightarrow 0101\ 1010$

$(39)_{10} \rightarrow 0011\ 1001 \rightarrow 0110\ 1100$

$+\ 0011\ 0011$

G_2	G_1
0101^{1}	1010
$+\ 0110$	$+\ 1100$
1011	$0110 \rightarrow X$

$G_1 \rightarrow$ Final carry $= 1$

$G_2 \rightarrow$ F.C $= 0$ $\rightarrow$ subtract 3.

add 3 to it

$1100\ \ 0110$

$-\ 0011\ \ 0011$

$\boxed{1001\ \ 1001}$ Ans.

LOGIC GATES

It is a physical device which performs logic operation on one or more logical inputs and produces a single logical output.

i) Basic gates $\rightarrow$ NOT, AND, OR

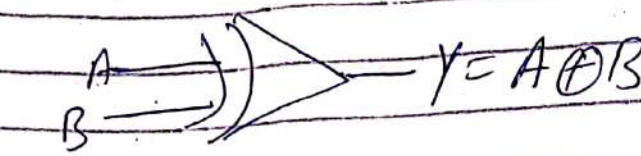
ii) Universal gates $\rightarrow$ NAND and NOR

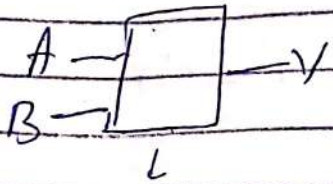
iii) Arithmetic gates $\rightarrow$ X-OR, X-NOR.

NAND and NOR

Associative ✗ Commutative ✓

Exclusive OR gate



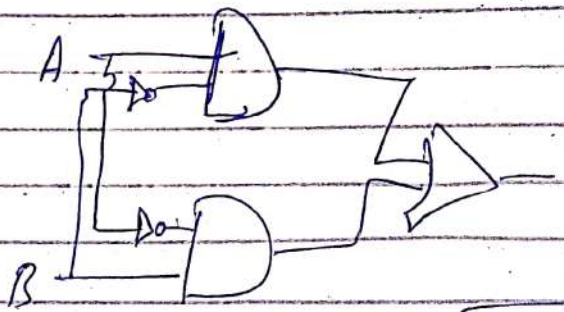


IEEE symbol

$Y = A \oplus B$

$= A \cdot B' + A' \cdot B$

or $(A+B) \cdot (A'+B')$



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$A \oplus A = 0$

$A \oplus A' = 1$

$A \oplus 0 = A$

$A \oplus 1 = A'$

If $A \oplus B = C$

$B \oplus C = A$

$A \oplus C = B$

$\Rightarrow A \oplus B \oplus C = 0$

$A \oplus A = 0$

$A \oplus A \oplus A = A$

$A \oplus A \oplus A + \cdots \; n = A$ if n = odd
$= 0$ if n = even.

XNOR Gate

A	B	A ⊙ B
0	0	1
0	1	0
1	0	0
1	1	1

when A = B o/p = 1
when A ≠ B o/p = 0

XNOR follows associative and commutative laws.

$A \odot A = 1$
$A \odot A' = 0$
$A \odot 0 = A'$
$A \odot 1 = A$

$Y = A \odot B = A \cdot B + A' \cdot B'$

$A \odot A = 1$
$A \odot A \odot A = A$

$A \odot A \odot A \odot \cdots \; n = A$ if n = odd
$= 0$ if n = even

XOR and XNOR → complement → even I/P
XOR and XNOR → same → odd I/P

→ $A \oplus B \oplus C = A \odot B \odot C$

$D \oplus A \oplus B \oplus C = A \odot B \odot C \odot D$

K' MAP (Karnaugh Map)

logic minimization.

	A	B	C	F
m_0	0	0	0	0
m_1	0	0	1	0
m_2	0	1	0	1
m_3	0	1	1	0
m_4	1	0	0	1
m_5	1	0	1	1
m_6	1	1	0	1
m_7	1	1	1	1

SOP → $F = A'BC' + AB'C' + AB'C + ABC' + ABC$

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

$C + C' = 1$

$F = A'BC' + AB' + AB$

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

$F = A + A'BC$

$A + A'B = A + B$

$F = A + BC'$

why? $(A + A') = 1$

A BC
0 00 m_0
0 01 m_1
0 10 m_2
0 11 m_3
1 00 m_4
1 01 m_5
1 10 m_6
1 11 m_7

Same 1 can be paired more than once.

Arrange the cells in such a way that only one variable changes at a time.

A \ BC	00	01	11	10
A=0	m_0	m_1	m_3	m_2
A=1	m_4	m_5	m_7	m_6

map rolling

A \ BC	00	01	11	10
0	0	0	0	1
1	1	1	1	1

II

I

no diagonal pairing → because two variables changing

A → MSB
C → LSB
(MSB to LSB)

Pairing of 2 1s → 4 1s → 8 1s → 16 1s

First check for 16 → 8 → 4 → 2

$F = I + II$ → groups of 1s (implicants)

Write the ones that do not change.

$F = A \cdot 1 \cdot 1 + 1 \cdot B \cdot C'$

$F = A + BC'$

Ex 1: $F(A,B,C) = \Sigma m(1,3,5,7)$

sop: m_1 m_3 m_5 m_7

① Find out no. of variables $n = 3$ (A, B, C)

② Find out no. of cells / total combinations in truth table $= 2^3 = 8$

A \ BC	00	01	11	10
0	0	1	1	0
1	0	1	1	0

↳ implicant

$F = C$ Ans.

Ex 2: $F(A,B,C) = \Sigma m(0,1,2,4,7)$

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	1	0

I, III, IV

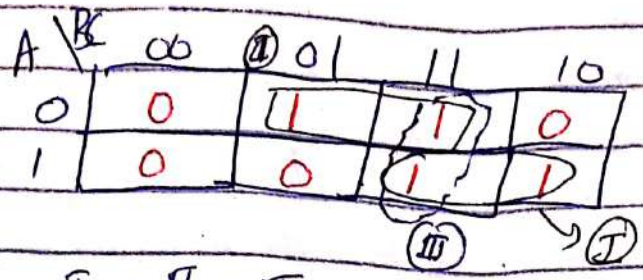
III → cannot be paired with others so alone

$F = I + II + III + IV$

$F = B'C' + A'B' + ABC + A'C'$

Ans

Ex 3 $F(A,B,C) = \Sigma m(1,3,6,7)$.



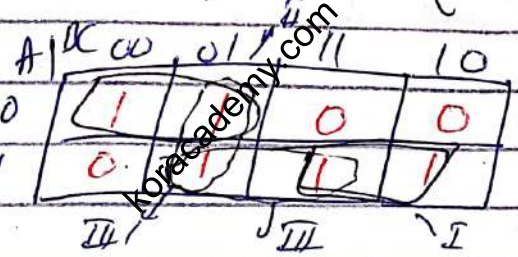
$F = I + II + III$

$F = AB + A'C + BC$

~~$F = AB$~~ Redundancy theorem

$\Rightarrow F = AB + A'C$. Ans

Ex 4 $F(A,B,C) = \Sigma M(0,1,5,6,7)$



$F = I + II + III$	$F = I + II + III'$
$F = AB + A'B' + AC$	$F = AB + A'B' + B'C$

The result is minimum but ~~can~~ may not be the same/unique.

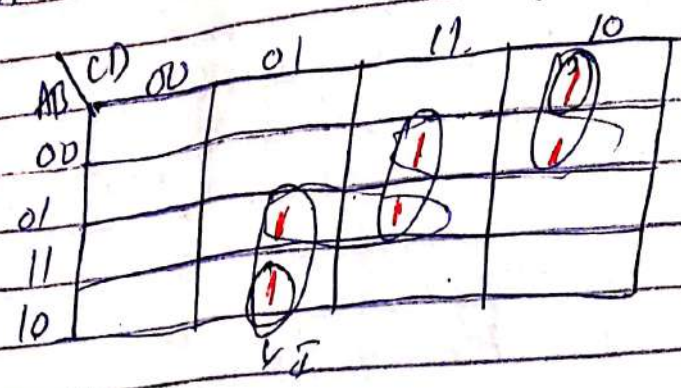
Implicant The group of 1's is called implicant.

It can be a grp of 1, 2, 4, 8, 16 ones.

Prime implicants Largest possible group of a I.

Essential Prime Implicant Grp of 1 in which at least there is single 1 which

Can't be united in any other way.

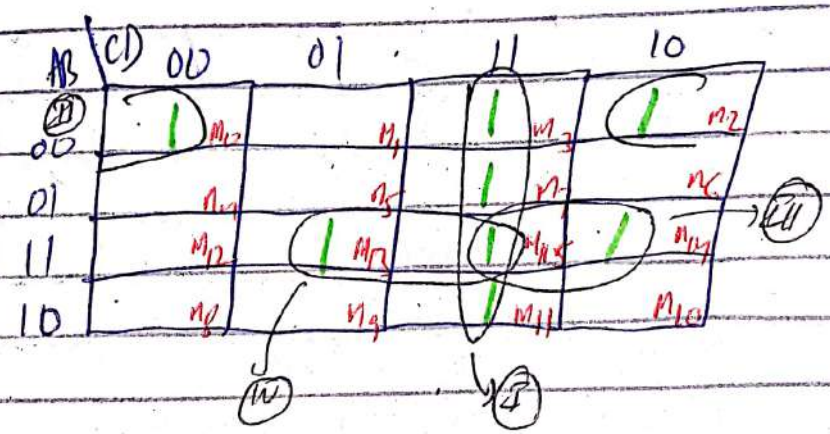


I → EPI II → NEPI

III → EPI

4 variables K MAP

Ex 1 $f(A,B,C,D) = \sum m(0,2,3,7,11,13,14,15)$



$F = I + II + III + IV$

$$F = CD + A'B'D' + ABC + ABD$$

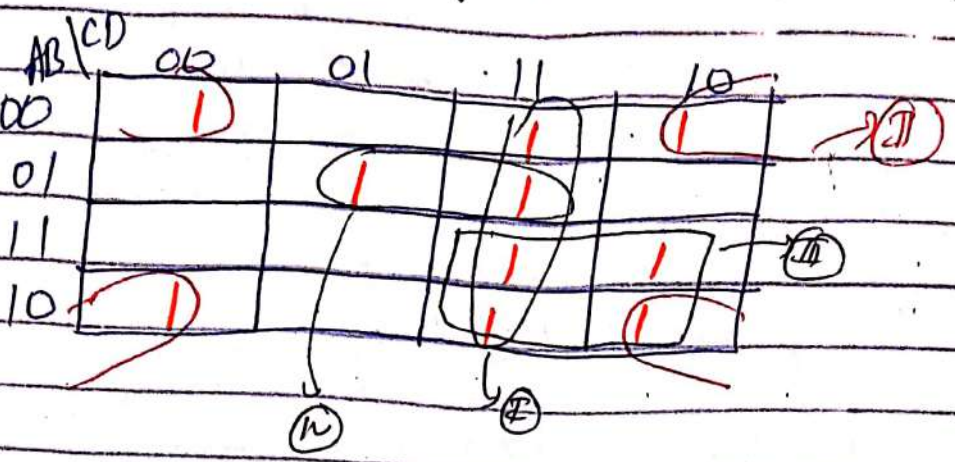
Combine 2 Is → 1 Term is reduced

4 Is → 2

8 → 3

16 → 4

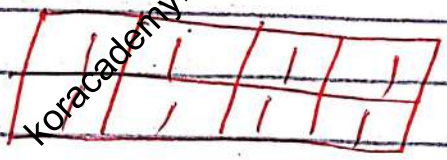
Ex 2 $F(A,B,C,D) = \Sigma m(0,2,3,5,7,8,10,11,14,15)$



$F = I + II + III + IV$

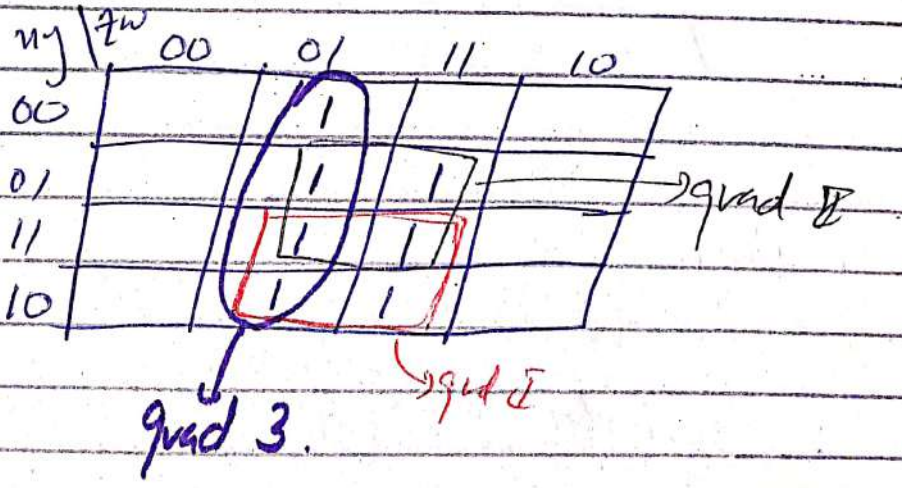
$F = CD + B'D' + AC + A'BD$ Ans

Example 3



$F = 1$

Ex 4 $F(x,y,z,w) = \Sigma m(1,5,7,9,11,13,15)$



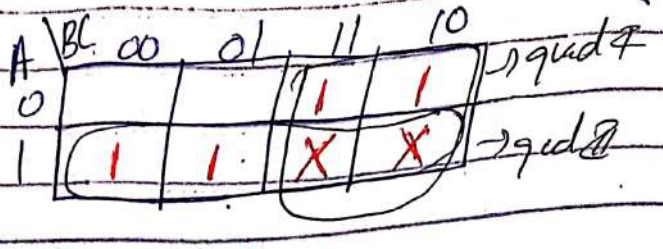
$F = \text{quad I} + \text{quad II} + \text{quad III}$

$F = xw + yw + z'w$ Ans.

Don't Care In K MAP

doesn't matter

Ex 1. $F(A,B,C) = \Sigma m(2,3,4,5) + \Sigma d(6,7)$



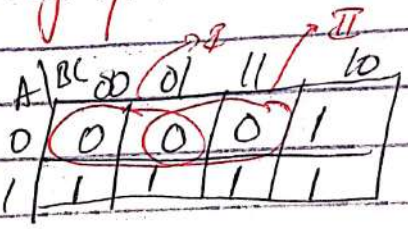
X → 0 for maxterms
X → 1 operations with minterms.

$F = A + B$

K MAP Using Maxterms

Here we group 0s

Ex 1



$\bar{F} = I + II$

$F' = A'B' + A'C$

De Morgan's law

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

$\Rightarrow F = (A'B')' \cdot (A'C)'$

Again De Morgan's law

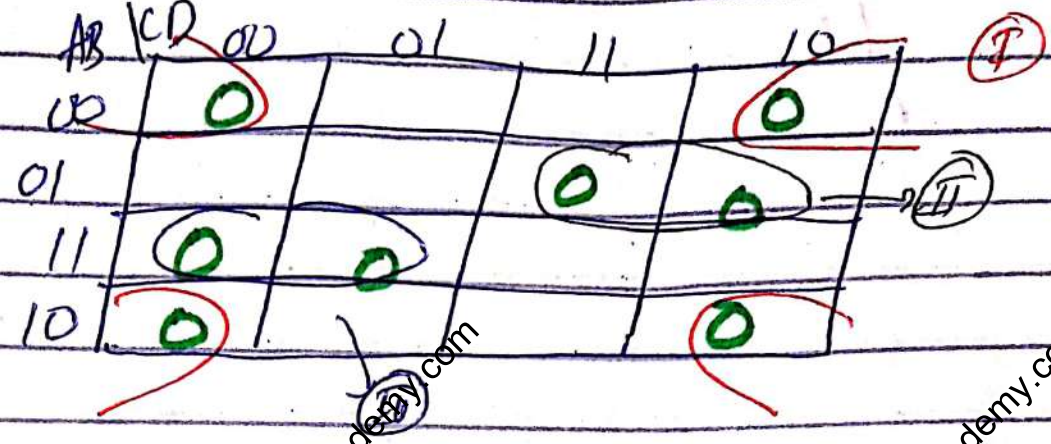
$F = (A + B) \cdot (A' + C')$ POS form

Alternative

$F = A + BC'$

Ex 2. $F(A,B,C,D) = \Sigma m(1,3,5,4,9,11,14,15)$

$F(A,B,C,D) = \pi M(0,2,6,7,8,10,12,13)$



In ~~group~~ Os
~~pairing~~ ~~adv~~ ~~is opposite of terms~~.

2 → 4 → 8 → 16

pairing means

as for O $F' = I + II + III$

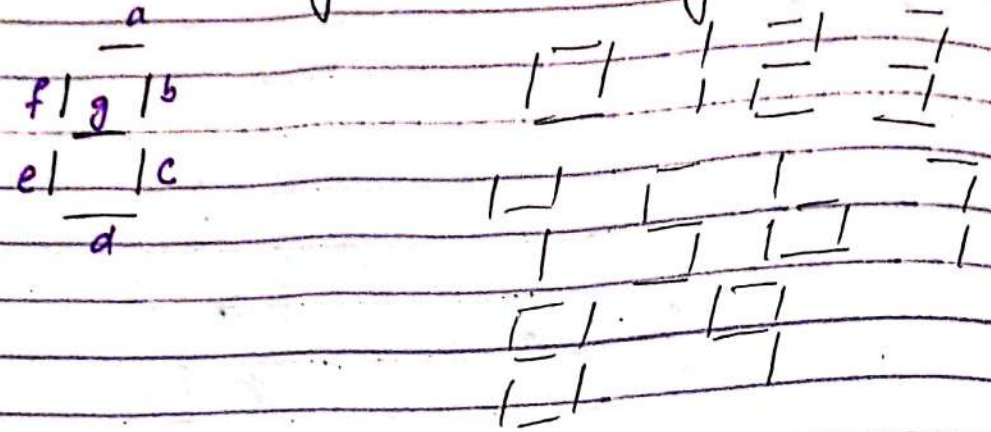
$F' = B'D' + A'BC + ABC'$

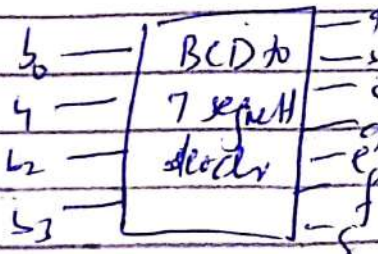
De Morgan's law.

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

Ans

Seven Segment Display Decoder



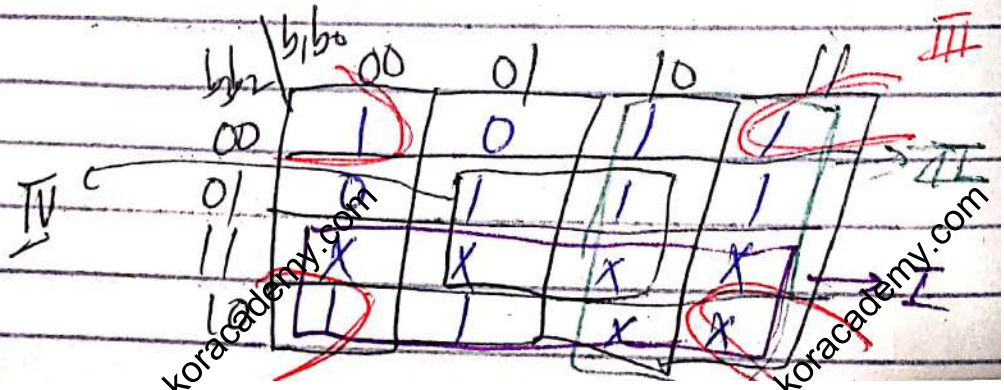


	b_3	b_2	b_1	b_0	a	b	c	d	e	f	g
(0) →	0	0	0	0	1	1	1	1	1	1	0
(1) →	0	0	0	1	0	1	1	0	0	0	0
(2) →	0	0	1	0	1	1	0	1	1	0	1
(3) →	0	0	1	1	1	1	1	1	0	0	1
(4) →	0	1	0	0	0	1	1	0	0	1	1
(5) →	0	1	0	1	1	0	1	1	0	1	1
(6) →	0	1	1	0	1	0	1	1	1	1	1
(7) →	0	1	1	1	1	1	1	0	0	0	0
(8) →	1	0	0	0	1	1	1	1	1	1	1
(9) →	1	0	0	1	1	1	1	1	0	1	1

Greater than 9 → don't care.

K Map

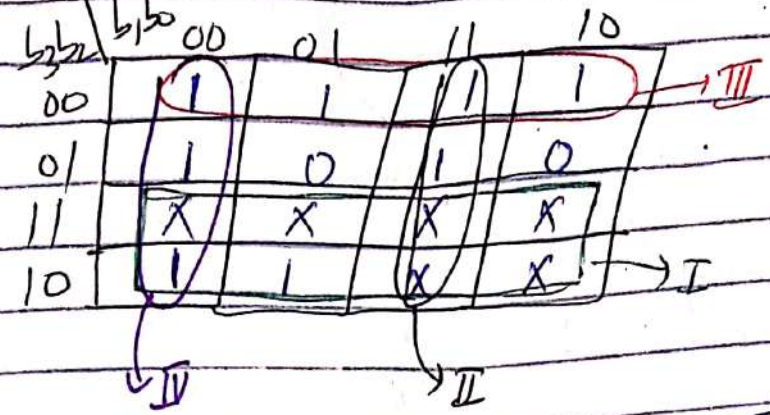
For (a)



$$a = b_3 + b_1 + b_2' b_0' + b_1 b_0$$

$$a = b_1 + b_3 + b_0 \odot b_2$$

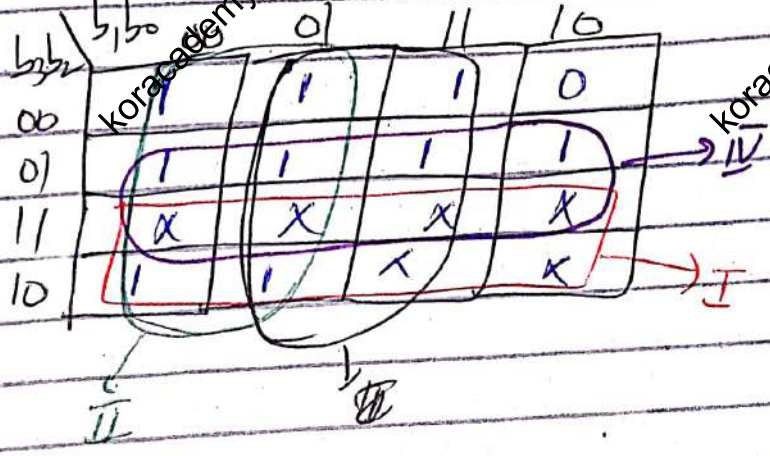
for b



exclusive NOR

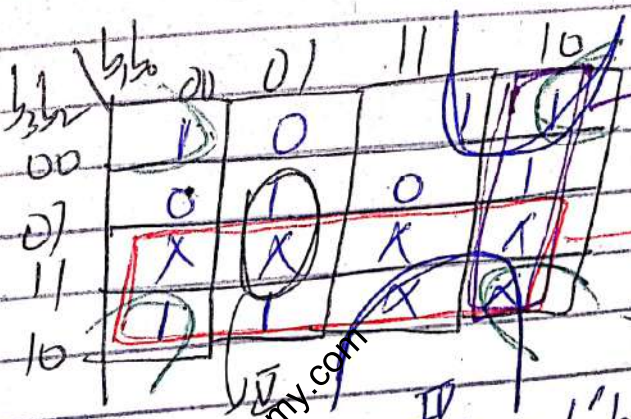
$$b = b_3 + b_1 b_0 + b_3 b_2' + b_1' b_0'$$

for c



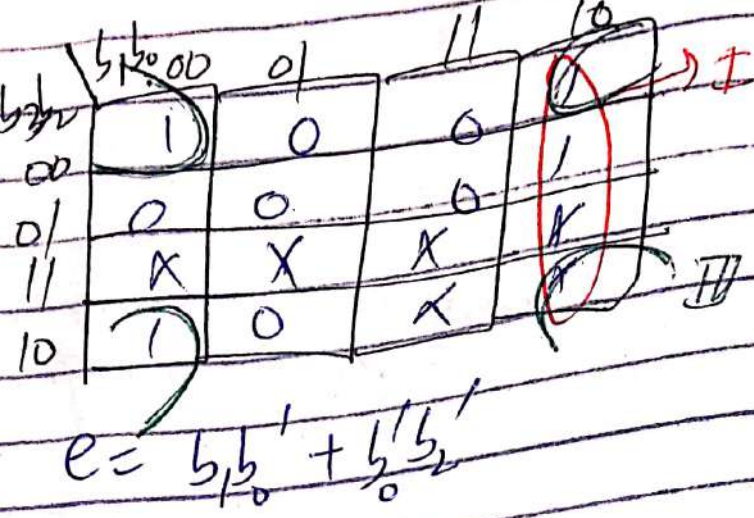
$$c = b_3 + b_2 + b_1' + b_0$$

for d



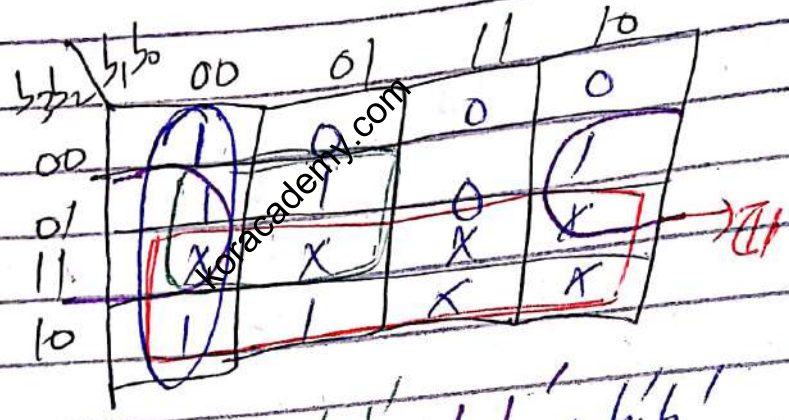
$$d = b_3 + b_1 b_0 + b_1' b_0' + b_2' b_1 + b_2 b_1' b_0$$

for e



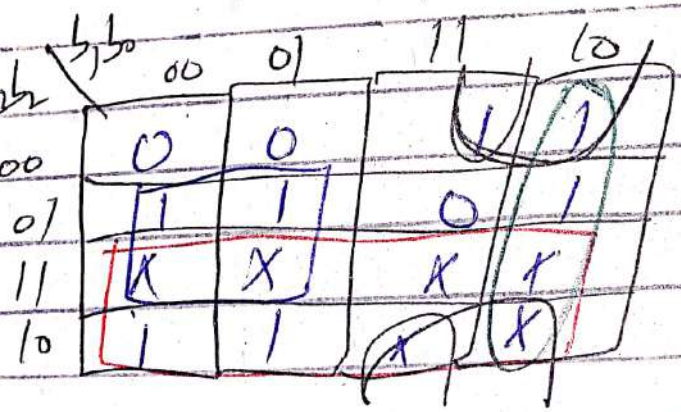
$e = b_1b_0' + b_0'b_2$

for f



$f = b_3 + b_2b_1' + b_2b_0' + b_1'b_0'$

for g

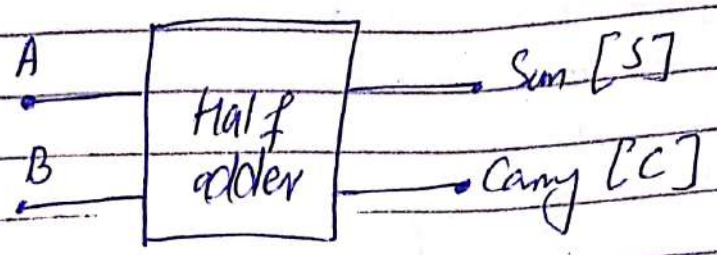


$g = b_3 + b_1b_0' + b_2b_1' + b_2'b_1$

Half Adder

Combinational ckt → output is only dependent on present input.

Sequential ckt → output depends on the present input as well as previous outputs.



→ Used to add single bit numbers.
→ It does not take carry from previous sum.

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

For Sum

A \ B	0	1
0	0	1
1	1	0

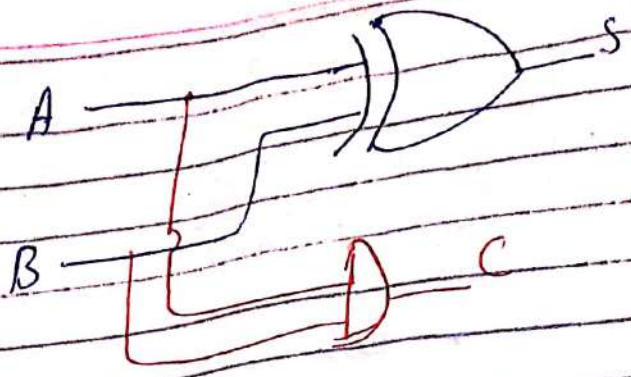
$A'B + AB' = A \oplus B$

Exclusive OR → odd no. 1 detector

Carry.

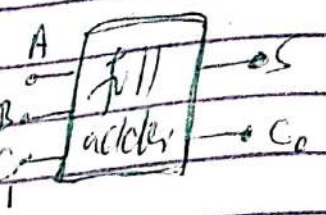
A \ B	0	1
0	0	0
1	0	1

$A \cdot B = Carry$



Full Adder

A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



KMAP $2^3 = 8$ cells

Sum →

A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Check board configuration

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

$S = A \oplus B \oplus C$

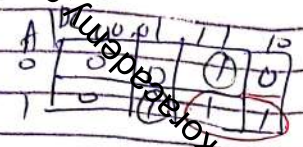
Carry

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_o = BC + AB + AC$

$C_o = AB + C(A \oplus B)$

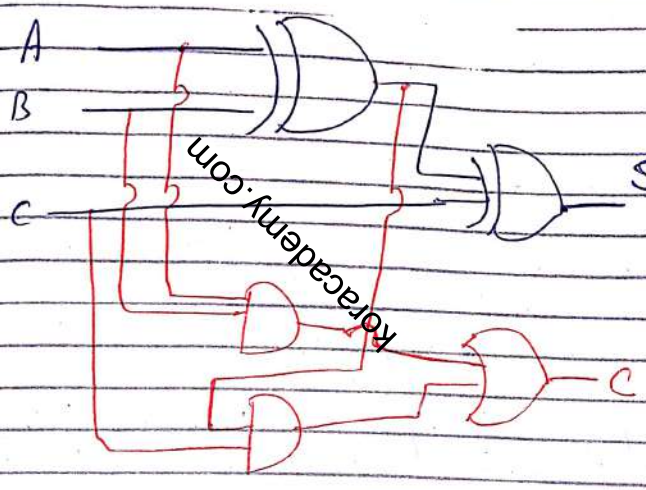
Method for carry



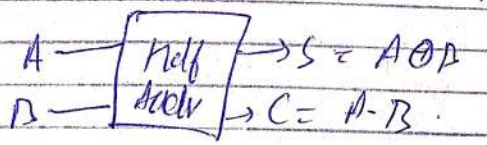
$C_o = AB + AB'C + A'BC$

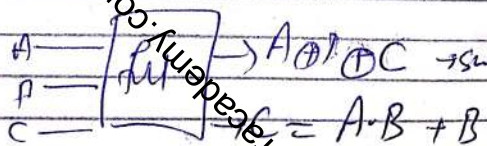
$C_o = AB + C(AB' + A'B)$

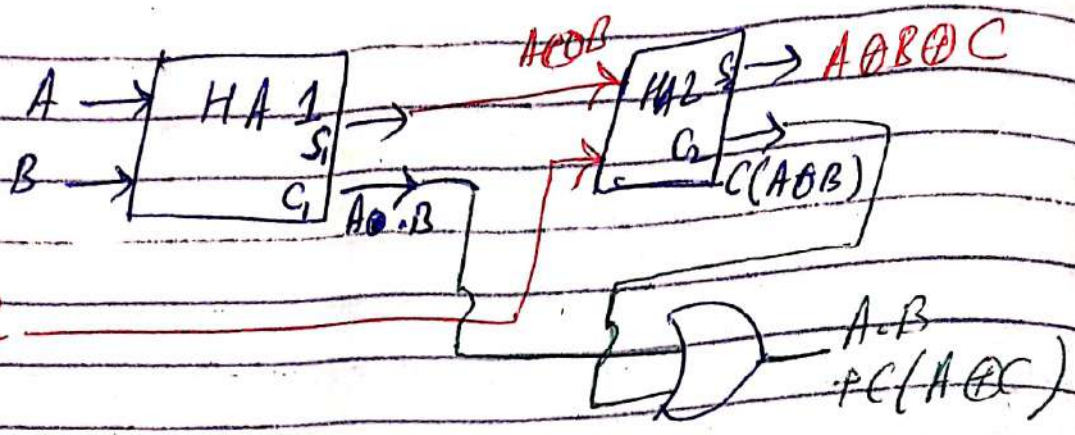
$C_o = AB + C(A \oplus B)$



Full Adder Using Half Adder







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

$$= A \cdot B + AB'C + A'BC$$

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

$A + A'B = A + B$

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

$$= AB + AC + A'BC$$

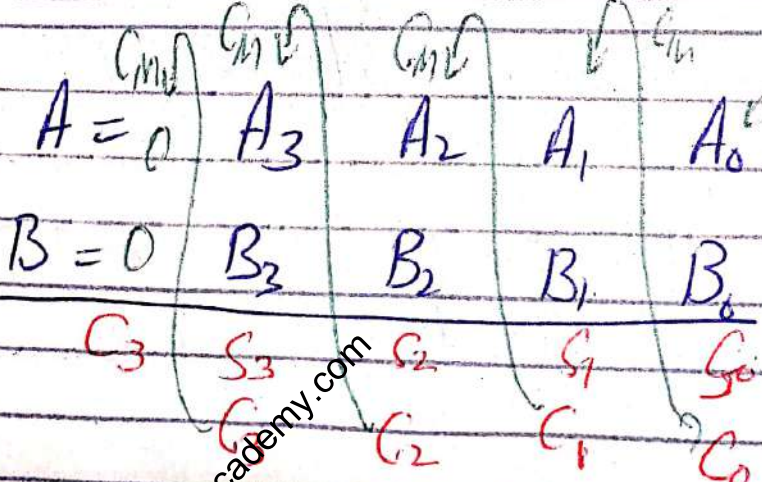
$$= AB + (A + A'B)C$$

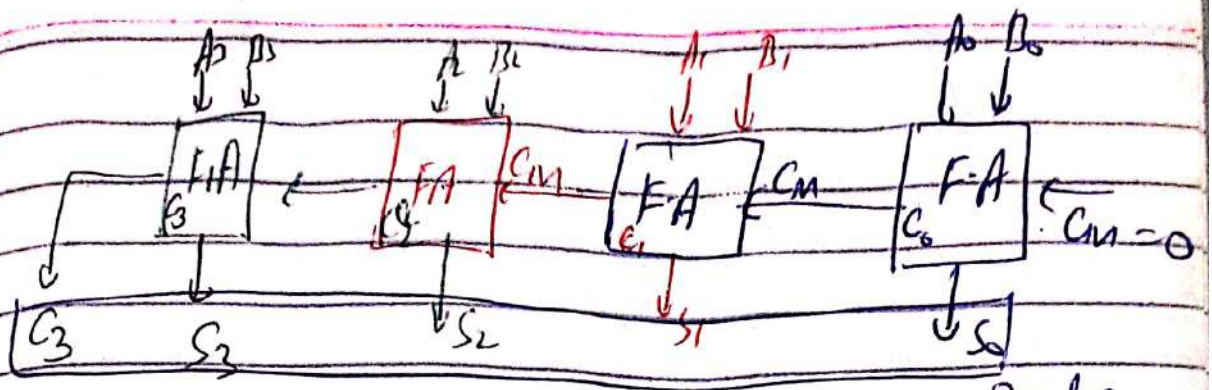
$$= AB + (A + B)C$$

$$= AB + AC + BC$$

Ans

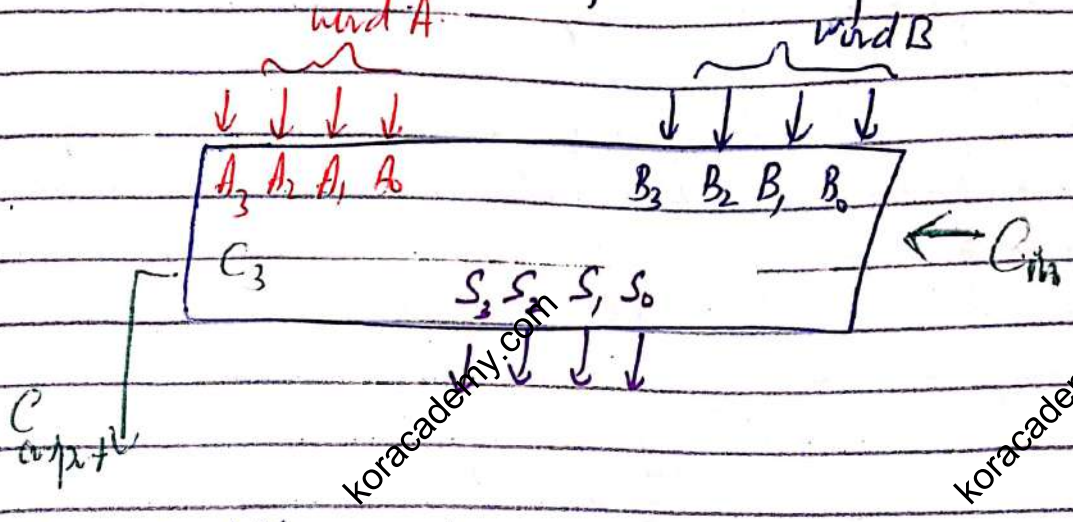
4 Bit Parallel Adder Using Full Adders





final sum.

74HC283 → IC for 4 bit full adder.

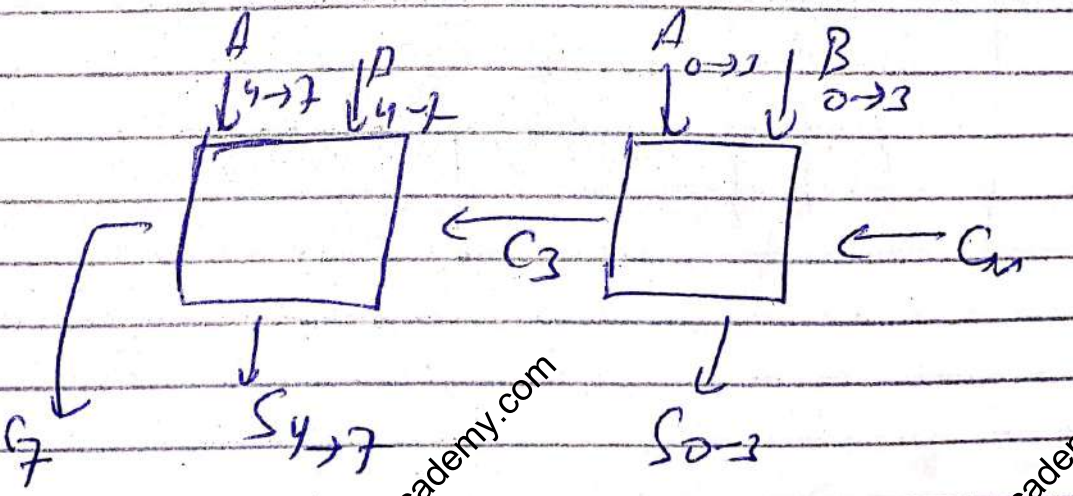


for 8 bit number
using two 4 bit full adders.

$A = A_7\ A_6 \cdots A_0$

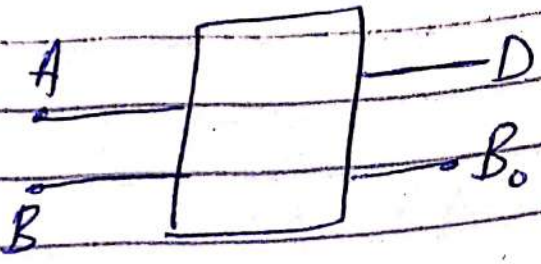
$B = B_7\ B_6 \cdots B_0$

$S = C_7\ S_7\ S_6 \cdots S_0$



Half Subtractor: Single bit numbers.

$$\begin{array}{r} 10 \\ -01 \\ \hline 01 \end{array}$$



A	B	D	B_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Diffe

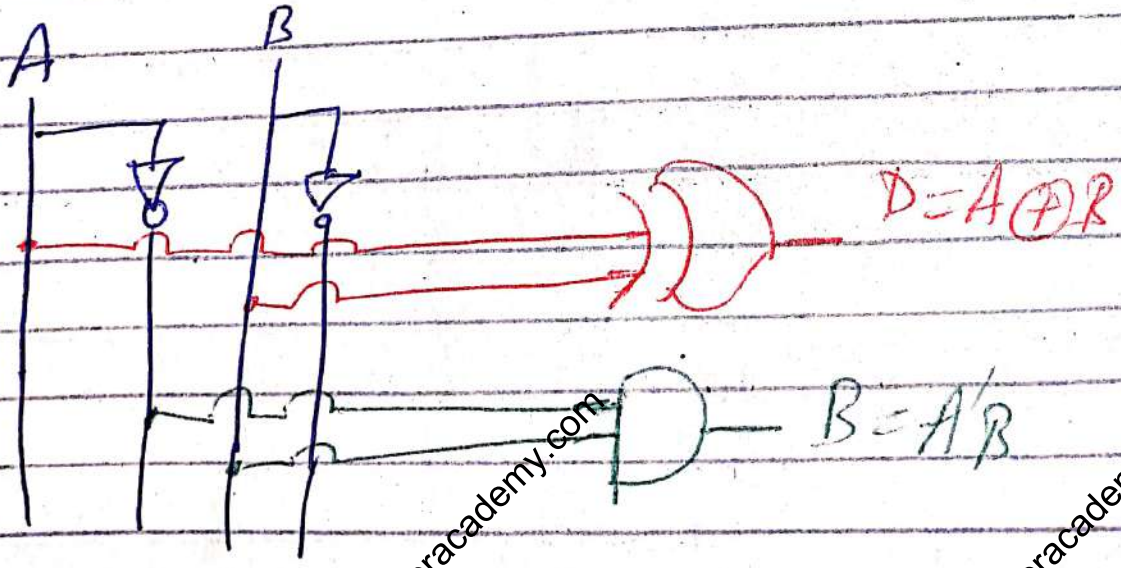
A \ B	0	1
0	0	1
1	1	0

$A'B + AB' = A \oplus B$

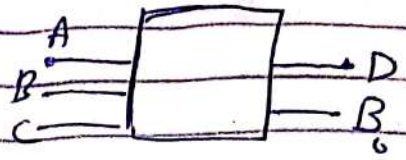
Borr

A \ B	0	1
0	0	1
1	0	0

$Bor = A'B$



Full Subtractor



A	B	C	D	B_0
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

(→ borrow from next stage)

1 − 1 = 0

0 − 1 = 1

Difference

A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

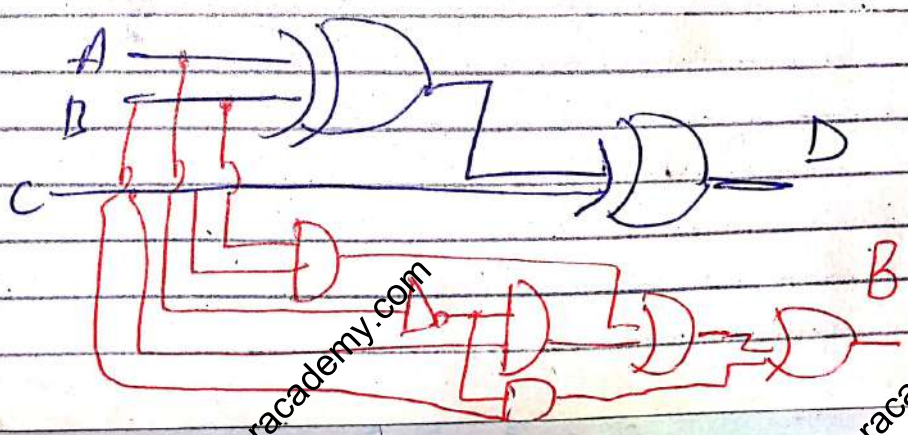
Check boarder configuration

$$D = A \oplus B \oplus C$$

Borrow

A \ BC	00	01	11	10
0	0	1	1	1
1	0	0	1	0

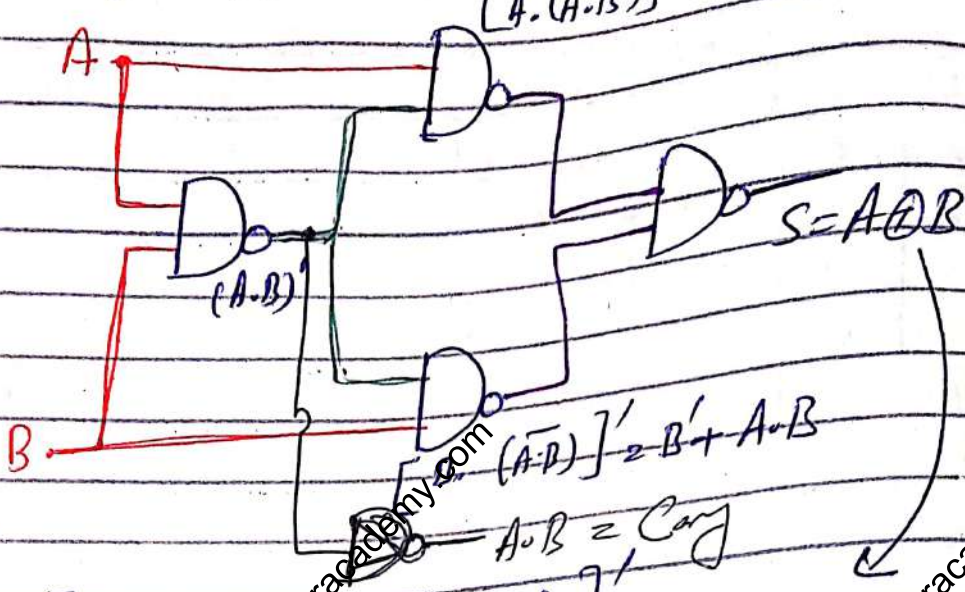
$$B_0 = BC + A'C + A'B$$



Half Adder Using NAND gate only

$S = A \oplus B$

$C_0 = A \cdot B$



$[(A' + A.B)(B' + A.B)]'$

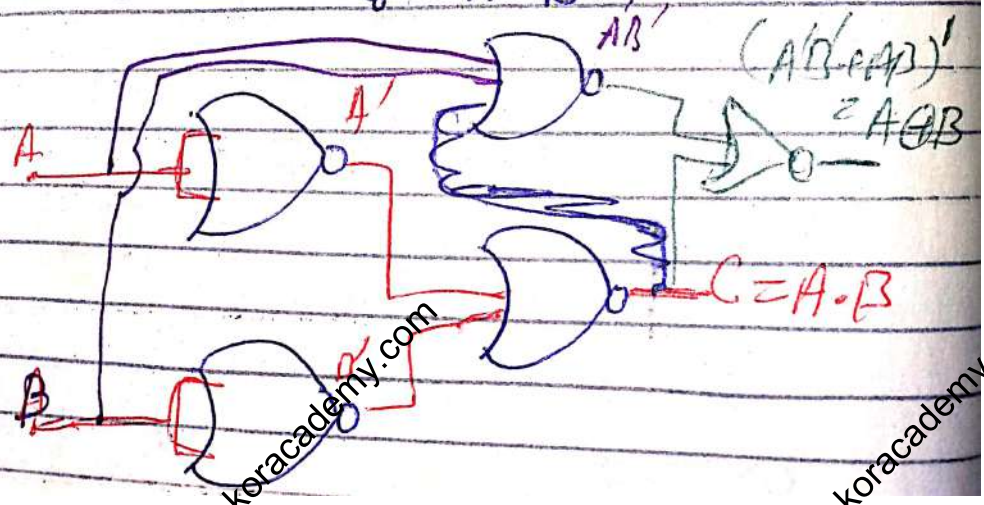
$(A'.B' + AB)'$

Distributive law $(A \odot B)'$ XNOR

$A \oplus B$ XOR

Half Using NOR gate

$S = A \oplus B$ $C_0 = A \cdot B$

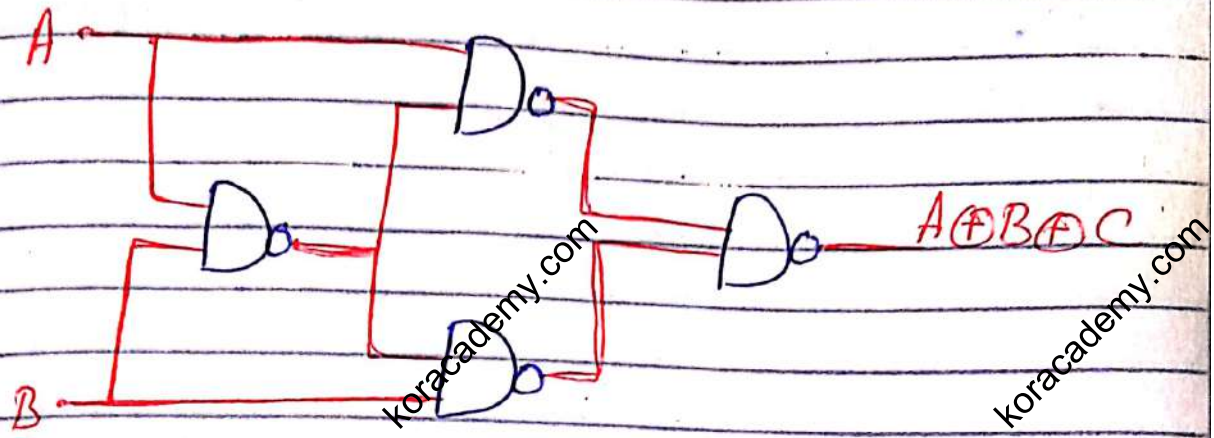


$S = A \oplus B = A'B + AB'$

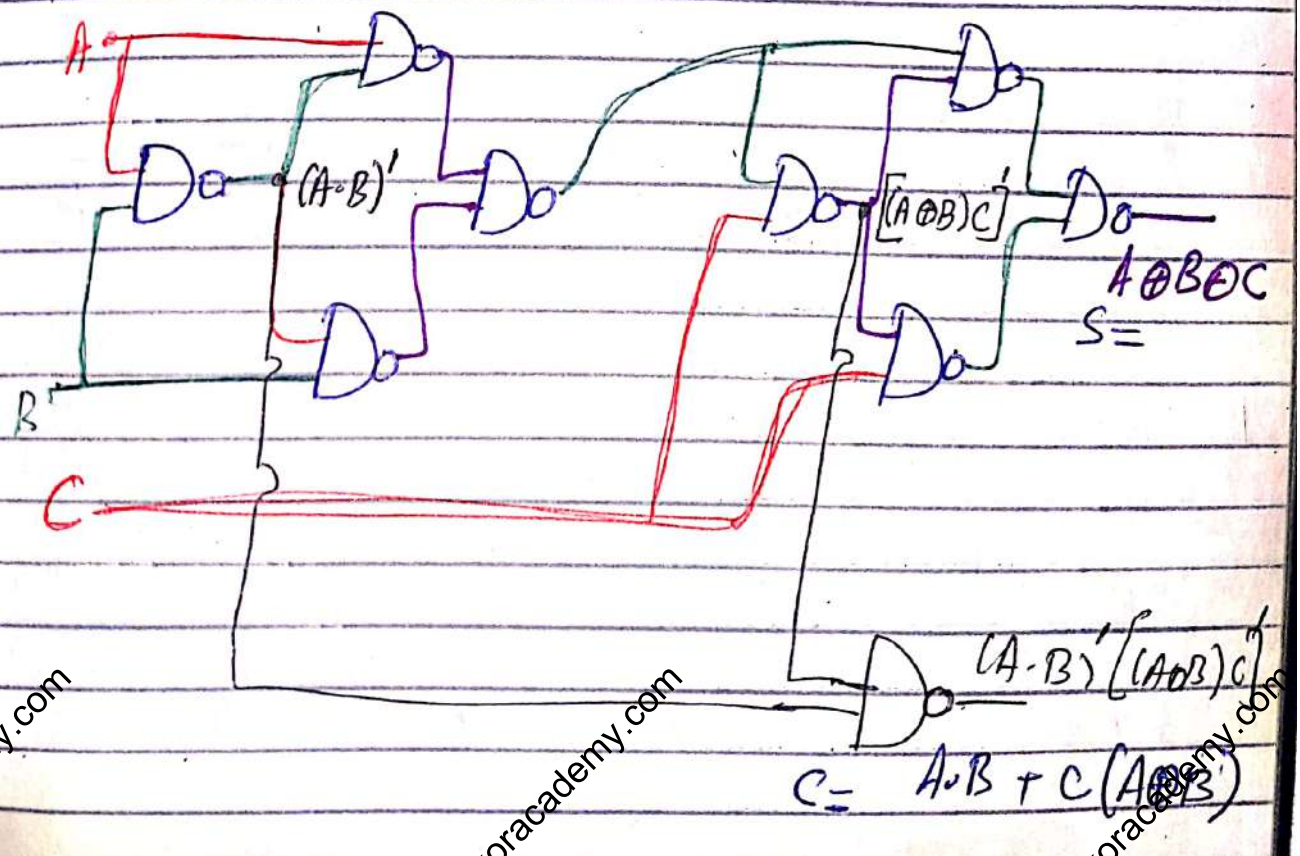
$= \overline{(A'B' + AB)}$

Full Adder using NAND gates

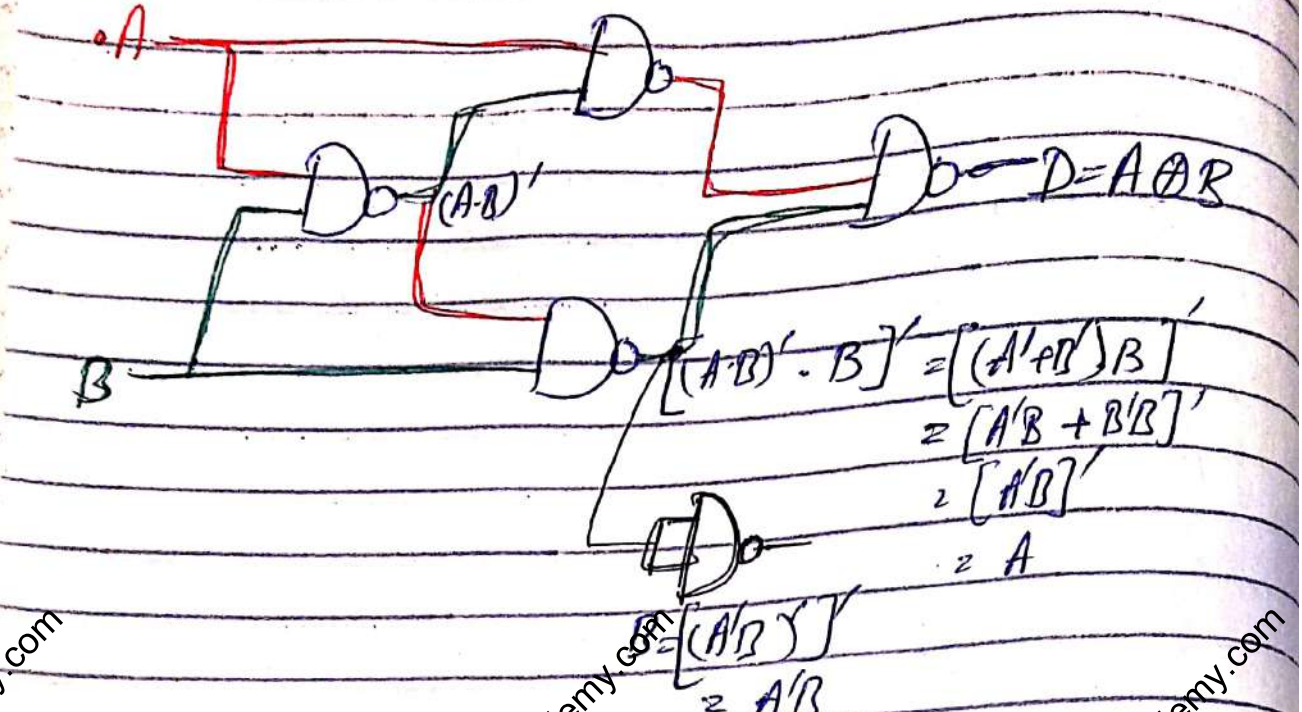
$S = A \oplus B \oplus C$ $\quad C_o = A \cdot B + C(A \oplus B)$



For $A \oplus B \oplus C$ ↓

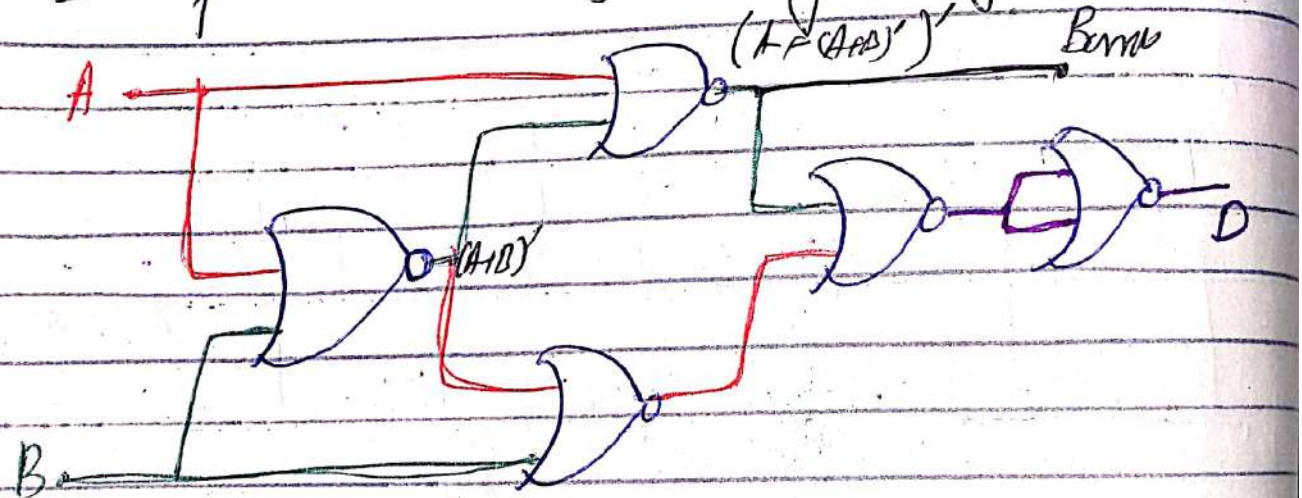


Half Subtractor using NAND gate only



D = A⊕B B = A'B.

Half Subtractor using NOR gate only

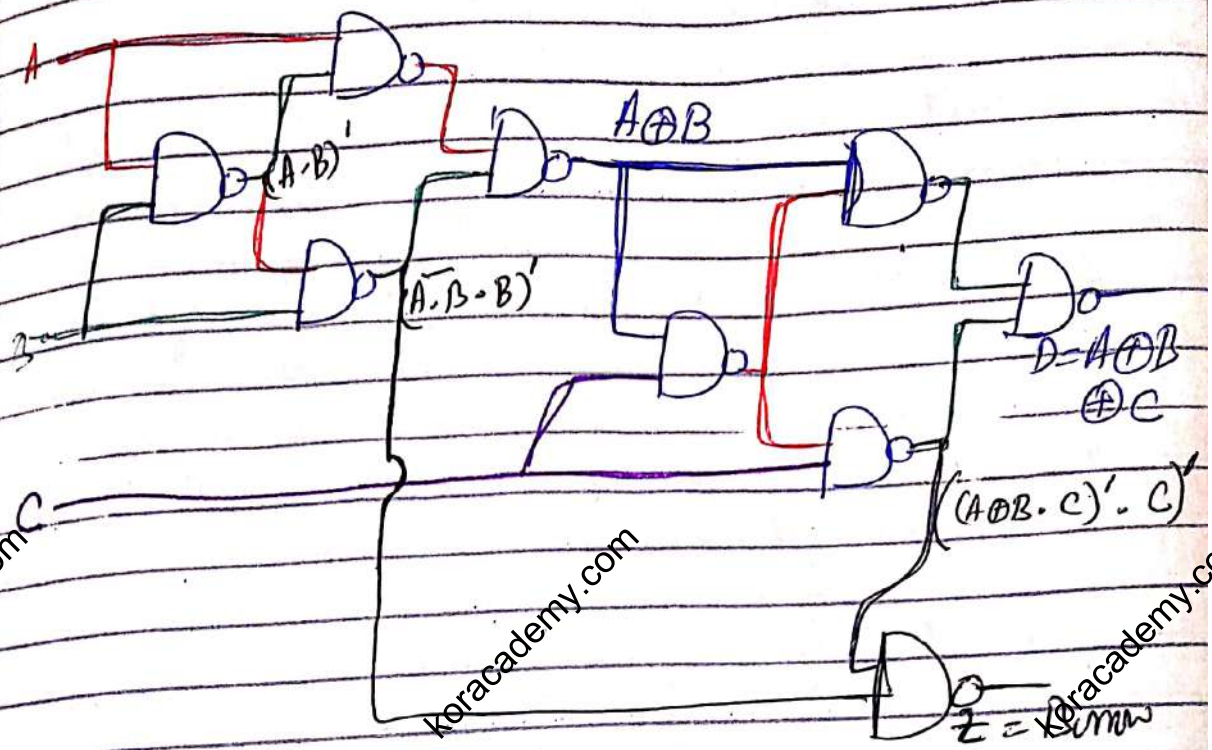


[A + (A+B)']' = [A + A'B]' = (A+B')' = A'B

Full Subtractor Using NAND gates Only

A → minuend B → subtrahend

C →



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

$= AB + B'$

$= A + B'$

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

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

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

$= (A\oplus B) + C$

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

$Z = (A+B')' + (A\oplus B+C)'$

$= A'\cdot B + \overline{A\oplus B}\cdot C$

XOR = XNOR

$= A'B + A\odot B\,C$

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

$= A'B + ABC + A'B'C \Rightarrow$ Borrow

speed → superior

Carry Look Ahead Adder

The time taken by logic gates to process input and give output is called propagation delay.

Propagation delay of sum and carry is different

We are more concerned about the propagation delay of carry

In CLA, we predict the carry before it is calculated.

A	B	C_{in}	C_0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

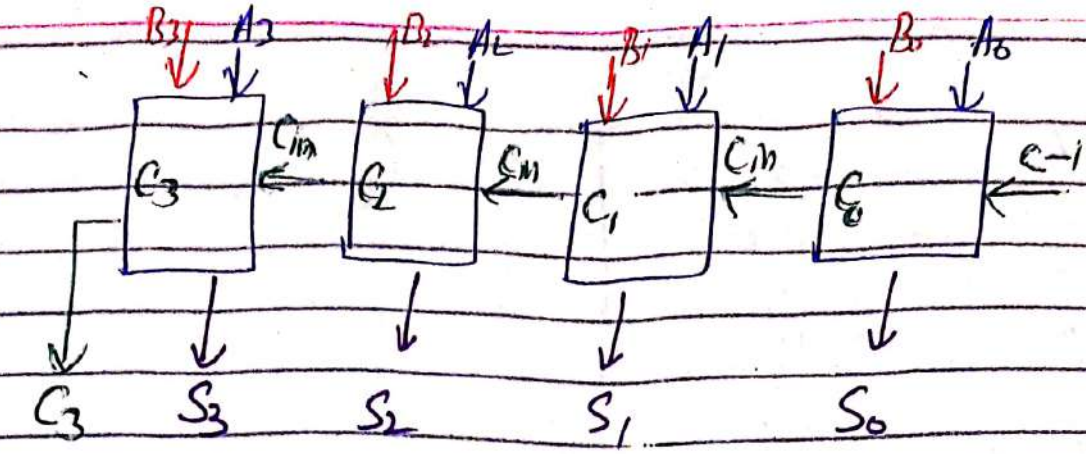
A⊕B = 1 and C_{in} = 1

A·B = 1

$$C_0 = \underbrace{A \cdot B}_{\text{carry generator} \rightarrow G} + \underbrace{(A \oplus B)}_{\text{carry propagator} \rightarrow P} C_{in}$$

$$C_0 = G + PC_{in}$$

↳ depends on C_{in}



$$C_i = G_i + P_i C_{i-1}$$

$$C_i = G_i + P_i C_{i-1}$$

If $i=0 \Rightarrow C_0 = G_0 + P_0 C_{-1} \rightarrow (1)$

$i=1 \Rightarrow C_1 = G_1 + P_1 C_0 \rightarrow (2)$

(1) in $(2) \Rightarrow C_1 = G_1 + P_1 (G_0 + P_0 C_{-1})$

$$C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$$

$i=2 \Rightarrow C_2 = G_2 + P_2 C_1$

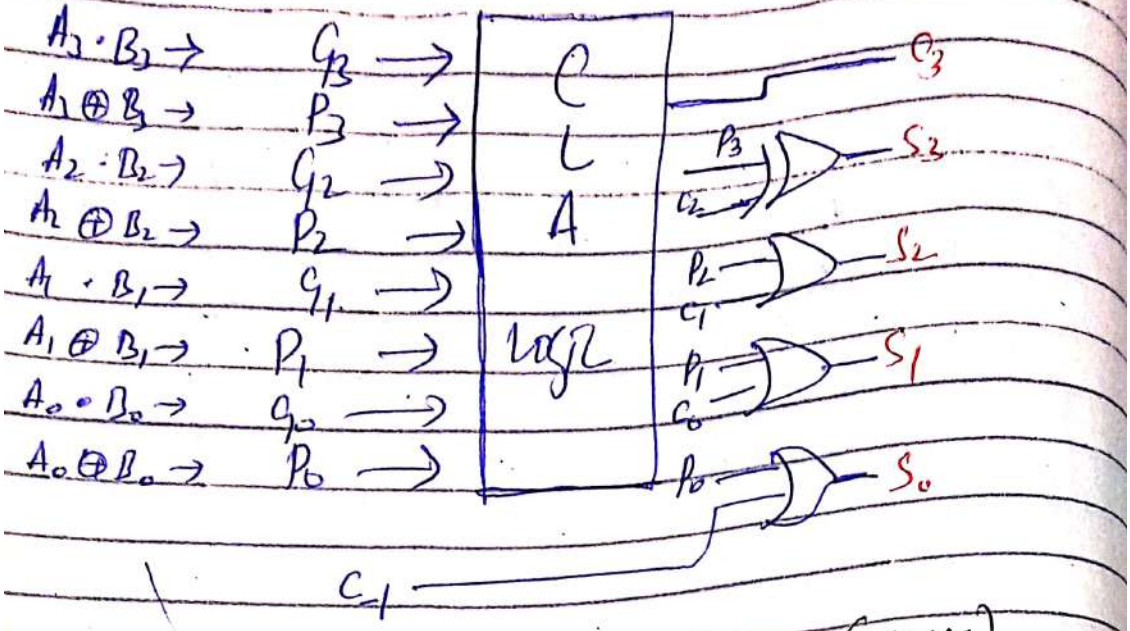
$C_2 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_{-1})$

$$C_2 = G_2 + P_2 G_1 + P_1 P_2 G_0 + P_0 P_1 P_2 C_{-1}$$

$i=3 \Rightarrow C_3 = G_3 + P_3 C_2$

$C_3 = G_3 + P_3 (G_2 + P_2 G_1 + P_1 P_2 G_0 + P_2 P_1 P_0 C_{-1})$

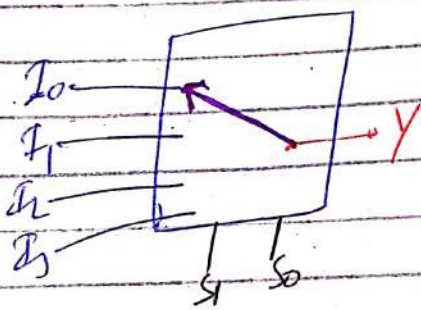
$$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1}$$



Introduction to Multiplexer (MUX)

It is a combinational circuit that selects binary information from one of many input lines and directs it to output line.

It is simply a data selector

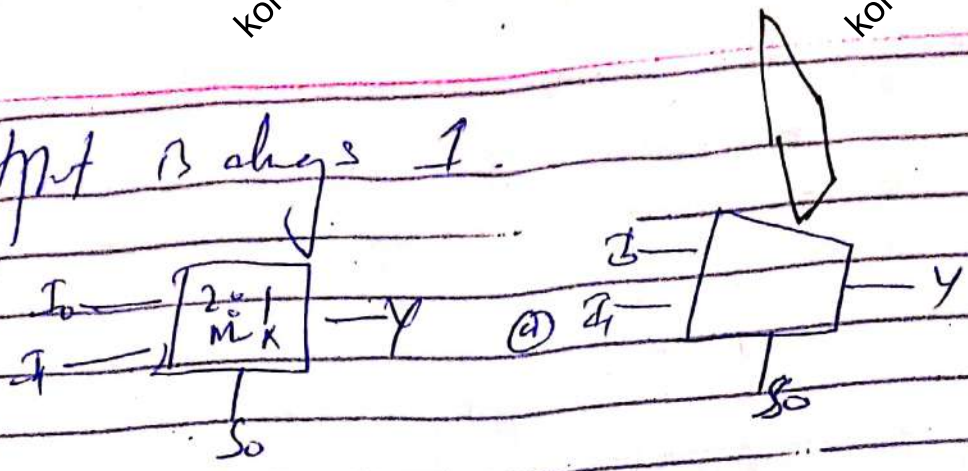


Select line / selective variables

$S_1 \rightarrow$ MSB $\quad S_0 \rightarrow$ LSB

S_1	S_0	
0	0	I_0 will be selected.
0	1	I_1 will be selected
1	0	I_2 will be selected for output.
1	1	I_3

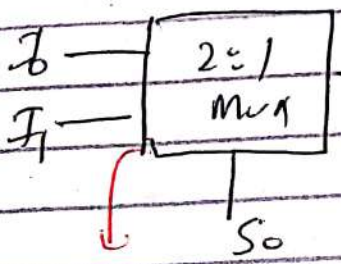
No. of input is always 1.



n → number of inputs m → no. of select lines

$n = 2^m$ (or) $m = \log_2 n$

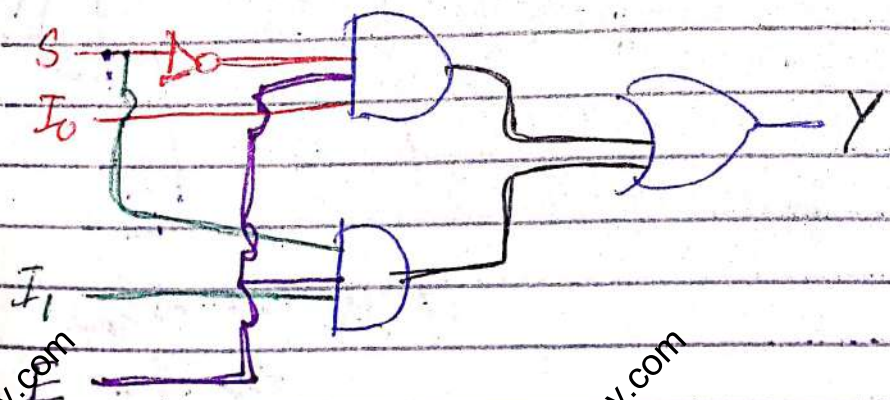
2:1 MUX:



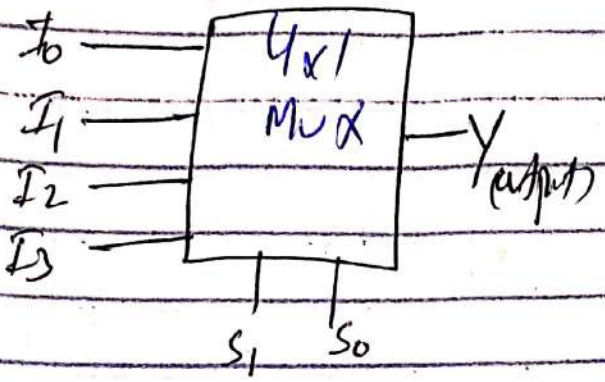
E	S	Y
0	X	0
1	0	I_0
1	1	I_1

$Y = E.S'.I_0 + E.S.I_1$

$Y = E(S'.I_0 + S.I_1)$



4 x 1 Multiplexer



$n = 4$

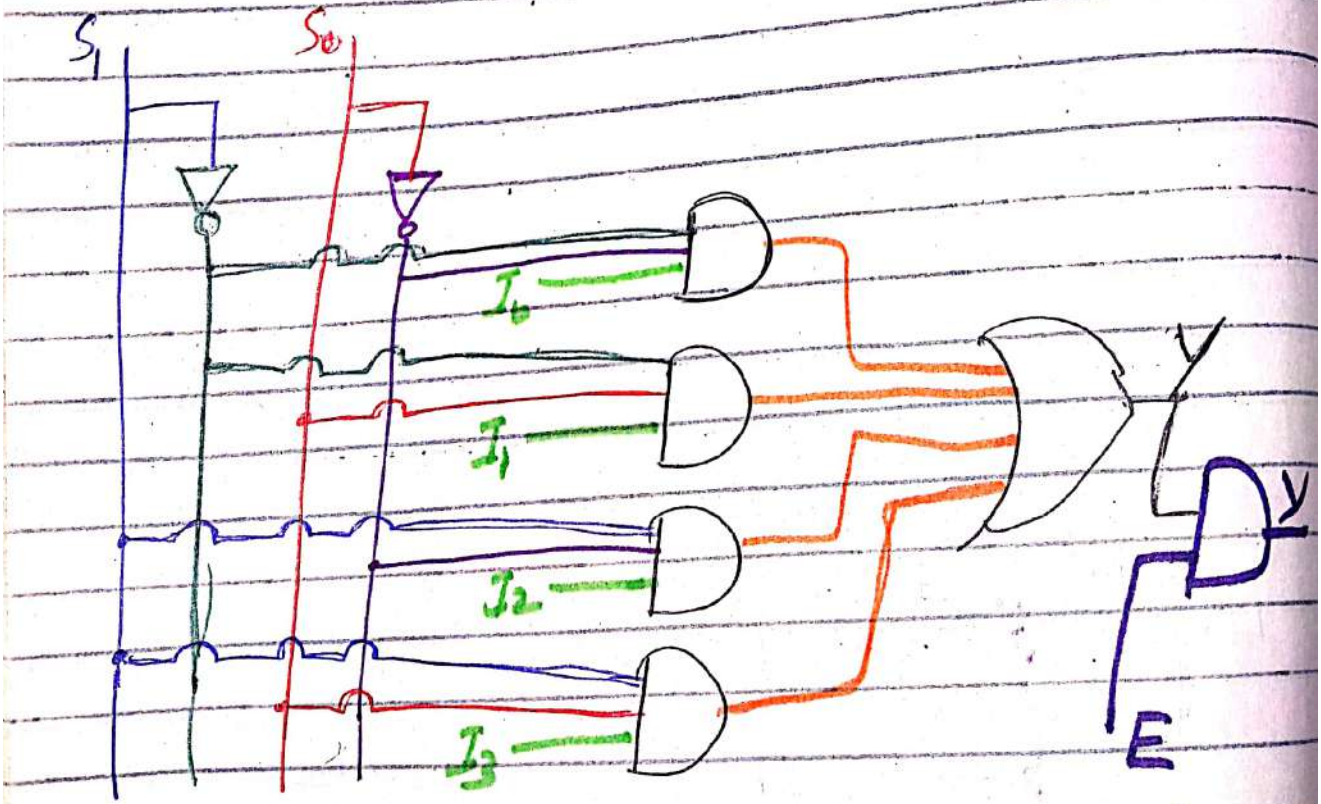
$m = \log_2 n$

$m = \log_2 4 = \log_2 2^2$

$\Rightarrow \log_2 2^2 = 2$

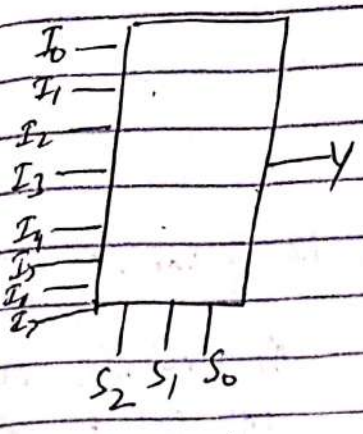
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$



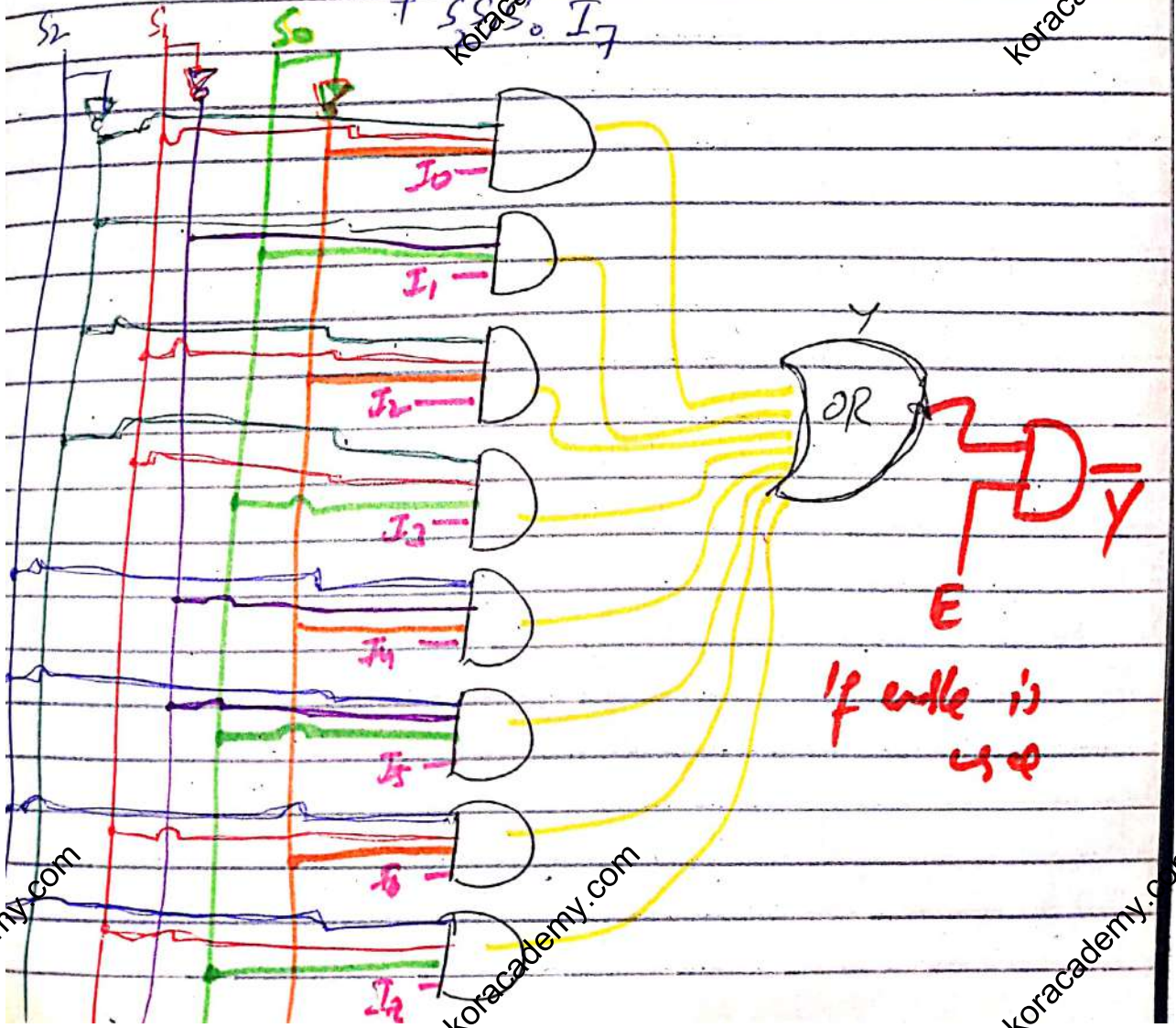
If you want to have enable

8 x 1 Multiplexer



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

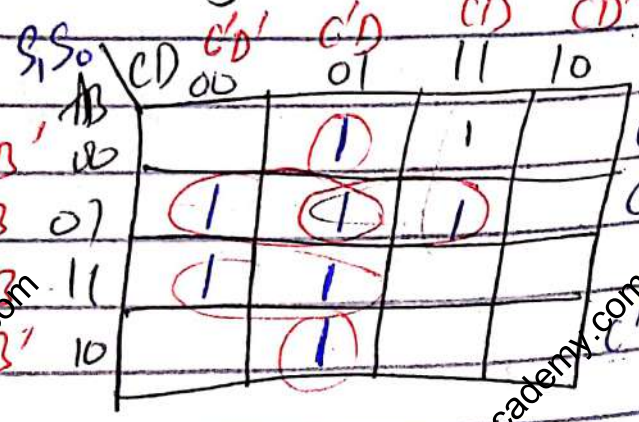
$$Y = S_2'S_1'S_0'I_0 + S_2'S_1'S_0I_1 + S_2'S_1S_0'I_2 + S_2'S_1S_0I_3 + S_2S_1'S_0'I_4 + S_2S_1'S_0I_5 + S_2S_1S_0'I_6 + S_2S_1S_0I_7$$

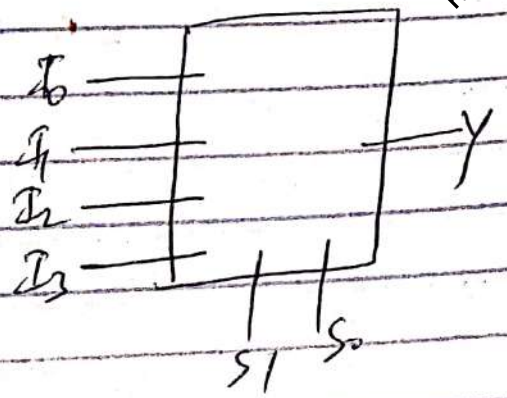


Implementation of Boolean Function using multiplexer

eg Implement $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,13)$ using 4X1 MUX.

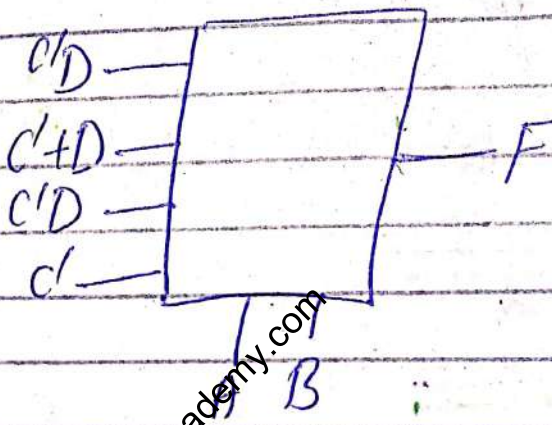
(i) No of variables 16 cell kmap
(ii) 4:1 MUX
(iii) select selective variables





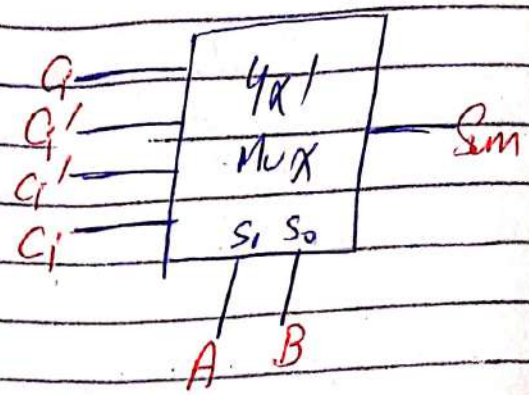
S_1	S_0	Y
0	0	$I_0 = C'D$
0	1	$I_1 = C'+D$
1	0	$I_2 = C'D$
1	1	$I_3 = C'$

let $S_0 = B$
$S_1 = A$

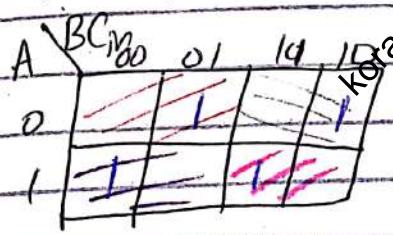


1 Bit Full Adder Using Multiplexer

A	B	C_{in}	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



IC MAP for Sum



$S_0 = B$

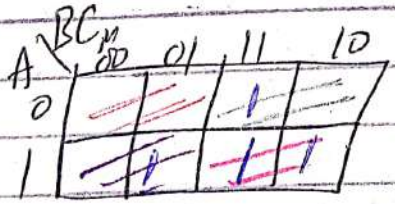
$S_1 = A$

Input only for C

→ A and B are select lines

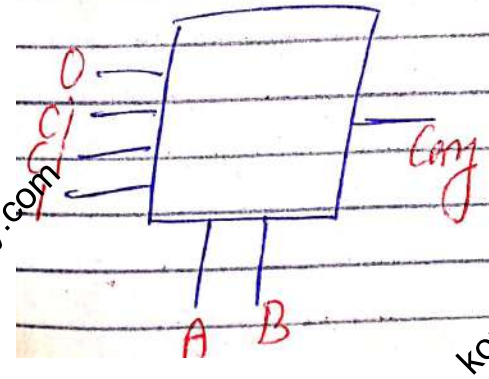
A	B	S
0	0	$I_0 = C_i$
0	1	$I_1 = C_i'$
1	0	$I_2 = C_i'$
1	1	$I_3 = C_i$

K MAP for Carry



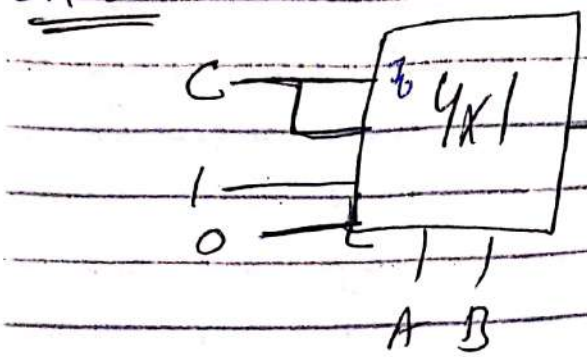
A	B	S
0	0	$I_0 = 0$
0	1	$I_1 = C_i$
1	0	$I_2 = C_i$
1	1	$I_3 = 1$

$C + C' = 1$



Logical Expression from MUX:

Ex 1



C	Y
0	0
0	1
1	0
1	1

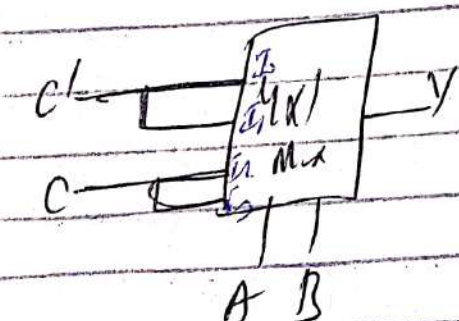
$Y_1 = A'B' \cdot C + A'B \cdot C + AB'(1) + AB(0)$

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

$B' + B = 1$

$\boxed{Y = A'C + AB'}$

Ex 2



A	B	Y
0	0	$I_0 = C'$
0	1	$I_1 = C'$
1	0	$I_2 = C$
1	1	$I_3 = C$

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

$Y = AC'(B + B') + AC(D + AB')$

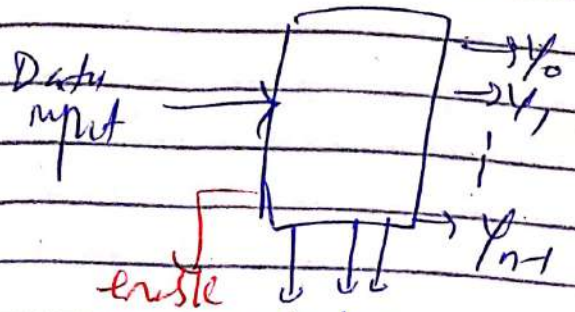
$Y = A'C' + AC$

$\boxed{Y = A \odot C}$

Introduction to Demultiplexer: (DEMUX)

→ one input and many output.
→ Reverse operation of multiplexer.
→ Use to many circuit as data distributer.

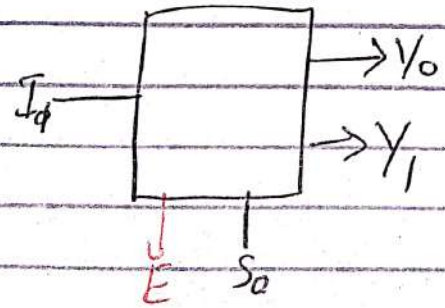
enable → 0 → all 0



n → output lines m → select lines.

$n = 2^m$ ⇔ $m = \log_2 n$

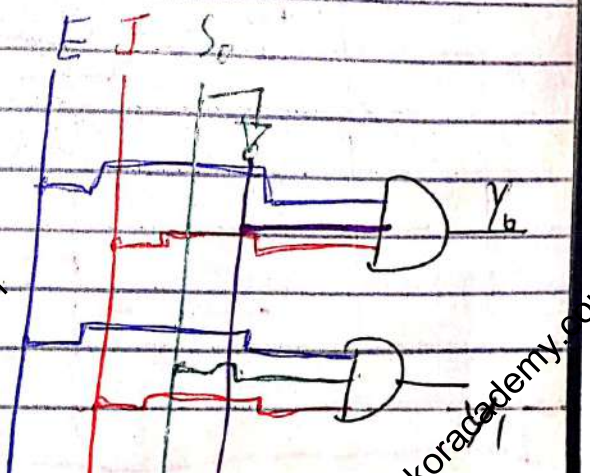
1 x 2 DeMux



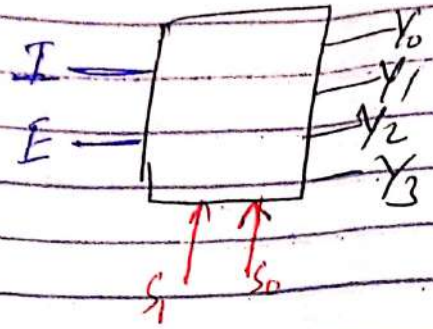
E	S_0	Y_0	Y_1
0	0	0	0
0	1	0	0
1	0	I	0
1	1	0	I

$Y_0 = ES_0'I$

$Y_1 = ES_0I$



1:4 DeMux



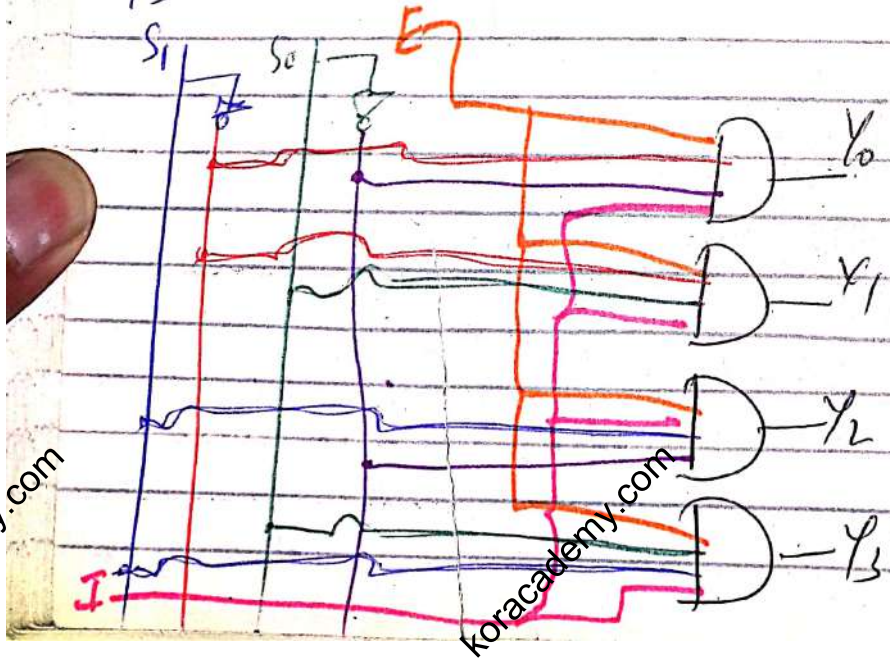
E	S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	I	0	0	0
1	0	1	0	I	0	0
1	1	0	0	0	I	0
1	1	1	0	0	0	I

$Y_0 = E S_1' S_0' I$

$Y_1 = E S_1' S_0 I$

$Y_2 = E S_1 S_0' I$

$Y_3 = E S_1 S_0 I$



Full Subtractor Using 1:8 DeMux

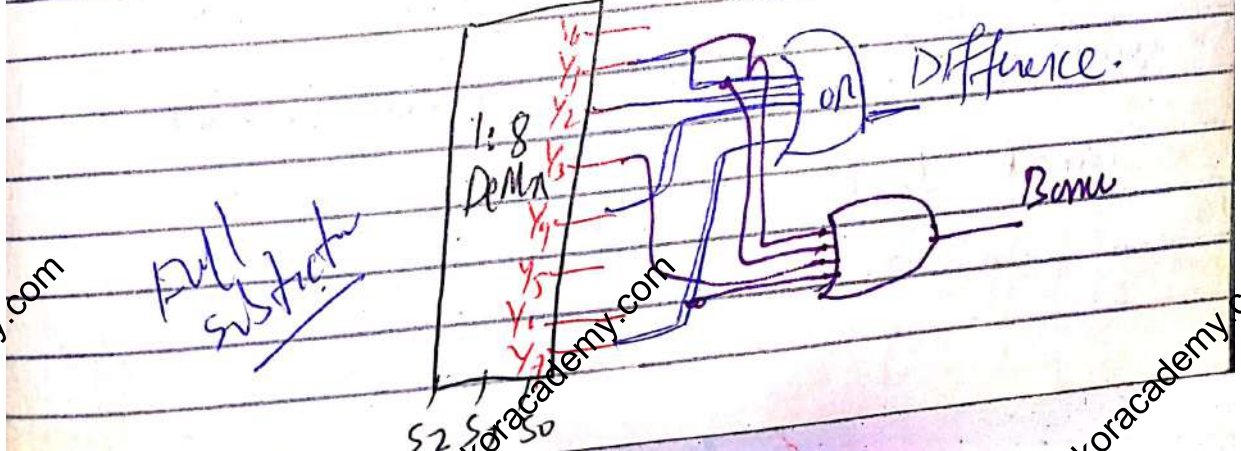
	A	B	B_i	D	B_o
M_0	0	0	0	0	0
M_1	0	0	1	1	1
M_2	0	1	0	1	1
M_3	0	1	1	0	1
M_4	1	0	0	1	0
M_5	1	0	1	0	0
M_6	1	1	0	0	0
M_7	1	1	1	1	1

$$D = (A, B, B_i) = \Sigma m(1, 2, 4, 7)$$

$$B_0 (A, B, B_i) = \Sigma m(1, 2, 3, 7)$$

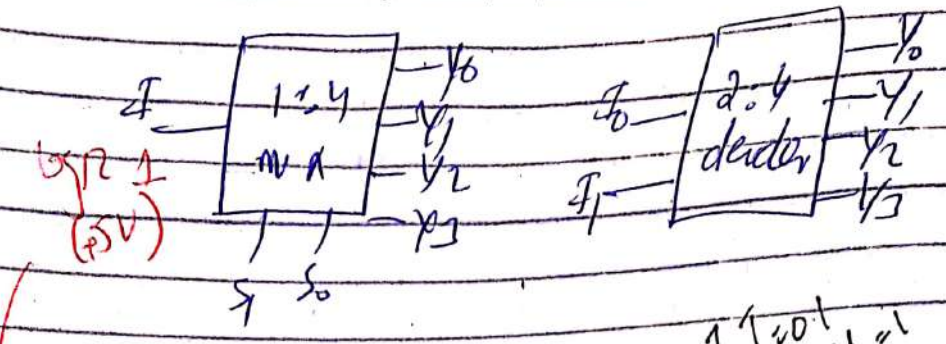
A B B_i

S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



Demultiplexer As Decoder

2:4 decoder from 1:4 DeMux



For decoder $S_1S_0 = 10$ $Y_2 = I$

$I_0 I_1 = 01$ $Y_1 = 1$

I_1	I_0	Y_0	Y_1	Y_2	Y_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

When I is united to 5V supply, the demux operates as decoder

For 3:8 Decoder

$S_0 = I_0$

$I = 1$ $S_1 = I_1$

$S_2 = I_2$

selection lines → no of outputs, selection variables

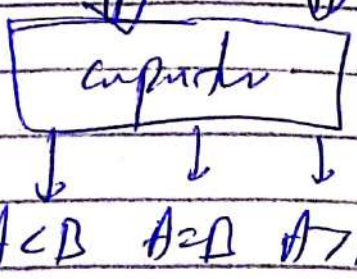
connect I to logic 1.

$S_0 = I_0$ $S_1 = I_1$

2. Bit Comparator

→ A digital comparator is a combinational circuit designed to compare two n-bit binary words.

→ Comparator has three outputs.

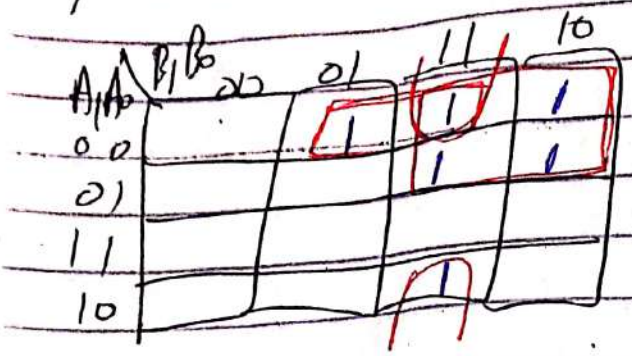


$A = A_1 A_0$

$B = B_1 B_0$

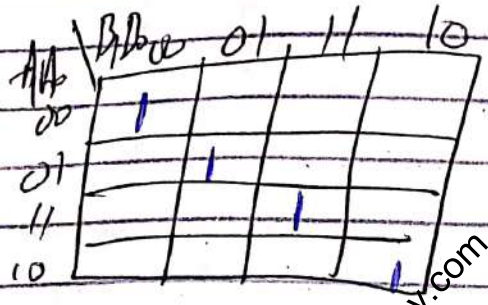
A_1	A_0	B_1	B_0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

for $A<B$



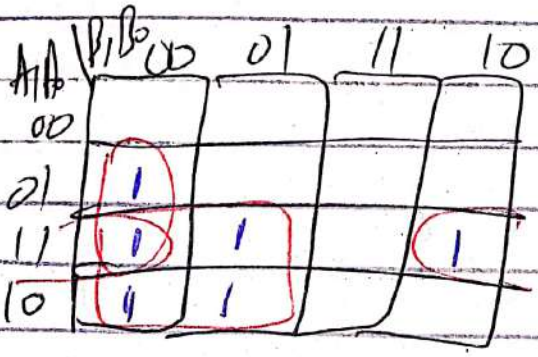
$A<B =$
$A_1' A_0' B_0 + A_1' B_1 + A_0' B_1 B_0$

for $A=B$



$A=B$
$(A_1 \odot B_1)(A_0 \odot B_0)$

for $A>B$-



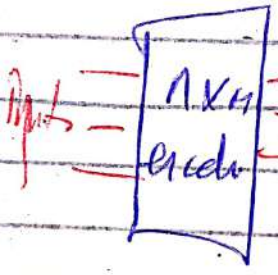
$A>B =$
$A_0 B_1' B_0' + A_1 A_0 B_0' + A_1 B_1'$

Introduction to Encoders and Decoders

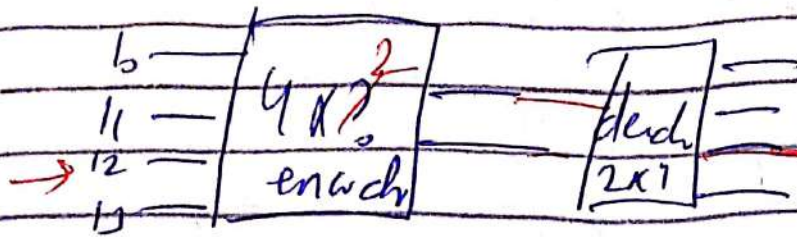
→ Combinational circuits

→ n inputs → m outputs

→ Function of decoder is opposite of encoder



MSI → Median scale integrated



encoder is used to minimize the no. of data lines.

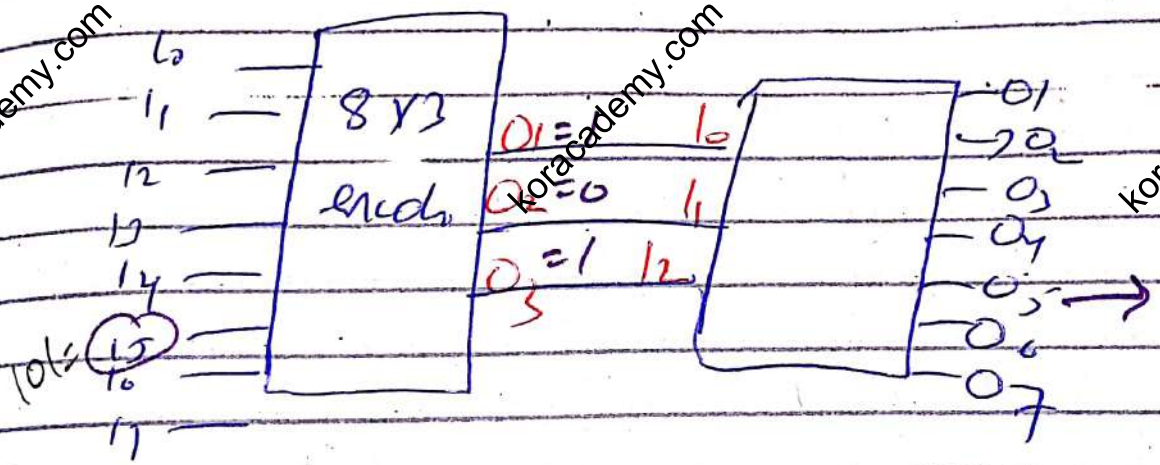
4 inputs → 2 bits

$n = 2^m$ encoders

$m = 2^n$ decoders

implementation of boolean functions

eg



eg $I_5 = 101$

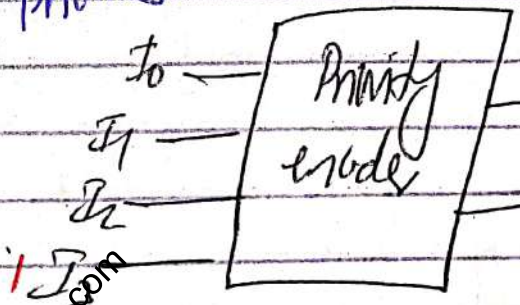
only one input is high here.

Priority Encoder

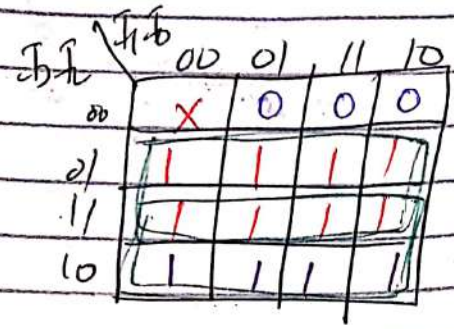
priority

assume I_7 → least priority

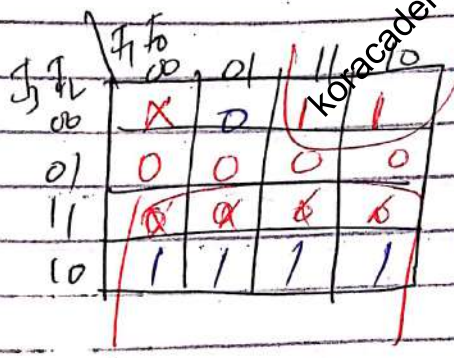
I_0 → highest priority



I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1



$$Y_1 = I_2 + I_3$$

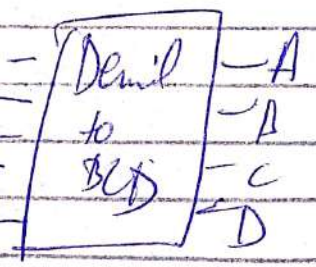


$$Y_0 = I_3 + I_1 I_2'$$

Decimal to BCD Encoder

Input	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

output

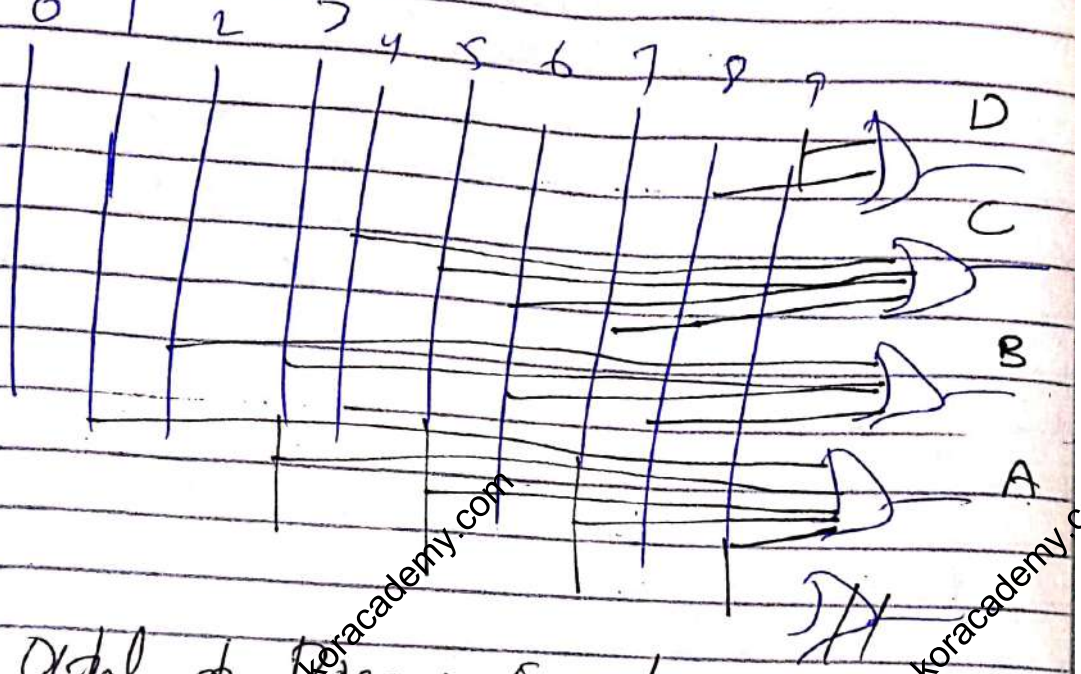


$D = 8 + 9$

$C = 4 + 5 + 6 + 7$

$B = 2 + 3 + 6 + 7$

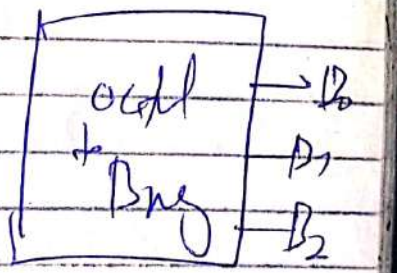
$A = 1 + 3 + 5 + 7 + 9$



Octal to Binary Encoder

8 input lines → 3 output lines.

Input	B_2	B_1	B_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7			



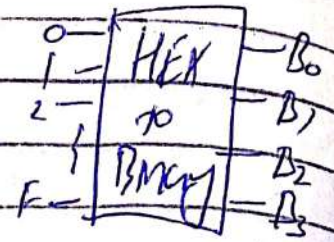
$B_2 = 4 + 5 + 6 + 7$

$B_1 = 2 + 3 + 6 + 7$

$B_0 = 1 + 3 + 5 + 7$

Hexadecimal to Binary Encoder

I/P H	B_3	B_2	B_1	B_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1



$N = 16 \rightarrow$ inputs

$16 = 2^M$ | $(2)^4 = (2)^M$

$\Rightarrow M = 4 \rightarrow$ outputs

$B_0 = 1 + 3 + 5 + 7 + 9 + B + D + F$

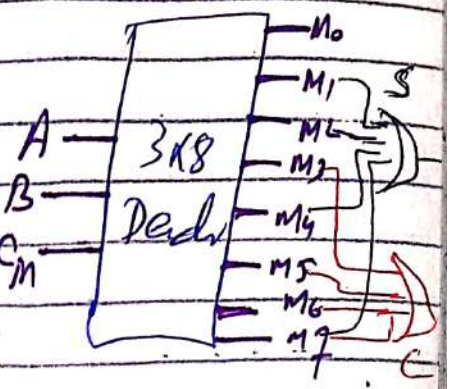
$B_1 = 2 + 3 + 6 + 7 + A + B + E + F$

$B_2 = 4 + 5 + 6 + 7 + C + D + E + F$

$B_3 = 8 + 9 + A + B + C + D + E + F$

Full Adder Implementation Using Decoder

A	B	C_M	S	C_0	
0	0	0	0	0	m_0
0	0	1	1	0	m_1
0	1	0	1	0	m_2
0	1	1	0	1	m_3
1	0	0	1	0	m_4
1	0	1	0	1	m_5
1	1	0	0	1	m_6
1	1	1	1	1	m_7



Advantage:
Cost IC

$F_S = \Sigma(m_1, m_2, m_4, m_7)$

$F_C = \Sigma(m_3, m_5, m_6, m_7)$

Octal Addition

Use → 8.

$A_1\ A_0$
$+\ B_1\ B_0$
C_0

eg $\begin{array}{r} 4 \\ 5 \\ \hline 9 \end{array}$ → 7

$C_0 = 1 \times 8 + 1$

$S = 1$

$C = 1$

If $C_0 \leq 7$
$Sum = C_0$
$Carry = 0$

If $C_0 > 7$
$C_0 = C \times 8 + S$
Carry Use Sum.

Carry always 1.

7
+7
14

$C_0 = 1 \times 8 + 6$

$C = 1 \quad S = 6$

Ex

$$\begin{array}{r} 2^{0}\ 4^{0}\ 3 \\ 2\ 1\ 3 \\ \hline 4\ 5\ 5 \end{array}$$ Ans

Ex

$$\begin{array}{r} {}^{1}5\ {}^{1}6\ 7 \\ 2\ 4\ 3 \\ \hline 1\ 0\ 3\ 2 \end{array}$$

$S = 1032$ Ans

$10 \rightarrow 7$

$4 \quad 1 \times 8 + 2$

$11 > 7$

$1 \times 8 + 3$

$877 \quad 1 \times 8 + 0$

Octal subtraction

$8 + 3 = 11$

$$\begin{array}{r} 7\ 4\ 3 \\ -\ 5\ 6\ 4 \\ \hline 1\ 5\ 7 \end{array}$$

$8 + 4 = 12$

$$\begin{array}{r} 6\ 2\ 1\ 4 \\ -\ 2\ 6\ 5 \\ \hline 3\ 3\ 7 \end{array}$$ Ans

Octal multiplication

$$\begin{array}{r} 7\ 2 \\ \times\ 1\ 2 \\ \hline 1\ 6\ 4 \\ 7\ 2\ X \\ \hline 1\ 1\ 0\ 4 \end{array}$$

$14 > 7$

$1 \times 8 + 6$

877

$1 \times 8 + 0$

$$\begin{array}{r} 3\;5\;7 \\ \times\quad 2\;3 \\ \hline 1\;3\;1\;5 \\ 7\;3\;6\;\times \\ \hline 1\;0\;6\;7\;5 \end{array}$$

$21 > 7$

$2\times8+5$

$17 = 2\times8+1$

$14>7 = 1\times8+6$

$11>7 = 1\times8+3$

$8>7 = 1\times8+0$

Hexadecimal addition

base (r) → 16

0 → 16−1

= 0 − 15

Ex

$$\begin{array}{r} 5\;6\;8\;9 \\ +\;4\;5\;7\;4 \\ \hline 9\;B\;F\;D \end{array}$$

Ex

$$\begin{array}{r} A\;D\;D \\ +\;D\;A\;D \\ \hline 1\;8\;8\;A \end{array}$$

$26 = 1\times16 + x$

$26 = 1\times16+10$

$24 = 1\times16+8$

Hexadecimal subtraction

$$\begin{array}{r} 9\;6\;5\;4 \\ -\;5\;3\;2\;1 \\ \hline 4\;3\;3\;3 \end{array}$$

16 16 16+11

$$\begin{array}{r} 9\;7\;4\;B \\ -\;5\;8\;7\;C \\ \hline 3\;E\;C\;F \end{array}$$

$3\;4\times16^1$ $\quad B\times16^0$

1×16^1

16×16^0

$16+13 = 27$

$\times16^0$

Hexadecimal multiplication

```
  ① 9 4
x   1 2
-------
  1 2 8
  9 4x
-------
  A 6 8
```

$18 = 1 \times 16 + 2$

```
      ①   ①
  ④ ① A ② B ② C
x       2   3
-------------
    2 0 3 4
  1 5 7 8 x
-------------
  1 7 7 B 4
```

$36 = 1 \times 16 + 18 > 16$

$36 = 2 \times 16 + 4$

$35 = 2 \times 16 + 3$

$32 = 2 \times 16 + 0$

$24 = 1 \times 16 + 8$

$23 = 1 \times 16 + 7$

$21 = 1 \times 16 + 5$

Hexadecimal Division

```
      577
   2 ) AEF
       A↓
       ---
       0E
        E↓
       ---
       0F
        E
       ---
        1
```

$q = 577$

$r = 1$.

$2 \times 1 = 2$

$2 \times 2 = 4$

$2 \times 3 = 6$

$2 \times 4 = 8$

$2 \times 5 = A$

$2 \times 6 = C$

$2 \times 7 = E$

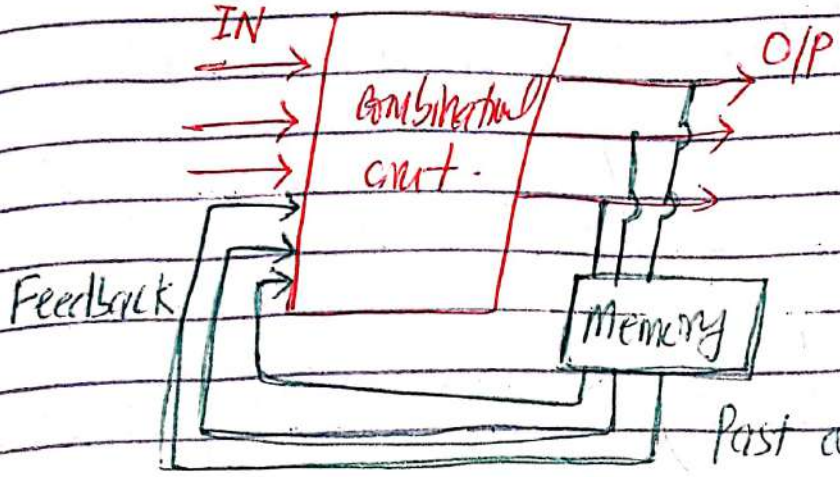
$2 \times 8 = 10$

$2 \times 9 = 12$

$2 \times A = 14$

Introduction to Sequential Circuits:

"In sequential circuits, the present output depends upon the present input as well as the past output (outputs)."



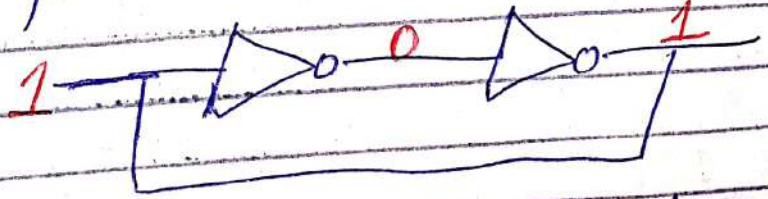
let say a counter counting from 0 to 3.

0 + 1
(1) + 1
(2) + 1
(3) + 1
↓

If it does not know the previous output so how will it add 1 to it. Therefore we need to store the previous output → memory

Memory is nothing but a flipflop cascaded

eg for 1 bit we use NOT gate.



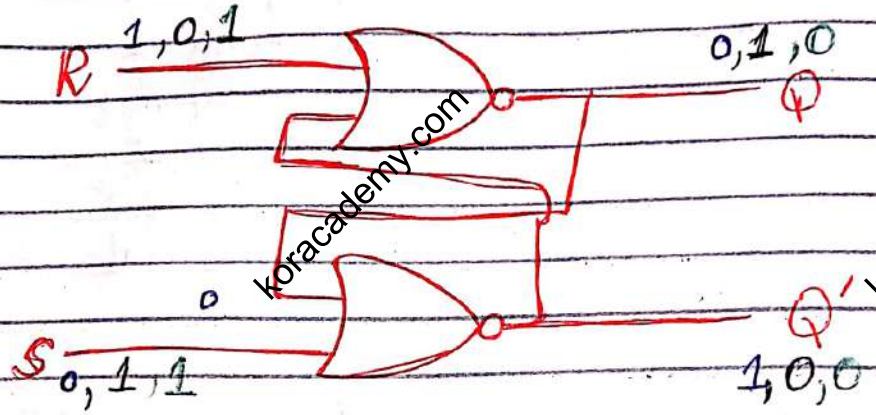
So the 1 is stored.

SR Latch

The basic storage element is called LATCH. As the name suggests it latches "0" or "1".

SR Latch → NOR
SR Latch → NAND

R → Reset ↓ Q=0
S → Set ↓ Q=1



Truth table of NOR.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Case I

S=0, R=1
Q=0 Q'=1
Remove inputs;
ie S=0 R=0
Q=0 Q'=1 → Memory

Case II

S=1 R=0
Q=1 Q'=0
Remove the inputs ie R=0 and S=0
Q=1 Q'=0 → Memory

Case III

$S = 1$ $R = 1$

$Q = 0$ $Q' = 0$

Removing inputs

$S = 0$ $R = 0$

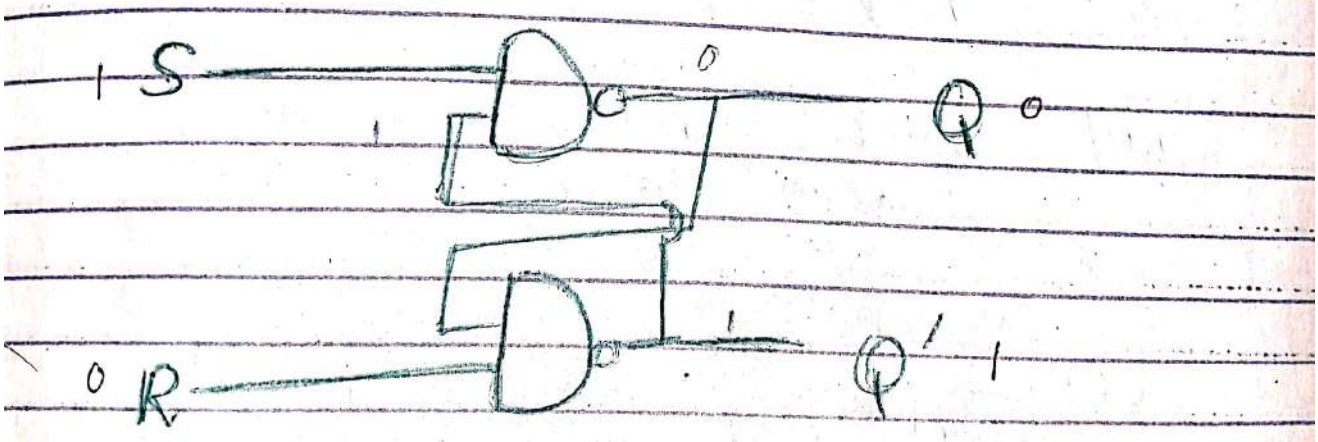
$Q = 0$ $Q' = 1$ → If start with Q

If stat with Q'

$Q = 1$ $Q' = 0$

Truth table for SR latch:

Inputs		Outputs		Previous output
S	R	Q	Q'	
0	0			Memory
0	1	0	1	
1	0	1	0	
1	1			Not used → Invalid



S	R	Q	Q'	
0	0			→ Not used
0	1	1	0	
1	0	0	1	
1	1			Memory

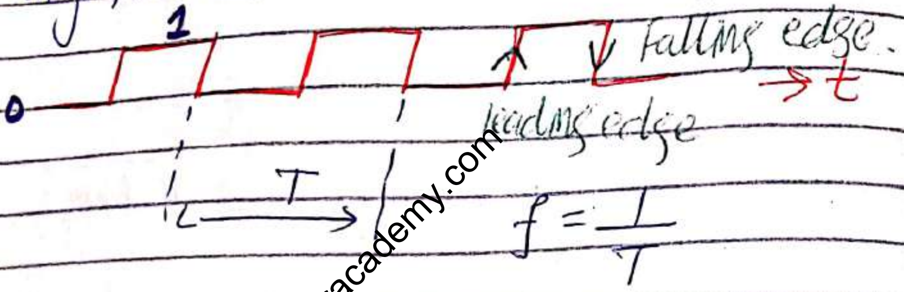
seq cir → may processes → input of one o/p to another → ...

What is a Clock

Control over process → Sequence of circuit
Decides time of the input.

We can decide speed of circuit through clock by changing clock frequency.

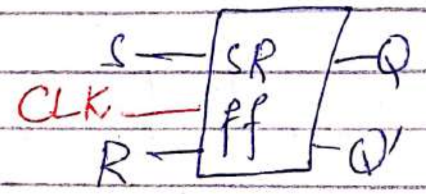
Clock is a signal that goes from low to high, then high to low and repeat.



$f = \frac{1}{T}$

Duty cycle of 50%.
Time of high = Time of low.

Circuit → faster ⇒ increase frequency
decrease time period



SR ff will work only when the clock is high.

Similarly you can design a ~~data ff~~ circuit which works on the leading edge.

Duty cycle

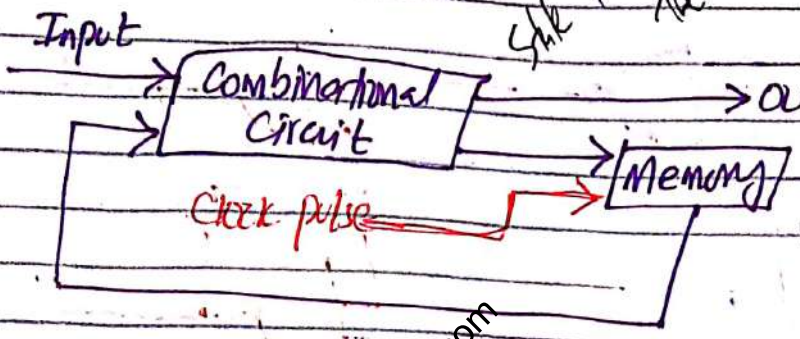
Ratio of time for which the signal is high ~~of~~ to the total time.

s/c ckt out o/p change randomly.

duty cycle = $\frac{\text{signal 1 time}}{t}$ $\frac{t/2}{t}$

50 % of this clock. $= \frac{1}{2}$

Triggering Methods:

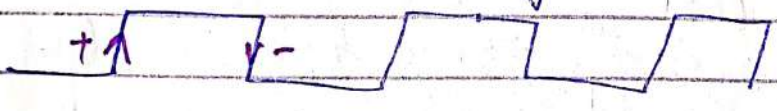


clock pulse = continuous signal

Triggering → Level
→ Edge

When clock is at high, changes will happen in circuit → level triggering.

Edge → Positive edge triggering
→ Negative " "



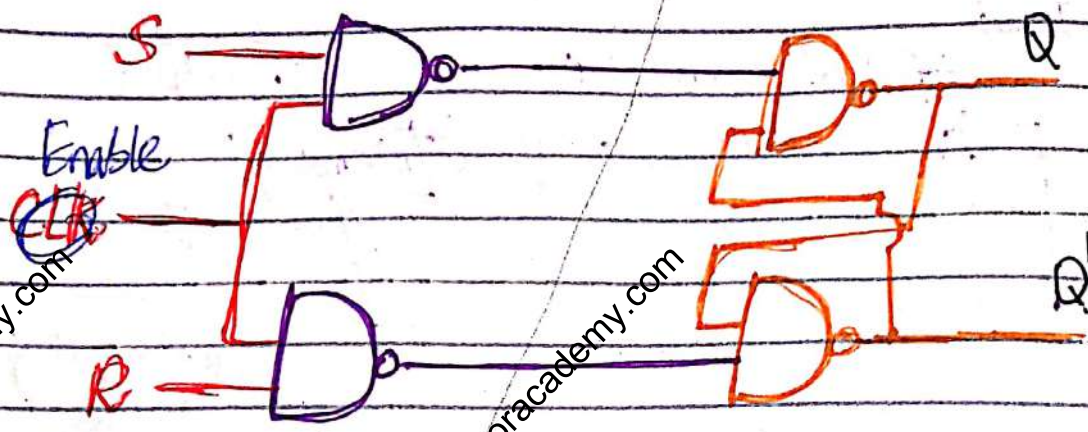
low to high → there will be change in memory element → positive edge trig

When clock goes from high to low → there will be transition in memory element → negative edge triggering.

no control signal → S and R can change intentionally & accidently.

Difference B/w Latch and Flip Flop

We need to introduce a control signal in the SR latch inorder to store Q and Q' until they are used. ↳ decide when to change S or R.



This circuit can act both as a latch and a flipflop. latch/ff depends on control I/P

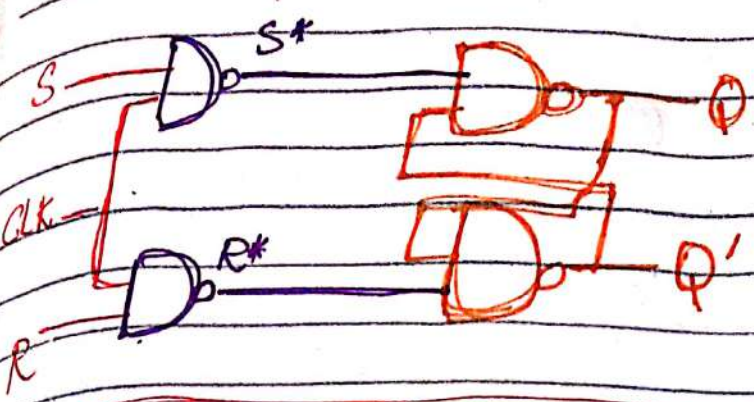
If the control input is not a clock, it is just enable, it is level triggered → It will act as a latch → This circuit will be operational if enable is high
SR latch with controlled I/P.

If we have a clock as control input, it will act as a ff → when there is edge triggering.

Latch is level sensitive.
FF is edge sensitive.

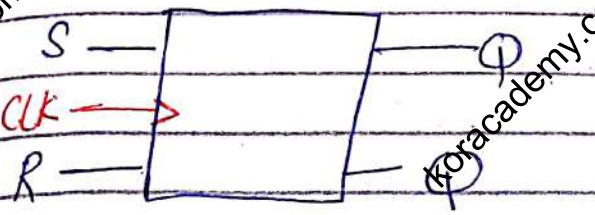
$E_n = 1$ $E_n = 0$
$S^n = S$ $S^n = 1$ $R^n = 1$
= Memory

Introduction to SR flip flop



S*	R*	Q	Q'
0	0	Not used	
0	1	1	0
1	0	0	1
1	1	Memory	

$$S^* = \overline{(S \cdot CLK)} = S' + CLK'$$
$$R^* = \overline{(R \cdot CLK)} = R' + CLK'$$



→ +ve edge triggered clock

Truth table

Inputs			outputs	
CLK	S	R	Q	Q'
0	X	X	Memory	
1	0	0	Memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Not used	

S^* & R^* = 1 ← $A + 1 = A$

CLK = 0 ⇒ CLK' = 1 ⇒ ~~S* = S'~~ $S^* = 1$

CLK = 0 ⇒ CLK' = 1 ⇒ ~~R* = R'~~ $R^* = 1$

When $\Rightarrow$ **CLK = 1, $S^* = S'$, $R^* = R'$**

CLK = 1, S = 0, R = 0 $\Rightarrow$ $S^* = 1$, $R^* = 1$ $\Rightarrow$ Memory

CLK = 1, S = 0, R = 1 : $S^* = 1$, $R^* = 0$

CLK = 1, S = 1, R = 0 : $S^* = 0$, $R^* = 1$

CLK = 1, S = 1, R = 1 : $S^* = 0$, $R^* = 0$

~~Characteristic~~

Truth table for SR ff

CLK	S	R	Q_{n+1}	
0	X	X	Q_n	[Memory]
1	0	0	Q_n	
1	0	1	0	
1	1	0	1	
1	1	1	Invalid	

Q_{n+1} → Next state Q_n → Present state

In characteristic table Q_{n+1} = ?

Next state is dependent upon your input (S and R) and also the present state (Q_n).

Characteristic Table for SR ff:

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

clk = 1

Excitation table for SR ff

Inputs → Q_n, Q_{n+1} Outputs → S, R

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

Finding Q_{n+1} from characteristic table;

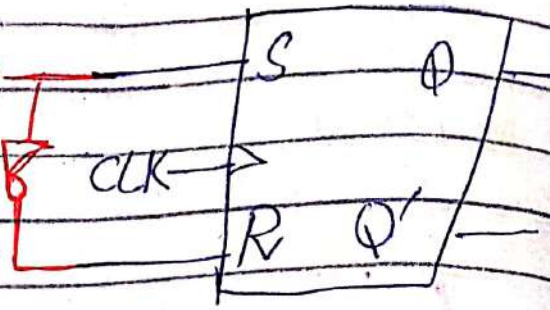
Q_n \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

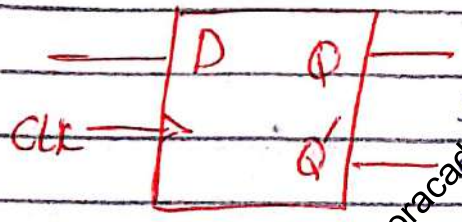
$$Q_{n+1} = S + Q_n R'$$

Introduction to D Flipflop

SR

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





CLK	D (Q_n)	Q_{n+1}
0	X	Q_n
S=1 R=0 →	1	0
S=0 R=1 →	0	1

Characteristic table

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

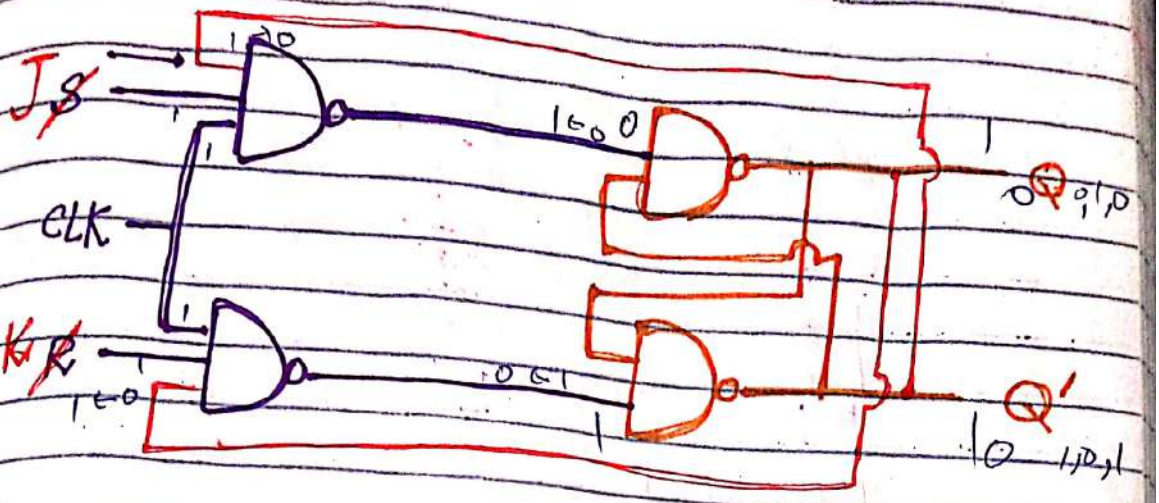
$Q_{n+1} = D$

Excitation table

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

$Q_{n+1} = D$

Introduction to JK Flipflop



CLK = 0 Memory

CLK = 1 J = 0 K = 0 Q = 1 Q' = 0
CLK = 1 J = 0 K = 1 Q = 0 Q = 1

CLK = 1 J = 1 K = 1
Assume Q = 0 Q' = 1.

output Q = 0, 1, 0, 1, 0, 1 -----
Q' = 1, 0, 1, 0, 1, 0 ----- Racing

$Q_{n+1} = Q'$

Truth table

CLK	J	K	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Q_n'

Characteristic table of JK

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

Excitation Table

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

For J

Q_n \ Q_{n+1}	0	1
0	0	1
1	X	X

For K

Q_n \ Q_{n+1}	0	1
0	X	X
1	1	0

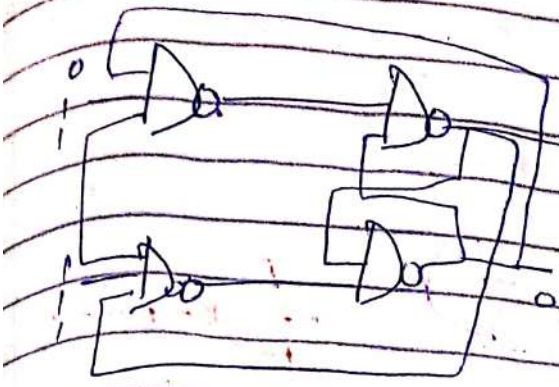
$J = Q_{n+1}$ $\qquad$ $K = Q_n$

Q_{n+1} from characteristic table.

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$Q_{n+1} = Q_n'J + Q_nK'$

Race Around Condition:

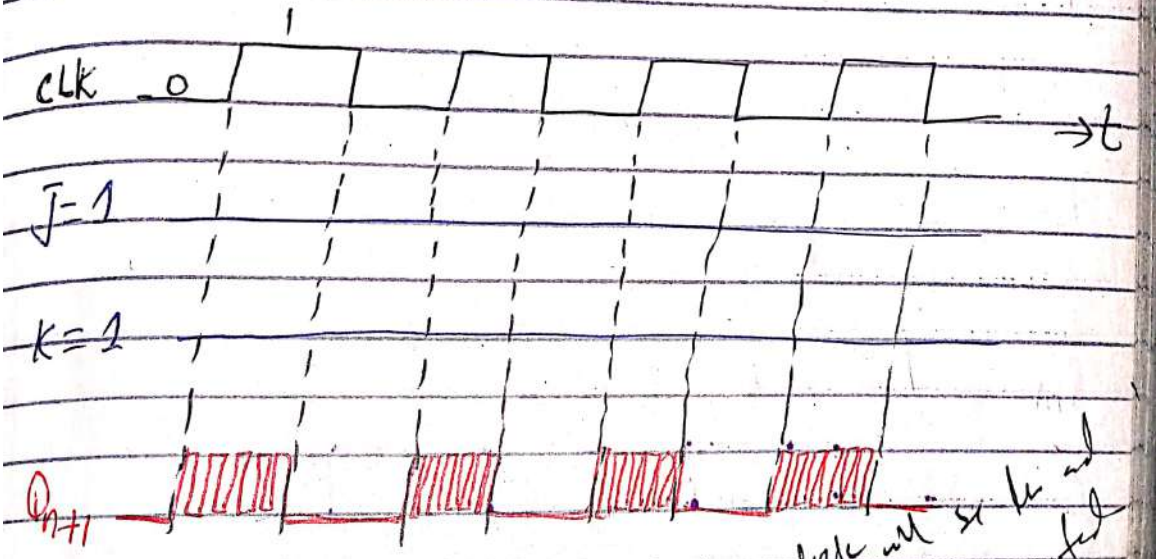


Assume Q=1 Q'=0

CLK	J	K	Q_{n+1}	Q_{n+1}'
0	X	X	Q_n	Q_n'
1	0	0	Q_n	Q_n'
1	0	1	0	1
1	1	0	1	0
1	1	1		

Continous changing of output 0→1→0→1 is called race around condition

Race Around → uncontrolled change
Toggling → controlled.



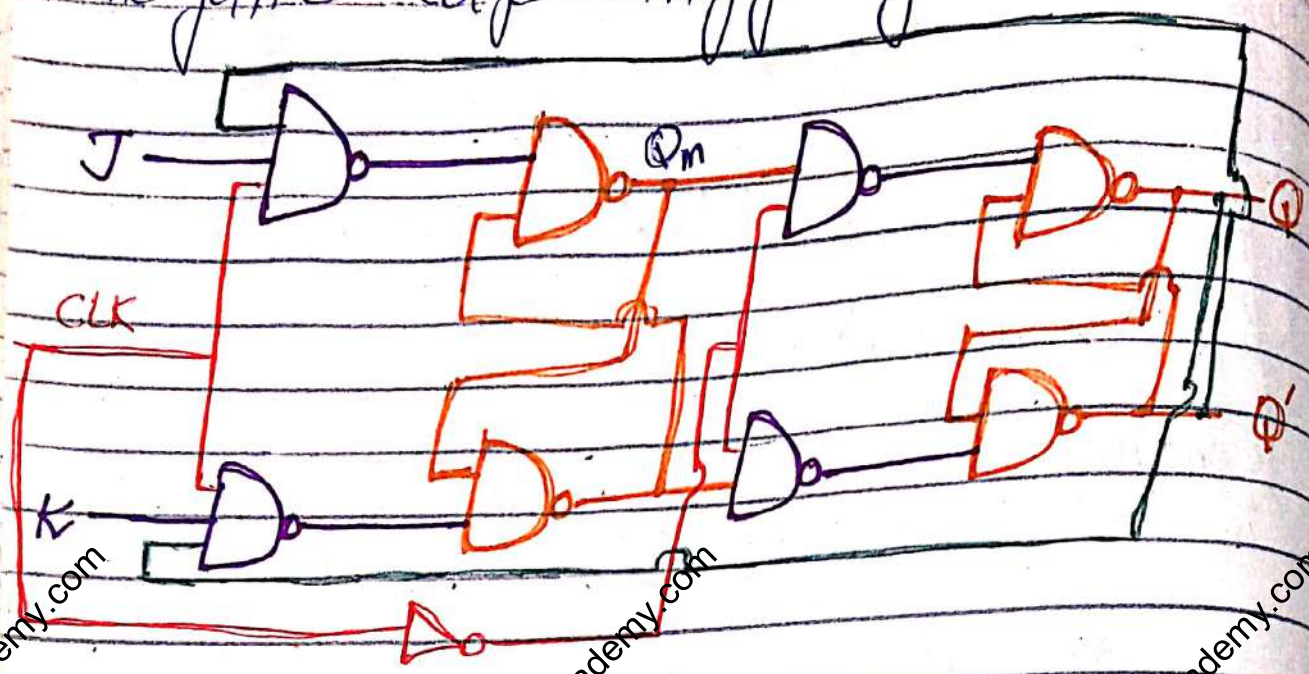
$\frac{T}{2}$ > delay

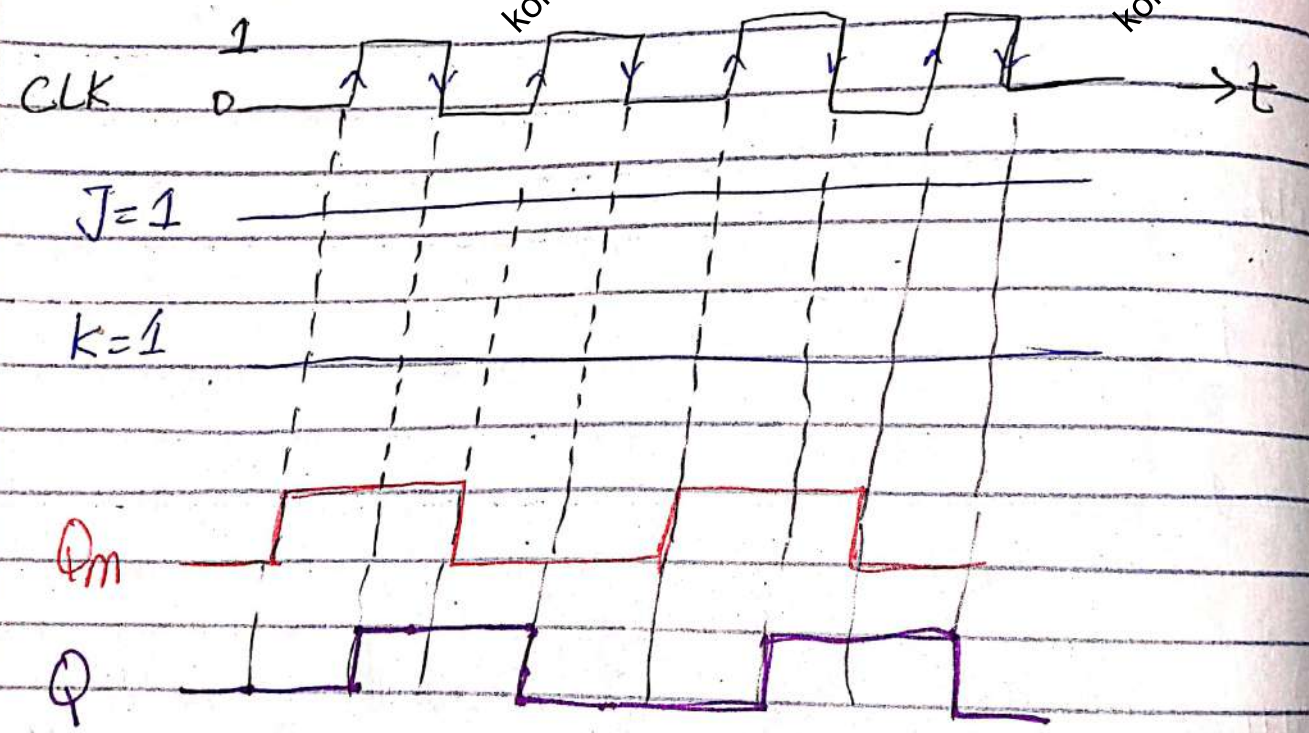
Conditions to overcome Racing:

i) T/2 < Propagation delay of ff.
ii) Edge triggering
iii) Master slave.

Master Slave JK Flipflop

Master slave operation is same as negative edge triggering.

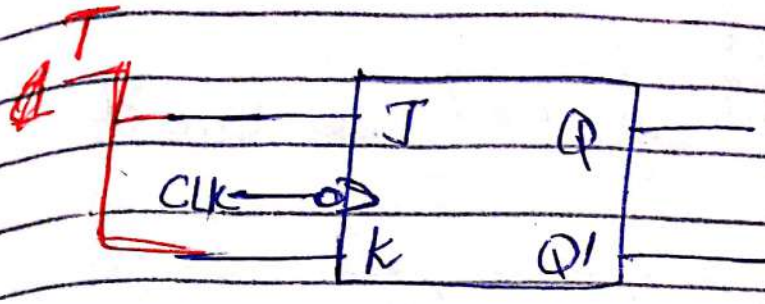




Clk	J	K	Q_{n+1}
1	1	1	Q_n' → toggling

Introduction to T flip flop

↳ toggle



CLK	T	Q_{n+1}	
0	X	Q_n	Memory
1	0	Q_n	
1	1	Q_n' → toggle	

Characteristic table

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

Q_n \ T	0	1
0	0	1
1	1	0

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

Excitation table

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

Flip Flop Conversion

Steps

1. Identify available and required flipflop.
2. Make characteristic table for required ff.
3. Make excitation table for available ff.
4. Write boolean expression for available ff.
5. Draw the circuit.

① J-K to D ff conversion

Available = JK Required = D

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

Characteristic of D

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

Excitation of JK

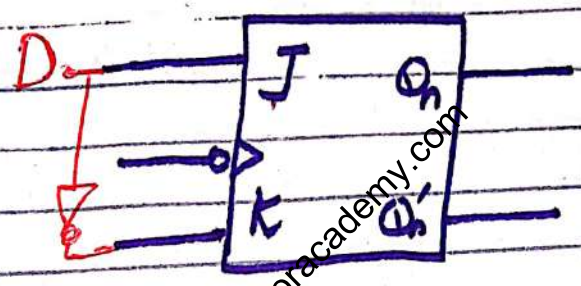
Q_n	Q_{n+1}	D	Q_{n+1}	J	K
0	0		0	0	X
0	1		1	1	X
1	0		0	X	1
1	1		1	X	0

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

Q_n \ D	0	1
0	0	1
1	X	X

Q_n \ D	0	1
0	X	X
1	1	0

$J = D$ $\quad K = D'$



(2) T ff to D ff

Available = T $\quad$ Required = D

Excitation

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

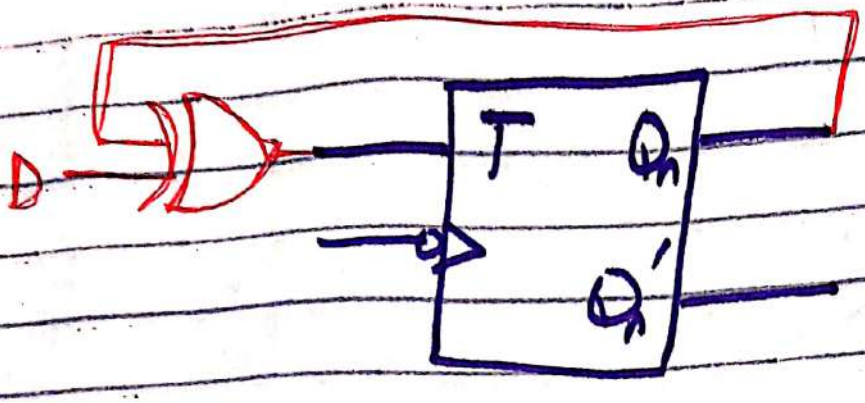
Characteristic

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

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

Q_n \ D	0	1
0	0	1
1	1	0

$T = Q_n \oplus D$



(3) SR to JK ff conversion

Available = SR → Excitation

Required = JK → characteristic

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

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

Match Q_n and Q_{n+1} from characteristic table with excitation table.

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

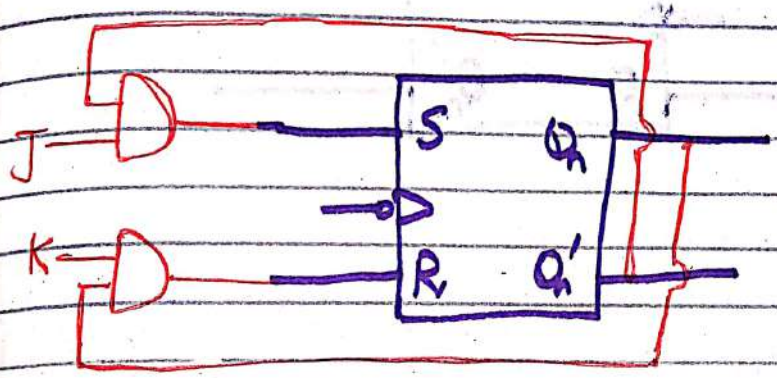
S	R
0	X
0	X
1	0
1	0
X	0
0	1
X	0
0	1

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	X	0	0	X

$S = Q_n' J$

Q_n \ JK	00	01	11	10
0	X	X	0	0
1	0	1	1	0

$R = Q_n K$



(4) SR to T ff.

Required → T Available → SR

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

Characteristic

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

Excitation

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

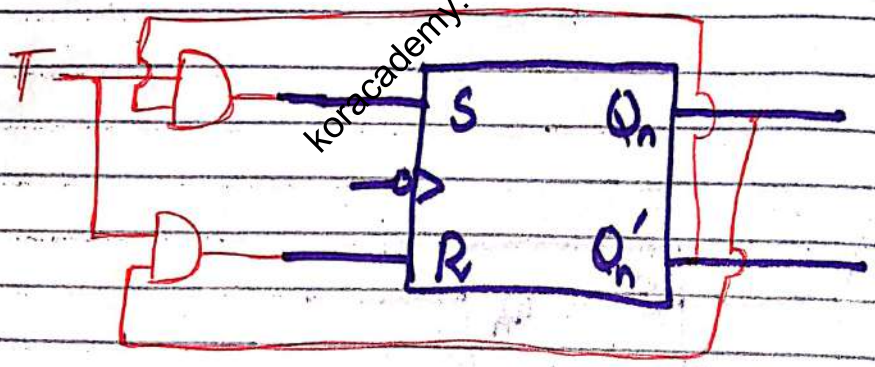
S	R
0	X
1	0
X	0
0	1

Q_n \ T	0	1
0	0	1
1	X	0

$S = Q_n' T$

Q_n \ T	0	1
0	X	0
1	0	1

$R = Q_n T$



(5) JK ff to SR Ff

Available = JK ↓ Excitation

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

Required = SR ↓ characteristic

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

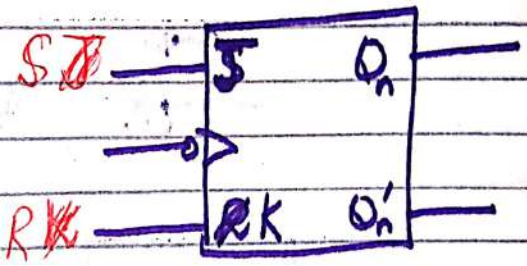
Q_n	S	R	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	X
0	1	1	X	X	X
1	0	0	1	X	0
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	X	X	X

Q_n \ SR	00	01	11	10
0	0	0	X	1
1	X	X	X	X

$J = S$

Q_n \ SR	00	01	11	10
0	X	X	X	X
1	0	1	X	0

$K = R$



(6) T ff to SR ff

Available = T
Excitation

Required = SR (Chat)

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

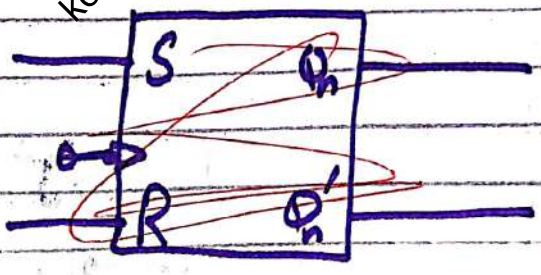
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X

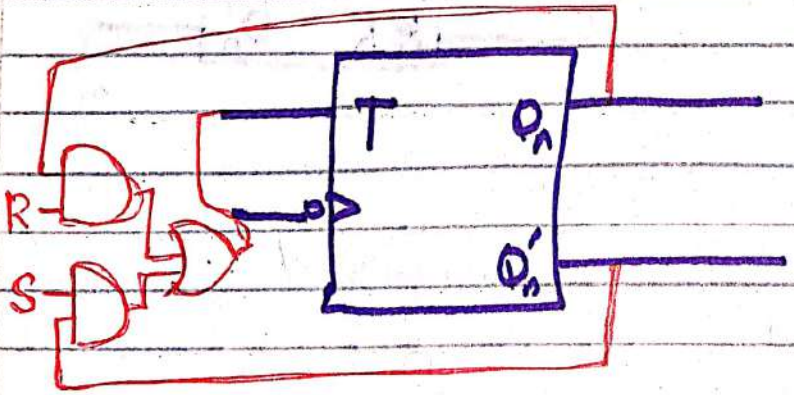
Q_n	S	R	Q_{n+1}	T
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	~~X~~	X
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	X	X

Q_n \ SR	00	01	11	10
0	0	0	X	1
1	0	1	X	0

~~$T = Q_n' S$~~

$T = Q_n' S + Q_n R$





(7) D ff to SR

Available = D → Excitation

Required = SR → Characteristic

Characteristic table of SR.

Combination table

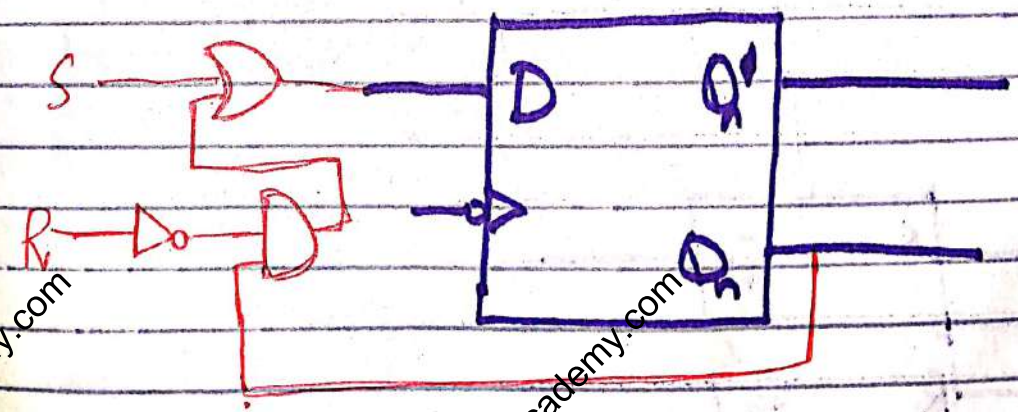
Q_n	S	R	Q_{n+1}	D
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	X	X
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	X	X

Excitation of D-

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

Q_n \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

$D = S + Q_n R'$

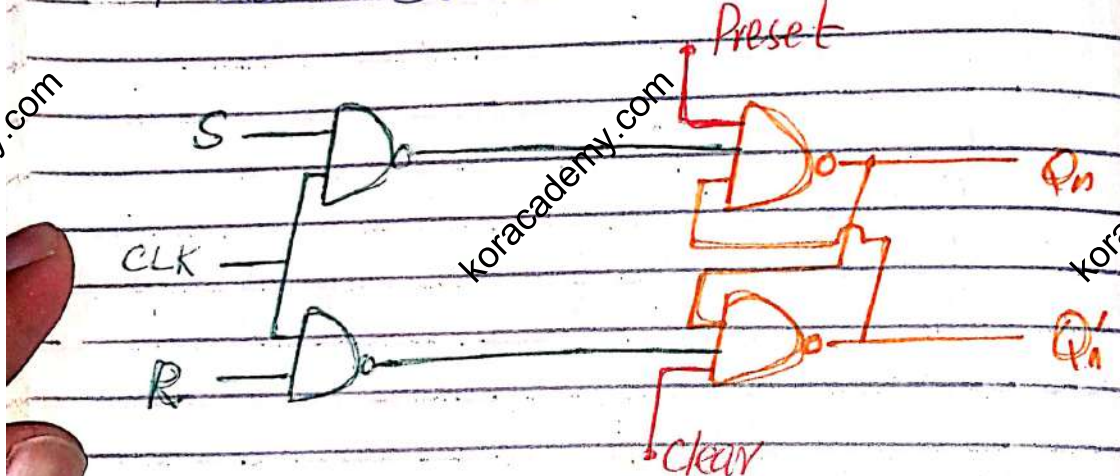


Preset and clear Inputs:

They are direct inputs or overriding inputs or asynchronous inputs.

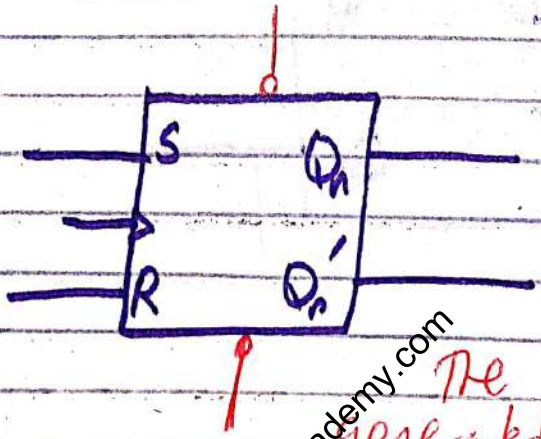
overriding → directly changes the output without the effect of any other input.

The synchronous inputs are S, R, J, Ic, T and D.



Preset = 0 ⇒ Qn = 1
Clear = 0 ⇒ Qn = 0

→ whatever be the value of clock and synchronous inputs.



Bubble (o) shows that it is active low signal → whenever / if it is low the output will be generated.

Preset	clear	Q_n
0	0	Not used.
0	1	1
1	0	0
1	1	FF will perform normally → no effect on ff.

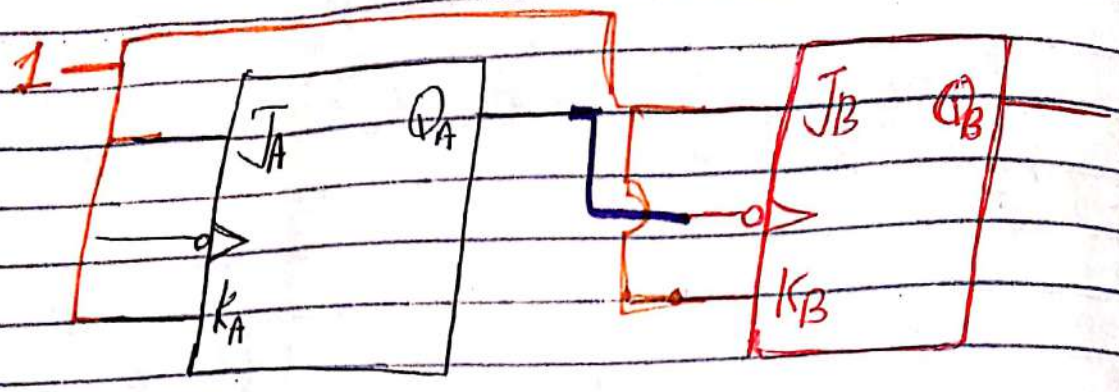
Difference B/w Synchronous and Asynchronous Sequential Circuits.

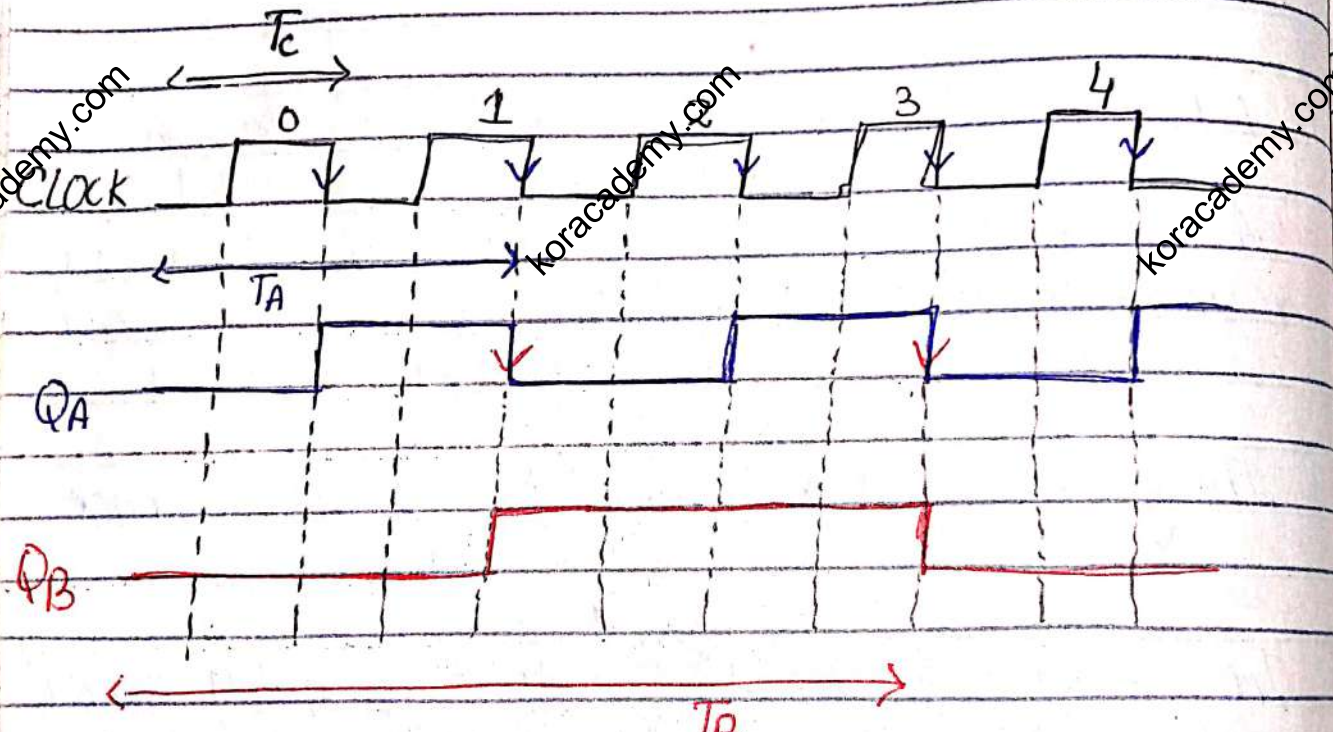
Synchronous	Asynchronous
1. Easy to design.	Difficult to design.
2. A clocked ff acts as memory element.	An unclocked ff or time delay element is used as memory element.
3. They are slower.	Faster due to absence of clock.
4. The states of the memory element is affected only at the active edge of clock, if input is changed.	The states of memory element will change at any time as soon as input is changed.
5. FFs are used.	Latches are used.

Introduction to Counters

Counter is a sequential circuit that counts.

FF as divide by 2





$T_A = 2T_C \Rightarrow \frac{1}{f_A} = \frac{2}{f_C} \Rightarrow f_C = 2f_A$

$f_A = \frac{f_C}{2}$

$T_B = 2T_A \quad \frac{1}{f_B} = \frac{2}{f_A} \Rightarrow f_B = \frac{f_A}{2} = \frac{f_C}{4}$

output = Q_A, Q_B.

$Q_A \rightarrow$ LSB $Q_B \rightarrow$ MSB.

If we have P no. of ff where J=1 and K=1 and is -ve edge triggered:

In total we are going to divide the frequency by 2^P

as here P=2 $2^2 = 4$

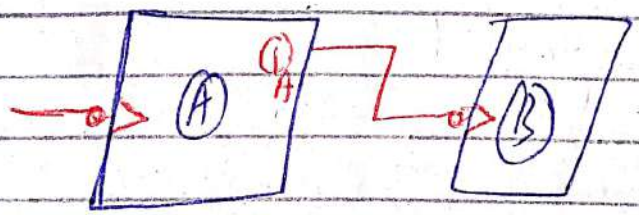
$\Rightarrow$ So $f_B = \frac{f_c}{4}$

CLK	Q_B	Q_A	
0	0	0	(0)
1	0	1	(1)
2	1	0	(2)
3	1	1	(3)

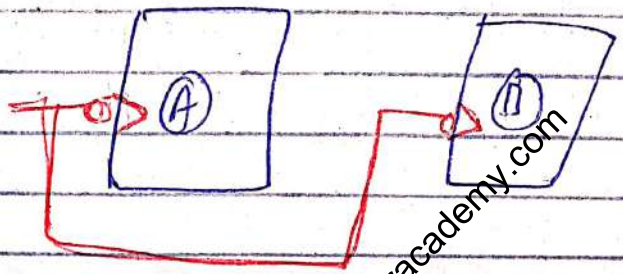
Counter counting from 0 to 3 (clock pulses).

Types of Counters:

1. Asynchronous / Ripple counters.
2. Synchronous.



Asynchronous



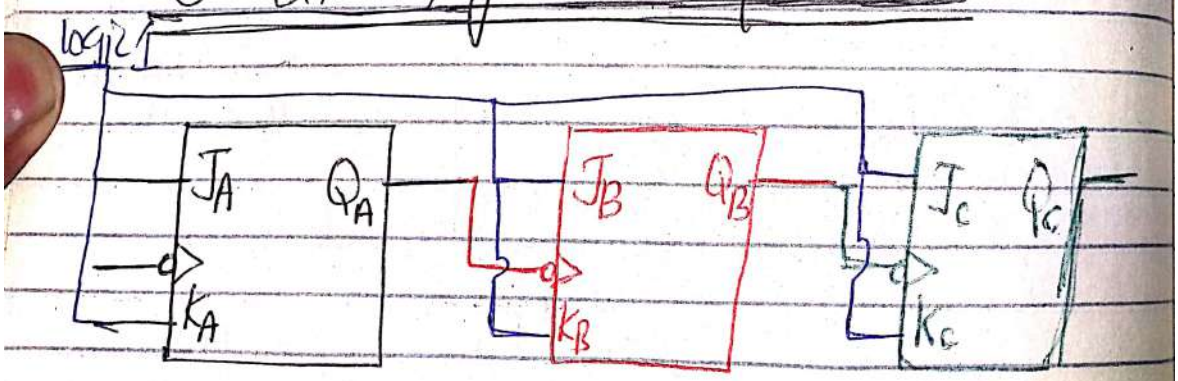
Synchronous

Asynchronous counter	Synchronous counter
1. FFs are connected in such a way that output of the first ff drives the clock of the next ff.	Clock is given simultaneously to all the flip flops.
2. Circuit is simple for more number of states.	Circuit becomes complicated as no. of states increases.
3. Speed is slow as clock is propagated through no. of stages.	Speed is high as clock is given at the same time.

Counters (As./S.)

- Up counter: 0 → 1 → 2 → 3
- Down counter: 3 → 2 → 1 → 0
- UP/down counter

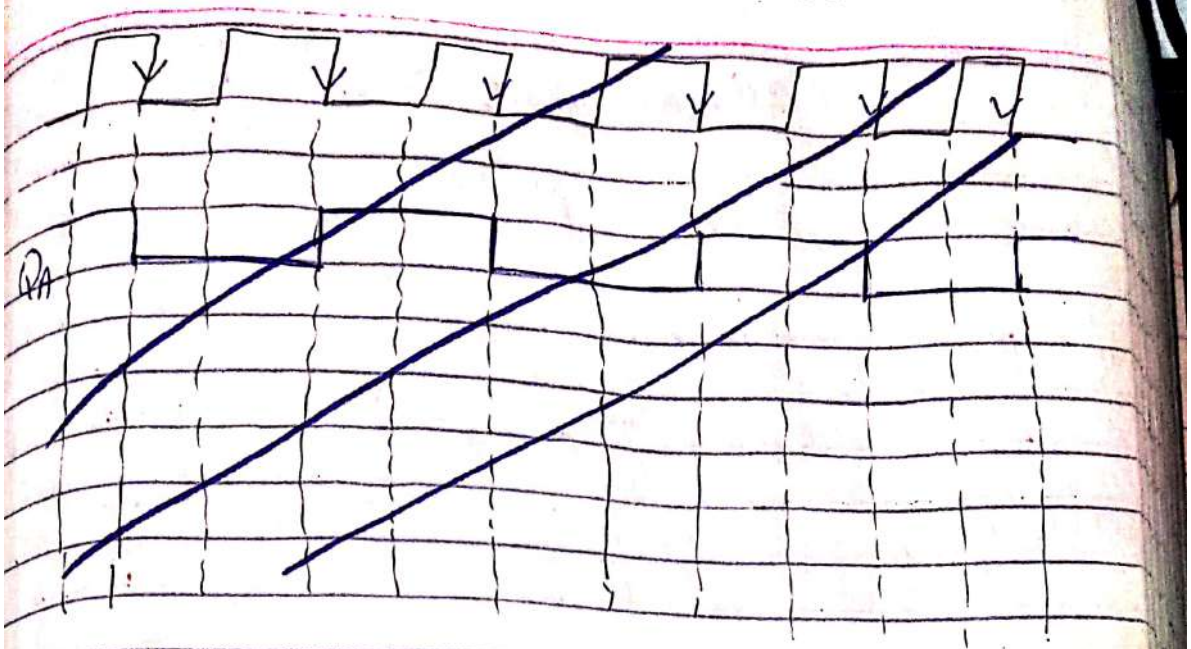
3 Bit Asynchronous Up Counter

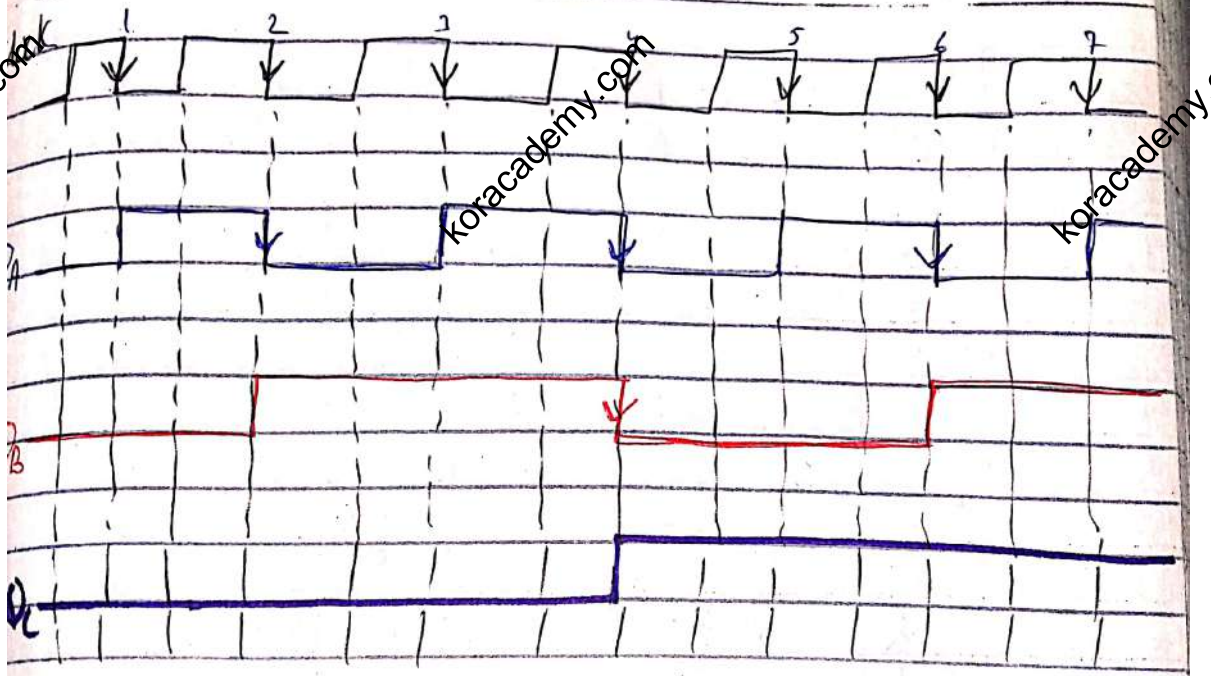


3 bit means 000 to 111
i.e. 0 to 7.

Q_A → LSB Q_C → MSB

$J_A = K_A = 1$. → toggle





CLK	Q_C	Q_B	Q_A	Decimal Eq.
Initially	0	0	0	0
1st ↓	0	0	1	1
2nd ↓	0	1	0	2
3rd ↓	0	1	1	3
4th ↓	1	0	0	4
5th ↓	1	0	1	5
6th ↓	1	1	0	6
7th ↓	1	1	1	7
8th ↓	0	0	0	0

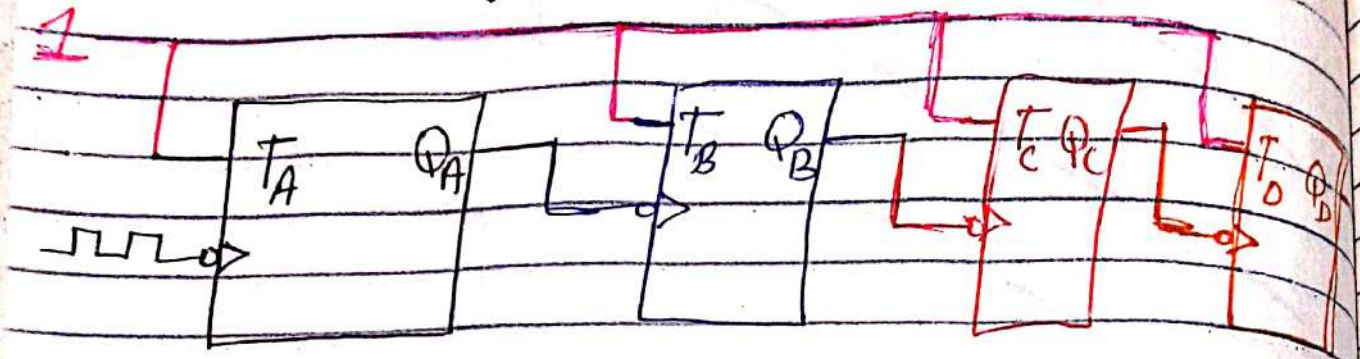
8 states

$2^n = 2^3$ states

n → no. of ffs

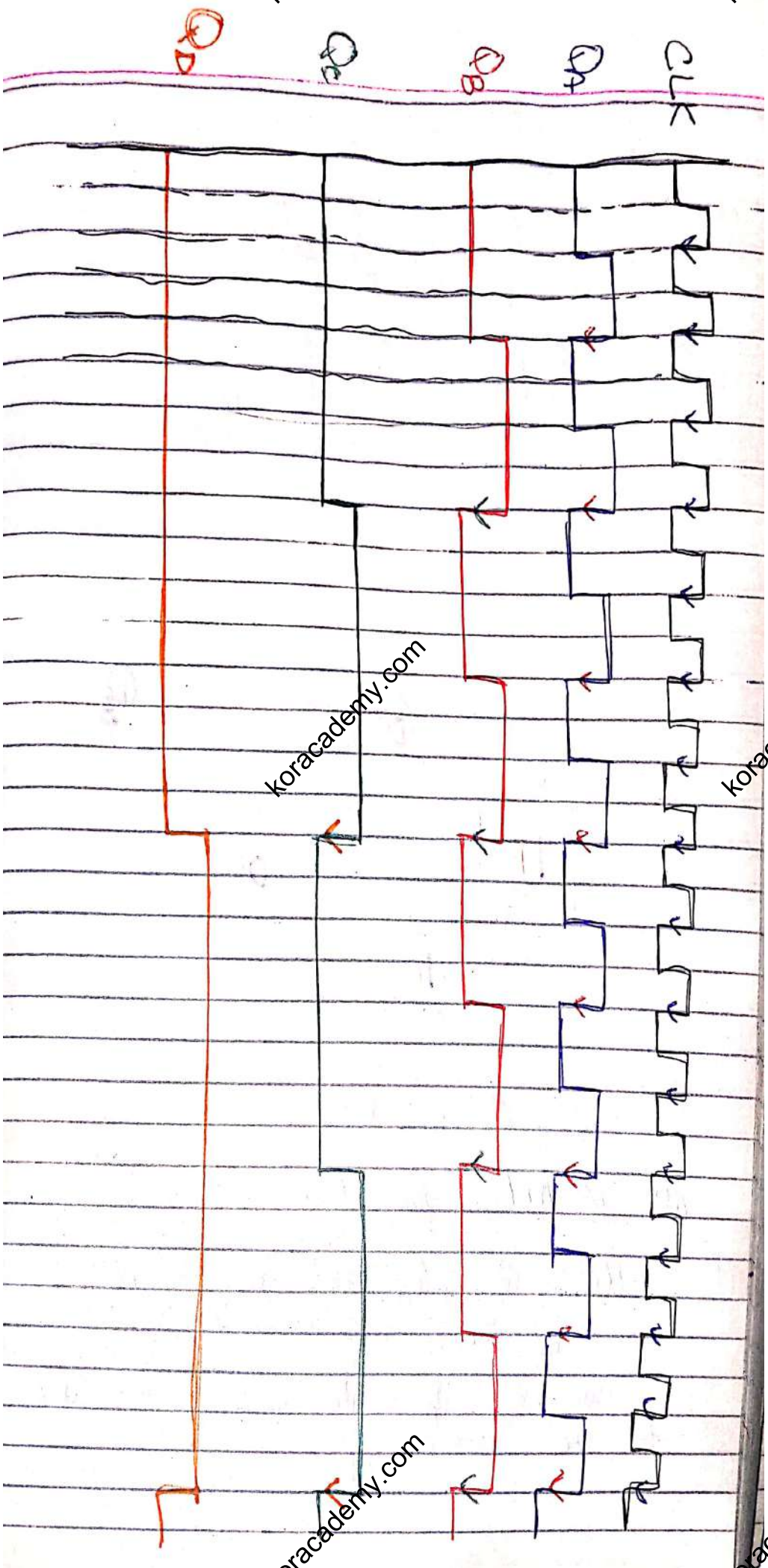
Maximum count = $2^n - 1$

4 Bit Asynchronous Up Counter



$Q_A \rightarrow$ LSB $Q_D \rightarrow$ MSB

CLK	Q_D	Q_C	Q_B	Q_A	D. E
Initially	0	0	0	0	0
1st ↓	0	0	0	1	1
2nd ↓	0	0	1	0	2
3rd ↓	0	0	1	1	3
4th ↓	0	1	0	0	4
5th ↓	0	1	0	1	5
6th ↓	0	1	1	0	6
7th ↓	0	1	1	1	7
8th ↓	1	0	0	0	8
9th ↓	1	0	0	1	9
10th ↓	1	0	1	0	16
11th ↓	1	0	1	1	11
12th ↓	1	1	0	0	12
13th ↓	1	1	0	1	13
14th ↓	1	1	1	0	14
15th ↓	1	1	1	1	15
16th ↓	0	0	0	0	0

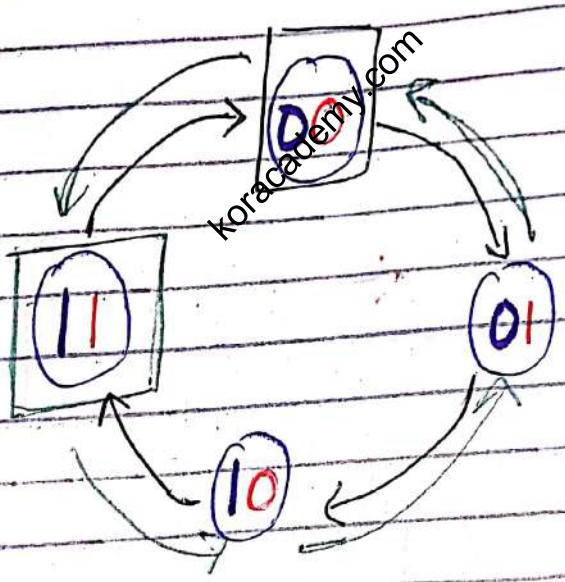


State Diagram of A Counter

let a 2 bit up counter;
let the output of FFs is $Q_B Q_A$.

cant from 00 → 11 0 → 3.
00 → 01 → 10 → 11 Up
00 ← 01 ← 10 ← 11 Down

State of a counter is nothing but its counts ie 00, 01 etc.



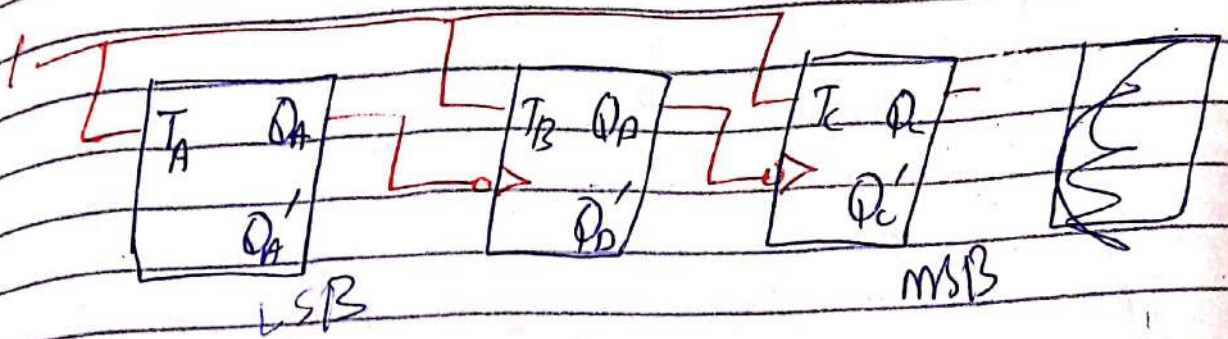
Q_B Q_A
Up Counting
Down Counting.

3 Bit Asynchronous Down Counter

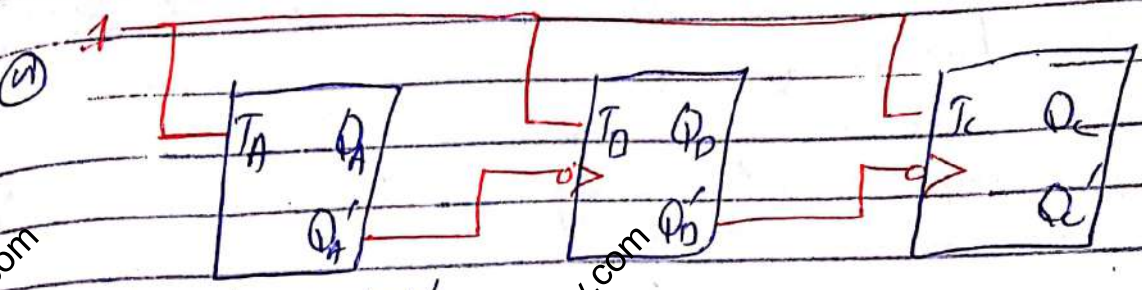
Two methods for down counting;

① Take Q_A', Q_B' and Q_C' as output.
Q_A' → LSB Q_C' → MSB
Provide Q_A', Q_B', Q_C' -- as clock to the next FF.

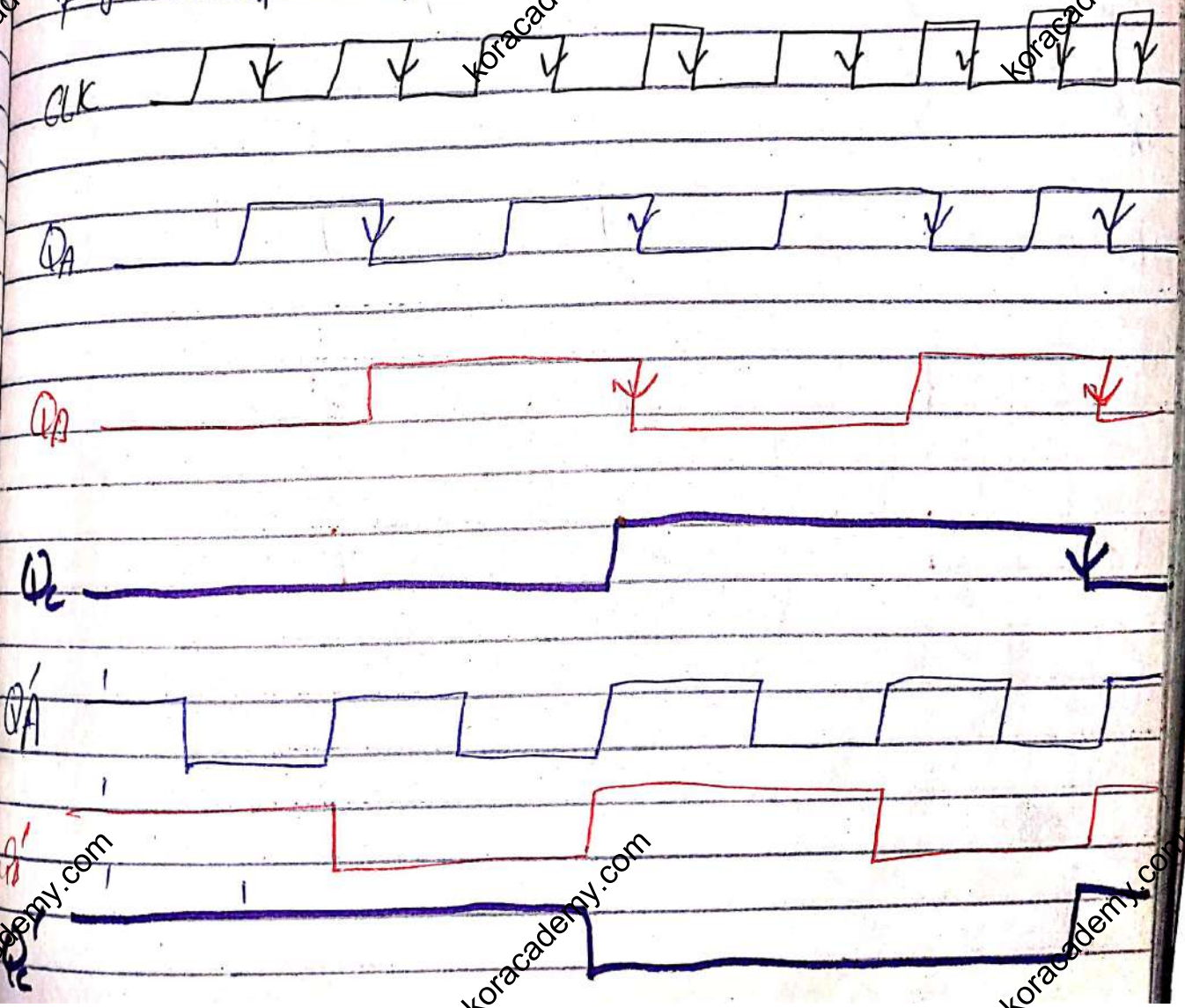
② If you want Q_A, Q_B and Q_C to be outputs, then give Q_A', Q_B' Q_C' ---- as clock to the next ff.



(a)

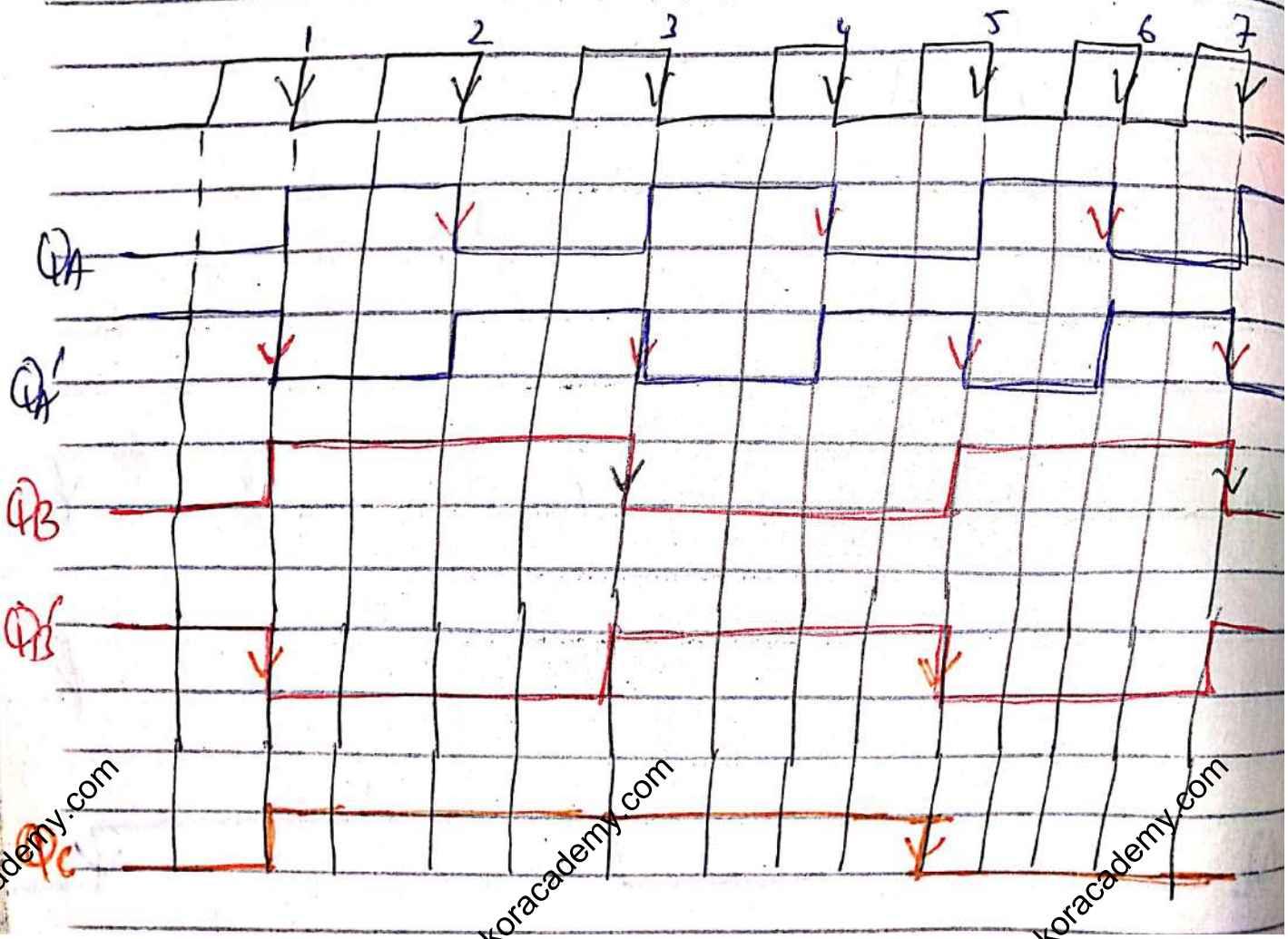


If you take Q_A' → -ve pt



CLK	Q_C	Q_B	Q_A	Q_C'	Q_B'	Q_A'	
Initally	0	0	0	1	1	1	(7)
1	0	0	1	1	1	0	(6)
2	0	1	0	1	0	1	(5)
3	0	1	1	1	0	0	(4)
4	1	0	0	0	1	1	(3)
5	1	0	1	0	1	0	(2)
6	1	1	0	0	0	1	(1)
7	1	1	1	0	0	0	(0)

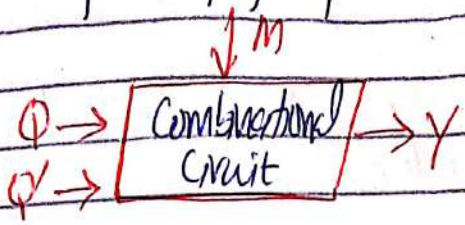
If you want Q_A as output of the counter as clock ↓.

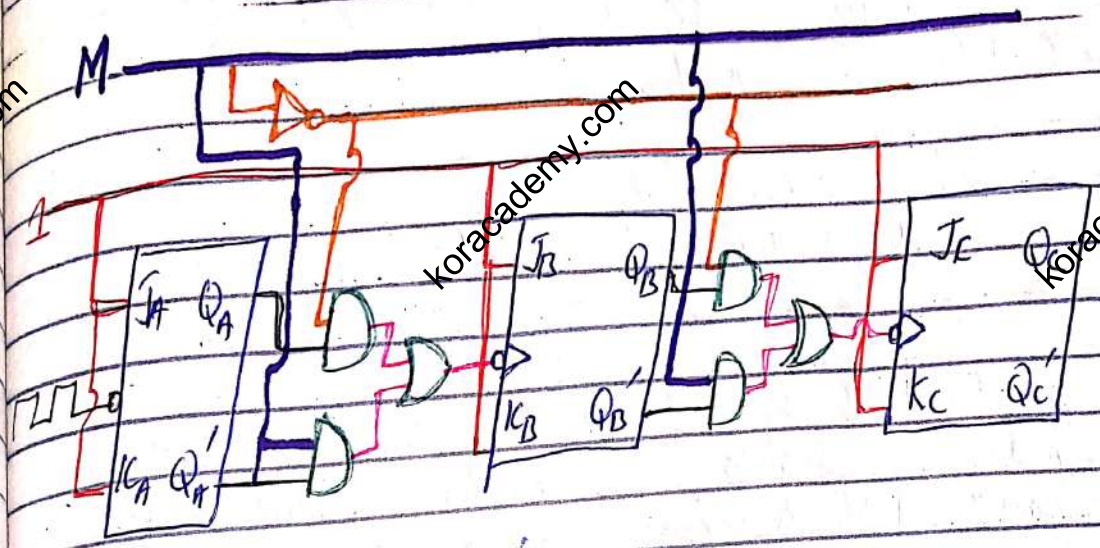


3 Bit Up Down Ripple Counter

A mode control input (M) is used to select either up or down mode.

A combinational circuit is required b/w each pair of flip flops.





M	Q	Q'	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

M=0 → up counting
Q is connected as clock.

M=1 down counting
Q' is connected as clock.

M → MSB Q' → LSB

M \ QQ'	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$Y = M'Q + MQ'$

Modulus of the Counter And Counting To A Particular Value:

2 Bit ripple counter is called MOD-4 or modulus 4 counter.

3 Bit ripple counter is called MOD-8 or modulus 8 counter.

let n → number of bits.
⇒ MOD number = 2^n
Also state = 2^n.

⇒ Modulus of a counter represents the number of states of a counter (how many counts it is going to do).

* Example
Design a MOD 6 counter using MOD 8 counter

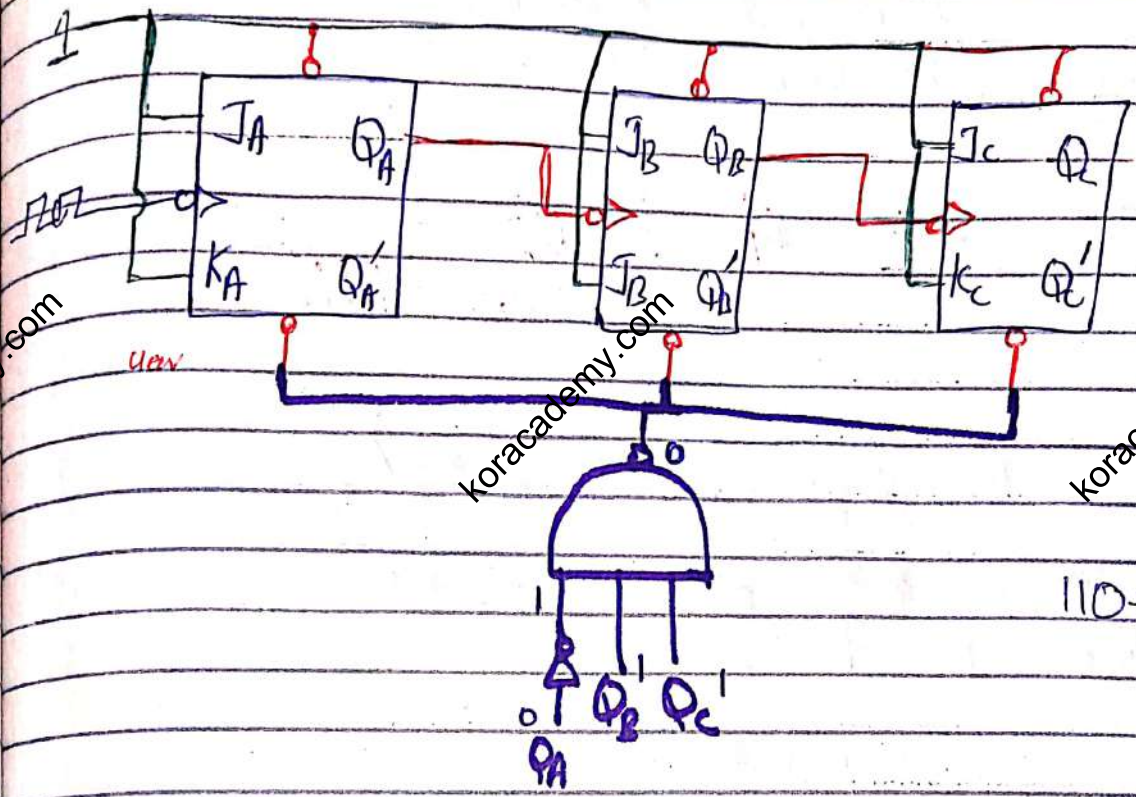
MOD 6 → states = 6 → max count = 6-1 = 5
⇒ 000 → 001 → 010 → 100 → 101

Mod 8 counter →
Preset → Clear = 0 = Q = 0
CLK → R

Reset = 0 ⇒ Q = 1 → Set

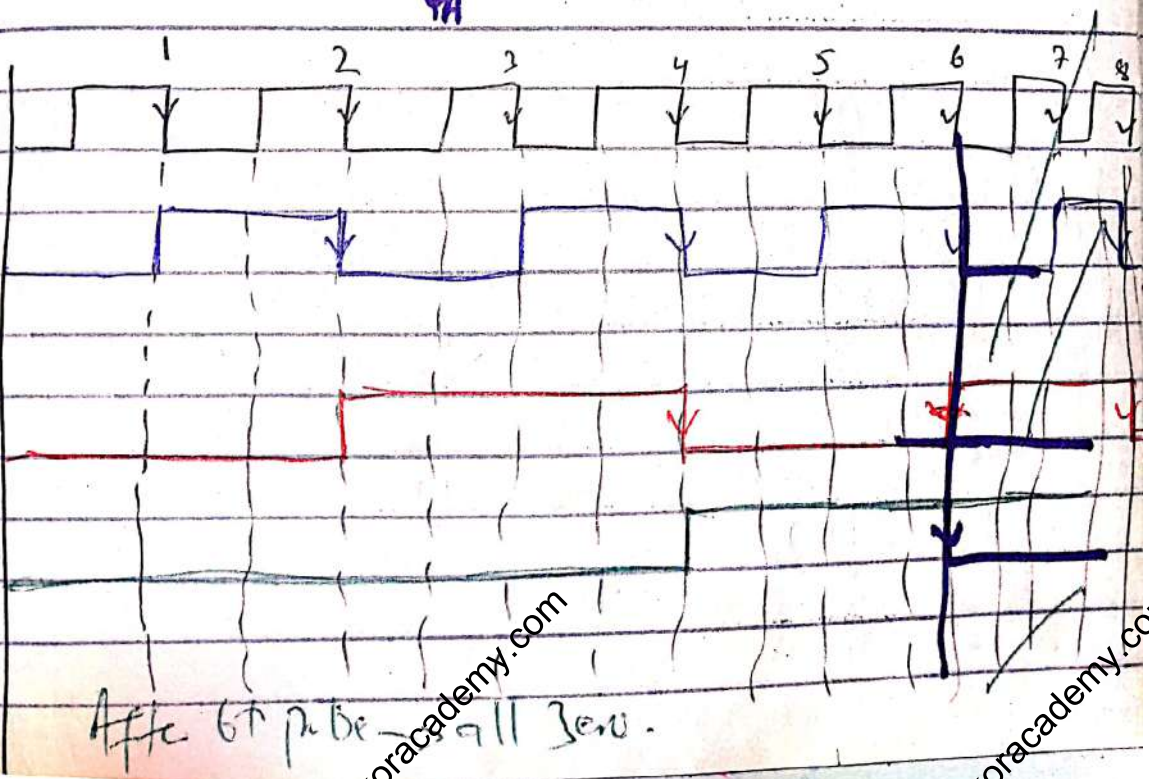
Clear = 0 ⇒ Q = 0 → Reset

As we dont want 6 and 7, so as soon as 6 is arrived we have to reset the flipflop; Use a NAND gate (three input) and provide the output to all the clears.



110 → 6

X



After 6th pulse all zero.

Important Points

(1) Negative Edge triggered ff:

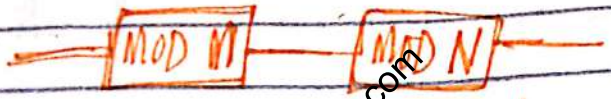
Q is clock → up counter.

Q' is clock → down counter.

(2) Positive Edge triggered ff:

Q' is clock → Up counter.

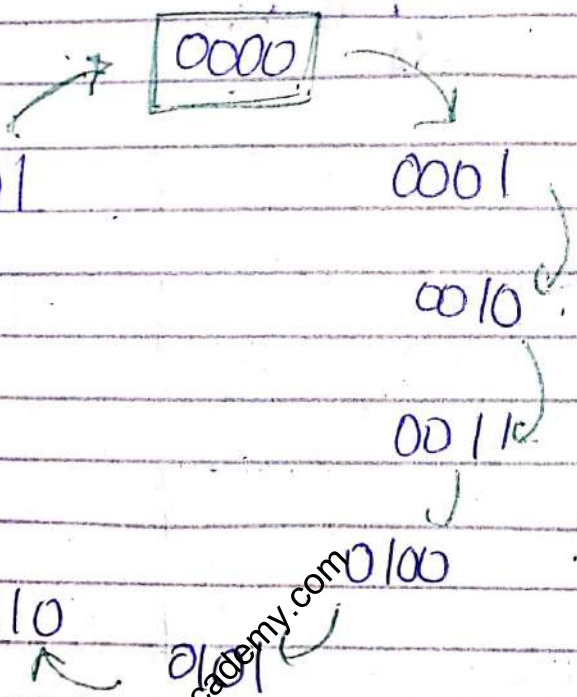
Q is clock → Down counter.



Decade (BCD) Ripple Counter

No. of states = 10 0 → 9

Max count = 9

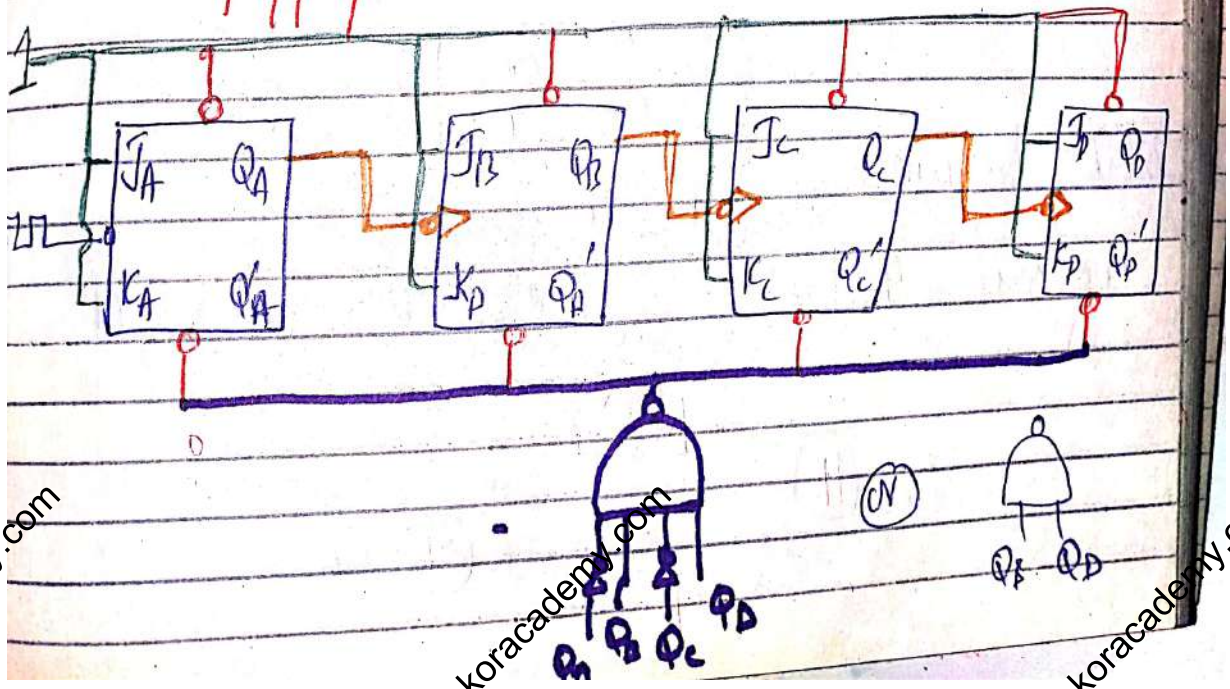


4 bit counter starts at 0000 and ends at 1111 but we want to stop at 1001.

CLK	Q_D	Q_C	Q_B	Q_A
Initially	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
	1	0	1	0
	1	0	1	1
	1	1	0	0
	1	1	0	1
	1	1	1	0
	1	1	1	1

→ Decade counter.

As soon as 1010 (10) is arrived, reset the flipflops.



* How to design Synchronous Counter

Step 1 Decide the number of flipflops and the flipflop to be used.

Step 2 Excitation table of flipflop.

Step 3 State diagram and circuit excitation table

Step 4 obtain simplified equations using kMaps.

Step 5 Draw the logic diagram

Question Design two bit synchronous up counter.

JK ff → 2.

Excitation table

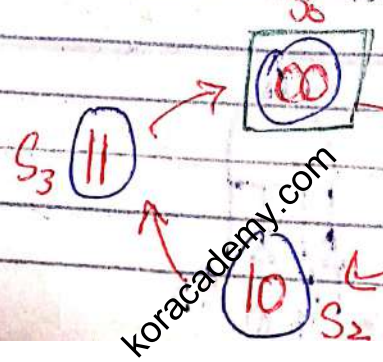
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State diagram

$2^n = 2^2 = 4$ states.

Max count = 4 − 1 = 3

00 → 01 → 10 → 11



Circuit Excitation Table

Present State		Next State		Flip excitation table			
Q_1	Q_2	Q_1^*	Q_2^*	J_1	K_1	J_2	K_2
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	1	1	X	0	1	X
1	1	0	0	X	1	X	1

Q_1 \ Q_2	0	1
0	0	1
1	X	X

$J_1 = Q_2$

Q_1 \ Q_2	0	1
0	X	X
1	0	1

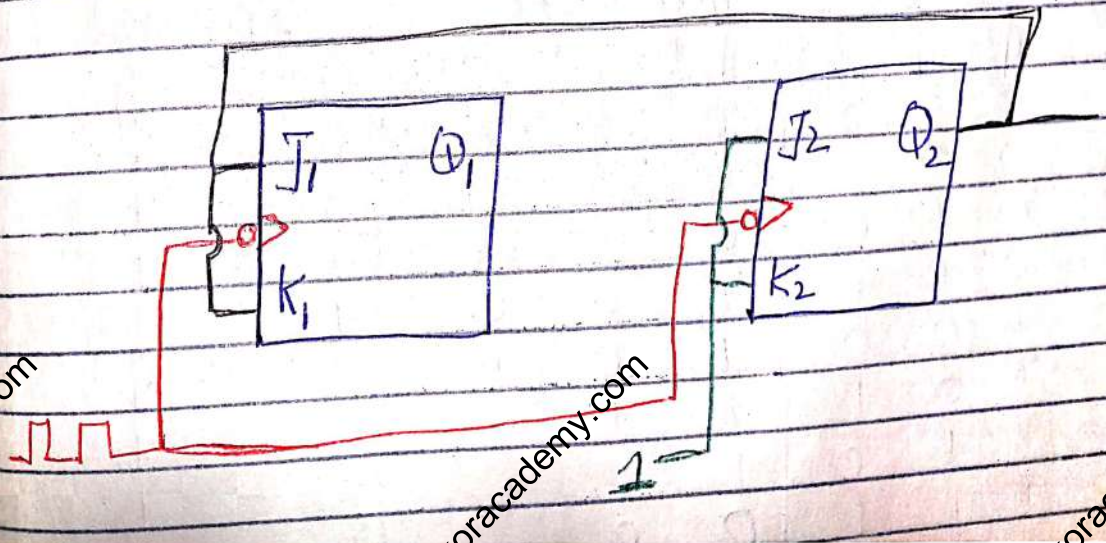
$K_1 = Q_2$

Q_1 \ Q_2	0	1
0	1	X
1	1	X

$J_2 = 1$

Q_1 \ Q_2	0	1
0	X	1
1	X	1

$K_2 = 1$



3 Bit Synchronous Up Counter

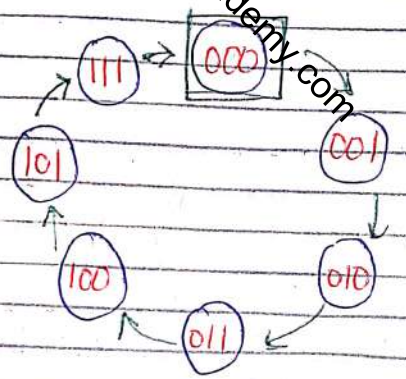
Step 1 → 3 ffs T ff

Step 2 Excitation table for T ff

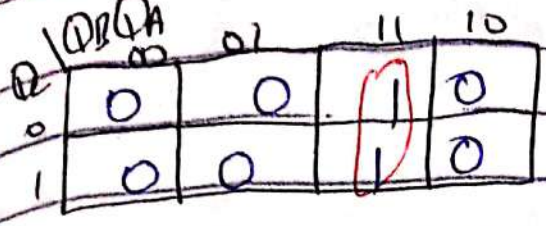
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Step 3 → state diagram $2^3 = 8$ states

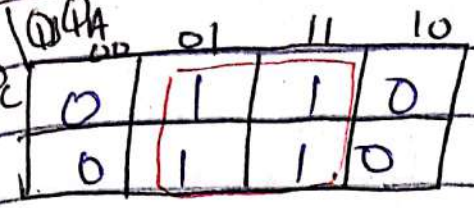
Max cnt = 8 − 1 = 7



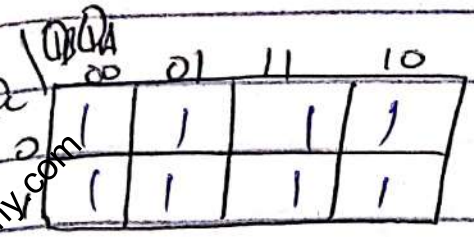
Present states Q_C	Q_B	Q_A	Next states Q_C^+	Q_B^+	Q_A^+	Flip-flop inputs T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



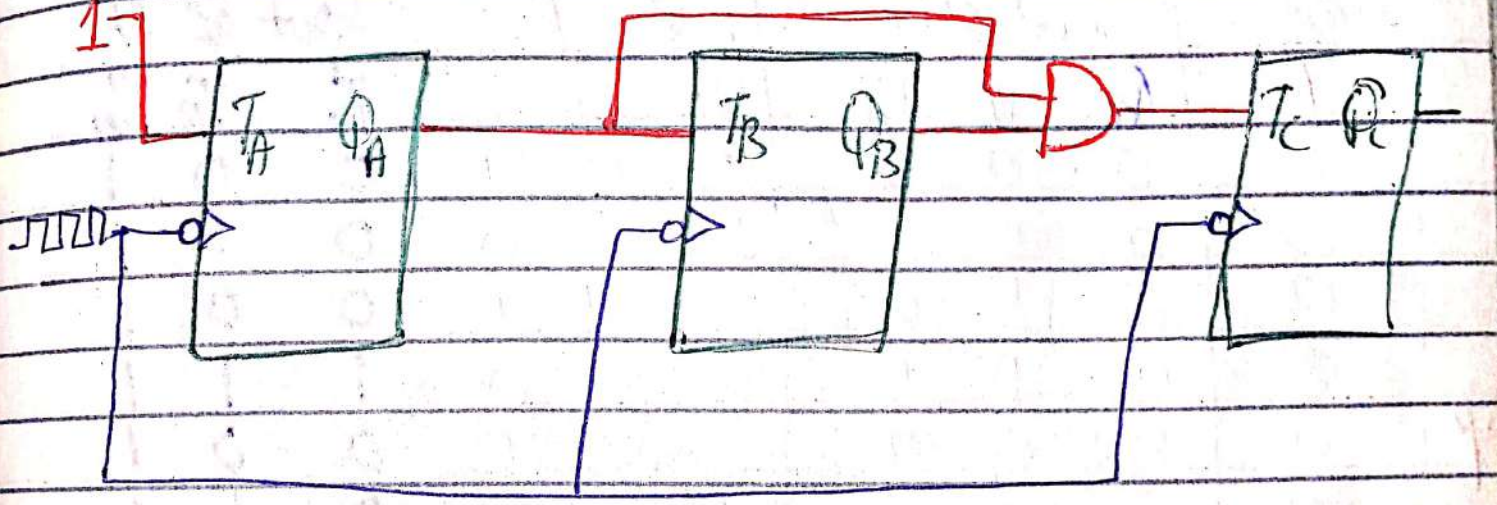
$T_C = Q_B Q_A$



$T_B = Q_A$



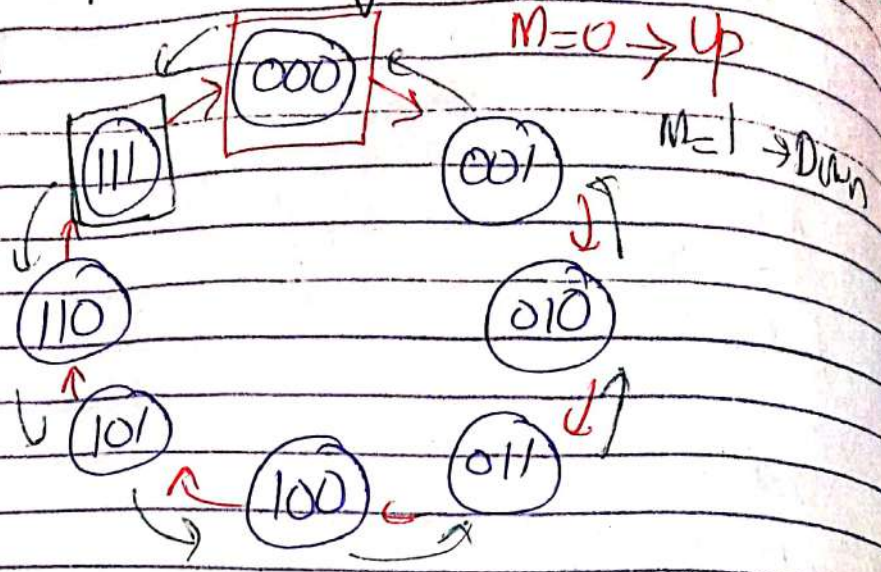
$T_A = 1$



$Q_C \rightarrow$ MSB $\quad Q_A \rightarrow$ LSB

$Q_C Q_B Q_A \rightarrow$ output of counter.

3 Bit Up down Synchronous Counter



$2^3 = 8$ states Max cnt = 8 - 1 = 7

Control Input	Present States			Next State			Input of FF		
M	Q_C	Q_B	Q_A	Q_C^+	Q_B^+	Q_A^+	T_C	T_B	T_A
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

Excitation of T

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

MQ_C \ Q_BQ_A	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	0	0	0
10	1	0	0	0

$T_C = MQ_B'Q_A' + M'Q_BQ_A$

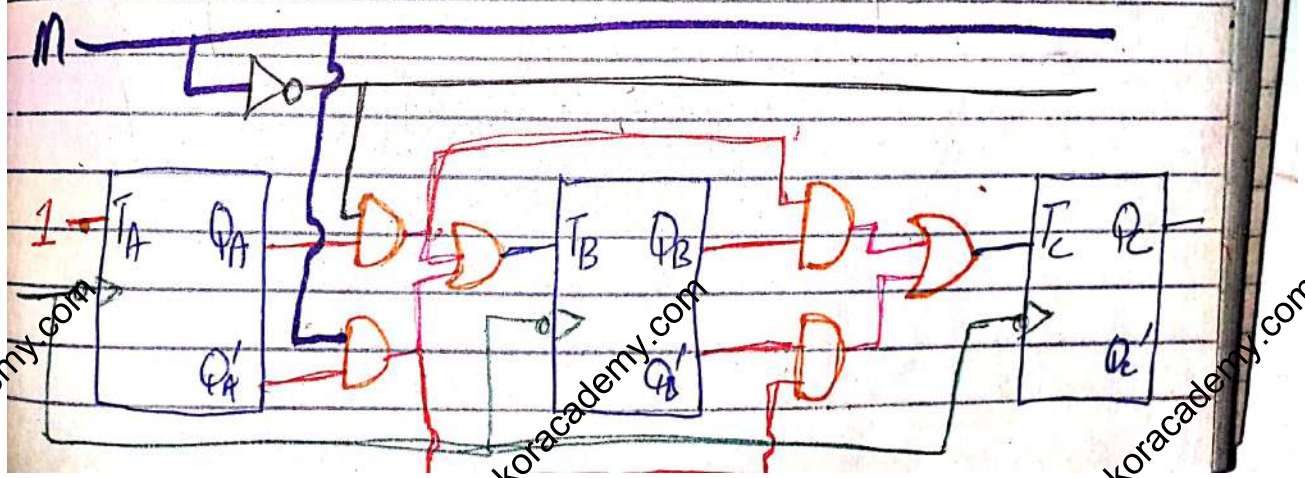
MQ_C \ Q_BQ_A	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	0
10	1	0	0	1

$T_B = M'Q_A + MQ_A'$

$= M \oplus Q$

MQ_C \ Q_BQ_A	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$T_A = 1$



If 4 ffs → same method → output from AND gate.

If JK ff → $J_A, K_A = 1$ output Q_A is connected to both J_B and K_B and similarly further.

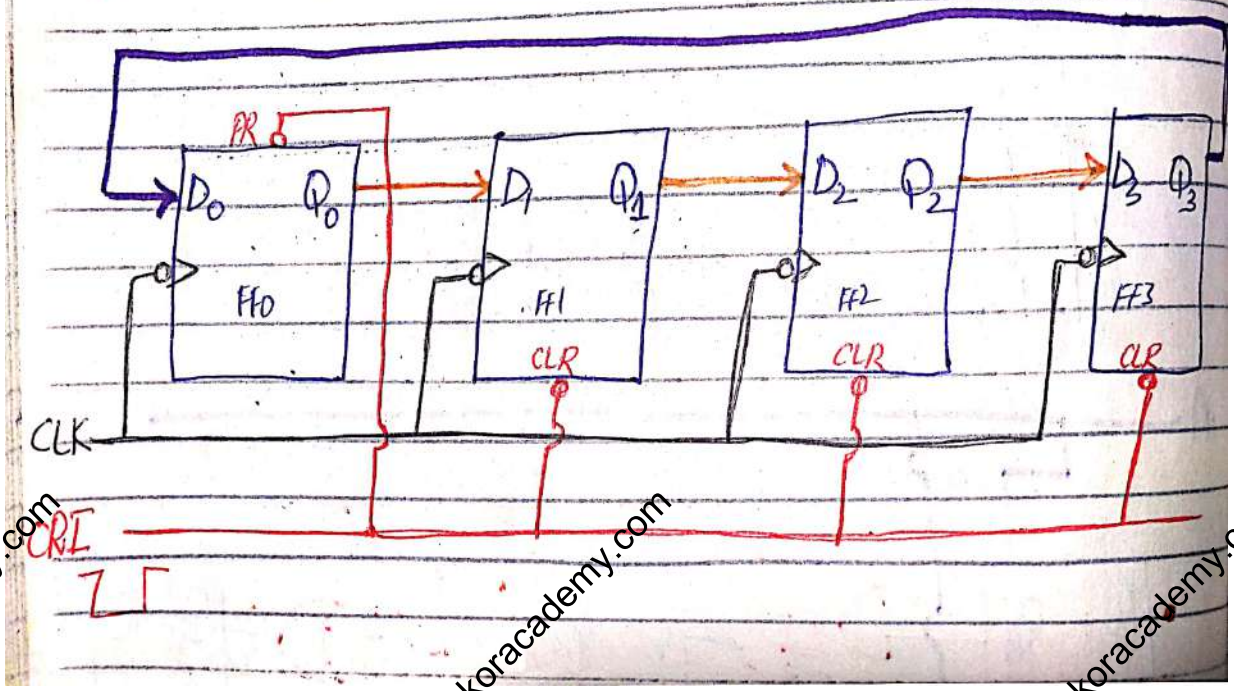
Ring Counter:

Ring counter is a typical application of shift register.

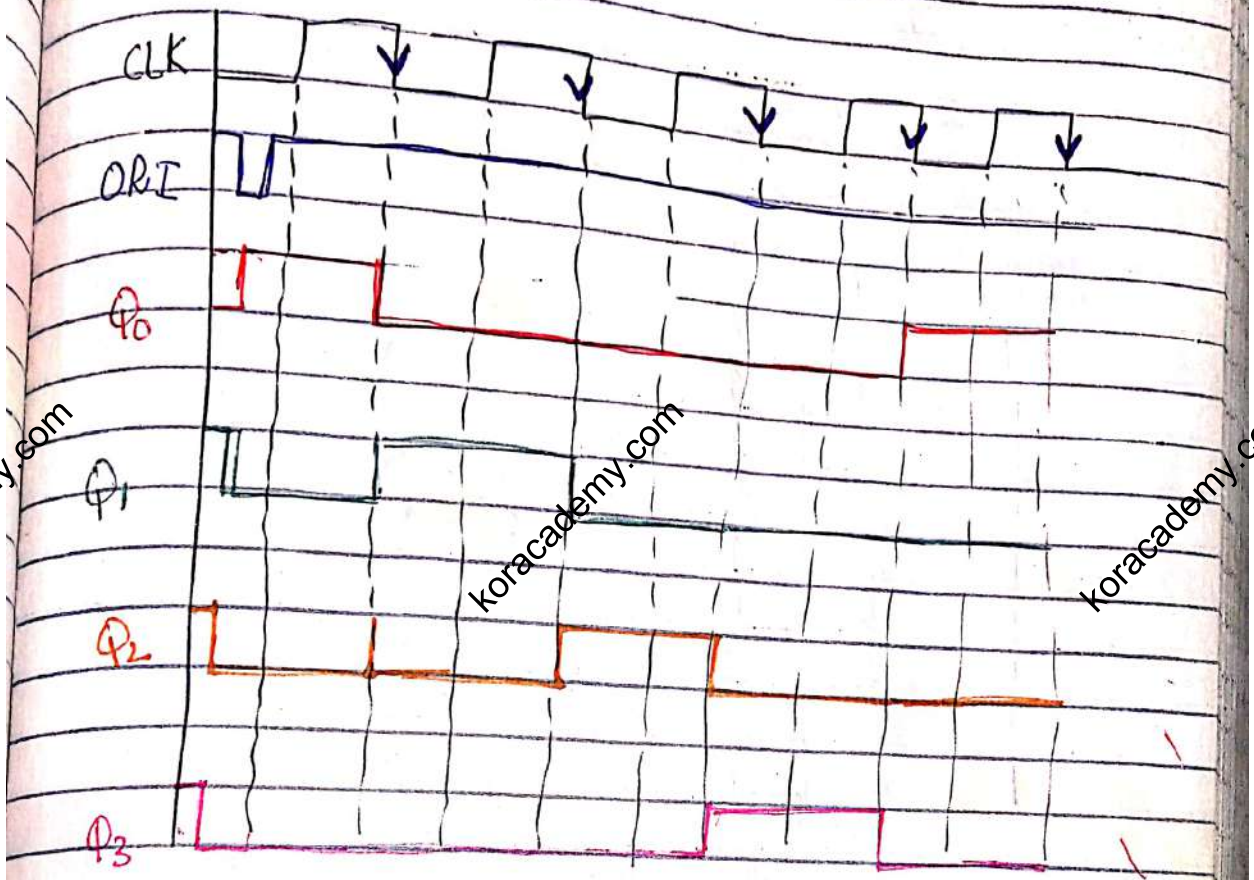
The change in ring counter and shift register is of the feedback.

No. of states = No. of ffs used

ORI → overriding inputs.
[PR=0 → Q=1 ⇒ set CLR=0 → Q=0 ⇒ Reset]



ORI	CLX	Preseted are Q_0	Q_1	Q_2	Q_3
⌐⌐	X	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	0	0	0	1
1	↓	1	0	0	0



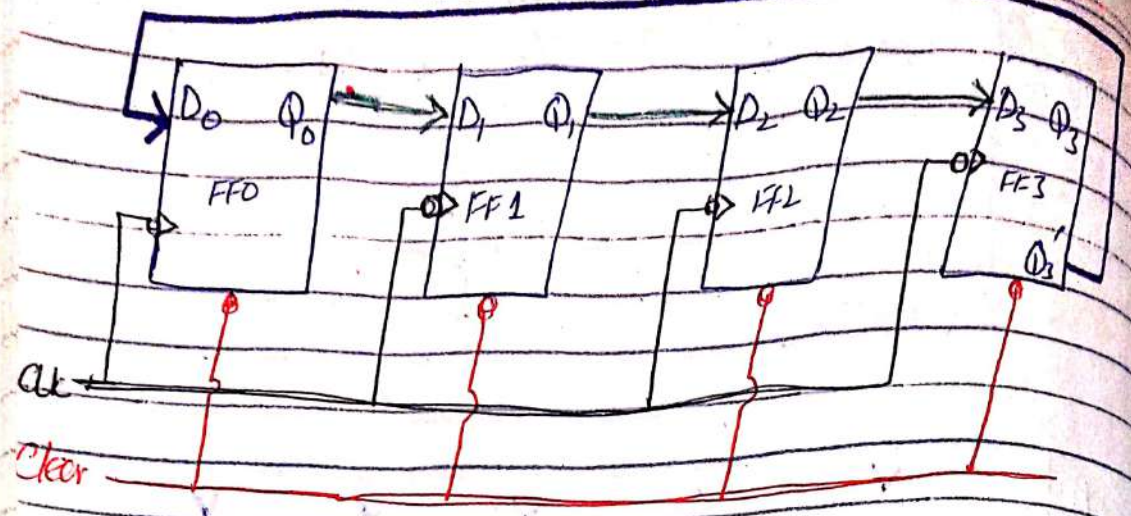
Johnson's Counter (Twisted / Switch tail Ring counter)

Much better than ring counter.

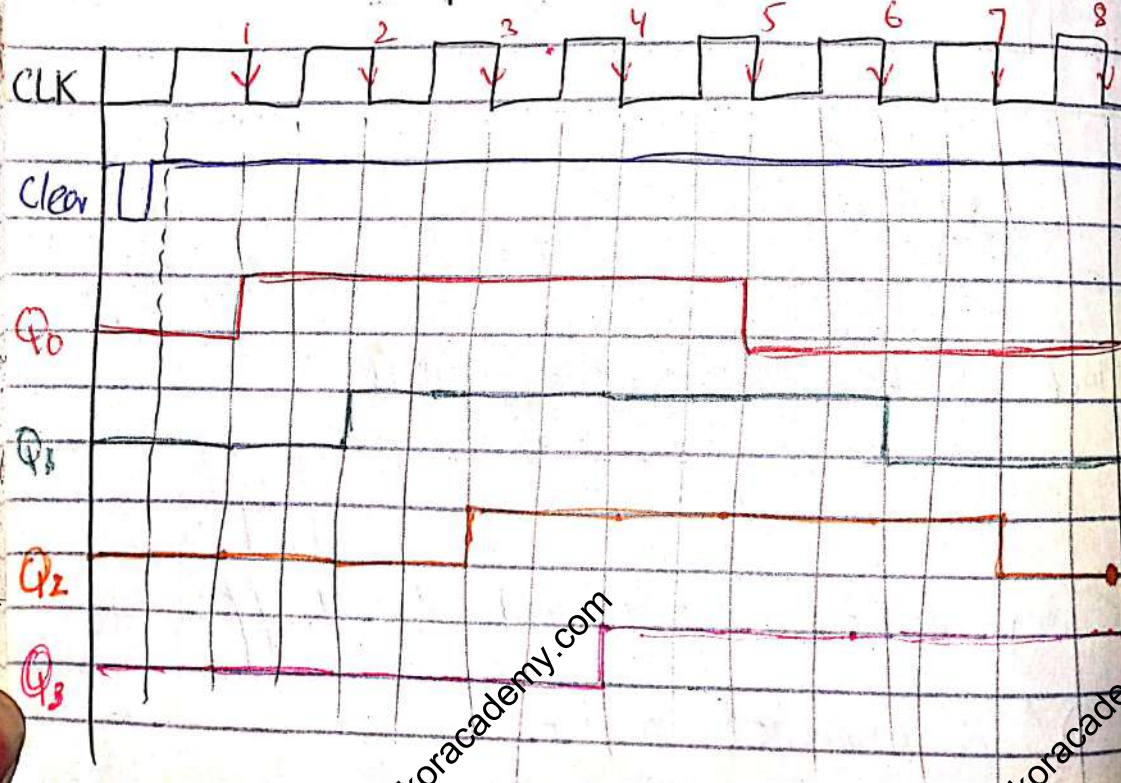
No. of states = 2 × No. of ffs

Clear input is connected to all ffs.

Q_3' is connected to D_0.



CLR	CLK	Q_0	Q_1	Q_2	Q_3
⊓	X	0	0	0	0
1	↓	1	0	0	0
1	↓	1	1	0	0
1	↓	1	1	1	0
1	↓	1	1	1	1
1	↓	0	1	1	1
1	↓	0	0	1	1
1	↓	0	0	0	1
1	↓	0	0	0	0



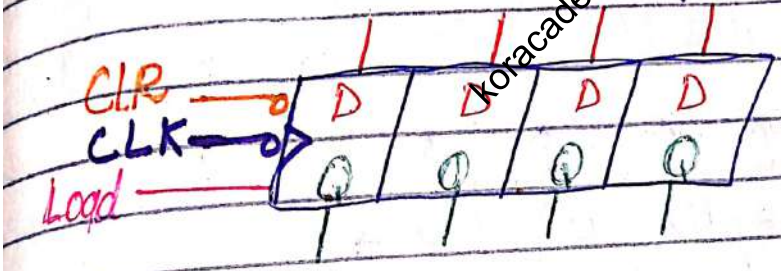
Introduction to Registers

Flipflop is a 1 bit memory cell.

To increase storage capacity (number of bits), we have to use a group of flipflops. This group of ffs is known as REGISTER.

The n bit register consist of n number of ffs and is capable of storing n bit word.

eg I want to store a 4 bit word, so I need 4 ffs eg D.



We are bound to follow the clock.

↳ Problem let $f = 1MHz \Rightarrow T = 1\mu s$.

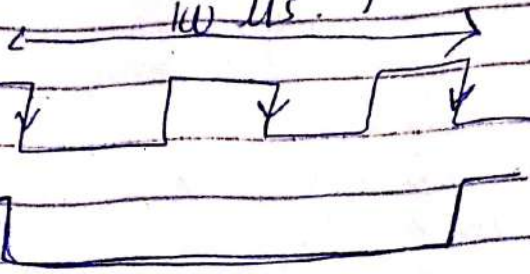
⇒ whatever we store will get change after 1 μs as negative edge will arrive after 1 μs.

If I want to store for 100 μs : for this we use an independent control called as load.

↳ Synchronous load → ff operated when CLK ↑ Load ↑

↳ Asynchronous → ff operated Load ↑ only.

Make load low for 100 µs.



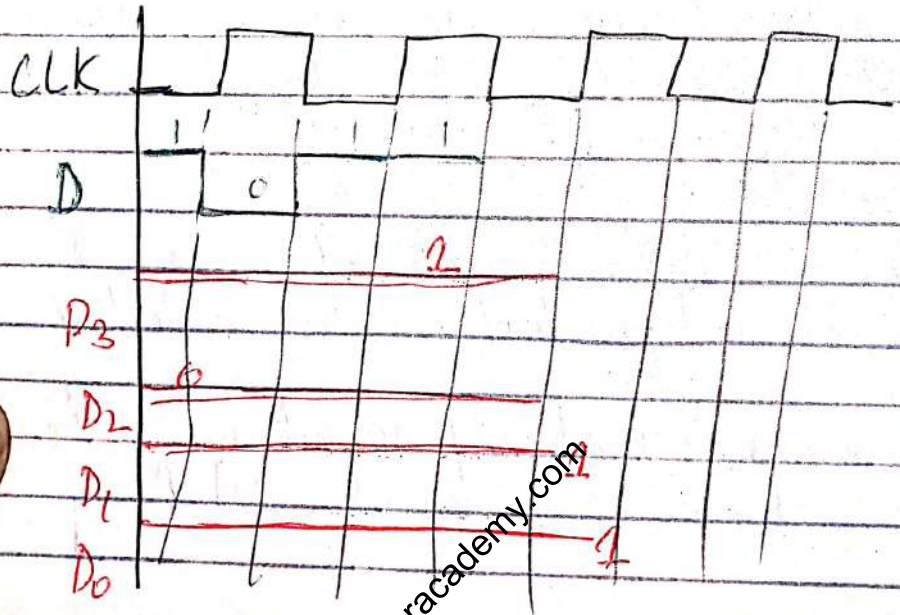
Clear input → Reset values of ffs.

Data Formats And Classification of Registers

Data can be entered in serial or parallel form.

Serial → 1 bit at a time → Temporal code
Parallel → All bits at a time → Spatial code

1 0 1 1 parallel



Similarly extract in serial or parallel form

Classification:

① Depending on Input And Output

(a) Serial Input Serial output (SISO).
(b) Serial input Parallel output (SIPO).
(c) Parallel Input serial output (PISO)
(d) Parallel Input Parallel output (PIPO).

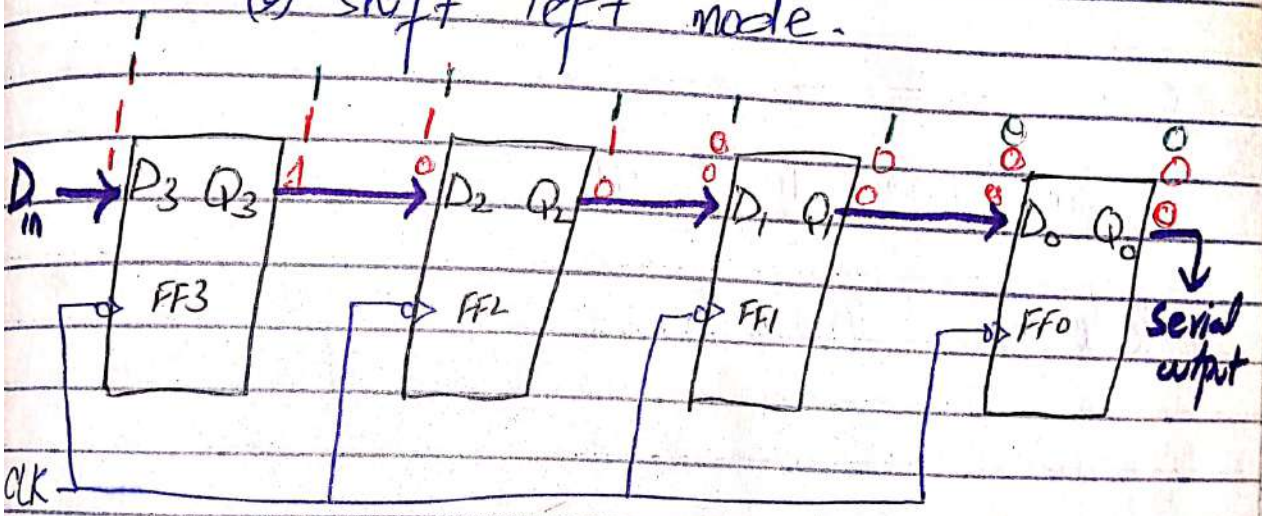
② Depending On Application

*(a) Shift register
(b) Storage register

Shift Register

① Serial In Serial Out Register

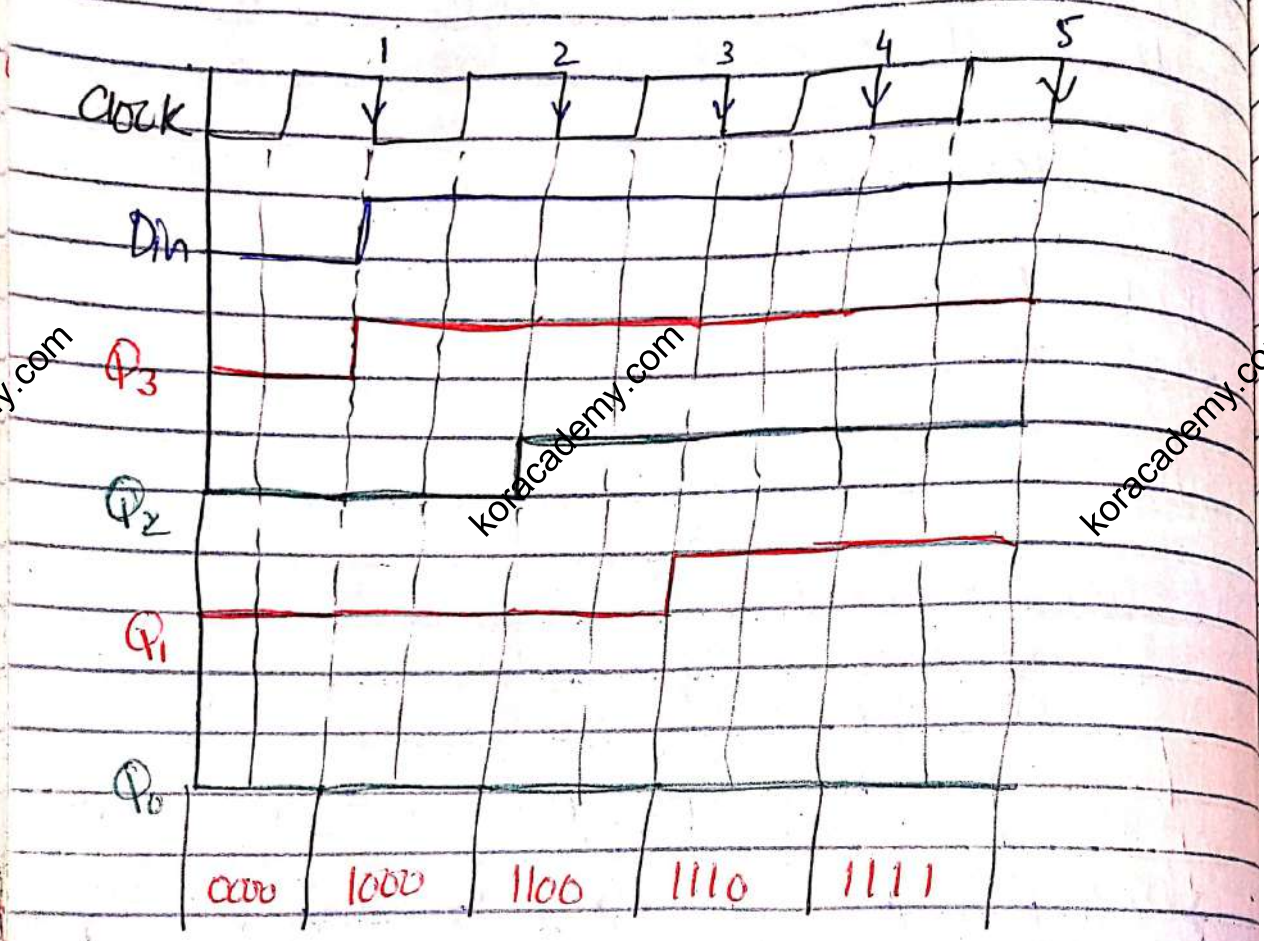
(a) shift right mode
(b) shift left mode.



Assume initially outputs are zero.

let we want to store 1111 (1) → LSB → given as input D_3.

CLK	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
	1	0	0	0
	1	1	0	0
	1	1	1	0
	1	1	1	1

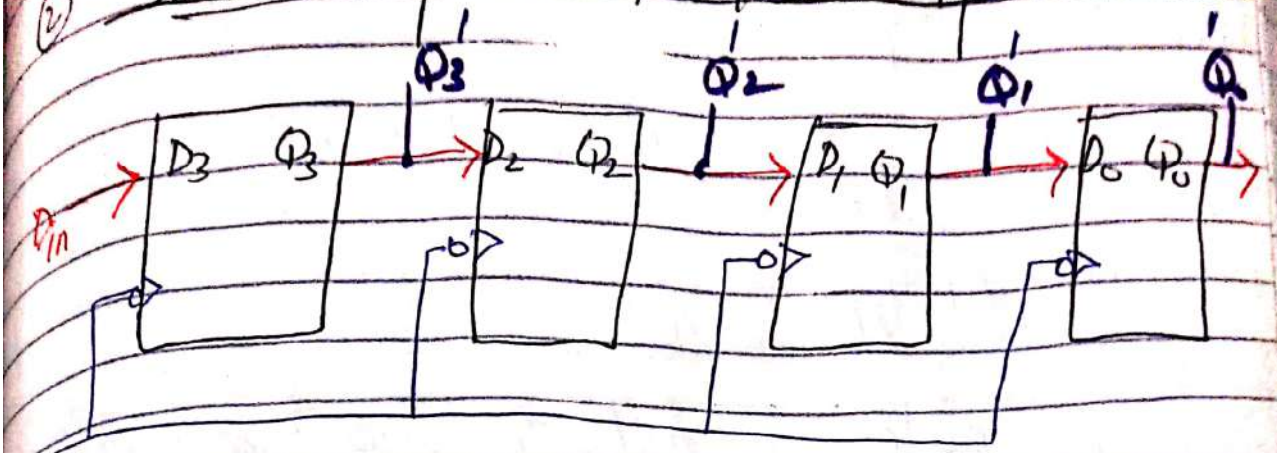


Application → Transmitter ⟶ Receiver
↳ Large distance → SISO

→ To store data in SISO and SIPO requires 4 clock pulses.

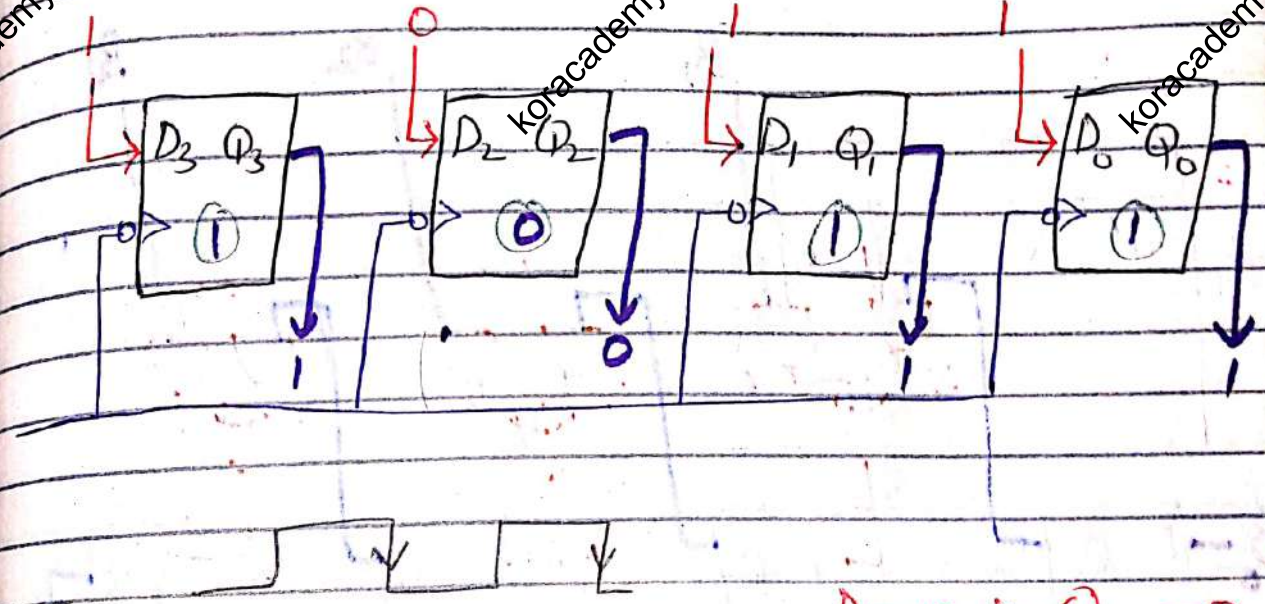
→ To give output → SISO more clock pulses
SIPO → less ⇒ 4

(2) Serial Input Parallel output



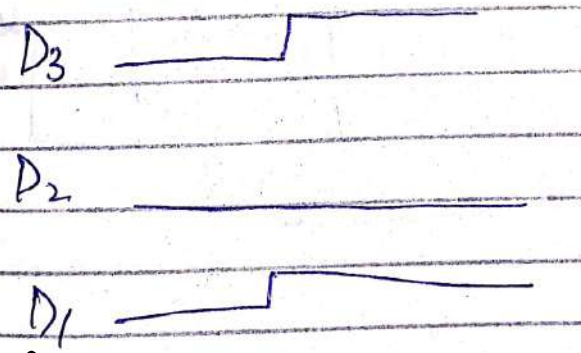
(3) Parallel Input Parallel output:

(Storage / Buffer register)



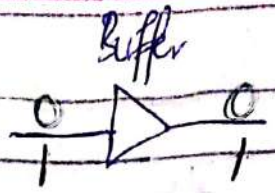
$D = 0 \Rightarrow Q_{n+1} = 0$

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



$CLK = 0 \Rightarrow Q_{n+1} = Q_n$
$D = X$

Again if we want to change → clock = 1 → input → on output



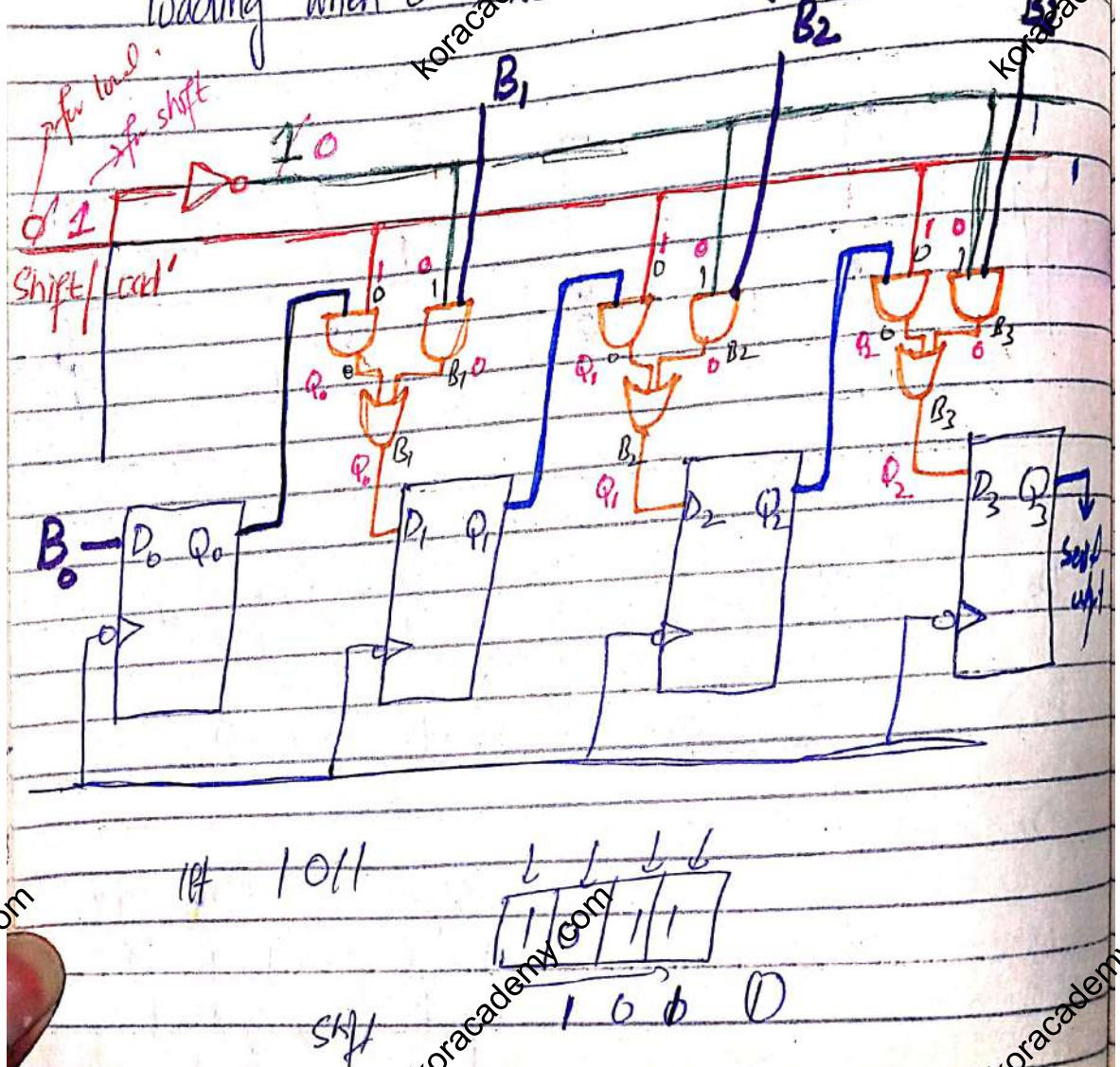
In PIPO we require 1 clock pulse to store the data.

⑨ Parallel Input Serial Output

Two modes in which this circuit can work;

(i) Load mode → Parallel input
(ii) Shift Mode → Serial output

Loading when 0 → Active low signal = load



left 1011

shift 1 0 0 0

Bidirectional Shift Register

Shift left → ×2

Shift right → ÷2

Let a 2 bit number, say 11

2^2 2^1 2^0

1 1 = 3

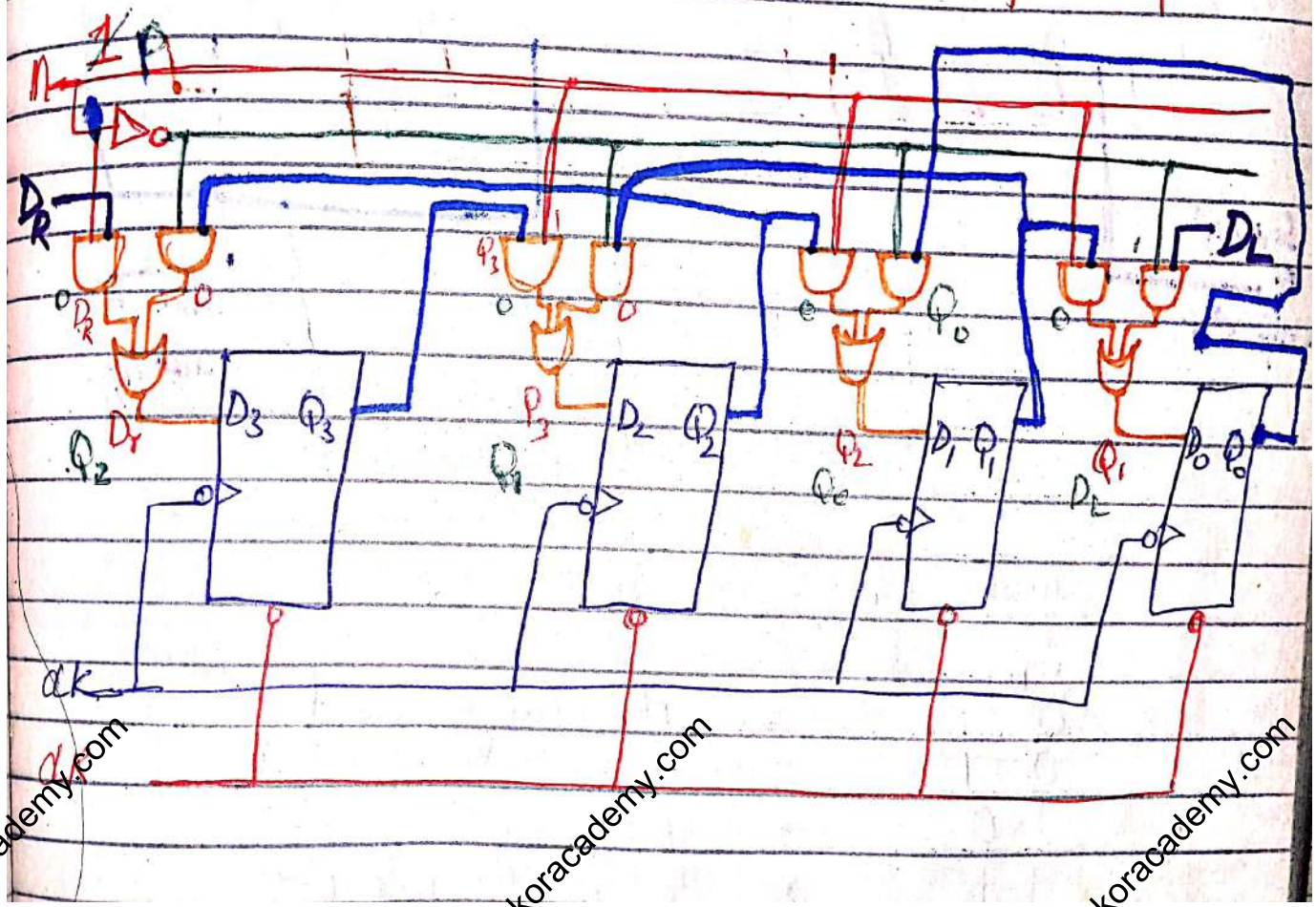
←

1 1 0 = 6

→

1 1 = 3

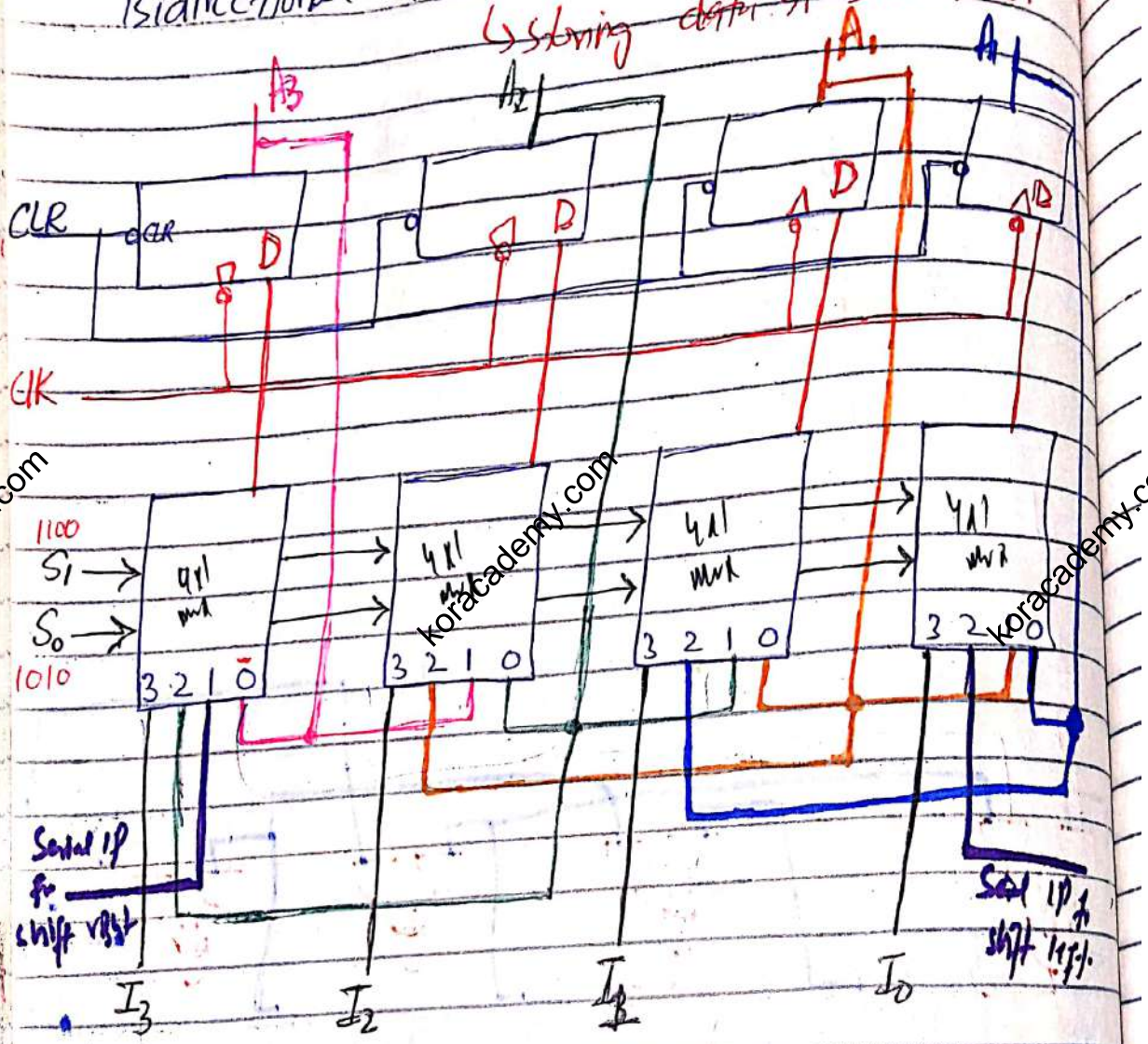
M=1 → shift right M=0 → shift left



Universal Shift Register

Bidirectional + Parallel loading = U-S-R

↳ storing data at same time.



MSB → I_3

LSB → I_1

Mode Control S_1 S_0		Register operation
0	0	→ No change / Hold
0	1	→ shift right (LSB ✓)
1	0	→ shift left (MSB ✓)
1	1	→ Parallel loading ✓

Require 4 ff to store 1011.

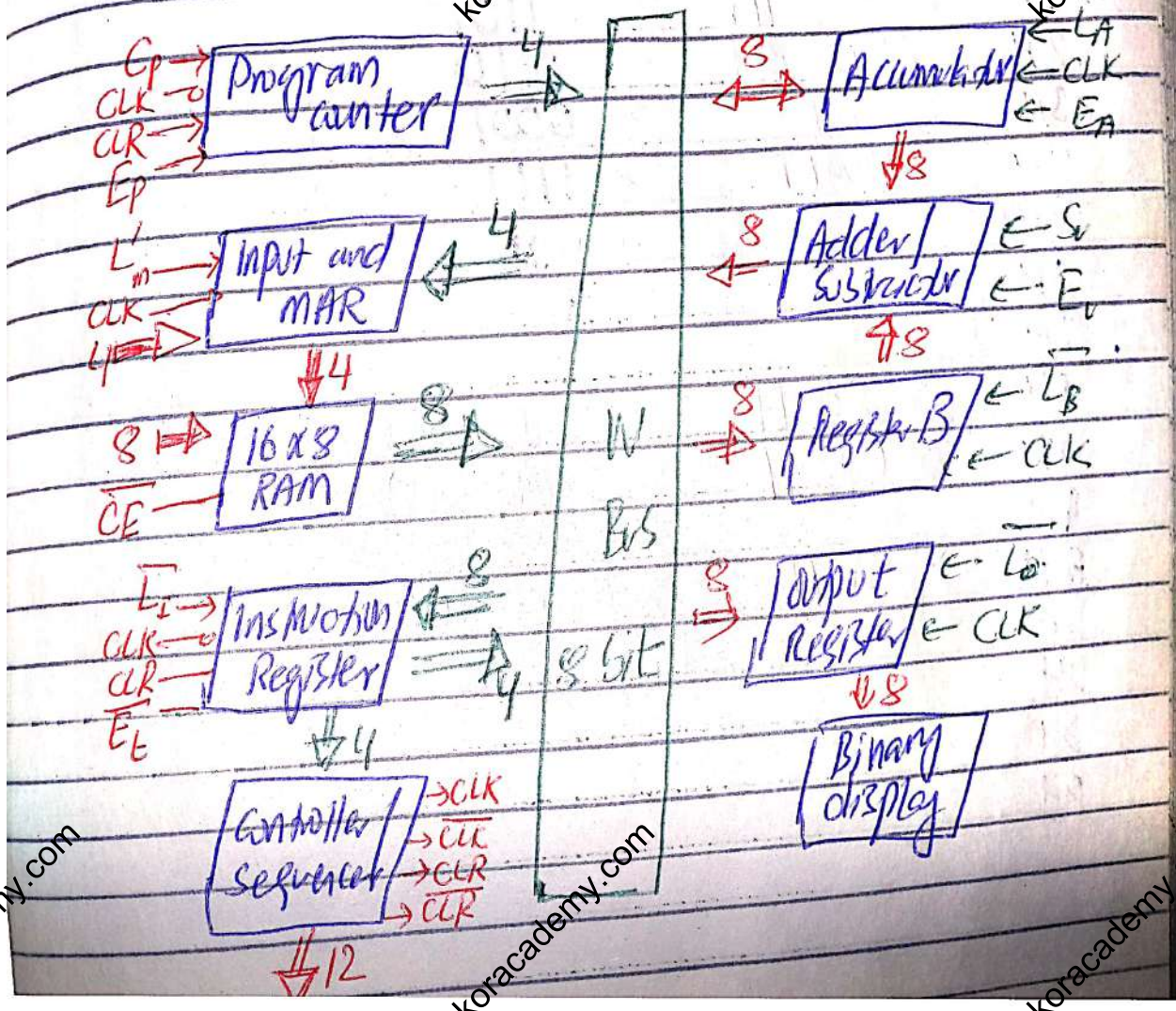
CLK	D	Q_{n+1}
0	X	Q_n
1	0	0
1	1	1

Simple As Possible (SAP 1)

Total five instructions;
LDA, ADD, SUB, OUT, HLT.

executed with the help of T states.
Total 6 T states to execute an instruction.

T_1, T_2, T_3 → common for all instructions



Instruction	Op code
LDA	1011
ADD	0110
SUB	1101
OUT	0001
HLT	1111

Q. Write a code for the following arithmetic problem in assembly language for SAP 1 architecture and translate it into equivalent machine language code using the given op codes.

21H − 02H + 0AH

Address → 8H, 9H, BH

Value → 21H, 02H, 0AH

0H → LDA 8H → 1011 1000
1H → SUB 9H → 1101 1001
2H → ADD BH → 0110 1011
3H → OUT → 0001 XXXX
4H → HALT → 1111 XXXX
5H
6H
7H
8H → 21H → 0010 0001
9H → 02H → 0000 0010
AH
BH → 0AH → 0000 1010
CH
DH
EH
FH

opcode