4th Semester	REC4C002	Digital Systems Design	L-T-P 3-0-0	3 CREDITS

MODULE – I (10 Hours)

Revision of Number System: Introduction to various number systems and their Conversion. Arithmetic Operation using 1's and 2`s Compliments, Signed Binary and Floating Point Number Representation Introduction to Binary codes and their applications.

Revision Boolean Algebra and Logic Gates: Boolean algebra and identities, Complete Logic set, logic gates and truth tables. Universal logic gates, Algebraic Reductionand realization using logic gates

MODULE – II (11 Hours)

Combinational Logic Design: Specifying the Problem, Canonical Logic Forms, Extracting Canonical Forms, EX-OR Equivalence Operations, Logic Array, K-Maps: Two, Three and Four variable K-maps, NAND and NOR Logic Implementations.

Logic Components: Concept of Digital Components, Binary Adders, Subtraction and Multiplication, An Equality Detector and comparator, Line Decoder, encoders, Multiplexers and De-multiplexers.

MODULE – III (8 Hours)

Synchronous Sequential logic Design: sequential circuits, storage elements: Latches (SR, D), Storage elements: Flip-Flops inclusion of Master-Slave, characteristics equation and state diagram of each FFs and Conversion of Flip-Flops. Analysis of Clocked Sequential circuits and Mealy and Moore Models of Finite State Machines**.**

MODULE – IV (9 Hours)

Binary Counters : Introduction, Principle and design of synchronous and asynchronous counters, Design of MOD-N counters, Ring counters. Decade counters, State Diagram of binary counters.
Shift resistors: Principle of 4-bit shift resistors. Shifting principle, Timing Diagram, SISO, SIPO, PISO and PIPO resistors.

Memory and Programmable Logic: Types of Memories, Memory Decoding, error detection and correction), RAM and ROMs. Programmable Logic Array, Programmable Array Logic, Sequential Programmable Devices.

MODULE – V (7 Hours)

IC Logic Families: Properties DTL, RTL, TTL, I^2L and CMOS and its gate level implementation. A/D converters and D/A converters**.**

College Level (20%)
Basic hardware description language: Introduction to Verilog/VHDL programming language, Verilog/VHDL program of logic gates, adders, Substractors, Multiplexers, Comparators, Decoders flip-flops, counters, Shift resistors.

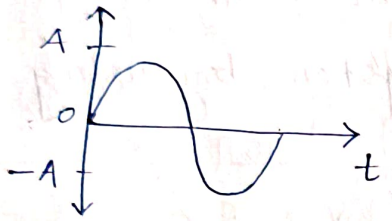
Digital Electronics is the field of electronics which deals with ~~electronics~~ digital signals. Eg: Smart phone, computer use or processed digital ir

Signal: Representation of some physical quantities which carries some data / information.

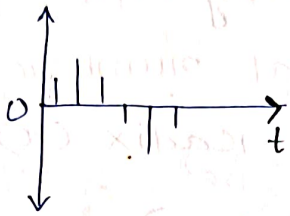
It is of three types.

- **Analog:** continuous in time & continous in amplitude
- **Discrete:** Discrete in time & discrete in amplitude
- **Digital:** continous in time & discrete in amplitude

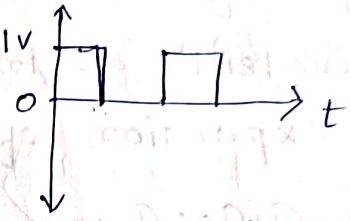
Eg:



(Analog)



(Discrete)

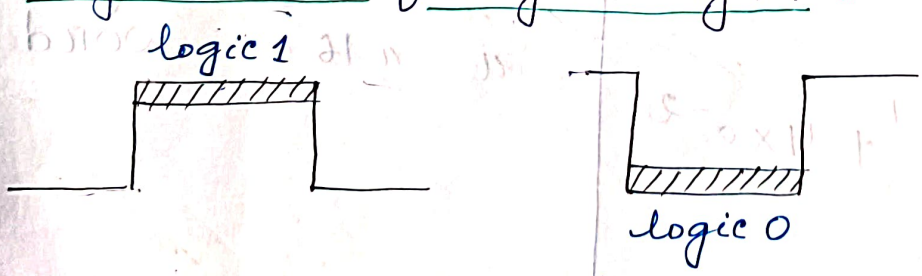


(Digital)

Advantages of digital signals over analog:

- Digital system is easier to design.
- Digital system is immune to noise signal.
- Digital signal has high information storing capability.
- Accuracy of digital data is high. It can be transmitted over long distances.
- Digital circuits can easily be fabricated in integrated circuit (ICs)

Logic levels of digital signal:



Number system: → A number system is a code that uses ordered set of symbol known as "digits".

1. Binary – 0, 1 (2)
2. Octal – (0–7) (8)
3. Decimal – (0–9) (10)
4. Hexadecimal – (0–F) (15)

- Positional Number System
- Non- " " "

Radix :- It is otherwise called as the base of the number system. It defines the number of element in that number system. It is represented by 'r'

11.05
↑
Decimal point / Radix point

~~Binary number system :~~

Conversion of any number system to decimal :

Any type of numer system can be converted to its equivalent decimal number system by using power series expansion of radix (r).

$(a_n \cdots a_1 a_0 \cdot a_{-1} \cdots a_{-m})_r$

$(a_n \times r^n) + \cdots + (a_1 \times r^1) + (a_0 \times r^0) + (a_{-1} \times r^{-1}) + (a_{-2} \times r^{-2}) \cdots + (a_{-m} r^{-m})$

→ Decimal no.

Ex :- Q. $(110.101)_2$

$= 1\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$= 4 + 2 + \frac{1}{2} + 0 + \frac{1}{8}$

$= (6.625)_{10}$

Q. $(32.14)_8$

$= 3\times8^1 + 2\times8^0 + 1\times8^{-1} + 4\times8^{-2}$

$= 24 + 2 + \frac{1}{8} + \frac{1}{16}$

$= (26.18)_{10}$

Q. $(A3.2)_{16}$

$= A\times16^1 + 3\times16^0 + 2\times16^{-1}$

$= 160 + 3 + \frac{1}{8} = (163.125)_{10}$

Group of 4bit – nibble
" " 8 " – byte
" " 16 " – word

Conversion of fractional point

Q. $(0.625)_{10} \longrightarrow (0.22)_4$

$0.625 \times 4 = 2.500$

$0.500 \times 4 = 2.000$

Q. $(0.625)_{10} = (0.3030\cdots)_5 = (0.\overline{30})_5$

$0.625 \times 5 = 3.125$

$0.125 \times 5 = 0.625$

$0.625 \times 5 = 3.125$

Binary to Decimal:

$$(1101.10110)_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4} + 0\times2^{-5}$$

$$= 8 + 4 + 0 + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32}$$

=

Decimal to Binary:

$(32.54)_{10} = (10000.10001\cdots)_2$

2 | 32
2 | 16 — 0
2 | 8 — 0
2 | 4 — 0
2 | 2 — 0
1

$(10000)_2$

$0.54 \times 2 = 1.08$

$0.08 \times 2 = 0.16$

$0.16 \times 2 = 0.32$

$0.32 \times 2 = 0.64$

$0.64 \times 2 = 1.28$

Octal to Binary:

$\pi = 8 = 2^3$

Q. $(753.146)_8 = (111101011.001100110)_2$

PTO

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 →	1	1	1

Binary to octal :

$(110110101 \cdot 100101101110)_2 = (665.4556)_8$

Q. $(001010 \cdot 101100)_2 = (12.54)_8$

Hexadecimal :-

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

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

Hexadecimal to binary :

Q. $(AF3.2B)_{16}$ =

$= (101011110011.00101011)_2$

Binary to Hexadecimal

Q. $(01011010.101100)_2$ =

$= (01011010.10110000)_2$

$= (5A.B0)_{16}$

Hexadecimal to octal

Q. $(AF3.2B)_{16}$

$= (101011110011.001010110)_2 = (5363.126)_8$

COMPLEMENTS

There are two types of complement.

- r's complement — r = Radix/base.
- (r-1)'s complement — r = 2 [1's complement, 2's complement.

r's complement (radix/true complement)

(r-1)'s complement (diminished radix complement)

1's complement

- In binary number system there is a exception for finding out the complements.
- For finding out 1's complement simply change all the 1's to zeros and 0's to 1's.

10110 —1's comp→ 01001

11111 —1's comp→ 00000

000 —1's comp→ 111

101 —1's comp→ 010

11111
10110
01001

2's complement:

- Step-1

10110 —1's→ 01001 + 1 = 01010 (2's)

11111 —1's→ 00000 + 1 = 00001

000 —1's→ 111 + 1 → 1000

101 → 010 + 1 → 011

- Step-2
  - Leave all the least significance zero's unchanged.
  - Leave the first least significant one unchanged.
  - change all higher significant 0's to 1's & 1's to 0's.

11010 → 00110

MSB LSB

MSB - more significant bit

LSB - Least " "

11112
11010
00110

0010 ⟶ 1110

000 ⟶ 000

110 ⟶ 010

111000 ⟶ 001000

11000 ⟶ 01000

11001 ⟶ 00111

9's complement (decimal number system):

- In decimal number system the (r-1)'s complement is obtained by subtracting the individual number from (r-1). 9's complement of a decimal number can be found by subtracting each digit from 9.

Q. 5327

Ans

```
  9999
  5327
------
  4672   ← 9's complement
```

10's complement: → $10^n - N$, n = no. of bit, N = given no. | 9's +1 = ((10^n - 1) - N) + 1

- It can be obtained by adding 1 to the 9's complement of the no. or the least significant non zero no is substracted from 10 and all others from 9.

Q. 5327 ans

```
  9999
  5327
------
  4672   ← 9's
+    1
------
  4673   ← 10's
```

or.

```
  99910
  5327
------
  4673   ← 10's
```

Q. 9's complement

```
  9999999
  5327000
---------
  4672999
```

10's complement

```
  99910
  5327000
---------
  4673000
```

Binary Codes:

- Different binary coding techniques are present in digital number system. The types are.

- **Weighted code:** The individual bit positions are assigned with a perticular bit.

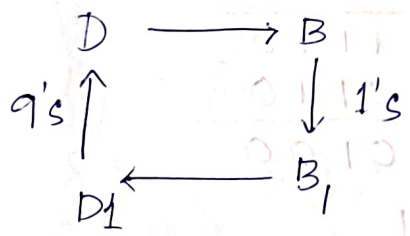
For:- ex: 8421, 2421 codes.

1010 = 10 (8421)

1010 = 4 (2421)

9's complement of Decimal ≡ 1's complement of the given code

- **Self complementary code:**



Xs-3 = 0101 (5)

original = (2)

1's complement

1010 → original decimal (7)

0101
1010
1111 → original (9)

9's of 5
$10^n - 1 - N$
$= 10 - 1 - 5$
$= 3$

- In this type of coding technique D is a decimal no which is coded in self complimentary form.
- If D is a decimal no, its corresponding binary no is B. B's 1's complement is B_1. From B_1 again its corresponding decimal no is obtained. Finally we will get that original decimal no D is the 9's complemented value of D_1.

- **Unit distance code:**

This type of code has successive decimal digit differs by only one bit.

Decimal	Binary
0 →	0 0
1 →	0 1
2 →	1 1
3 →	1 0

4. BCD (Binary coded Decimal)

$(734)_{10}$ → [0111 0011 0100]

BCD coding

$(359)_{10}$ → [0011 0101 1001]

- It is obtained by representing the individual digits of the decimal no in its four bit binary equivalent form.
- Then the no of used codes → 0000 to 1001

  The no of unused codes → 1010 to 1111

```
 29   →   0010 1001
 25       0010 0101
----     -----------
 54       0100 1110
               0110
         -----------
          0101 0100
            5    4
```

5. 2421 code :

In this type of coding some decimal no can be represented in two ways.

eg :- 2421

5 → 0101

5 → 1011

6. Excess - 3 code :

```
     1  3  2
+3   ↓  ↓  ↓
    (4  6  5)
```

0, 1, 2 } unused.

0+3 → 3 ... 9+3 → 12 } Used code.

13, 14, 15 } unused.

In excess-3 coding each decimal no is added with 3 & corresponding binary no is transmitted.

Used codes are → 0011 (3) to 1100 (12)

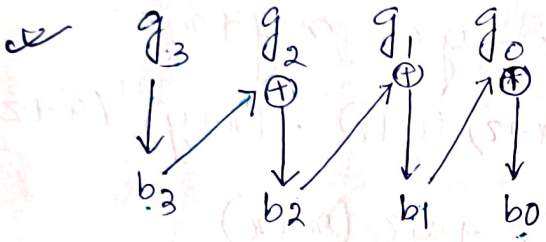
Unused codes are → 0000 to 0010 } , 1101 to 1111 }

excess-3

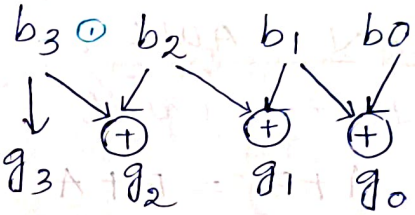
0 → 0011
1 → 0100
2 → 0101
3 → 0110
4 → 0111
⋮
9 → 1100

7. Gray Code : It is a non-weighted code. The main feature is that it exhibits only a single bit change from one code to next code.

Gray to Binary



Binary to Gray



eg :- 1 1 1 0 — Gray
1 0 1 1 — (Binary)

eg :- 1 0 1 1 ← Binary
1 1 1 0 (Gray)

BOOLEAN ALGEBRA :

Boolean Algebra is developed by Judge Bool in the year of 1854. The basic logic operation in Boolean Algebra is AND or NOT.

The system of algebra that operates on Boolean variable is called Boolean Algebra.

- AND operation is also known as logical conjuction ⇒ POS Expression.
- OR operation is known as logical disjunction. ⇒ SOP expression.

A = A·A 0 = 0·A
A = A+A A = 1·A
0 = A·A A = 0+A
1 = A+A 1 = 1+A

Variable Terms & Literals :

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

Variables, Terms, Literals

BOOLEAN LOGIC OPERATIONS:

In boolean logic operation boolean laws & theorems are included.

Boolean Addition	Multiplication
$0+0=0$	$0\cdot0=0$
$0+1=1$	$0\cdot1=0$
$1+0=1$	$1\cdot0=0$
$1+1=1$	$1\cdot1=1$

In binary $1+1=10$

BOOLEAN LAWS:

Commutative Law:

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

Associative Law:

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

Distributive Law:

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

Absorption Law:

In absorption law the two variable term is removed.

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

BOOLEAN THEOREMS:

A is any variable whose value is either 0 or 1.

$A\cdot0=0$ ✓ $A\cdot1=A$ ✓ $A+0=A$ $A+1=1$

$A\cdot A=A$ ✓ $A+A=A$ $A\cdot\bar{A}=0$ ✓ $A+\bar{A}=1$

$\bar{\bar{A}}=A$

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

$(A+B)(A+C)=A\cdot A+A\cdot C+B\cdot A+BC$
$=A+AC+BA+BC$
$=A(1+C)+BA+BC$

Consensus Law

① $xy+\bar{x}z+yz=xy+\bar{x}z$

② $(x+y)(\bar{x}+z)(y+z)=(x+y)(\bar{x}+z)$

① $xy+\bar{x}z+yz(x+\bar{x})$
$=xy+\bar{x}z+xyz+\bar{x}yz$
$=xy(1+z)+\bar{x}z(1+y)$
$=xy+\bar{x}z=RHS$

② Assignment

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

$(x+y)(\bar{x}+z)$
$=x\bar{x}+xz+\bar{x}y+yz$
$=$ (LHS=RHS)

$= A + BC$

DE-MORGAN'S THEOREM:

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

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

Transposition Law

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

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

eg: $x+0=x$ ⇔ (duality) $x \cdot 1 = x$

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

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

DUALITY THEOREM:

To findout the duality of one Boolean expression two steps as follows.

→ Interchange AND and OR.

→ Interchange 1's & 0's.

Dual of X is $\bar{X}$.

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

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

Ex:- $F = AB\bar{C} + A\bar{B}C$

$F = A \cdot 1$

$A + 0$

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

→ $\bar{A}B + A\bar{B} = (\bar{A}+B)(A+\bar{B}) = \bar{A}\bar{B} + AB$

Minimization or Simplification of Boolean Algebra.

The boolean Algebra is minimized so that for designing it less number of logic gates are required which reduces the hardware complexity, cost, space.

Ex:- Tv, radio, Mobile (the hardware circuit insite all the above devices are decreasing day by day. i.e due to advance fabrication process with minimized logic expressions.)

1. $A+A+B+B+B = ?$

$= A+B$

2. $A \cdot A \cdot B \cdot B \cdot A = ?$

$= A \cdot B$

3. $A = XY + XYZ + \bar{X}Y + X\bar{Y}Z = Y + XZ\bar{Y}$

4. $F = AB + A\bar{C} + A\bar{B}C\,(AB+C) = 1$

5. $F = \overline{X\bar{Y} + XYZ + X(Y + X \cdot \bar{Y})} = 0$

LHS

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

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

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

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

$= xy + \bar{x}z = RHS.$

3. $XY + XY\bar{Z} + \bar{X}Y + X\bar{Y}Z$

$= XY(1+\bar{Z}) + \bar{X}Y + X\bar{Y}Z$

$= XY + \bar{X}Y + X\bar{Y}Z$

$= Y(X+\bar{X}) + X\bar{Y}Z$

$= Y + X\bar{Y}Z$

4. $F = AB + \bar{A}C + A\bar{B}C(AB+C)$

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

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

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

$= 1 + 1 + 1 = 1$

5. $F = \overline{\overline{X\bar{Y} + XYZ} + X(Y + X\bar{Y})}$

$= \overline{\overline{X\bar{Y} + XYZ}} \cdot \overline{X(Y + X\bar{Y})}$

$= X\bar{Y} + XYZ + \bar{X} + (\bar{Y} + X\bar{Y})$

$= (X\bar{Y} + XYZ) \cdot \{\bar{X} + (\bar{Y} \cdot X\bar{Y})\}$

$= X\bar{Y}$

Q. $F = \overline{A + B\bar{C} \cdot (\bar{A} + \bar{B} + \bar{C})} + X\bar{Y}$

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

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

$= A + 1 + \bar{C} + X\bar{Y}$

$= 1 + X\bar{Y}$

$= T$

$= 0$

LOGIC GATES:

- Logic gates are the electronic circuits that perform a boolean algebric function.
- Logic gates are generally divided into three types.

(i) Basic logic gates:

NOT, BUFFER, OR, AND

(ii) UNIVERSAL LOGIC GATES:

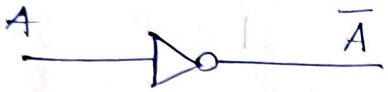
NAND, NOR

(iii) EXCLUSIVE LOGIC GATES:

EX-OR, EX-NOR. (X-OR, X-NOR)

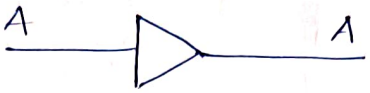
NOT :

I/P	O/P
0	1
1	0



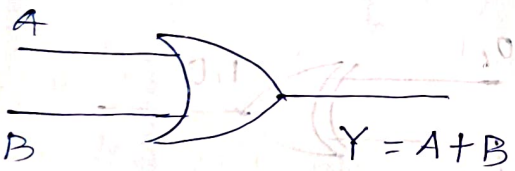
BUFFER :

I/P	O/P
0	0
1	1



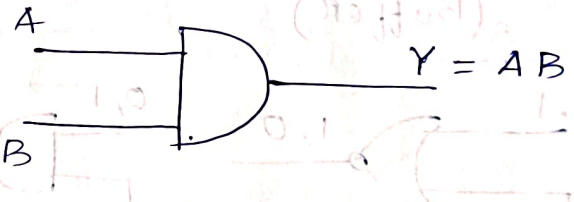
OR :

I/P		O/P
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1



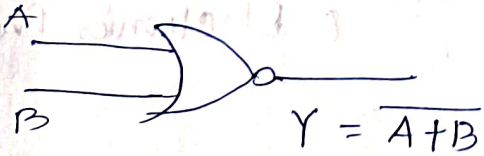
AND :

A	B	Y = AB
0	0	0
0	1	0
1	0	0
1	1	1



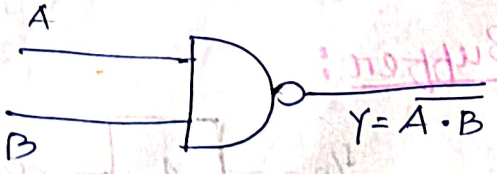
NOR :

A	B	$Y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



NAND :

A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



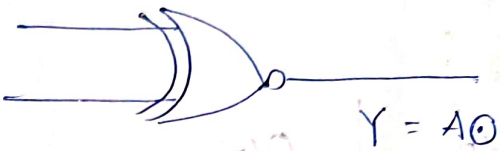
Ex-OR

A	B	$Y = A \oplus B = \bar{A}B + A\bar{B}$
0	0	0
0	1	1
1	0	1
1	1	0

$Y = A \oplus B$

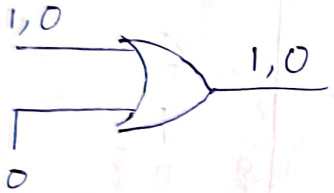
Ex-NOR

A	B	$Y = \overline{A \oplus B} = A \odot B = \bar{A}\bar{B} + AB$
0	0	1
0	1	0
1	0	0
1	1	1

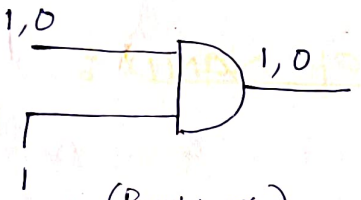


$Y = A \odot B$

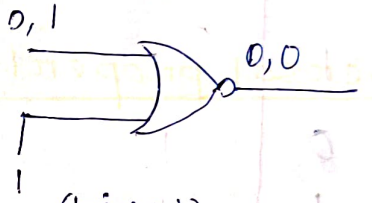
Applications of Logic gates:



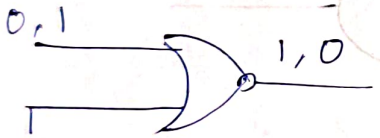
(Buffer)



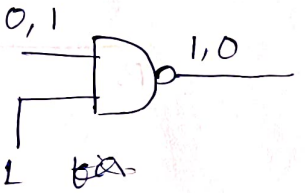
(Buffer)



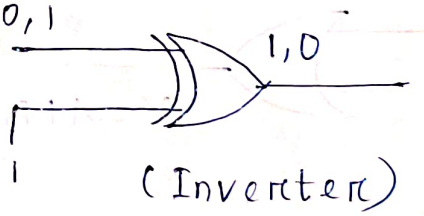
(fixed)



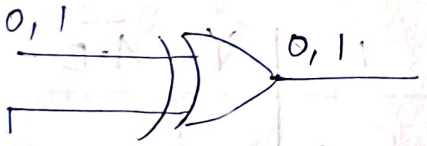
(Inverter)



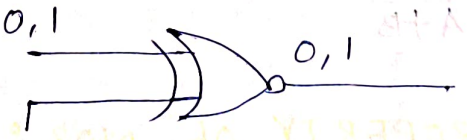
(Inverter)



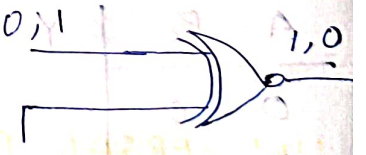
(Inverter)



(Buffer)



(Buffer)

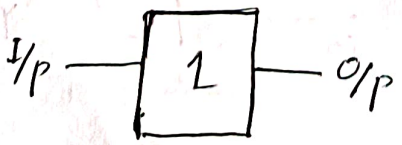


(Inverter)

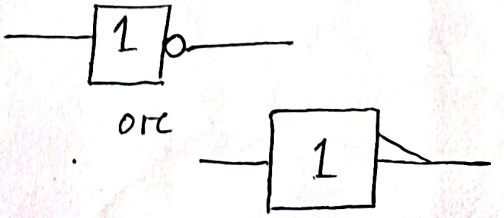
IEEE Symbols of Logic gates:

Institute Of Electrical & Electronics Engg

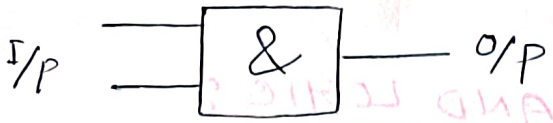
Buffer:



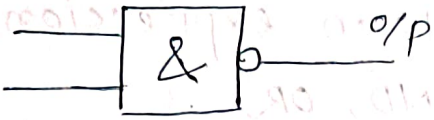
NOT :-



AND :-

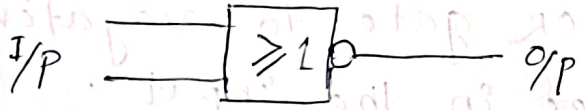


NAND :-

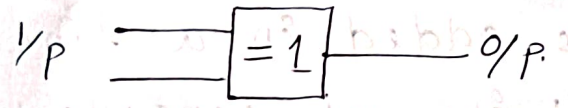


OR :-

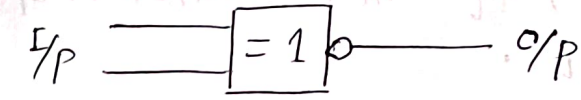
NOR :-



Ex-OR :-



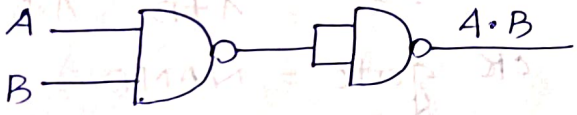
Ex-NOR :-



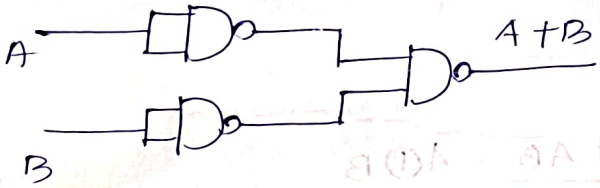
Universal property of NAND :

NOT → A → $\overline{A}$

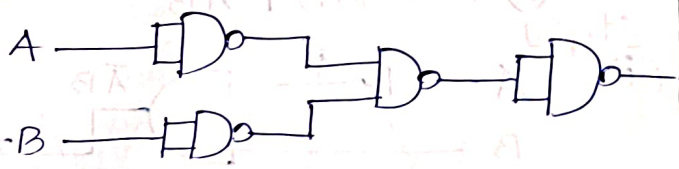
AND :



OR : →

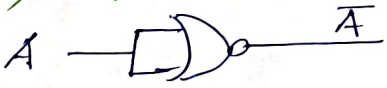


NAND :

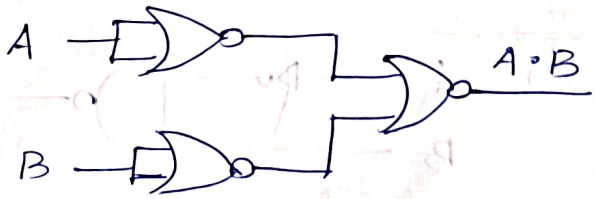


UNIVERSAL PROPERTY OF NOR :

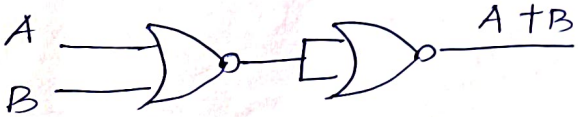
NOR → A → $\overline{A}$



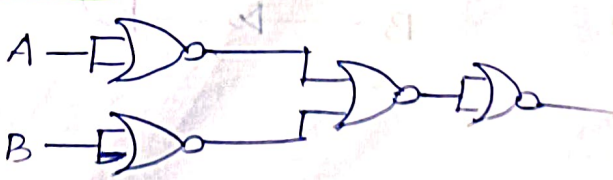
AND :



OR



NAND :



NAND LOGIC:

Steps to implement NAND LOGIC:

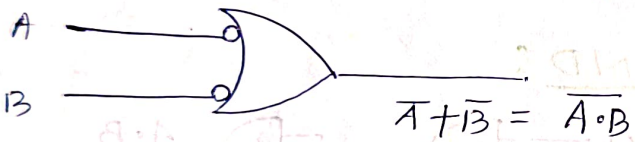
(i) Design the given boolean expressions using basic logic gates (NOT, AND, OR).

(ii) convert all the AND gate to NAND gate by adding a bubble in the output.

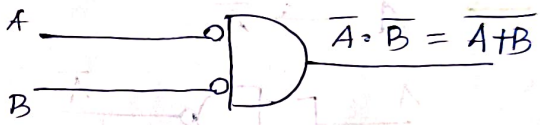
(iii) convert all the OR gate to negative OR by adding the bubbles in the input.

(iv) check the bubbles added in a line if it is cancelled or not; if not cancelled add a NOT gate to cancel it.

(v) Convert all the gates into NAND Gate.



-Ve OR gate = NAND



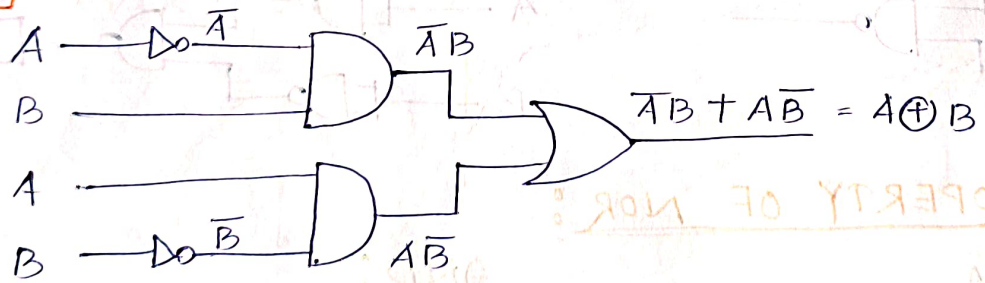
-Ve AND = NOR gate.

Ex-OR

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

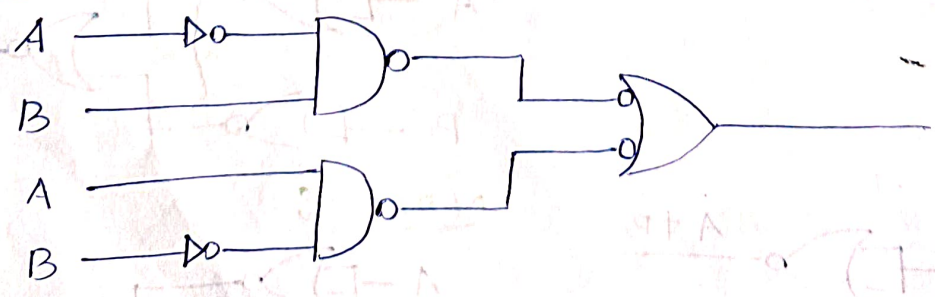
Step-1

(i)



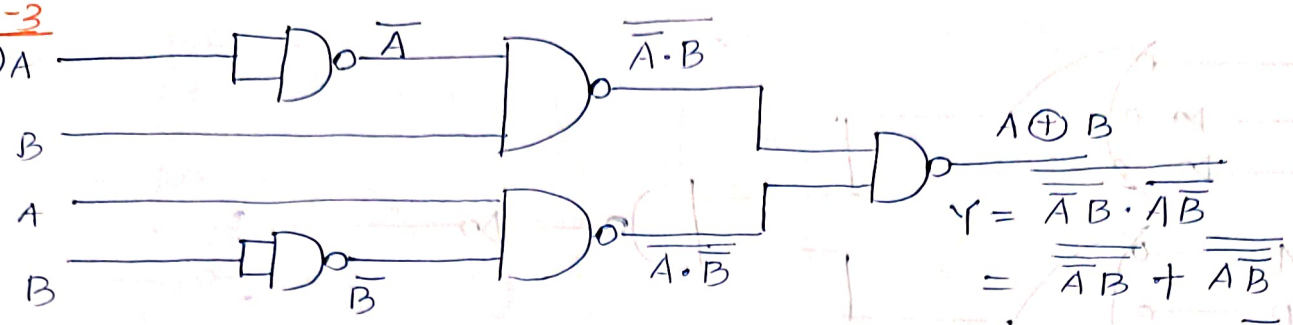
Step-2

(ii)



Step -3

(iii) A



$A \oplus B$

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

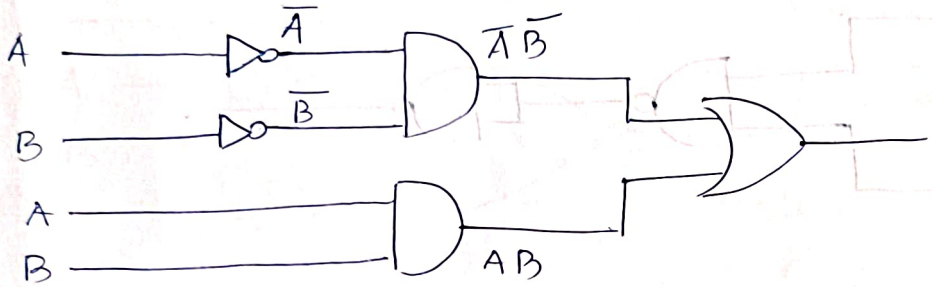
$= \overline{\overline{A}B} + \overline{A\overline{B}}$

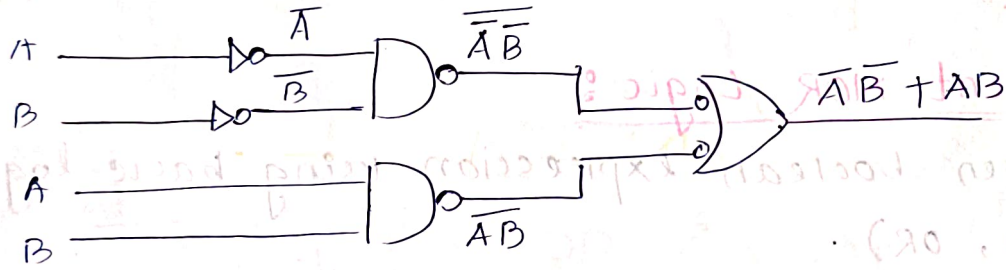
$= \overline{A}B + A\overline{B}$

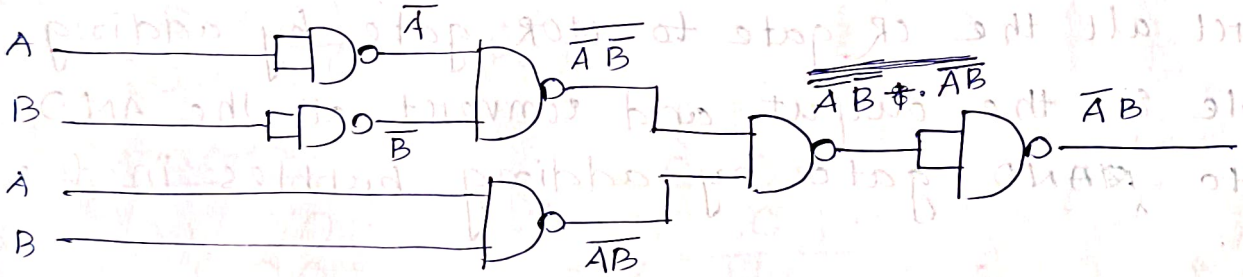
$= A \oplus B$

EX-NOR

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

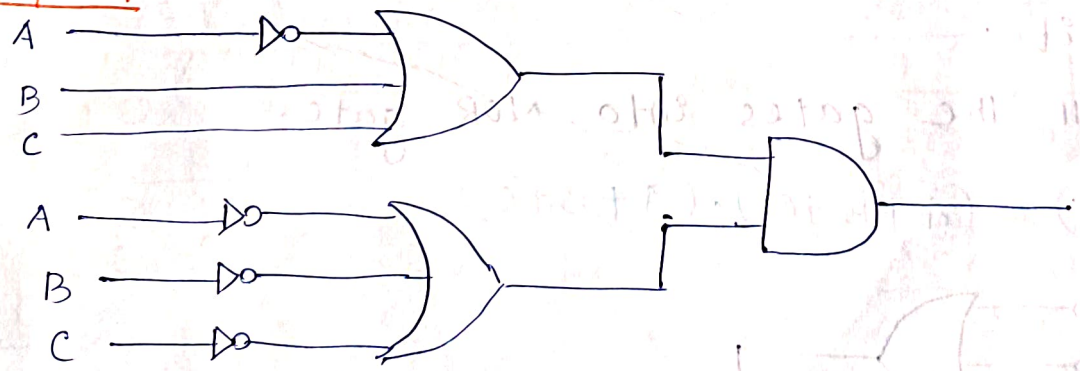




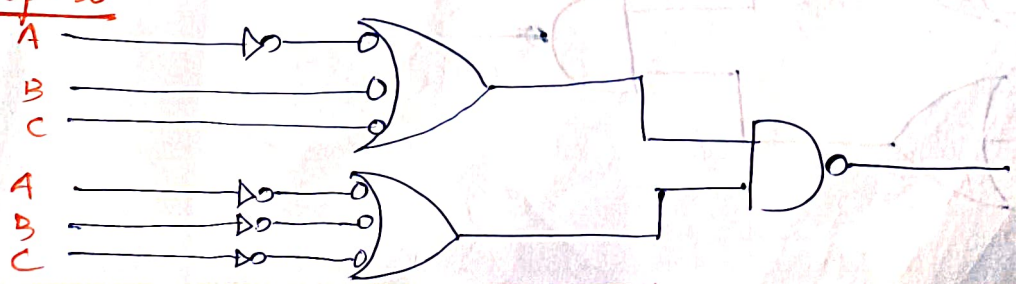


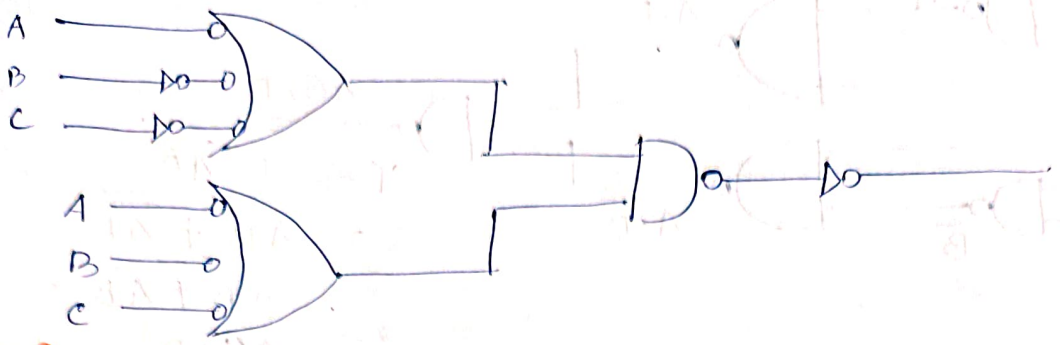
Q. $F(A,B,C) = (\overline{A}+B+C)\cdot(\overline{A}+\overline{B}+\overline{C})$

Step -1

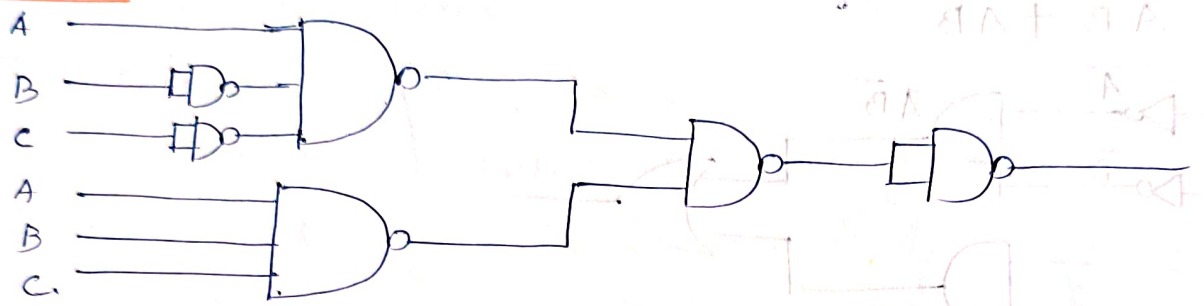


Step -2





Step-3



NOR LOGIC :

Steps to implement NOR Logic :

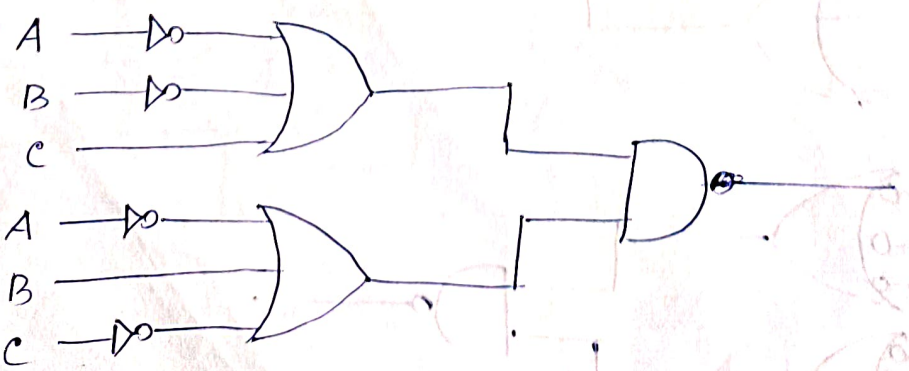
(i) Design the given boolean expression using basic logic gates (NOT, AND, OR).

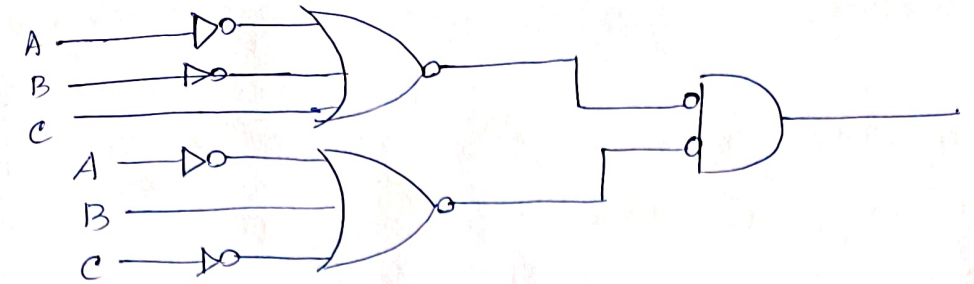
(ii) convert all the OR gate to NOR gate by adding a bubble in the output and convert all the AND gates to (-ve) AND gate by adding bubbles in the input.

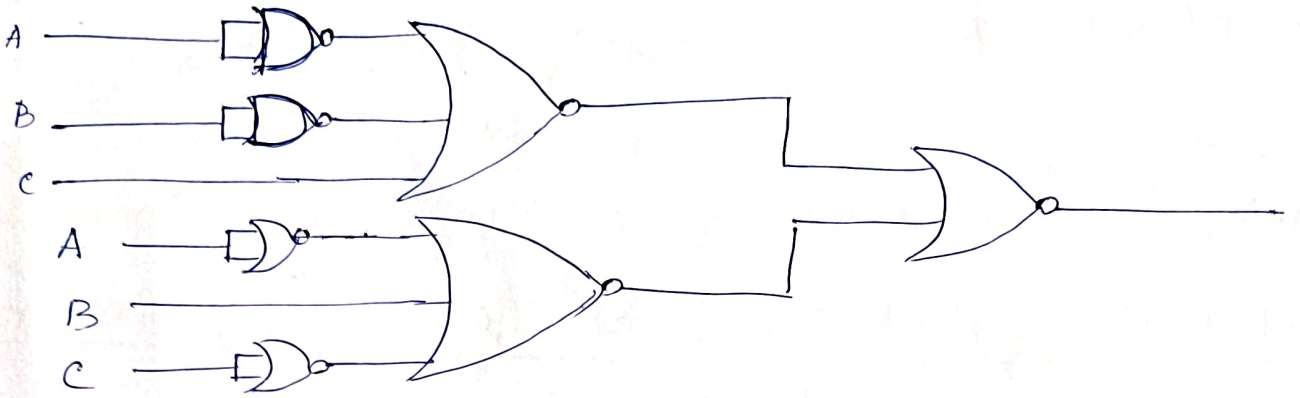
(iii) check the bubbles added in a line if it is cancelled or not; if not cancelled add a NOT gate to cancel it.

(iv) convert all the gates into NOR gate.

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







Q. Design Ex-OR & Ex-NOR using NOR gate:

Addition

eg:- 10111
01101
100100

1001
1110
10111

Subtraction

1110001
111001
0111000

100100
− 01101
1101

Multiplication

10111
101
10111
00000
10111
1100011

Division

11
1000 | 11000 |
1000
1000
1000
0

Ans. (11)2

Standard Boolean Expressions

There are two types of standard Boolean Expressions are present.

1. SOP → Sum of product → Eg: $F = \bar{A}B + AB$
2. POS → Product of sum → Eg: $F = (\bar{A}+B)\cdot(A+B)$

SOP → AND, OR & LOGIC

POS → OR AND LOGIC

SOP: It is a type of Boolean expression where the product terms are added.

Standard form of SOP (canonical form)

If in a boolean function each term contains all the variables of that function i.e called standard form or canonical form.

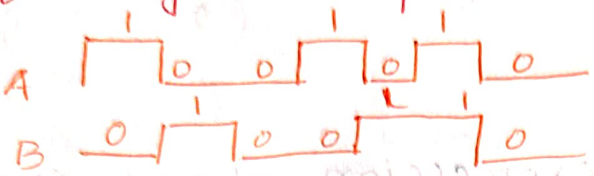
Eg: $F(A,B,C,D) = \bar{A}B\bar{C}D + \bar{A}\bar{B}CD + ABC\bar{D}$ (standard form)

$F = \bar{A} + \bar{B}\bar{C} + A\bar{D}$ (Non standard form)

Q. What is the only condition under which an OR gate output is zero.

Q. Design EX-OR gate using NOR gate.

Q. Draw the output waveform for the NAND & NOR gate for given input waveforms.



Q. Design Ex-OR gate using minimum no of NAND gate

Conversion of Non standard SOP to canonical Form

Q. $F = AB + B\bar{C}$ (Non standard form)

$= AB \cdot 1 + B\bar{C} \cdot 1$

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

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

$= ABC + AB\bar{C} + \bar{A}B\bar{C}$ (canonical form)

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

$= A \cdot 1 + \bar{B}\bar{C} \cdot 1 + D \cdot 1$

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

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

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

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

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

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

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

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

$D \Rightarrow D(\bar{A} + A)$

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

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

$= \bar{A}BCD + \bar{A}\bar{B}CD + ABCD + A\bar{B}CD + \bar{A}B\bar{C}D$

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

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

$= ABCD + A\bar{B}CD + AB\bar{C}D + A\bar{B}\bar{C}D + AB\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}BCD + \bar{A}B\bar{C}D + A\bar{B}\bar{C}D$

POS :

It is a type of Boolean expression where sum terms are multiplied.

Ex :- $(\bar{A}+B)\cdot(A+B)$

Standard form of POS : If all the term of POS expression contains all the variables of that function then that is called standard POS.

Conversion of non standard POS to canonical form.

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

$= (A+B+0)\cdot(B+C+0)\cdot(C+A+0)$

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

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

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

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

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

$A+\bar{B} \Rightarrow (A+\bar{B}+0)$

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

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

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

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

$\bar{A} \Rightarrow (\bar{A}+B\cdot\bar{B})$

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

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

Proof.

$(A+B)(A+C)$

$AA + AC + BA + BC$

$A + AC + BA + BC$

$A(1+C) + BA + BC$

$A + BA + BC$

$A(1+B) + BC$

$= A + BC$

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

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

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

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

Truth table Representation of SOP & POS :

- In SOP expression each term is equal to 1.

e.g:- $F = A\bar{B}\bar{C} + AB\bar{C} + \bar{A}\bar{B}C$

$= 1\ 1\ 0 \quad 1\ 1\ 1 \quad 0\ 0\ 1$

This is called binary equivalent of SOP terms.

- In POS expression each term is equal to 0.

e.g:- $F = (A+B+\bar{C})\cdot(\bar{A}+\bar{B}+C)\cdot(A+B+C)$

$= 0\ 0\ 1 \quad 1\ 1\ 0 \quad 0\ 0\ 0$

This is called binary equivalent of POS terms.

Truth table :

A	B	C	Y_1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

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

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

Min Term :- (Σ)

The standard SOP terms are called Min terms. or the product of Boolean expression were all possible variables appear in complemented or uncomplemented manner in each term. (m_i)

Max Term :- (Π)

Standard POS expressions are called max terms. (M_j)

F = A	B	O/P	Min	Max
0	0	0	m_0	M_0
0	1	1	m_1	M_1
1	0	1	m_2	M_2
1	1	0	m_3	M_3

$F = \Sigma(m_1, m_2)$ ← Minterm

$\Pi(M_0, M_3)$ ← Max term

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

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

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

A	B	C	O/P	min	Max
0	0	0	0	m_0	M_1
0	0	1	1	m_1	M_2
0	1	0	1	m_2	M_3
0	1	1	1	m_3	M_4
1	0	0	1	m_4	M_5
1	0	1	0	m_5	M_6
1	1	0	0	m_5	M_7
1	1	1	0	m_6	M_8

$F = \Sigma(m_1, m_2, m_3, m_4)$

$\Pi(M_1, M_6, M_7, M_8)$

Q. $F = \bar{A}BC + A\bar{B}C + ABC + \bar{A}\bar{B}C$. Find out the corresponding binary sop expression. and find out min & max term.

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

$011 \quad 101 \quad 111 \quad 001$

$3 \quad 5 \quad 7 \quad 1$

Minterm = $\Sigma m(3, 5, 7, 1)$

or $m_1 + m_3 + m_5 + m_7$

Max term = $\Pi M(0, 2, 4, 6)$

or $M_0 \cdot M_2 \cdot M_4 \cdot M_6$

$000 \quad 010 \quad 100 \quad 110$

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

K-Map (Karnaugh Map)

Karnaugh Veitch Map

It is a method of simplification of Boolean function in a systematic manner. There are different terms.

Cell :- The smallest unit of K-map corresponding to one row of the truth table is called cell.

Pair :- A group of two adjecent cells in a K-map is called pair. A pair cancels one variable in a K-map simplification.

Quad :- It is a group of four adjecent cells which cancels two variables from simplification.

Octate :- It is a group of eight adjecent cells which cancels three variable from simplification.

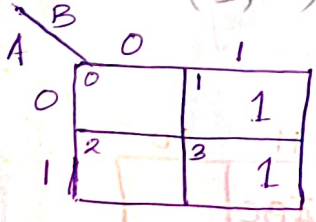
K-Map method of simplification is a graphical approach for simplifying Boolean expressions without using any Boolean rules and laws. K-map is Abstract of Venn. Diagram.

Two variable K-Map :

If the number of variables are two in a function, then the total no of cells will be 2^n i.e $n = 2 \Rightarrow 2^2 = 4$.
n is the no of variable.

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

$\Sigma m(3,1) = B$

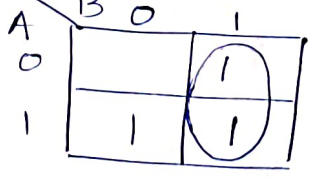


$F = A + B$

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

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

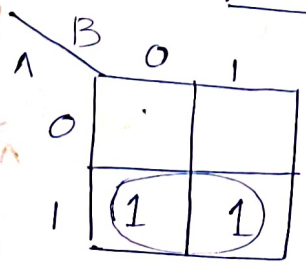
$\Sigma m(2,3) = A$

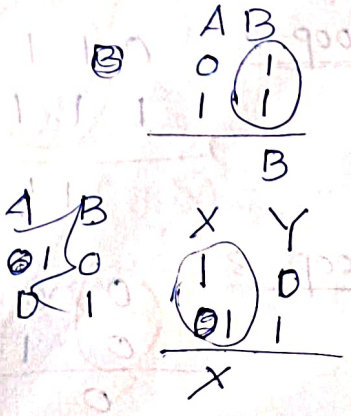


Q. $F = X + X\bar{Y}$

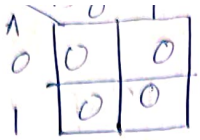
$= X\bar{Y} + XY + X\bar{Y}$

$= XY + X\bar{Y}$



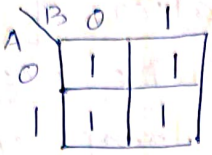


Q. F = 0 → $= A\bar{A}$



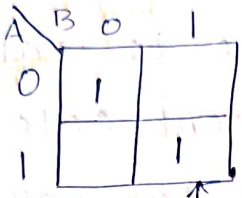
$= 0$

F = 1 →



$= 1$

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

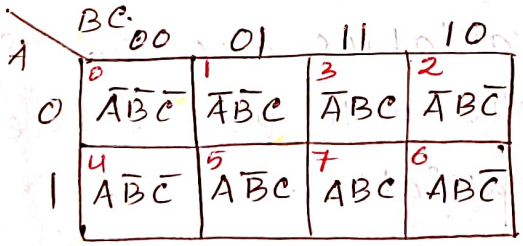


↑ singleton / single term.

Simplification not possible.

Three Variable K-Map:

- Any three variable boolean expression can be simplified by using three variable K-Map. The minterms are not arranged in binary sequence, but in a sequence gray code.
- In three variable K-map total 8 cells are present (2^3)
- In three variable K-map the no of combination selection starts from octate, then check for quad & then pair.

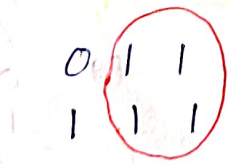


Q. Simplify the given boolean expression using K-Map.

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

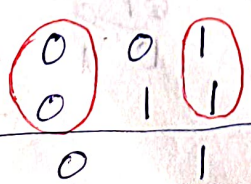
Loop-1 = $A\bar{B}\bar{C}$

Loop-2

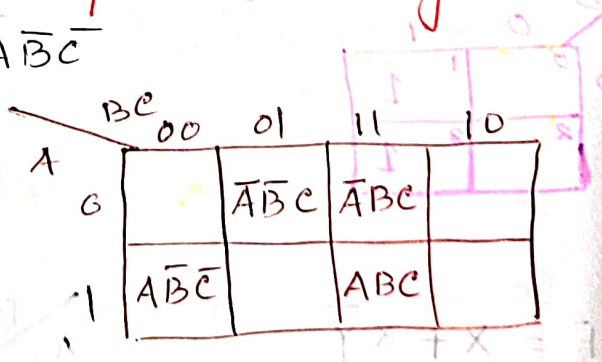


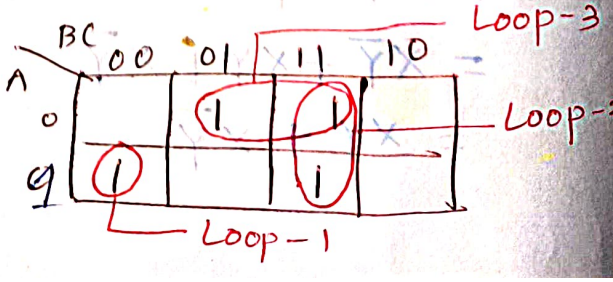
11 ⇒ BC

Loop-3



0 1 ⇒ $\bar{A}C$

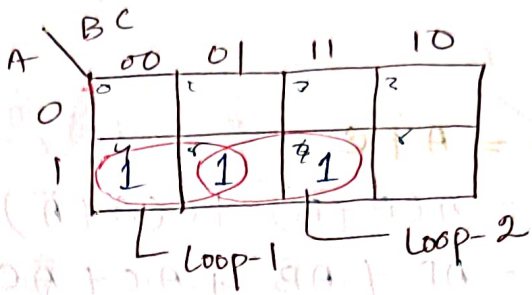




$\Rightarrow \boxed{F = BC + \bar{A}C + A\bar{B}\bar{C}} \Rightarrow$ Simplified Boolean expression.

Q. $F = ABC + A\bar{B}C + A\bar{B}\bar{C}$

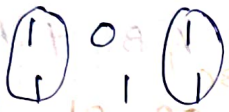
$\Sigma m(4,5,7)$



Loop-1 :

$10 \Rightarrow A\bar{B}$

Loop 2 :



$11 \Rightarrow AC$

$\Rightarrow \boxed{F = A\bar{B} + AC}$ Simplified Boolean expression.

Q. $F = AB + C$ Represent in K-MAP.

$= \cancel{AB \Rightarrow AB(C+\bar{C})}$

$= AB(C+\bar{C}) + C$

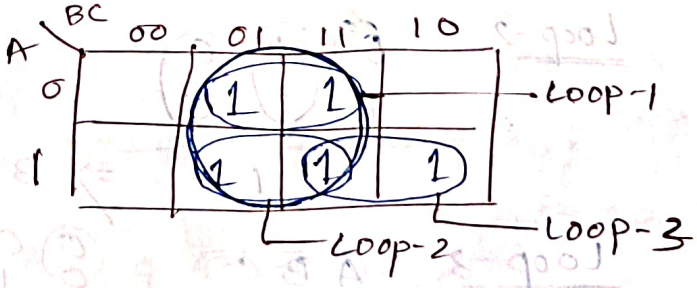
$= ABC + AB\bar{C} + C(A+\bar{A})$

$= ABC + AB\bar{C} + AC + \bar{A}C$

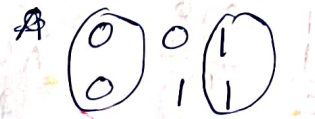
$= ABC + AB\bar{C} + AC(B+\bar{B}) + \bar{A}C(B+\bar{B})$

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

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

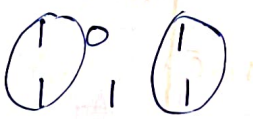


Loop-1



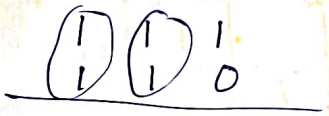
$01 \Rightarrow \bar{A}C$

Loop-2



$11 \Rightarrow AC$

Loop-3



$11 \Rightarrow AB$

$F = \bar{A}C + AC + AB$

$= BC(\bar{A}+A) + AB$

$\boxed{F = C + AB}$ ✓

Q. $F = A + C$

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

$F = B$

Ans $F = A + C$ $\Sigma m(1,3,4,5,6,7) = A + C$

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

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

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

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

Loop-1

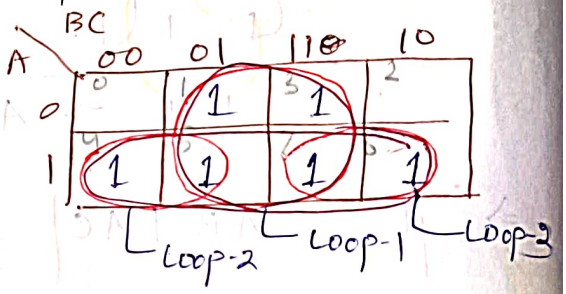
0	0	1
0	1	1
1	0	1
1	1	1

$C \Rightarrow C$

Loop-3

1	1	1
1	1	0

$AB \Rightarrow AB$



Loop-2

0	0	0
1	0	0

$A\bar{B} \Rightarrow A\bar{B}$

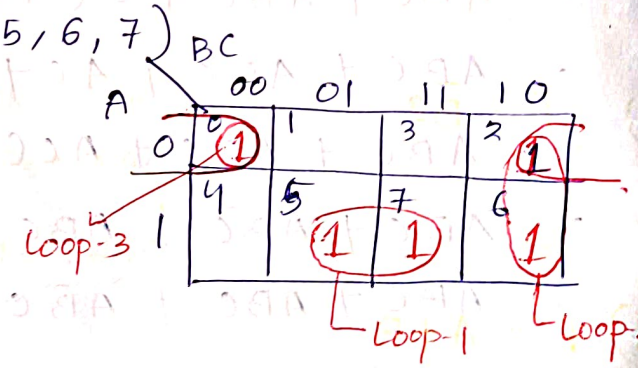
$F = C + A\bar{B} + AB$

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

Q. Simplify given min term using K-map and find out its corresponding SOP & POS.

$F = \Sigma M(0,2,5,6,7)$

let $F(A,B,C) = \Sigma M(0,2,5,6,7)$



Loop-1

1	0	1
1	1	1

$1\ 1 \Rightarrow AC$

Loop-2

0	1	0
1	1	0

$1\ 0 \Rightarrow B\bar{C}$

000, 010 → $\bar{A}\bar{C}$; 101, 111 → AC; 010, 110 → $B\bar{C}$

$\bar{A}\bar{C} + B\bar{C} + AC$ Ans.

Loop-3 ~~$A\bar{B}\bar{C}$~~

0	0	0
0	1	0

$\bar{A}\ \ \bar{C} \Rightarrow \bar{A}\bar{C}$

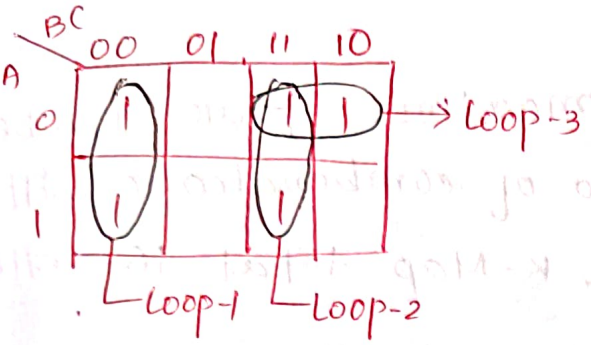
$F = AC + B\bar{C} + \bar{A}\bar{B}\bar{C}$

$F = \Pi M(1,3,4)$

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

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

Q.



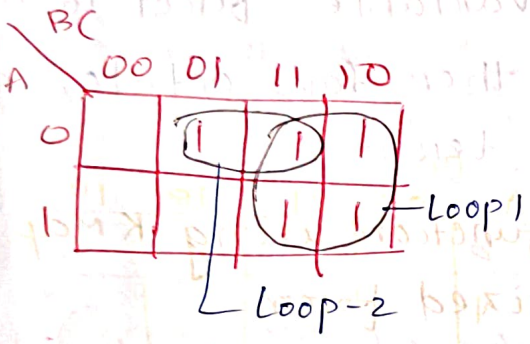
Loop-1 ⇒ 0 0 0
1 0 0
$\bar{B}\bar{C}$

Loop-3 ⇒ 0 1 1
1 1 1
BC

Loop-3 ⇒ 0 1 1
0 1 0
$\bar{A}B$

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

Q.



Loop-1

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

B

Loop-2

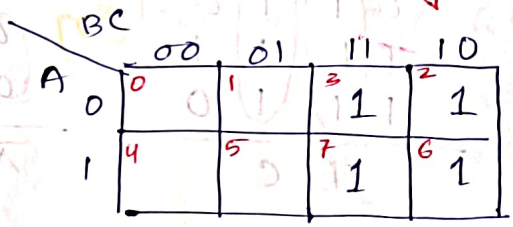
0	0	1	
1	1	1	

C

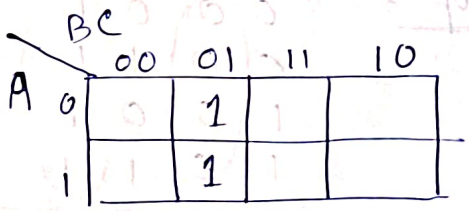
$F = B + C$.

Q. Represent the following function of ABC by K-Map.

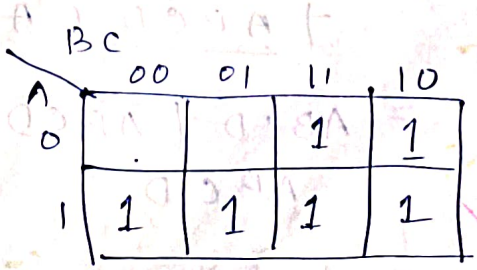
(i) $F = B$
$= (A + \bar{A})B$
$= AB + \bar{A}B$
$= ABC + AB\bar{C} + \bar{A}BC + \bar{A}B\bar{C}$



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



(iii) $F = A + B$
$= A\bar{B} + AB + AB + \bar{A}B$
$= A\bar{B}C + A\bar{B}\bar{C} + ABC + AB\bar{C} + ABC + AB\bar{C} + \bar{A}BC + \bar{A}B\bar{C}$
$= ABC + AB\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + \bar{A}B\bar{C}$

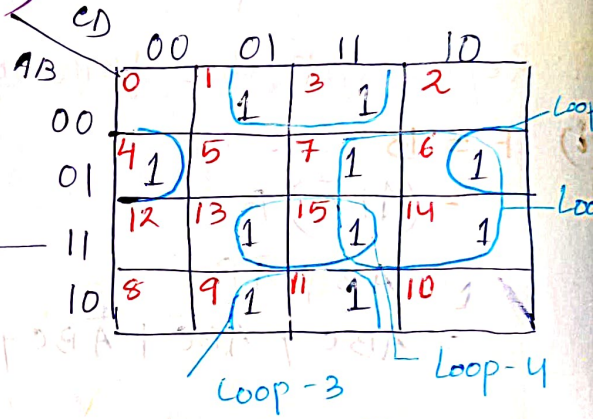


4-Variable K-Map :

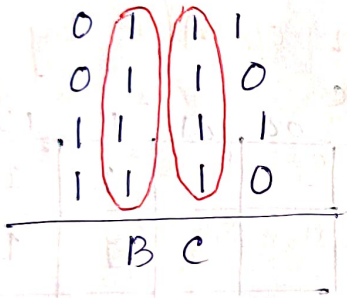
- The 4 variable K-map uses maximum four number of variables. So the total no of combinations will be $2^4 = 16$. So in 4 variable K-Map total 16 cells are present.
- In this K-Map for grouping of variable, first 16 no of combinations are checked, then for octate, then quads, then pair then single term.

ex :- (i) Represent the following function using K Map and simplify it to get minimized form.

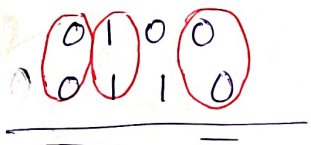
1. $F = \Sigma m(1,3,4,7,6,9,11,13,14,15)$



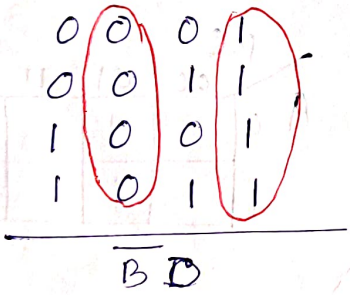
Loop-1



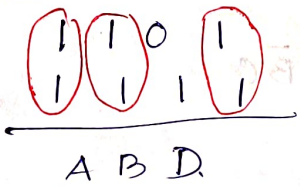
Loop-2



Loop-3



Loop-4



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

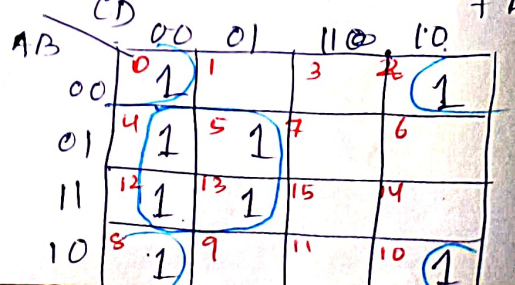
2. $Y = A\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{D} + AB\bar{C}D + A\bar{C}\bar{D} + \bar{A}B\bar{C}$

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

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

(1) $\Sigma m(0,2,4,5,8,10,12,13)$

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



Loop-1

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

$B\bar{C}$

Loop-2

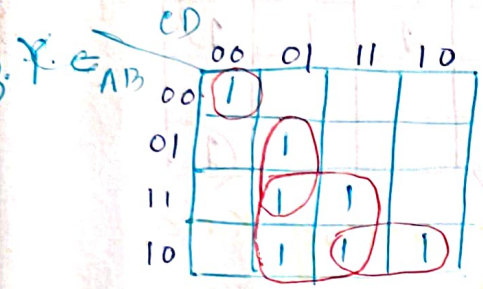
0 0 0 0
0 0 1 0
1 0 0 0
1 0 1 0

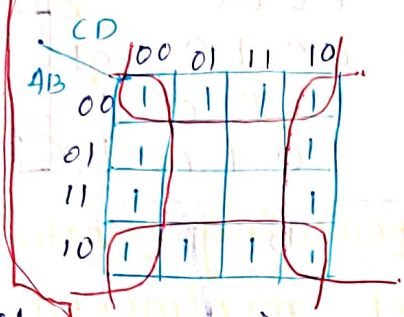
$\bar{B}\bar{D}$

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

(a) $\Sigma m(1,3,5,7,9,12,13)$

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





Loop-1

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

$\bar{B}$

Loop-1

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

A D

Loop-1

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

$A\bar{B}$

Loop-2

1 0 0 1
1 0 0 0

A B

Loop-3

0 1 0 1
1 0 0 1

$B\bar{C}D$

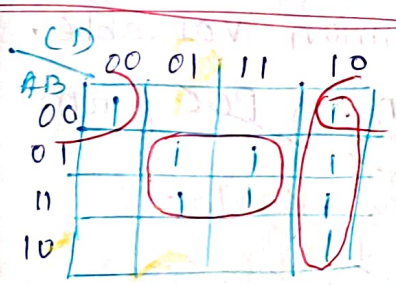
Loop-2

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

$\bar{D}$

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

(b) $\Sigma m(0,2,5,7,6,13,15,14,10)$



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

Loop-1

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

B D

Loop-2

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

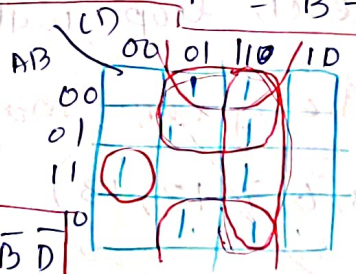
$C\bar{D}$

Loop-3

0 0 0 0
0 0 1 0

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

(c) $\Sigma m(1,3,5,7,12,15,9,11)$



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

Loop-1

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

A D

Loop-2

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

$C\bar{D}$

Loop-3

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

$\bar{B}\bar{D}$

Loop-4

$AB\bar{C}\bar{D}$

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

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

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

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

Loop-1

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

$\bar{A}$

Loop-2

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

$\bar{B}\cdot\bar{D}$

AB \ CD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11				
10	1			1

$F = \bar{A} + \bar{B}\bar{D}$ // $\Sigma m(0,1,2,3,4,5,6,7,8,10)$

K-Map of POS Expression:

If the boolean expression is in POS format for representing K-Map the following steps are adopted.

(i) combine the zero's to form the pairing in K-Map.
(ii) Read the zero's and identify the common variable.
(iii) the result of each loop will be written in POS form

(2) Q. Simplify the given max term using K-Map.

$F = \Pi M(0,2,3,6,7,12,13)$

Loop-1

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

$A + \bar{C}$

Loop-2

1 1 0 0
1 1 0 1

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

Loop-3

0 0 0 0
0 0 1 0

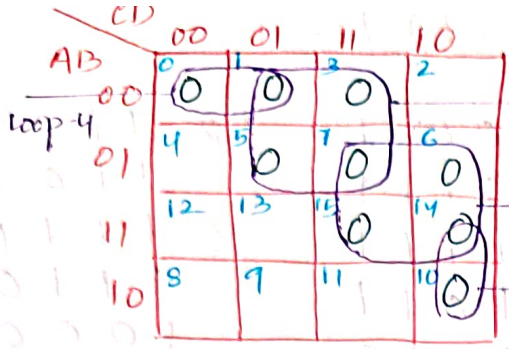
$A+B+D$

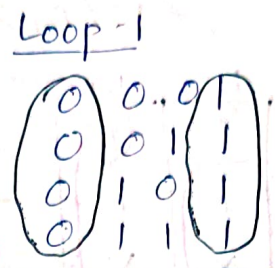
AB \ CD	00	01	11	10
00	0	1	0	0
01	4	5	0	0
11	0	0	15	14
10	8	9	11	10

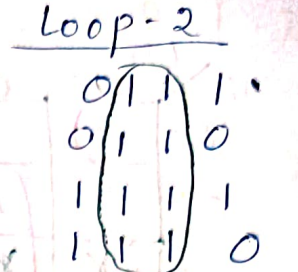
Loop-3
Loop-1
Loop-2

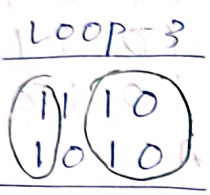
$F = (A+\bar{C})\cdot(\bar{A}+\bar{B}+C)\cdot(A+B+D)$

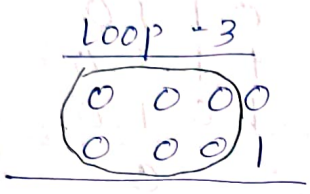
Q.











(3) $\pi M(0,1,3,5,6,7,10,14,15)$

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

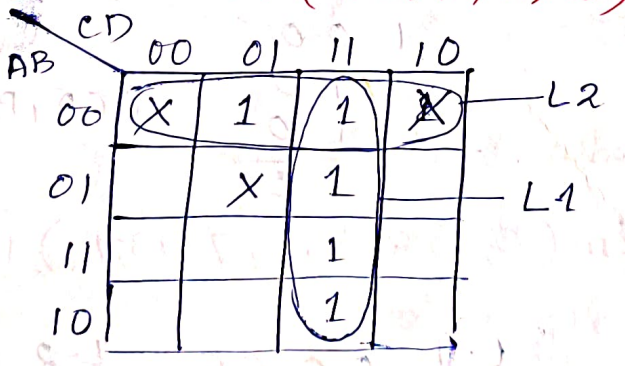
(1) $\pi M(2,3,4,6,7) = B(A+B+C)$

Don't care :-

Don't care condition is denoted by 'd' or 'x'. For any digital system design the don't care value can be 1 or 0 depending on situation (input & output condition).

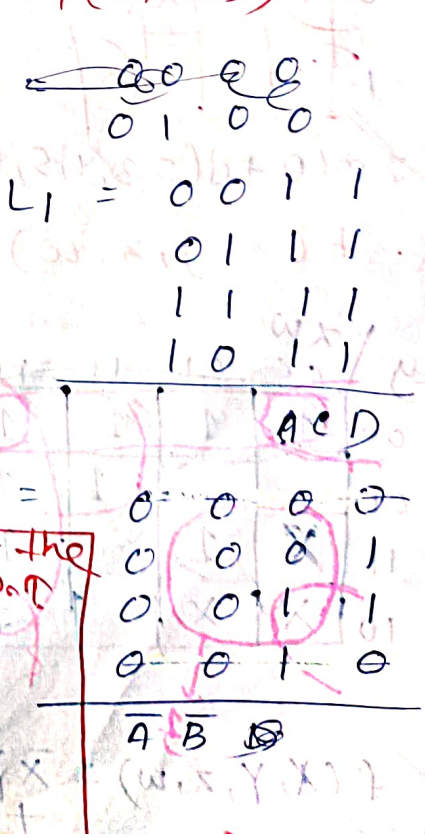
- To minimize the expression we can use the don't care values along with 1 to maximize the group
- Never consider a group of don't care's only.

Q. $F(A,B,C,D) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,5)$



P.I := 3
E.P.I = 2

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

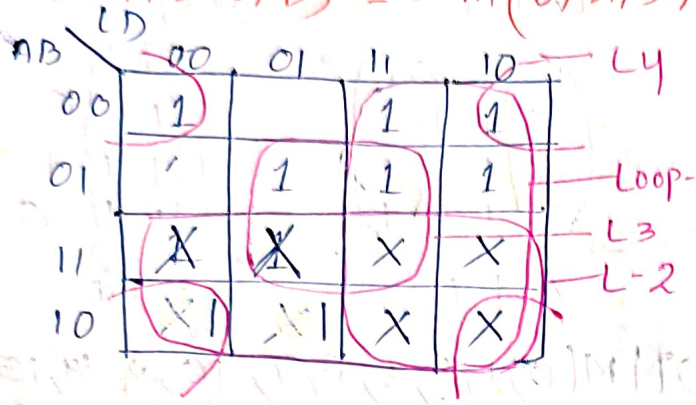


EPI: The groups that cover at least one of the minterms that can't get covered by another P.I

P.I:- possible no. group is called P.I.

Q. $f(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,9) + \Sigma d(10,11,12,13,14)$

(2)



L-1: 0 0 1 1 / 0 1 1 1 / 1 1 1 1 / 1 0 1 1 / 0 0 1 0 / 0 1 1 0 / 1 1 1 0 / 1 0 1 0 → C

L-2: 1 1 0 0 / 1 1 0 1 / 1 1 1 1 / 1 1 1 0 / 1 0 0 0 / 1 0 0 1 / 1 0 1 1 / 1 0 1 0 → A

L-3: 0 0 0 0 / 1 0 0 0 / 0 0 1 0 / 1 0 1 0 → $\bar{B}\ \bar{D}$

L-4: 1 1 0 1 / 1 1 1 1 / 0 1 0 1 / 0 1 1 1 → $B\ D$

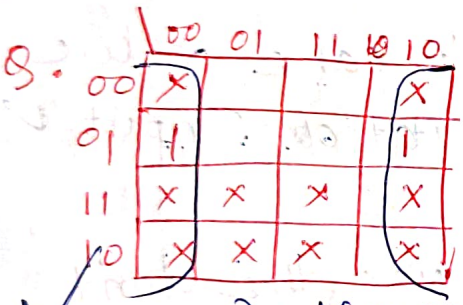
$f(A,B,C,D) = C + A + \bar{B}\bar{D} + BD$

F(A,B,C,D)
$= \Sigma m(1,3,7,11,15) + d(0,2,5)$
$= \bar{A}\bar{B} + CD$
$F(A,B,C,D) = \Sigma m(2,3,5,6,7,8,9,10,11,12,13,14,15) + d(0,1)$

A+B

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

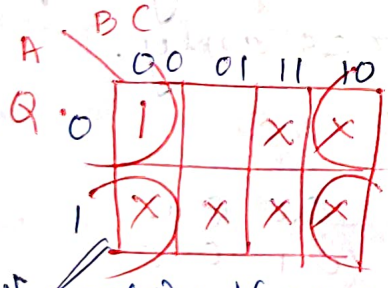
Q.



$\Sigma m(4,6) + d(0,2,8,9,10,11,12,13,14,15)$

Loop-1: 0 0 0 0 / 0 0 1 0 / 0 1 1 0 / 1 1 1 0 / 0 0 1 0 / 0 1 1 0 / 1 1 1 0 / 1 0 1 0 → $\bar{\omega}$

Q.

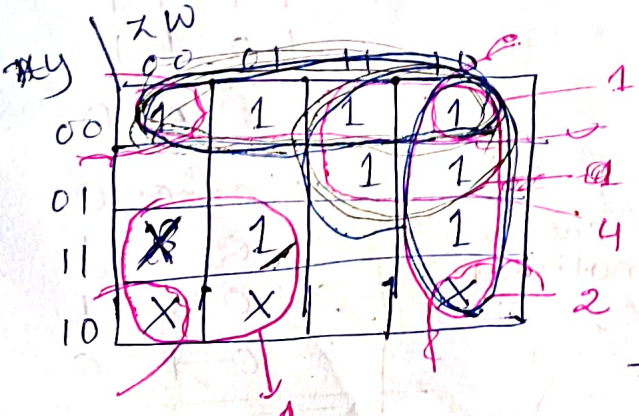


$\Sigma m(0) + d(2,3,4,5,6,7)$

0 0 0 / 1 0 0 / 0 1 0 / 1 1 0 → $\bar{C}$

$f(A,B,C,D) = \bar{C}$

Q. $F(x,y,z,\omega) = \Sigma m(0,1,2,3,6,7,13,14) + \Sigma d(8,9,10,12)$



L-1: 0 0 0 0 / 0 0 0 1 / 0 0 1 1 / 0 0 1 0 → $\bar{X}\bar{Y}$

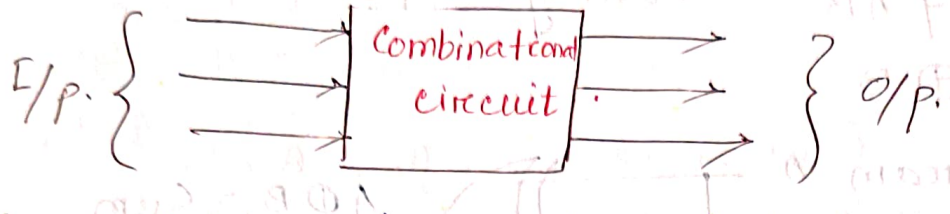
L-2: 0 0 1 0 / 0 1 1 0 / 1 1 1 0 / 1 0 1 0 → $z\bar{w}$

L-3: 1 1 0 0 / 1 1 0 1 / 1 0 0 0 / 1 0 0 1 → $x\ \bar{z}$

L-4: D 0 1 1 / 0 0 1 0 / 0 1 1 1 / 0 1 1 0 → $a\bar{x}\ z$

$f(X,Y,z,w) = \bar{X}\bar{Y} + z\bar{w} + x\bar{z} + \bar{x}z$

Combinational Logic :



Eg :- Adder, Subtractor, Multiplexor etc.

The combinational circuit are the logic circuits where output depends on the present input values. Mathematically we can say output is fun^n of input.

output = f(input).

— In this circuit there is no requirement of memory.

For designing any combinational circuit we have to follow the steps as below,

(i) Draw the block diagram showing the no of input & o/p & give proper name.

(ii) Make the truth table

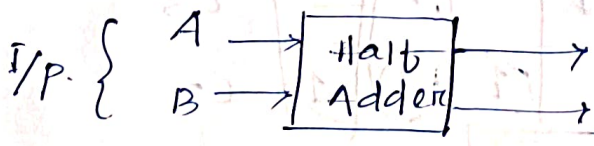
(iii) Derive the logic expression for output in terms of input (K-Map can be used)

(iv) Draw the logic circuit diagram.

Adder :

It is a combinational circuit which adds two binary numbers. There are types of adder.

Half Adder (Add^n of two single bit nos)



Truth table

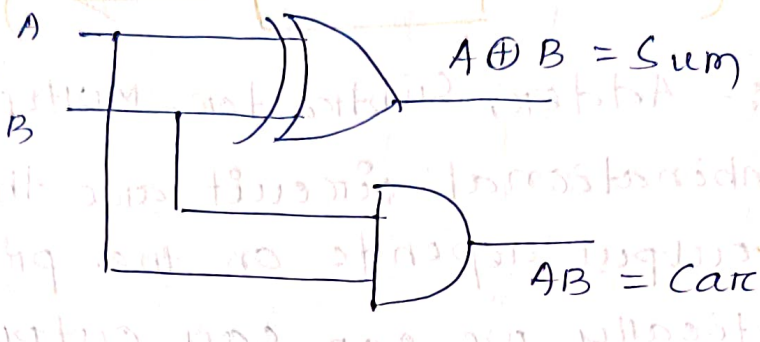
I/p		o/p	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic expression.

$Sum = \bar{A}B + A\bar{B}$ $\quad$ carry $= AB$

$= A \oplus B$

circuit diagram



Full Adder:

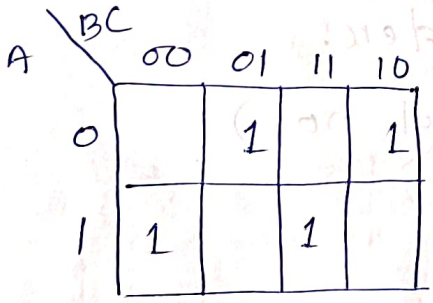
Truth table

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

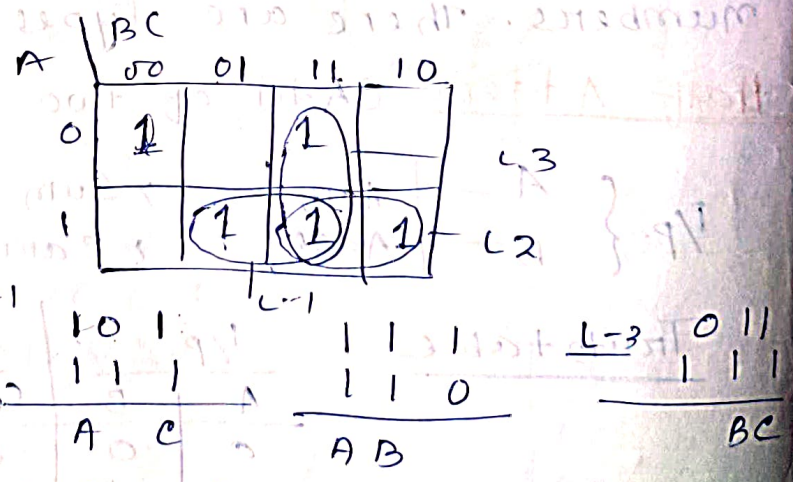
$Sum = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

$= A \oplus B \oplus C$

K-Map



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



$Cout = AB + BC + AC$

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

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

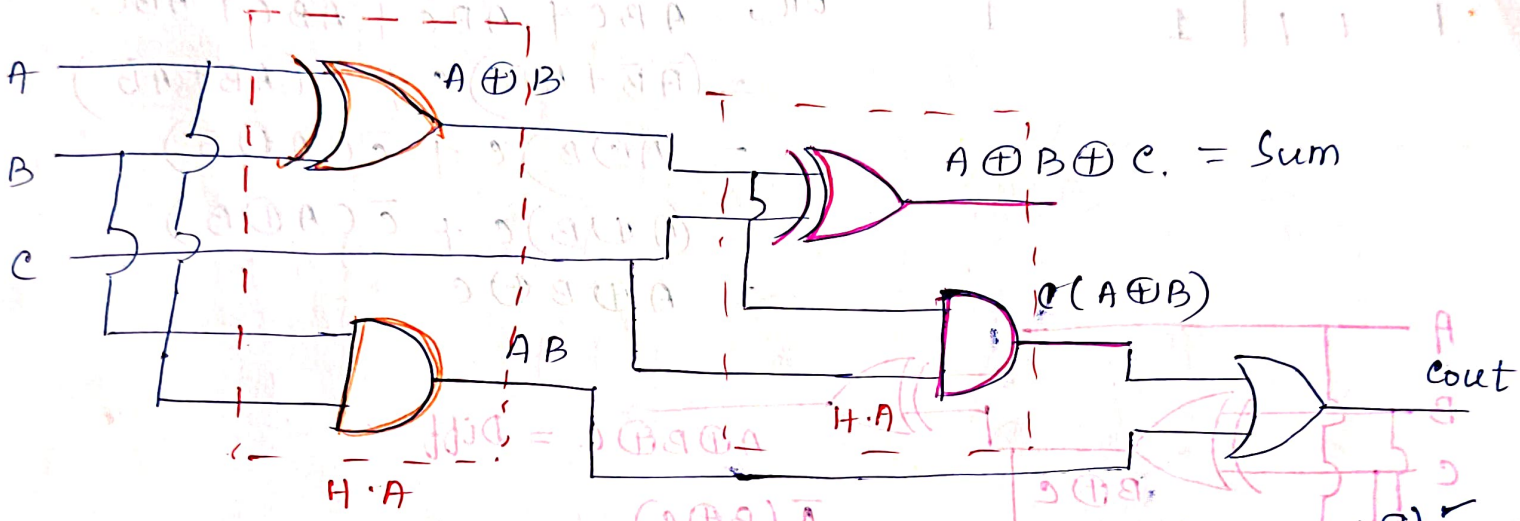
$= BC + A(B \oplus C)$

or. $Cout = C(\bar{A}B + A\bar{B}) + AB(C + \bar{C})$

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

$Sum = A \oplus B \oplus C$

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



(9)

Q. How many nand gates are required to design full Adder.

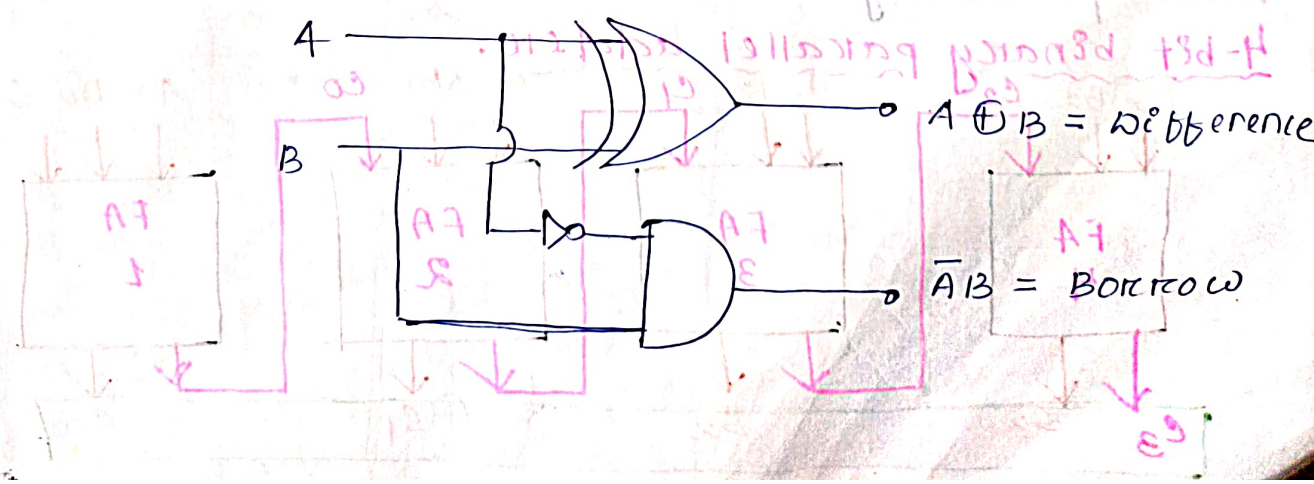
Q. How many NOR gates are required to design full Adder

Half Subtractor:

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

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

$Borrow = \bar{A}B$



Full subtractor:

A	B	C	Diff	Borrow
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

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

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

$= A \oplus B \oplus C$

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

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

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

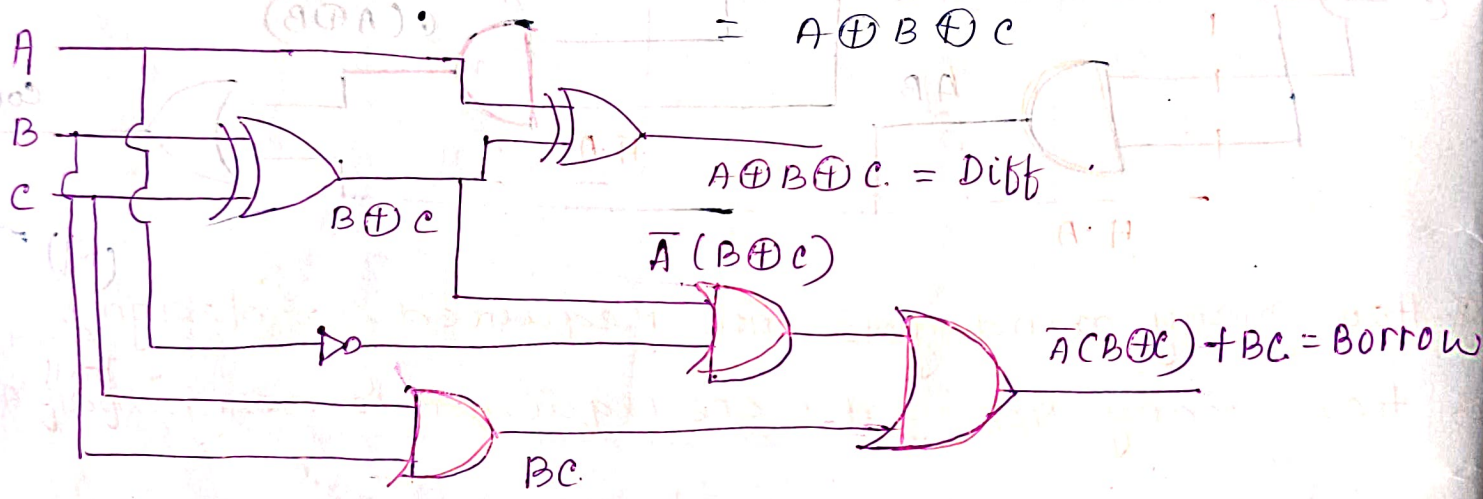
OR, $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

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

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

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

$= A \oplus B \oplus C$

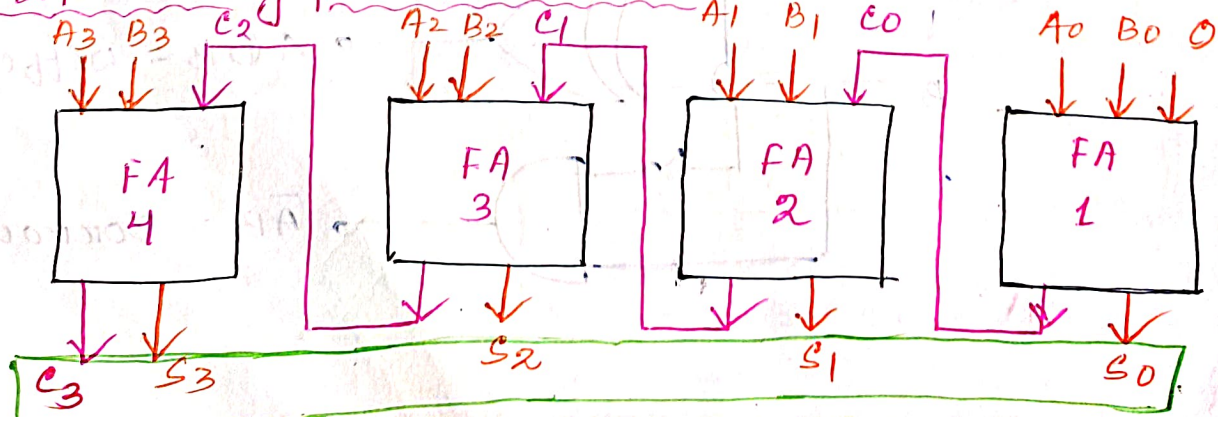


Binary Parallel Adder:

1. Ripple Carry Adder / Binary Adder:

The binary parallel adders are the combinational logic circuits used to add n number of binary data parallelly.

4-bit binary parallel adder.



Here the carry transmission from one stage to another in form of a wave shape. So it is called ripple carry adder.

C_2 C_1 C_0
A_3 A_2 A_1 A_0
B_3 B_2 B_1 B_0

C_3 S_3 S_2 S_1 S_0 ← Sum

Magnitude Comparator

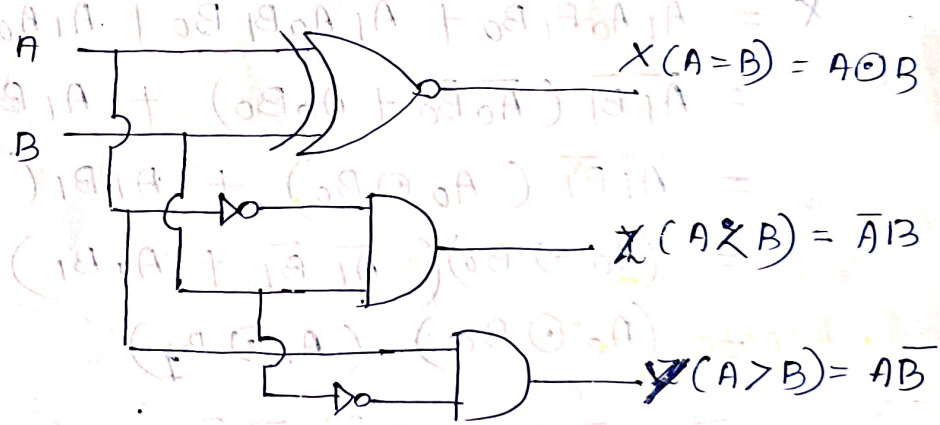
1. 1 Bit comparator:

I/ps A	I/ps B	O/ps X = (A = B)	O/ps Y (A > B)	O/ps Z (A < B)
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

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

$Y = A\overline{B}$

$Z = \overline{A}B$



2. 2 Bit comparator:

A		B		X	Y	Z
A_1	A_0	B_1	B_0	A = B	A > B	A < B
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1

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

$X = \bar{A_1}\bar{A_0}\bar{B_1}\bar{B_0} + \bar{A_1}A_0\bar{B_1}B_0 + A_1\bar{A_0}B_1\bar{B_0} + A_1A_0B_1B_0$

$Y = \bar{A_1}A_0\bar{B_1}\bar{B_0} + A_1\bar{A_0}\bar{B_1}\bar{B_0} + A_1\bar{A_0}\bar{B_1}B_0 + A_1A_0\bar{B_1}\bar{B_0} + A_1A_0\bar{B_1}B_0 + A_1A_0B_1\bar{B_0}$

$Z = \bar{A_1}\bar{A_0}\bar{B_1}B_0 + \bar{A_1}\bar{A_0}B_1\bar{B_0} + \bar{A_1}\bar{A_0}B_1B_0 + \bar{A_1}A_0\bar{B_1}\bar{B_0} + \bar{A_1}A_0B_1B_0 + A_1\bar{A_0}\bar{B_1}B_0$

Simplify

$X = \bar{A_1}\bar{A_0}\bar{B_1}\bar{B_0} + \bar{A_1}A_0\bar{B_1}B_0 + A_1\bar{A_0}B_1\bar{B_0} + A_1A_0B_1B_0$

$= \bar{A_1}\bar{B_1}(\bar{A_0}\bar{B_0} + A_0B_0) + A_1B_1(\bar{A_0}\bar{B_0} + A_0B_0)$

$= \bar{A_1}\bar{B_1}(A_0 \odot B_0) + A_1B_1(A_0 \odot B_0)$

$= (A_0 \odot B_0)(\bar{A_1}\bar{B_1} + A_1B_1)$

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

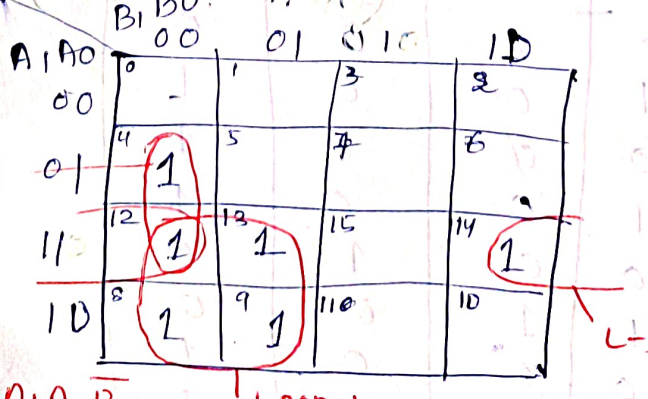
$Y = \bar{A_1}A_0\bar{B_1}\bar{B_0} + A_1\bar{A_0}\bar{B_1}\bar{B_0} + A_1\bar{A_0}\bar{B_1}B_0 + A_1A_0\bar{B_1}\bar{B_0} + A_1A_0\bar{B_1}B_0 + A_1A_0B_1\bar{B_0}$

$= \bar{B_1}\bar{B_0}(A_1\bar{A_0} + \bar{A_1}A_0) + A_1A_0(\bar{B_1}\bar{B_0} + B_1B_0)$

L-1 1000, 1001, 1100, 1101 → $A_1\bar{B_1}$

L-2 0100, 1000 → $A_0\bar{B_1}\bar{B_0}$

L-3 1100, 1110 → $A_1A_0\bar{B_0}$



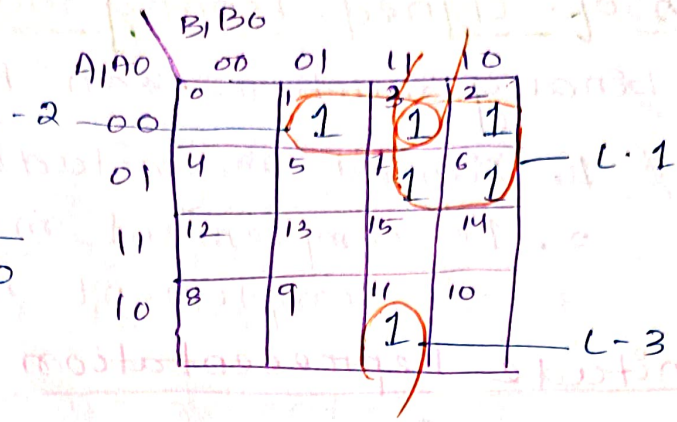
$Y = A_1\bar{B_1} + A_0\bar{B_1}\bar{B_0} + A_1A_0\bar{B_0}$

$Z = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,B_0 + \overline{A_1}\,\overline{A_0}\,B_1\,\overline{B_0} + \overline{A_1}\,\overline{A_0}\,B_1\,B_0 + \overline{A_1}\,A_0\,B_1\,\overline{B_0} + \overline{A_1}\,A_0\,B_1\,B_0$
$+ A_1\,\overline{A_0}\,B_1\,B_0$.

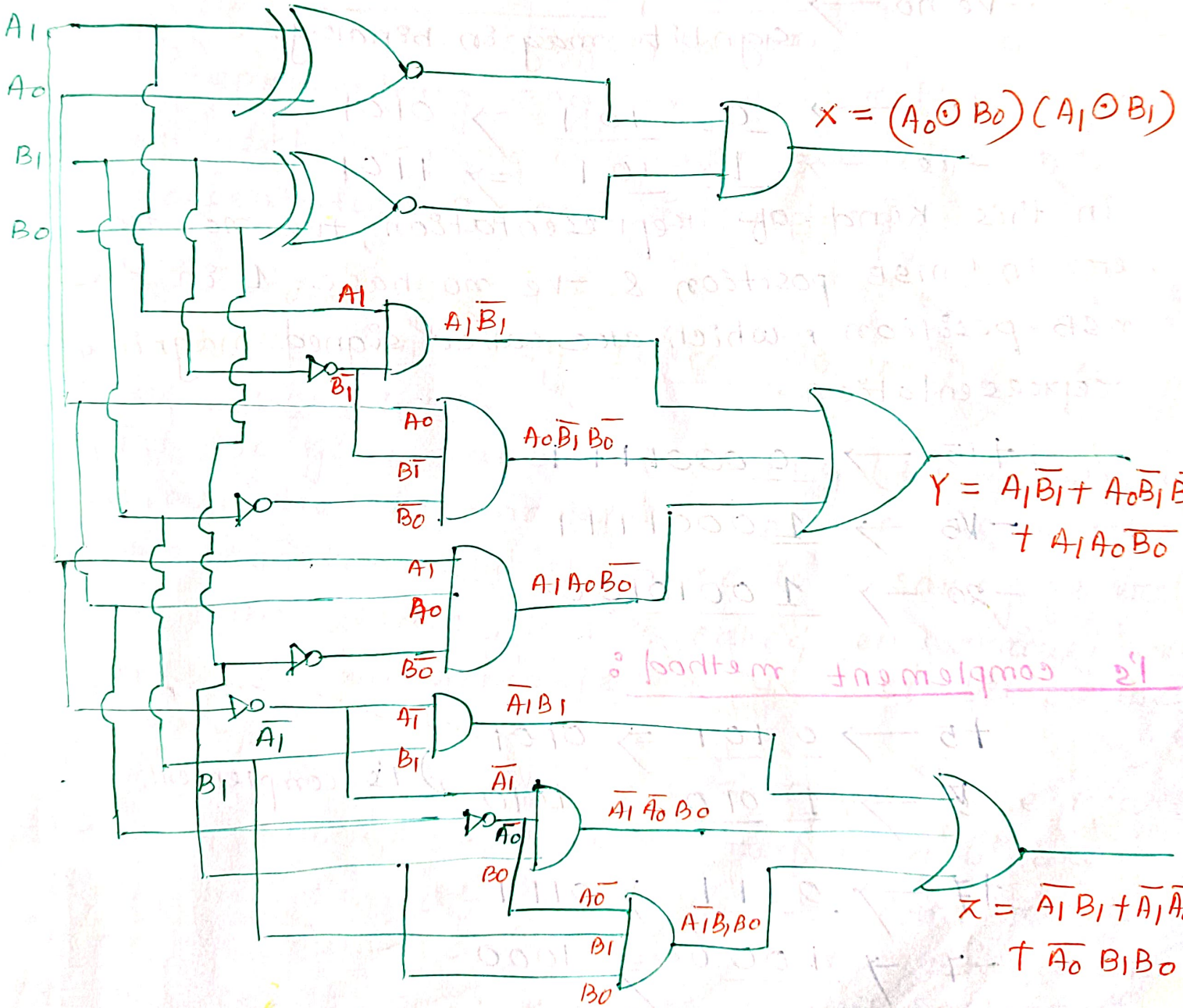
L-1
0 0 1 1
0 0 1 0
0 1 1 1
0 1 1 0
$\overline{A_1}\ B_1$

L-2
0 0 0 1
0 0 1 1
$\overline{A_1}\,\overline{A_0}\ B_0$

L-3
0 0 1 1
1 0 1 1
$\overline{A_0}\,B_1\,B_0$



$Z = \overline{A_1}\,B_1 + \overline{A_1}\,\overline{A_0}\,B_0 + \overline{A_0}\,B_1\,B_0$



Number system

Representation of signed binary numbers

The signed binary numbers can be represented in three ways.

1. signed magnitude Representation.
2. 1's complement method
3. 2's complement method.

Signed magnitude Representation.

+ve no → 0 | mag in binary.
(0 = sign bit)

-ve no → 1 | mag in binary.
(1 = sign bit)

+5 → 0 101 ⇒ 0101

-ve → 1 101 ⇒ 1101

In this kind of representation, +ve no has zero in MSB position & -ve no has 1 in the MSB position, which are called signed magnitude representation.

+15 → 0 0001111

-15 → 1 0001111

-20 → 1 0010100

1's complement method:

+5 → 0 101 ⇒ 0101

-5 → 1 010 ⇒ 1010 ← 1's complement.

+7 → 0 111 ⇒ 0111

-7 → 1 000 ⇒ 1000

+15 → 0 0001111

-15 → 1 1110000

2's complement method

$-5 \rightarrow$ 2's (+5)

$+5 \rightarrow 0101$

$-5 \Rightarrow 1011$ (2's complement of +5)

$+15 \rightarrow 00001111$

$-15 \rightarrow 11110001$

Notes

* Positive number representation is same for three types of signed binary representation.
* The negative number representation are different in all the three representation.
* For representing any -ve no first positive of that particular no is taken ~~fore~~ and then the representation method is applied.

Subtraction of Binary nos:

Binary subtraction using 1's complement:

Steps for subtraction:

- convert the given number to binary (4bit, 8bit, 16bit)
- Find out the 1's complement of the -ve number.
- Do the addition and if there is a carry generated at the final state, that is called end around carry
- Add the carry generated to the LSB bit of the sum value.
- check the MSB bit. if it is 'zero', then the result is a +ve number & it is the required binary no.
- check the MSB. if it is 'one', then the result is a -ve number, take the one's complement of the result which will give the original ans (-ve number).

Q. Subtract 4 from 5.

$5-4=?$

$5+(-4)=?$

(i) $+5 \rightarrow 0101$

$+4 \rightarrow 0100$

$-4 \rightarrow 1011$ (1's (+4))

(ii)

$+5 = 0101$

$-4 = 1011$

10000

End around carry. ← (1) → $+ \quad 1$

$00001 \Rightarrow +1$ (Ans)

MSB bit. mag.

Q. Subtract 5 from 4:

$4-5=?$

$4+(-5)=?$

$+4 = 0100$

$+5 = 0101$

$-5 = 1010$

$+4 = 0100$

$-5 = 1010$

1110

1's $\Rightarrow 0001 \Rightarrow -1$

Ans -1

Q. Subtract 19 from 30.

$30-19=?$

$30+(-19)=?$

$+30 = 00011110$

$+19 = 00010011$

$-19 = 11101100$

00011110

11101100

100001010

$\rightarrow 1$

00001011

Ans : +11

Q. $15-18=?$

$15+(-18)=?$

$+15 = 00001111$

$+18 = 00010010$

$-18 = 11101101$

00001111

11101101

11111100

$\Rightarrow 00000011 \Rightarrow -3$

15: $2|15$, $2|7$ — 1, $2|3$ — 1, 1 — 1

18: $2|18$, $2|9$ — 0, $2|4$ — 1, $2|2$ — 0, 1 — 0

Binary subtraction using 2's complement

Steps

- convert the given decimal no to binary.
- convert the -ve no in its corresponding 2's complement format.
- Do the addition of the no's and discard the carry generated at the result.
- check the MSB bit of it zero, then no is a +ve number & the added result is a original ans. If MSB is '1', then take 2's complement of the result to get the original no which is a -ve number.

Q. Subtract 25 from 8 ?

8 - 25 = ?

8 + (-25) = ?

+8 = 00001000

+25 = 00011001

-25 = 1's 11100110

2's 11100111

2 | 25
2 | 12 - 1
2 | 6 - 0
2 | 3 - 0
2 | 1 - 1
0 | 1

2 | 8
2 | 4 - 0
2 | 2 - 0
1 - 0

00001000
11100111
11101111

⇒ 00010000
+ 1
00010001 ⇒ -17

$1\times2^4+0\times2^3+0\times2^2+0\times2^1+1\times2^0$

$= 16+1 = 17$

Q. Add 15 & -9

Q. Add -12 & -3

Q. Subtract 35 from 48

Q. Add -17 & -27.

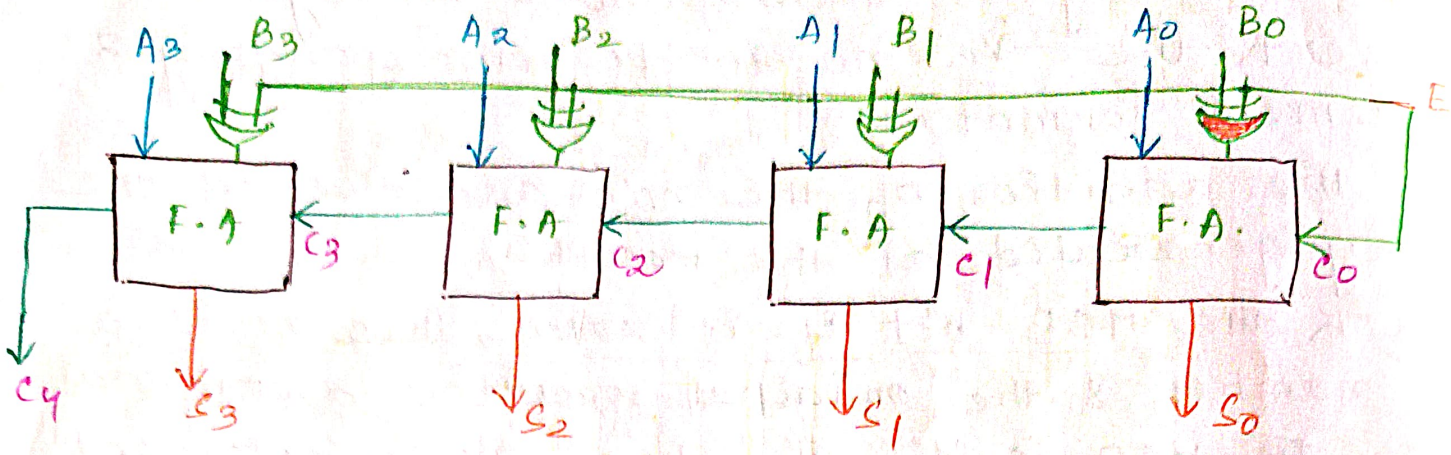
Q. Sub -9 & +13

Q. Add 23 from 29

} ⇒ 2's complement

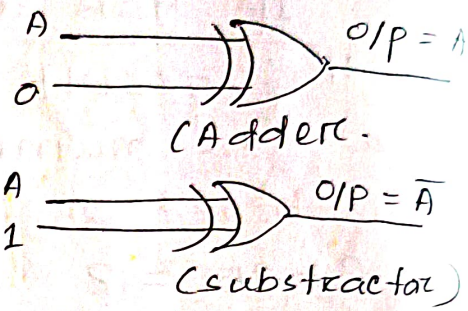
⇒ 1's complement

2's complement Adder / subtractor :



When E = 0,

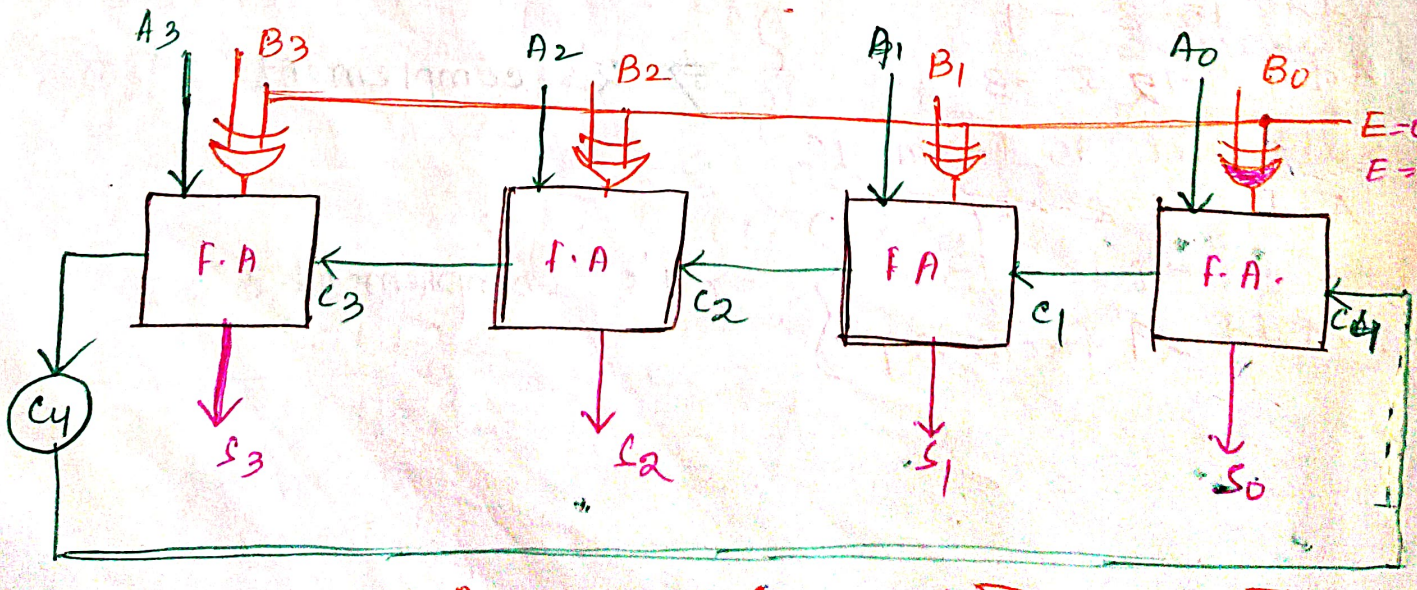
The above circuit works as adder. So the output of the adder = C_4, S_3, S_2, S_1, S_0

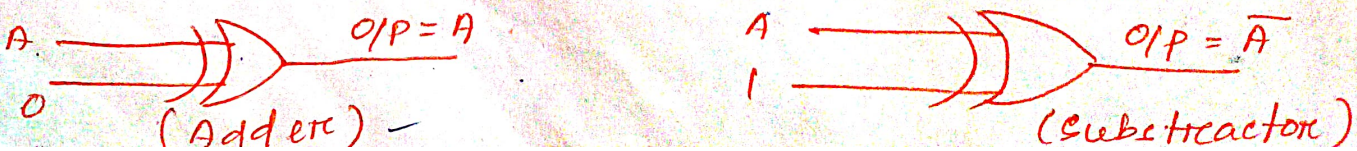


When E = 1,

The above circuit behaves as subtractor, the output of Ex-OR gate in this condition gives the complemented value of the second input. When Co is added to the complimented value, then the result is 2's complemented of the second no.

1's complement Adder / subtractor :-





Binary Multiplier.

It is a combinational circuit used in digital systems to perform the multiplication of two binary numbers.

2-bit Multiplier

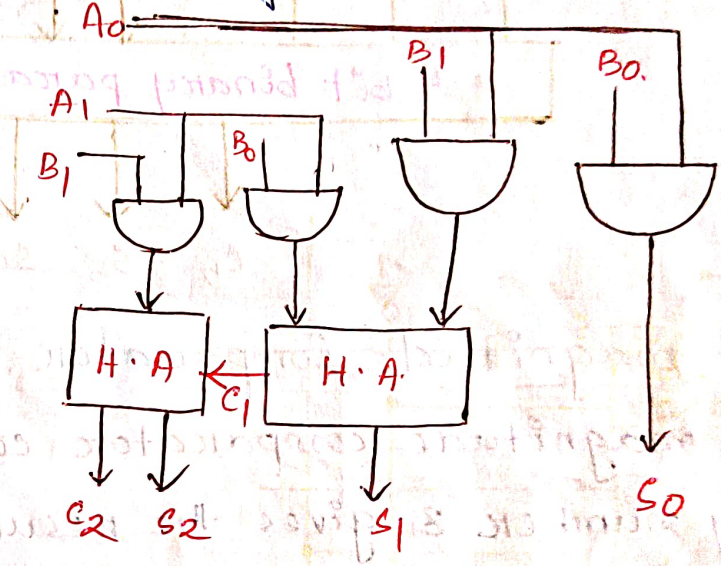
$A \rightarrow A_1 \; A_0$

$B \rightarrow B_1 \; B_0$

$A_1B_0 \quad A_0B_0$

$A_1B_1 \quad A_0B_1$

$A_1B_1 \quad (A_1B_0 + A_0B_1) \quad A_0B_0$



4 bit to 3 Bit Binary Multiplier:

$A \rightarrow A_3 \; A_2 \; A_1 \; A_0$

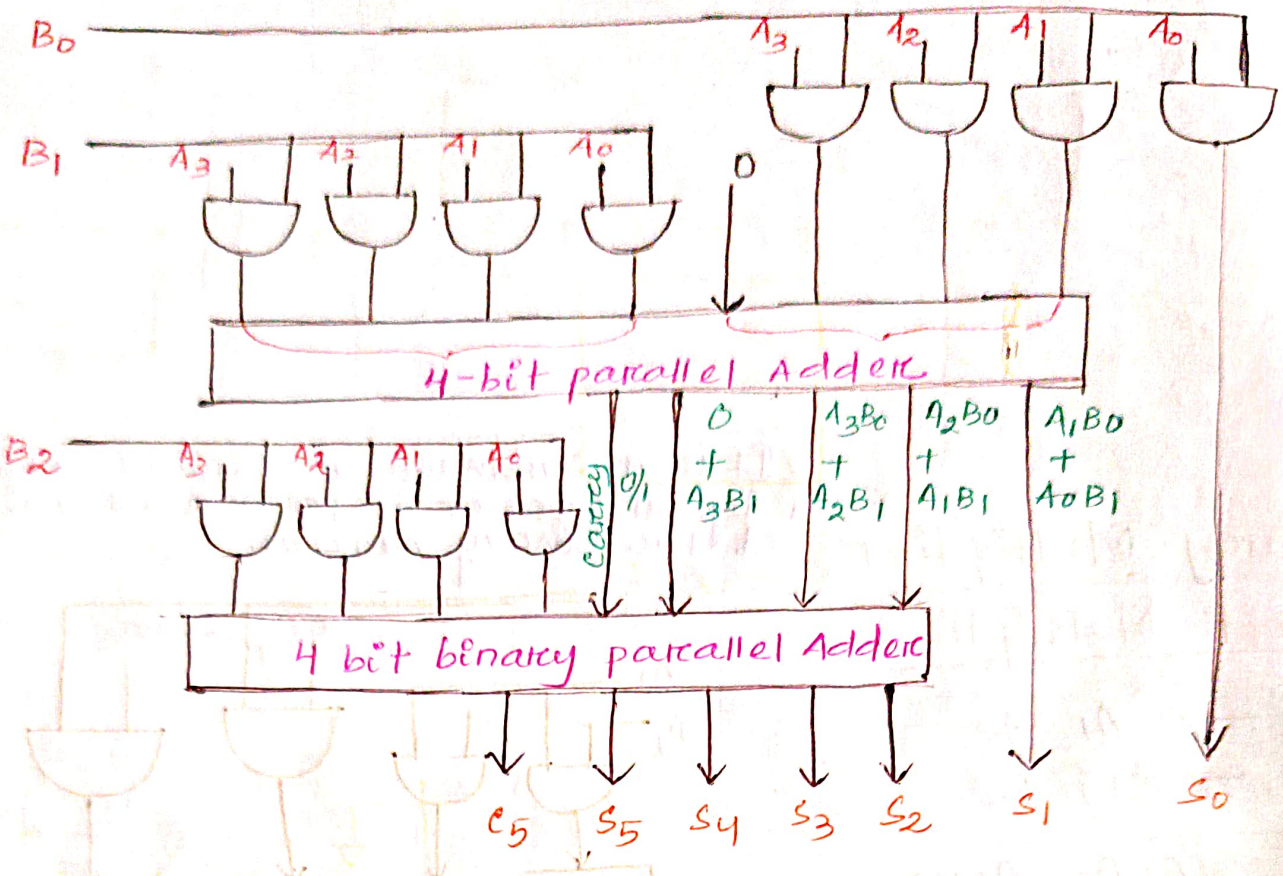
$B_2 \; B_1 \; B_0$

$A_3B_0 \quad A_2B_0 \quad A_1B_0 \quad A_0B_0$

$A_3B_1 \quad A_2B_1 \quad A_1B_1 \quad A_0B_1$

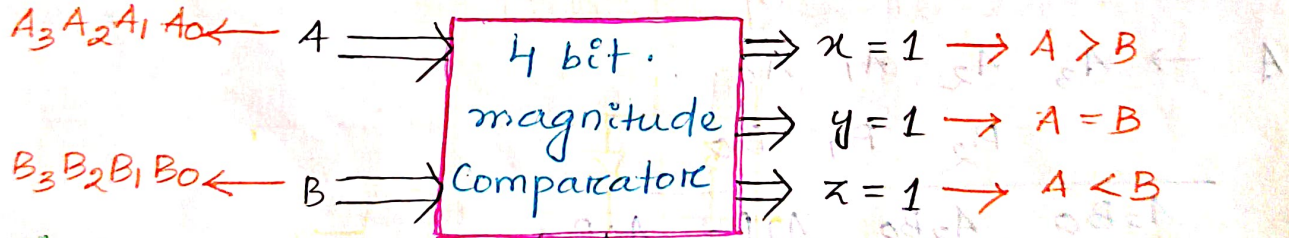
$A_3B_2 \quad A_2B_2 \quad A_1B_2 \quad A_0B_2$

$C_5 \; S_5 \; S_4 \; S_3 \; S_2 \; S_1 \; S_0$



4-bit magnitude Comparator.

4 bit magnitude comparator compares two four bit binary numbers & gives the result to show three conditions i.e equality, greater than (>), small than (<) condition.



Y = 1 (Equality condition / A = B)

$$y_3 = A_3 \odot B_3$$

$$y_2 = A_2 \odot B_2$$

$$y_1 = A_1 \odot B_1$$

$$y_0 = A_0 \odot B_0$$

$$\boxed{y = y_3\, y_2\, y_1\, y_0}$$

X = 1 (A > B)

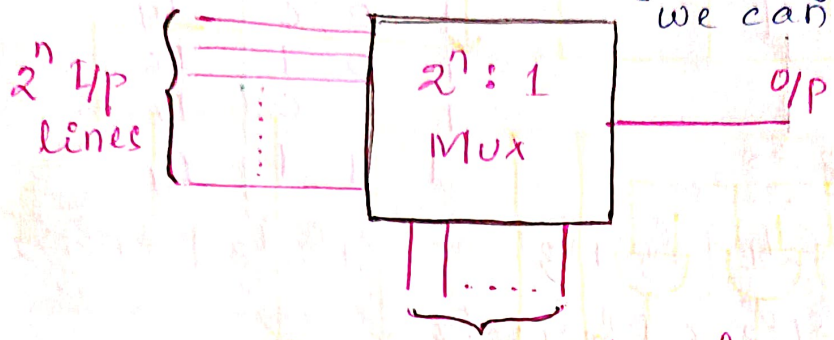
$$x = A_3 \cdot \overline{B_3} + y_3 A_2 \overline{B_2} + y_3 y_2 A_1 \overline{B_1} + y_3 y_2 y_1 A_0 \overline{B_0}$$

$x = 1 \ (A < B)$

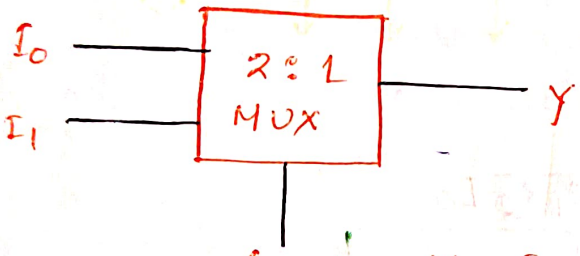
$$x = \bar{A}_3 B_3 + y_2 \bar{A}_2 B_2 + y_2 y_3 \bar{A}_1 B_1 + y_3 y_2 y_1 \bar{A}_0 B_0$$

Multiplexer

Mux is called data selector. By using lower order Mux we can also design higher order mux.



2:1 MUX

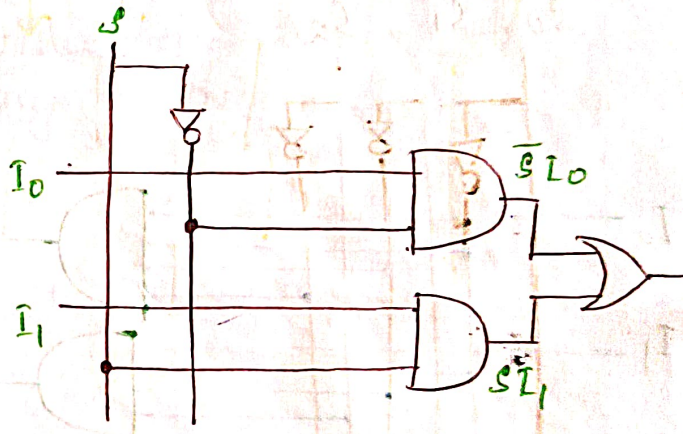


$S = 0, \ Y = I_0$

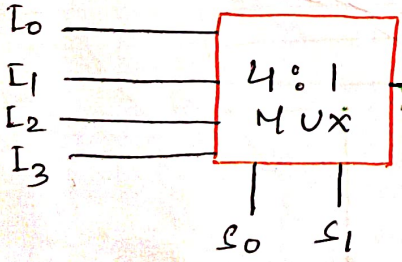
$S = 1, \ Y = I_1$

S	Y
0	I_0
1	I_1

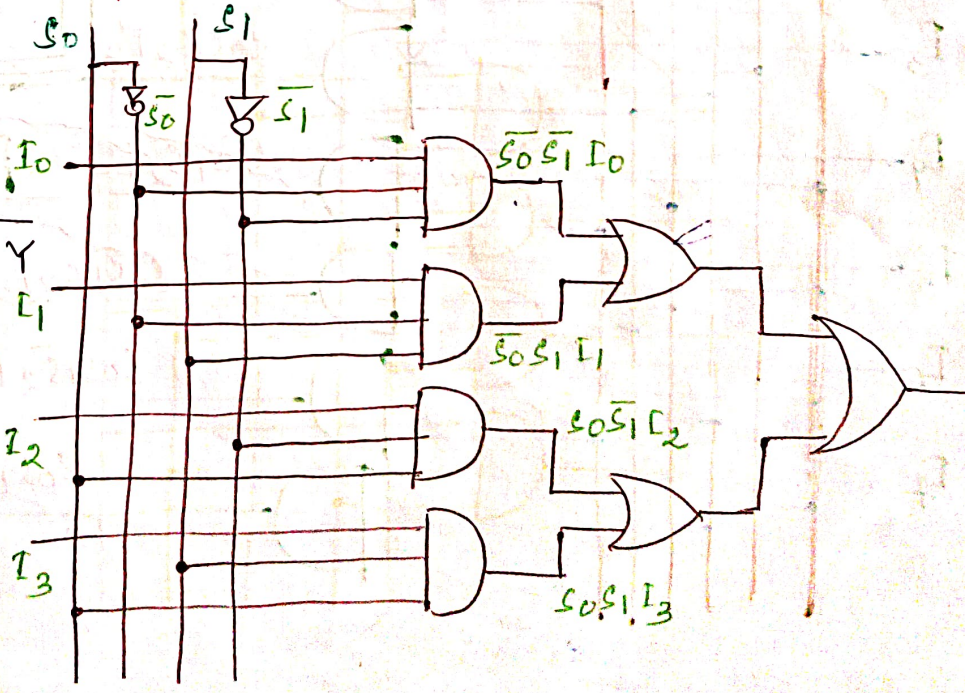
$$Y = \bar{S} I_0 + S I_1$$



4:1 MUX

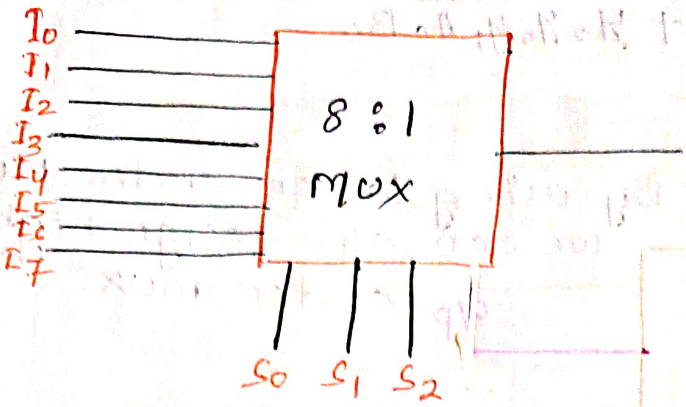


S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



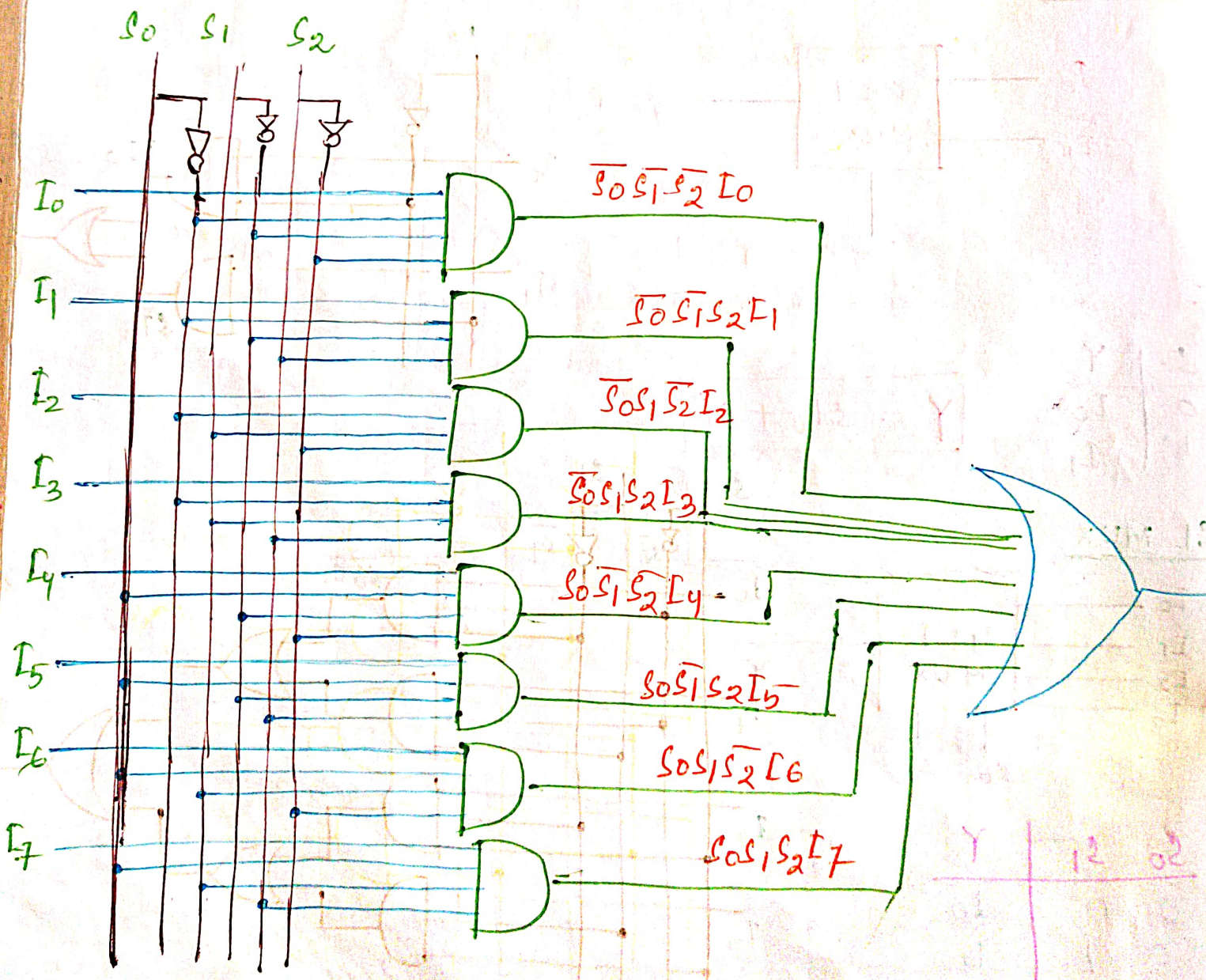
$$Y = \bar{S}_0 \bar{S}_1 I_0 + \bar{S}_0 S_1 I_1 + S_0 \bar{S}_1 I_2 + S_0 S_1 I_3$$

8:1 MUX

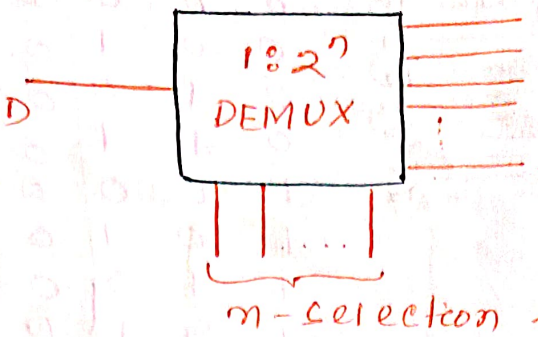


S_0	S_1	S_2	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

$$Y = \bar{S_0}\bar{S_1}\bar{S_2}I_0 + \bar{S_0}\bar{S_1}S_2I_1 + \bar{S_0}S_1\bar{S_2}I_2 + \bar{S_0}S_1S_2I_3 + S_0\bar{S_1}\bar{S_2}I_4 + S_0\bar{S_1}S_2I_5 + S_0S_1\bar{S_2}I_6 + S_0S_1S_2I_7$$

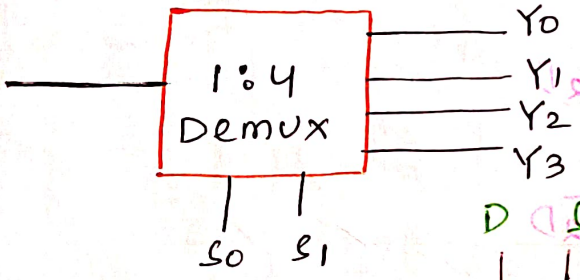


DE-MULTIPLEXER



- Depending on selection line value the single data is distributed across the 2^n output lines. So it is called data distributor.
- By varying 'n' the number of output will change as well as the number of input will change.
- This is also called data distributor & serial to parallel converter & one to many circuit.

1:4 DEMUX



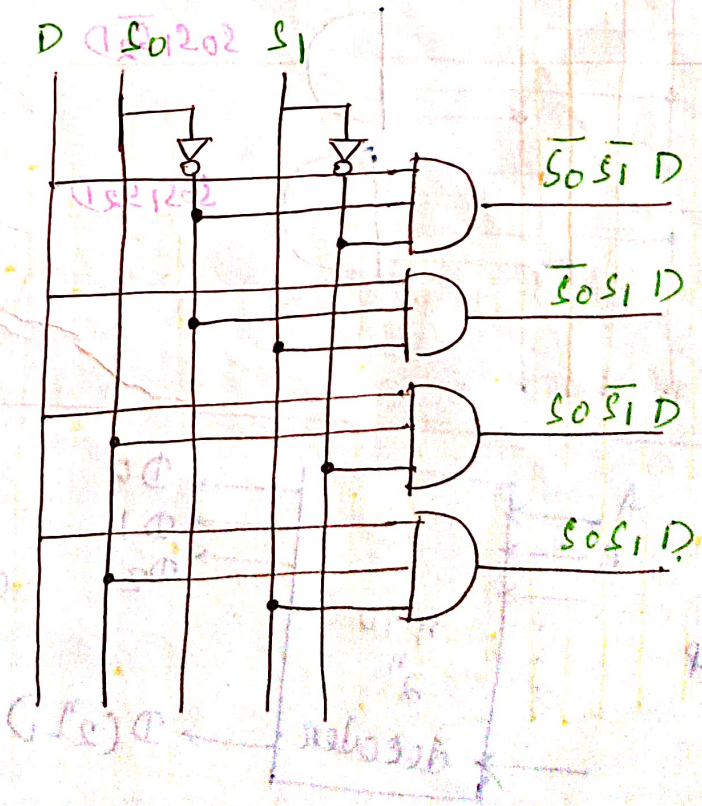
S_0	S_1	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

$Y_0 = \bar{S_0}\bar{S_1}D$

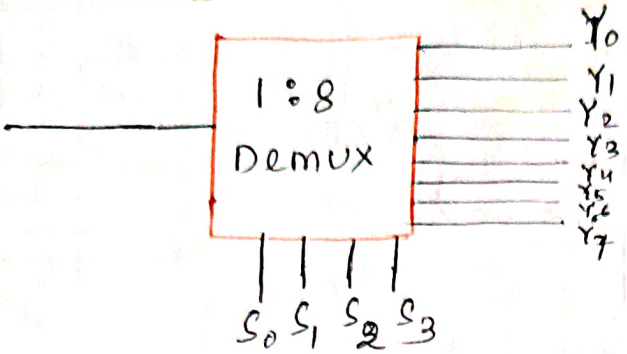
$Y_1 = \bar{S_0}S_1D$

$Y_2 = S_0\bar{S_1}D$

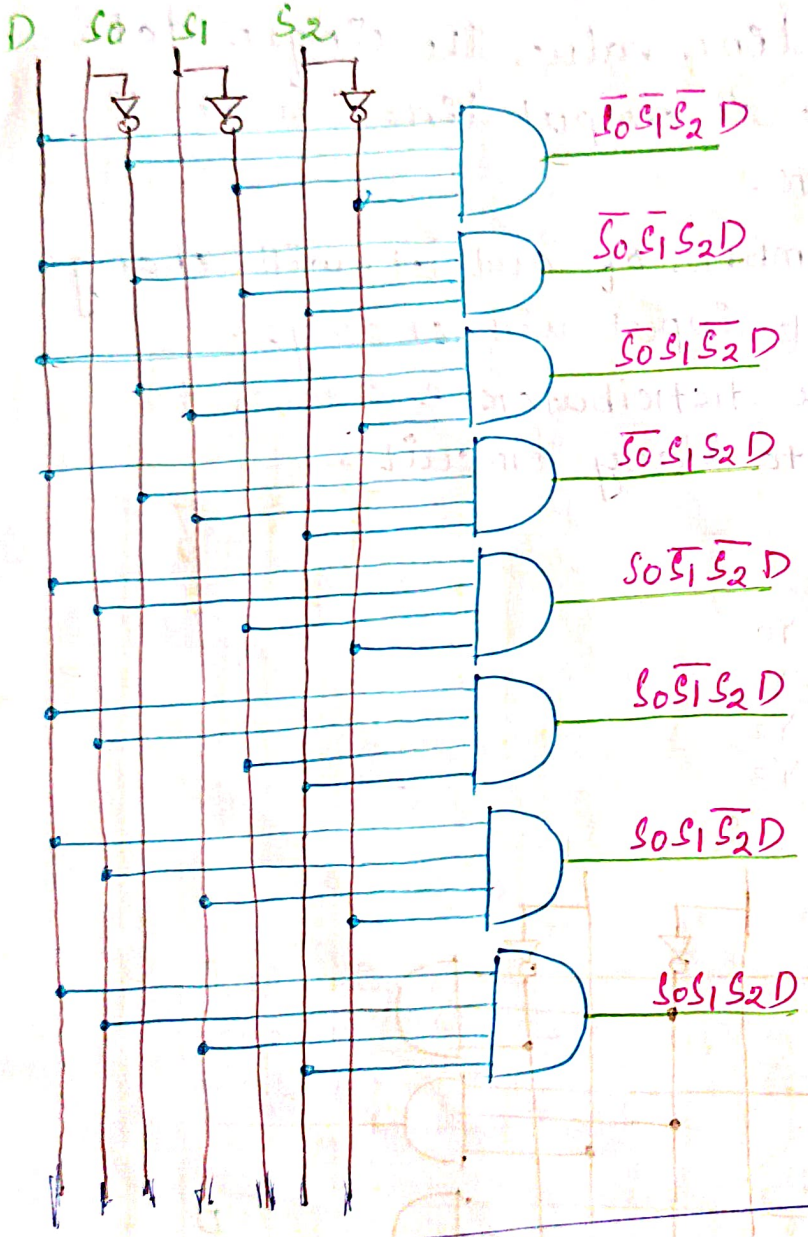
$Y_3 = S_0S_1D$



1:8 DEMUX



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



$Y_0 = \bar{S_0}\bar{S_1}\bar{S_2}D$

$Y_1 = \bar{S_0}\bar{S_1}S_2D$

$Y_2 = \bar{S_0}S_1\bar{S_2}D$

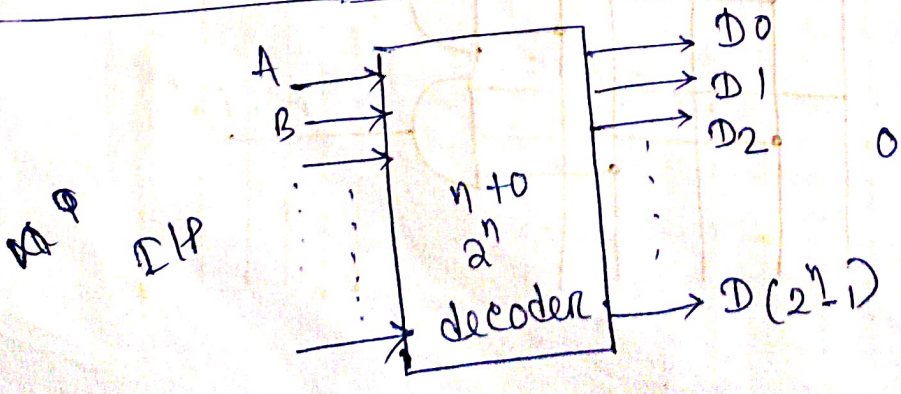
$Y_3 = \bar{S_0}S_1S_2D$

$Y_4 = S_0\bar{S_1}\bar{S_2}D$

$Y_5 = S_0\bar{S_1}S_2D$

$Y_6 = S_0S_1\bar{S_2}D$

$Y_7 = S_0S_1S_2D$



The digital decoder are extensively used in several application such as decoding of data, seven segment display, multiplexer, demultiplexing, memory operation etc.

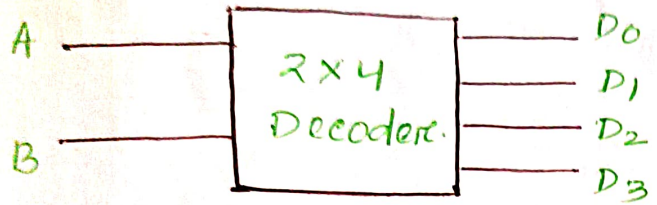
DECODER

It is a combinational logic circuit which is capable of converting coded data into other form (Hexadecimal, Decimal & other form)

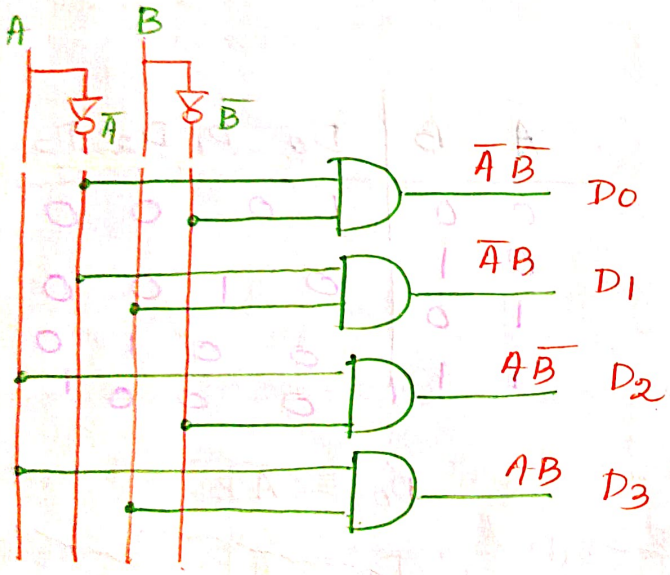
- Decoder has small n number of input & maximum 2^n number of output.

2x4 Decoder: (Active high)

- It is also called two line to four line decoder.



I/p lines		O/p lines.			
A	B	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

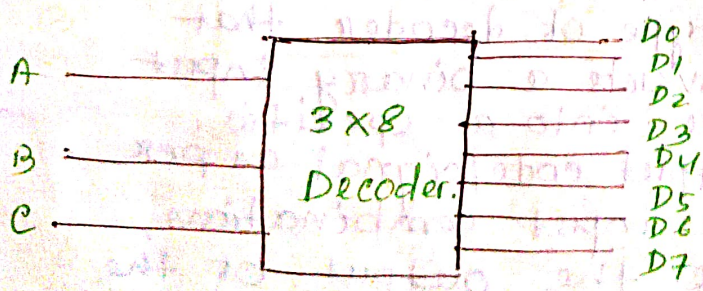


$D_0 = \bar{A}\bar{B}$ $D_2 = A\bar{B}$

$D_1 = \bar{A}B$ $D_3 = AB$

3x8 Decoders:

- It is also called three line to eight line decoder.



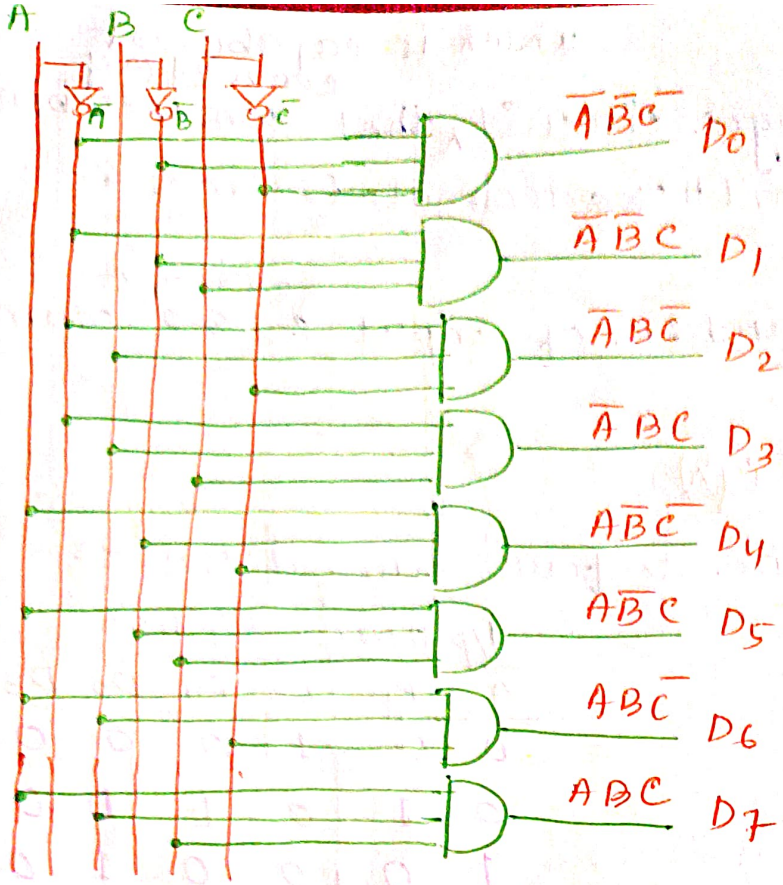
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = \bar{A}\bar{B}\bar{C}$ $D_4 = A\bar{B}\bar{C}$

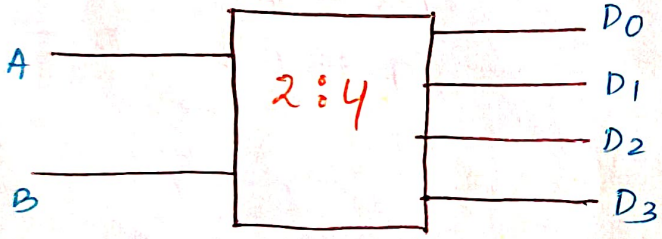
$D_1 = \bar{A}\bar{B}C$ $D_5 = A\bar{B}C$

$D_2 = \bar{A}B\bar{C}$ $D_6 = AB\bar{C}$

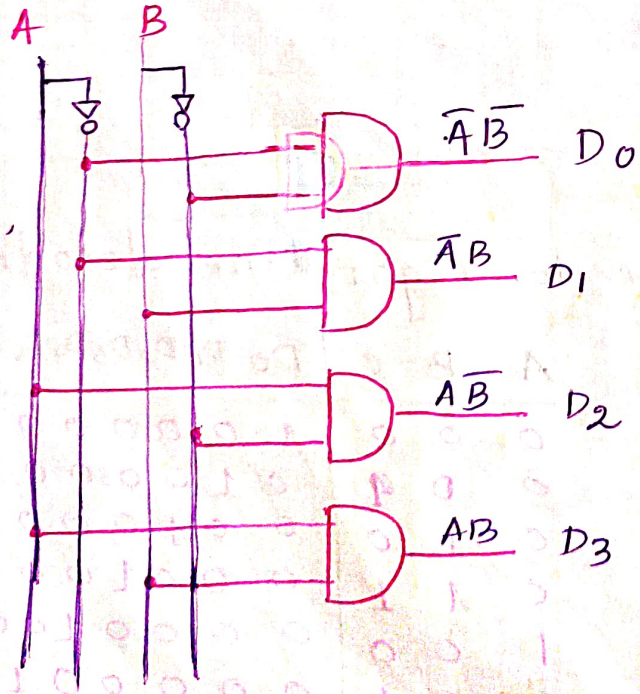
$D_3 = \bar{A}BC$ $D_7 = ABC$



Active High Decoder.



A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



$D_0 = \bar{A}\bar{B}$

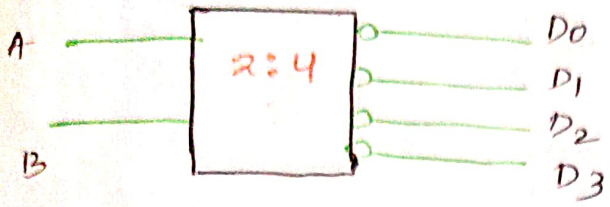
$D_1 = \bar{A}B$

$D_2 = A\bar{B}$

$D_3 = AB$

A type of decoder that converts a binary input code into a specific output code/signal as per the input combination, where the output of the decoder is considered active or ON when it is in the logic 1 state, it is called active high decoder.

Active Low decoder.



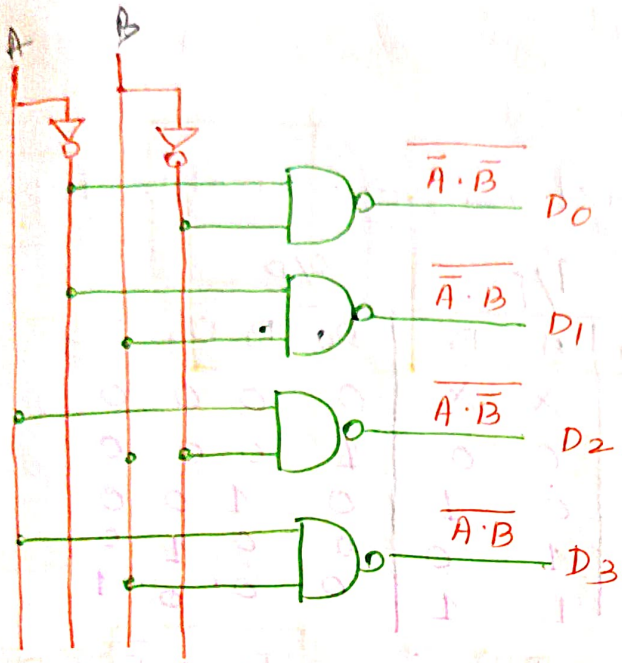
I/P		O/P			
A	B	D_0	D_1	D_2	D_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

$D_0 = A + B = \overline{\overline{A+B}} = \overline{\overline{A} \cdot \overline{B}}$

$D_1 = A + \overline{B} = \overline{\overline{A} \cdot B}$

$D_2 = \overline{A} + B = \overline{A \cdot \overline{B}}$

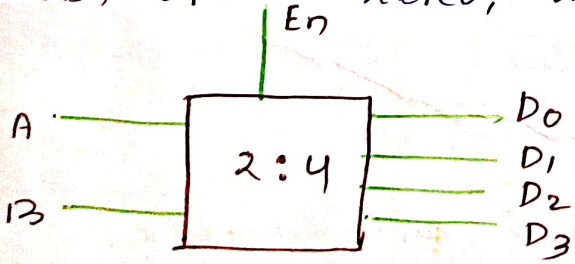
$D_3 = \overline{A} + \overline{B} = \overline{A \cdot B}$



Active low decoders are implemented using NAND gates.

Active High decoder with Active high Enable I/p.

Enable input is the control input of the device. when the enable input is 1, the device works & when it is zero, the device doesn't work.



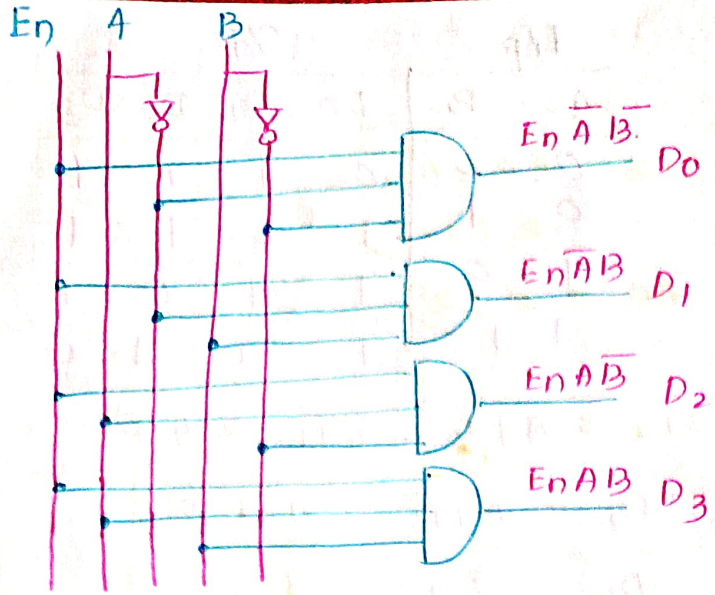
I/P			O/P			
En	A	B	D_0	D_1	D_2	D_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$D_0 = En\,\overline{A}\,\overline{B}$

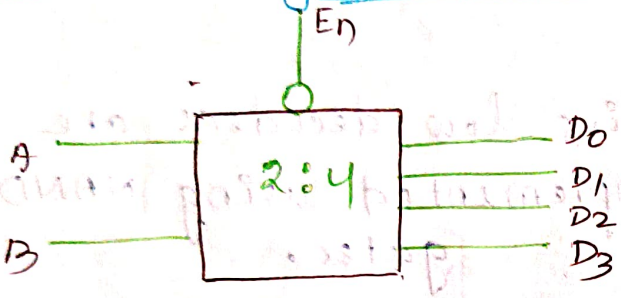
$D_1 = En\,\overline{A}\,B$

$D_2 = En\,A\,\overline{B}$

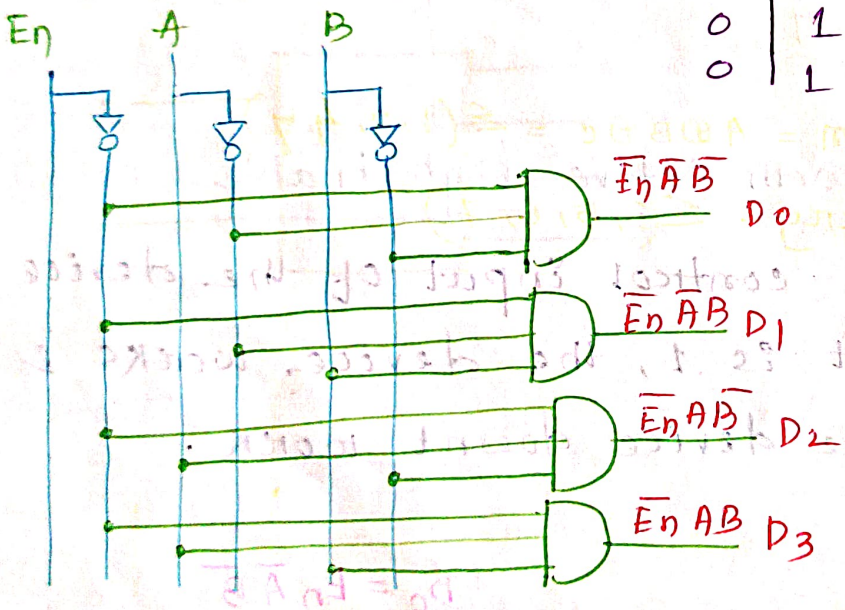
$D_3 = En\,A\,B$



Active high Decoder with Active low Enable Input



I/P			O/P			
En	A	B	D_0	D_1	D_2	D_3
1	X	X	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1



$D_0 = \bar{E}n\bar{A}\bar{B}$

$D_1 = \bar{E}n\bar{A}B$

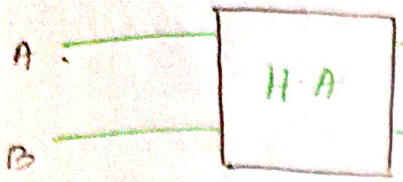
$D_2 = \bar{E}nA\bar{B}$

$D_3 = \bar{E}nAB$

Q. Design 2:4 active low decoder with active high enable input.

Q. Design 2:4 active low decoder with active low enable input.

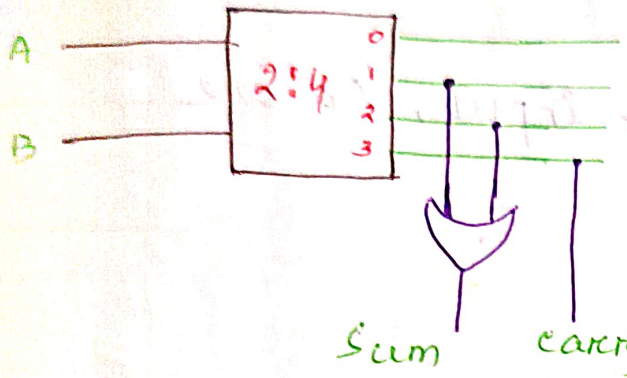
Implement a Half Adder using Decoder circuit



Sum = $A \oplus B = \bar{A}B + A\bar{B}$
$= \Sigma(1,2)$

Carry = $AB = \Sigma(3)$

As the half Adder has two input, so we use 2:4 decoder.



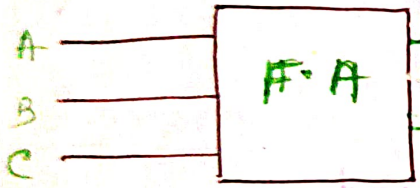
$D_0 = \bar{A}\bar{B} = m_0$

$D_1 = \bar{A}B = m_1$

$D_2 = A\bar{B} = m_2$

$D_3 = AB = m_3$

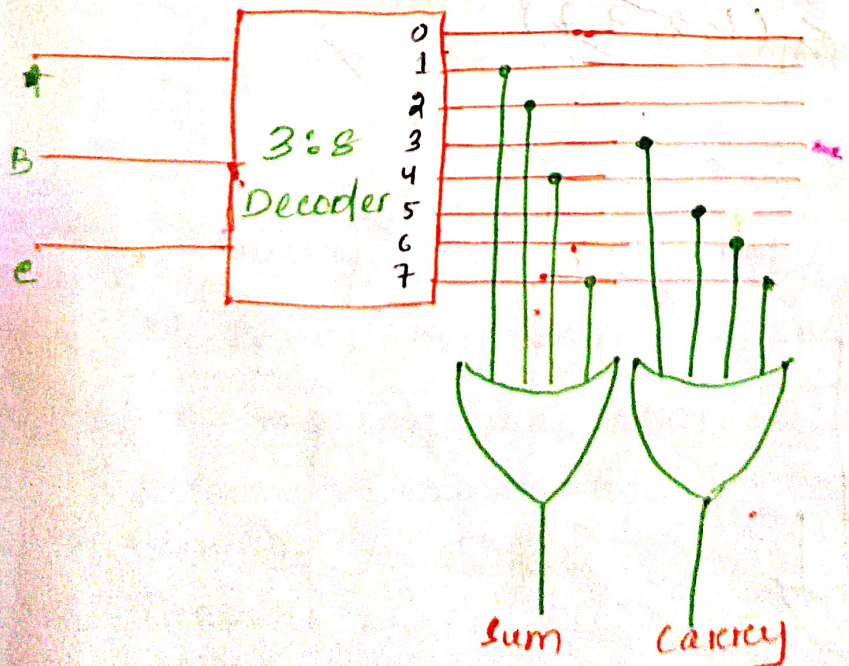
Implement a Full adder using Decoder circuit



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

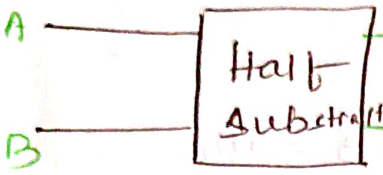
Carry = $\Sigma(3,5,6,7)$

As the full Adder has three inputs, so we need 3:8 decoder.



Implement Half Substractore using Decoder.
Implement Full substractor using Decoder.
Implement the function $Y(A,B,C,D) = \sum m(2,5,6,11,1$
by using decoder.

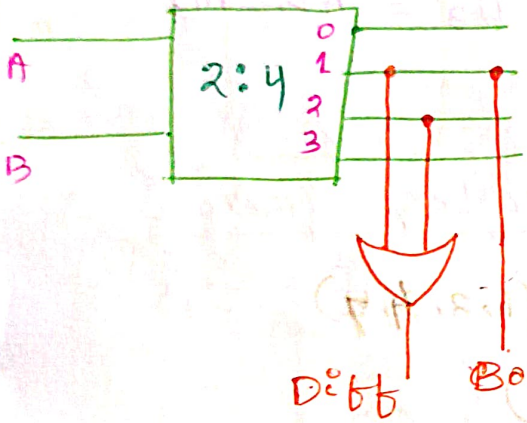
Ans. Implement Half Substractor using Decoder.



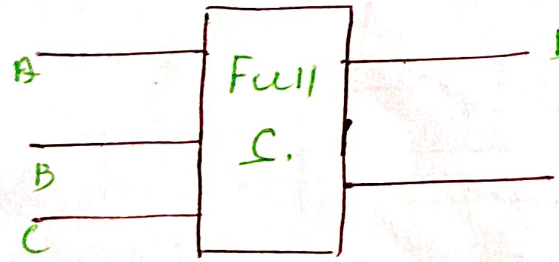
$Diff = \sum(1,2)$

$Borrow = \sum(1)$

As the half substractor two input, so we need 2:4 decoder.

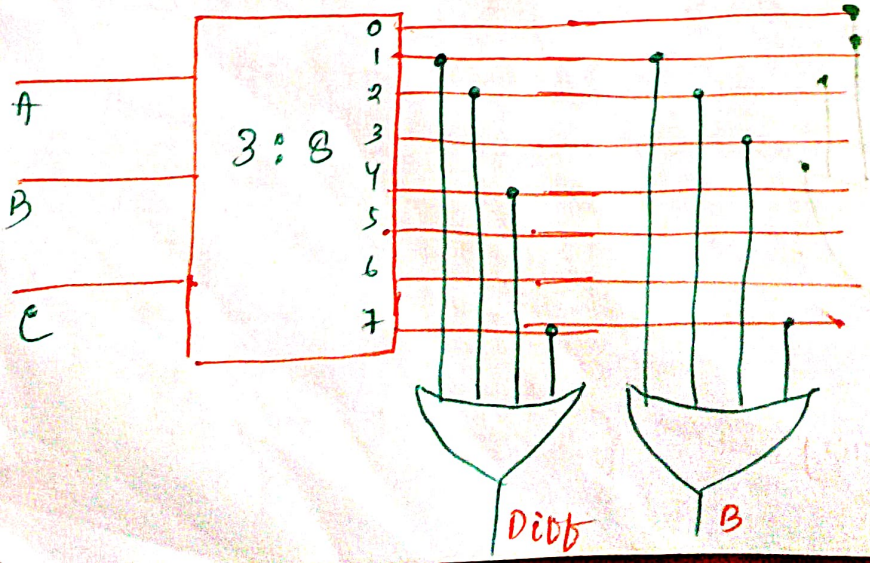


Implement Full Substractor using Decoder

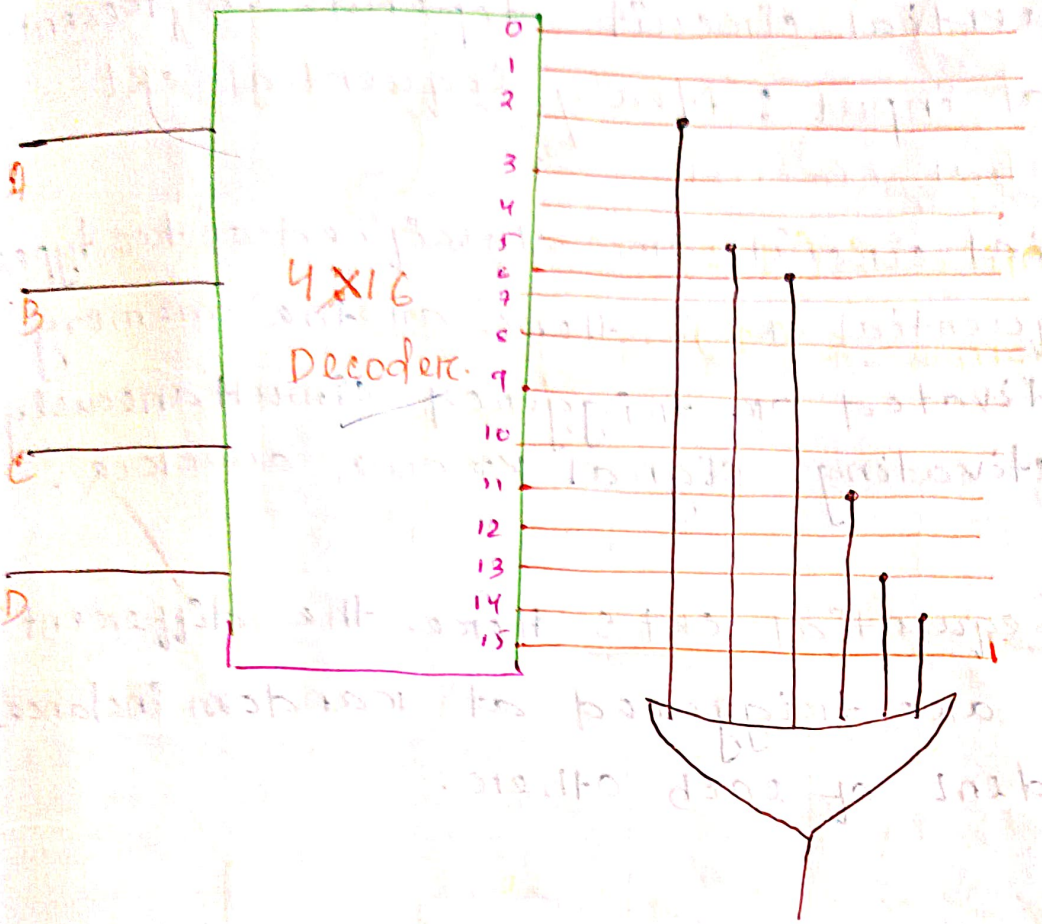


$Diff = \sum(1,2,4,7)$

$B_O = \sum(1,2,3,7)$

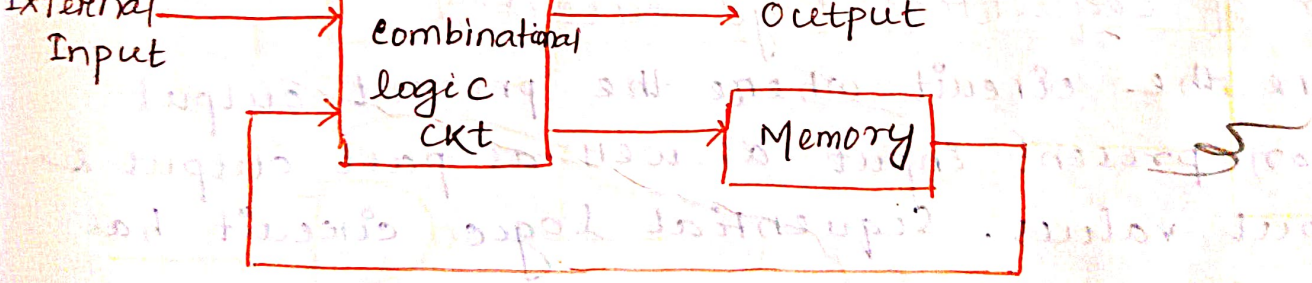


Implement the function $Y(A,B,C,D) = \Sigma(2,5,6,11,13,14)$ by using decoder.



Sequential logic circuit :

Block diagram of sequential logic circuit is as follo



- The binary data stored in the memory is known as state of the memory element.
- Depending on the states, it is classified as three type.

(i) Previous state → Q_{n-1}

(ii) Present state → Q_n

(iii) Next state → Q_{n+1}

- output of sequential circuit depends on present state that is called : Moore Sequential ckt
- output of sequential circuit depends on present state & external input : Mealy Sequential ckt

Types of sequential circuit:

All the sequential circuit are classified as two type

1. Synchronous Sequential ckt : Here all the memory elements are activated or triggered simultaneously by a common activating signal known as clock pulse.
2. Asynchronous Sequential ckt : Here the different memory elements are triggered at random instance of time independent of each other.
3. Clock Signal

- It is always periodic in nature.
- It may be a square wave form or rectangle wave form.

Definition of sequential logic circuit :

These are the circuit where the present output depends on present input as well as past output & past input values. Sequential logic circuit has a feedback network.

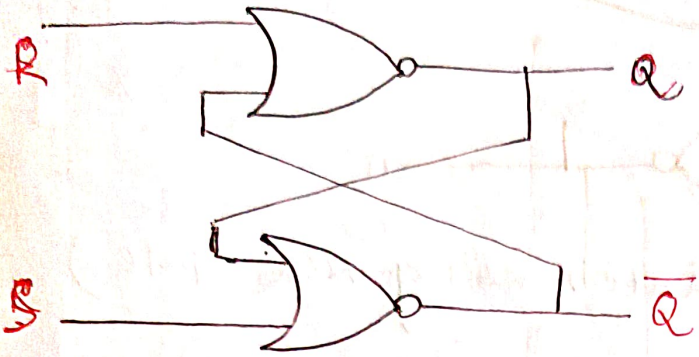
Latch :

- Latch is a logic element that can follow data variation and transfer these changes to output.
- It is a bistable temporary storage element used to store 1 or 0.

S-R latch (S-C Latch)

There are two type of implementation in S-R latch.

1. NOR Implementation: (Active High Input)



NOR		
0	0	1
0	1	0
1	0	0
1	1	0

any input 1, o/p = 0

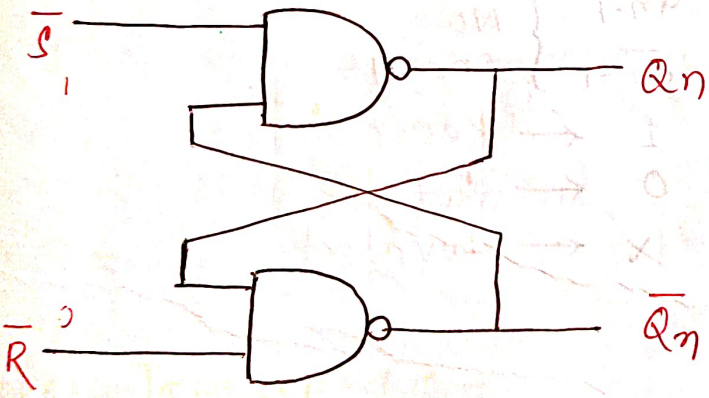
The above NOR implementation of S-R latch has active high input signal.

Latch is a sequential circuit that checks all the inputs continuously and changes its output at any time independently of a clocking signal.

Truth table

S	R	Q_n	$\bar{Q}_n$	State
0	0	Q_{n-1}	$\bar{Q}_{n-1}$	← No change (NC)
0	1	0	1	← Reset
1	0	1	0	← Set
1	1	x	x	← Invalid

2. NAND Implementation: (Active low Input)



$\bar{S}$	$\bar{R}$	
0	0	→ 1
0	1	→ 1
1	0	→ 1
1	1	→ 0

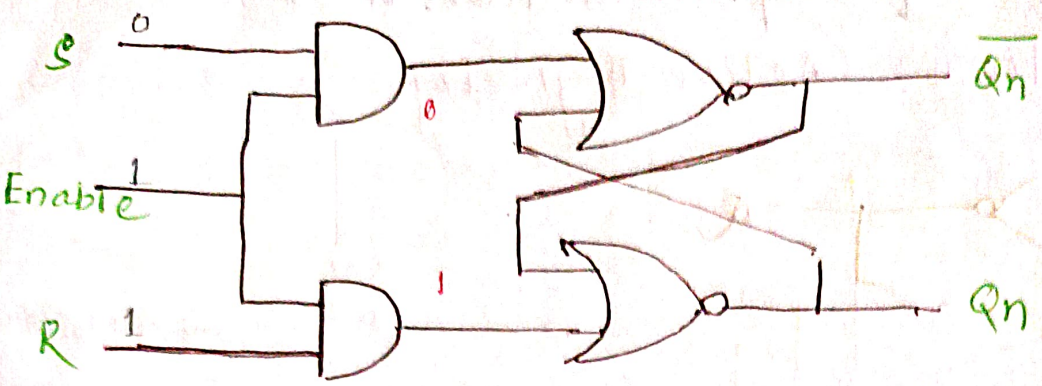
Truth table

$\bar{S}$	$\bar{R}$	Q_n	$\bar{Q_n}$	State
1	1	Q_{n-1}	$\bar{Q}_{n-1}$	No change
1	0	0	1	Reset
0	1	1	0	Set
0	0	x	x	Invalid

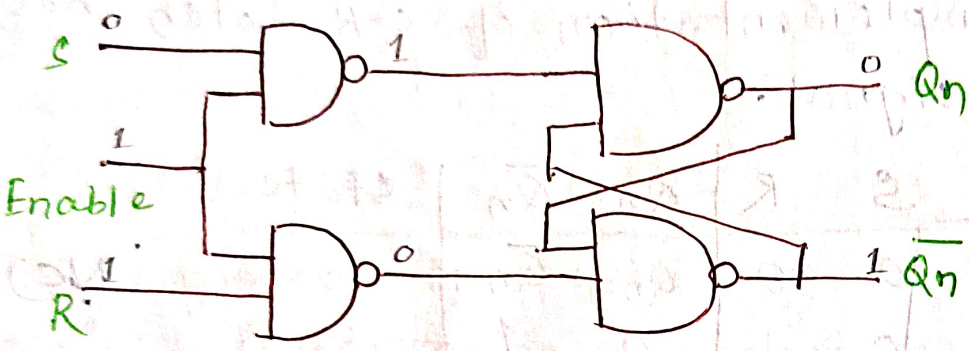
Note → In nand gate if any of the input is zero, output is 1.

→ In NOR gate if any of the input is 1, output is zero.

Gated S-R latch



(Enable / Triggered S-R latch using NOR gate)

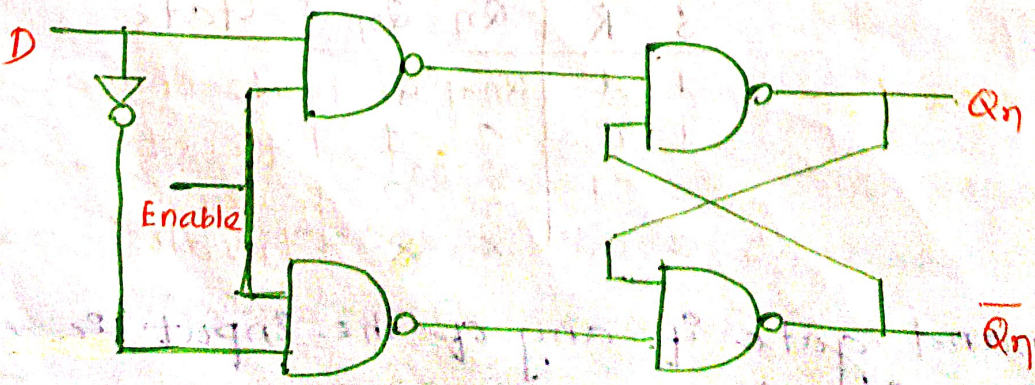


(Enable / Triggered S-R latch using NAND gate)

Truth table for both

E	S	R	Q_n	$\overline{Q_n}$	state
0	X	X	Q_{n-1}	$\overline{Q_{n-1}}$	No change
1	0	0	Q_{n-1}	$\overline{Q_{n-1}}$	No change
1	0	1	0	1	← Reset
1	1	0	1	0	← Set
1	1	1	X	X	← Invalid

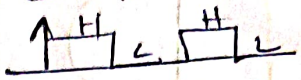
D-latch (Delay latch)



Truth table

E	D	Q_n	$\overline{Q_n}$	
0	X	Q_{n-1}	$\overline{Q_{n-1}}$	← No change
1	1	1	0	← Set
1	0	0	1	← Reset

Level Trigger
The circuit is triggered during the high level or low level of the clk pulse.

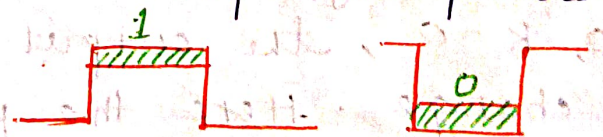


FLIP-FLOP :

- Flip-flop is a non-transparent device which requires a clock signal for its operation.
- It is a device with two stable states. (bistable)
- Flip-flop maintains its output state until directed by an input signal to change its state.

Latch	Flip-flop
- It is transparent.	- It is non transparent.
- Latch is level triggered.	- F/F is edge triggered.
- It may or may not required clock signal.	- It definite require clock signal.
- Less number of components are required.	- More number of components are required
- Due to less component processing faster.	- Due to large number of component processing is slower

Level Triggered :-



Edge Triggered :-

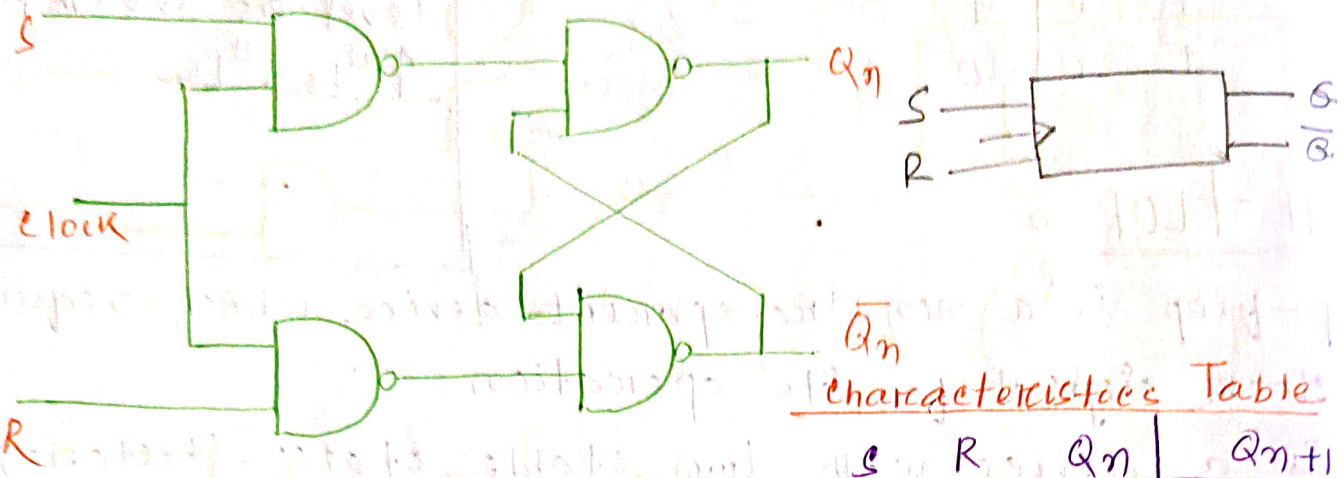
Sequential circuit which
- changes its state either at the positive edge (Rising edge) or at negative edge (Falling edge)
- is triggered only during a signal transition. from 0 to 1 from 1 to 0 and is disabled during the rest of the clk pulse duration.

+ve edge
Rising edge
Leading edge

-ve edge
Falling edge
Trailing edge.

S-R Flip-flop

- It is also known as Set-Reset / Set-clear flip-flop.



Truth table (Function table)

Clk	S	R	Qn State.
↓	X	X	Qn-1 } No change
↑	0	0	Qn-1 } No change
↑	0	1	0 ← Reset
↑	1	0	1 ← Set
↑	1	1	X ← Invalid

Characteristics Table

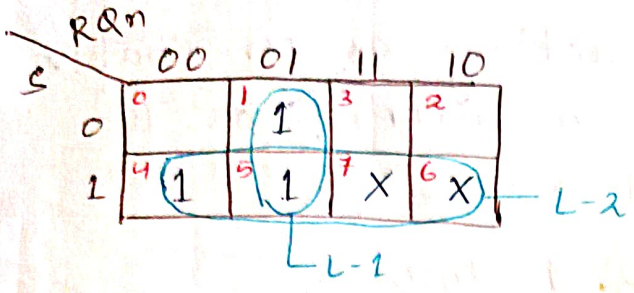
S	R	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Operation

- When the clock signal is deactivated (↓) for any value of S & R the flip-flop will store previous output value. So it is no change condition.
- When the clock signal is activated (↑).
  (a) For S = 0, R = 0, the output of previous stage will not change. Here the flip-flop will act as memory element.
  (b) When S = 0, R = 1, the output of F/F is zero. which is Reset condition.
  (c) When S = 1, R = 0, th Qn = 1, which is a set condition for the flip-flop.
  (d) In 1-1 input (S = 1, R = 1) the output of the flip-flop can be 0 & 1, which is an invalid condition. To avoid

this invalid condition, J-K flip flop is used.

characteristic equation for characteristic table using K map:



L-1

0	0	1
1	0	1

$\bar{R}Q_n$

L-2

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

S.

$$Q_{n+1} = S + \bar{R}Q_n$$ – characteristic eqn of S-R F/F.

Excitation-Table.

A table showing the required input condition for every possible transition of a flip flop output.

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

A characteristic table defines the logical properties of a F/F by describing its operation in tabular form

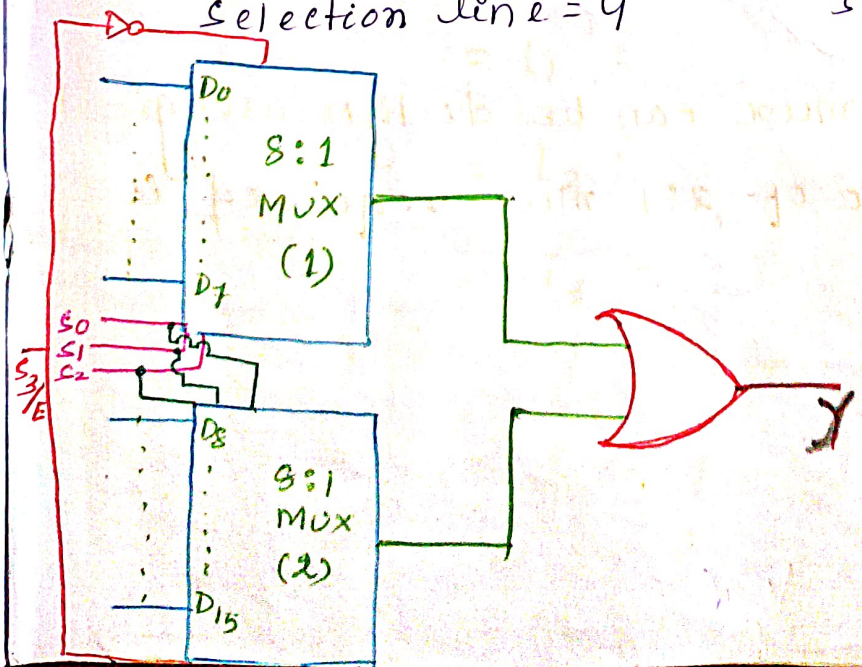
Q(n)	S	R	Q(n+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X

Multiplexer Tree

Design 16:1 MUX using 8:1 MUX:

16:1
$2^4 : 1$
$n = 4$
I/p line = 16
o/p line = 1
selection line = 4

8:1
$2^3 : 1$
$n = 3$
I/p line = 8
o/p line = 1
selection line = 3



MSB S_3	S_2	S_1	LSB S_0	O/P
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	0	D_{12}
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

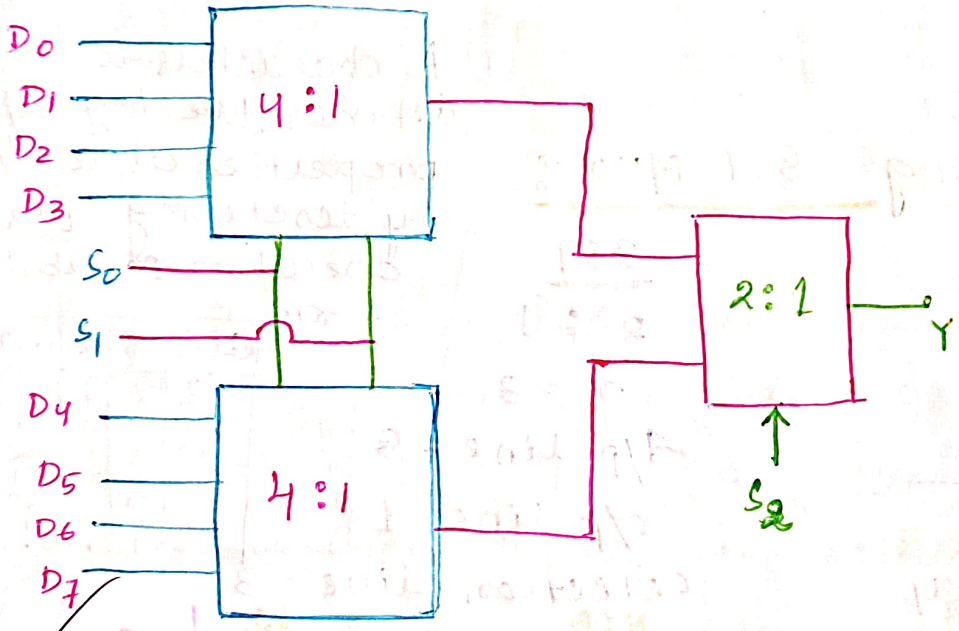
- For designing 16:1 MUX, two 8:1 MUX required
- For both the 8:1 mux S_0, S_1, S_2 are common selection line.
- The selection line S_3(MSB) is acting as an enable line for 8:1 mux, but it is considered as the selection line for final 16:1 mux.

Design 8:1 MUX using 4:1 & 2:1 mux.

For.

8:1	4:1	2:1
$2^3 : 1$	$2^2 : 1$	$2^1 : 1$
$n = 3$	$n = 2$	$n = 1$
I/p line = 8	I/p line = 4	I/p line = 2
output line = 1	o/p line = 1	o/p line = 1
Selection line = 3	S.L = 2	S.L = 1

For 8:1 MUX, 2@ two 4:1 & One 2:1 mux is required



MSB S_2	S_1	S_0	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Note: Any higher order mux can be design using 2:1 mux only. So the no of 2:1 mux required is always $2^n - 1$.

Eg: 64:1 $\xrightarrow{64-1=63}$ 2:1

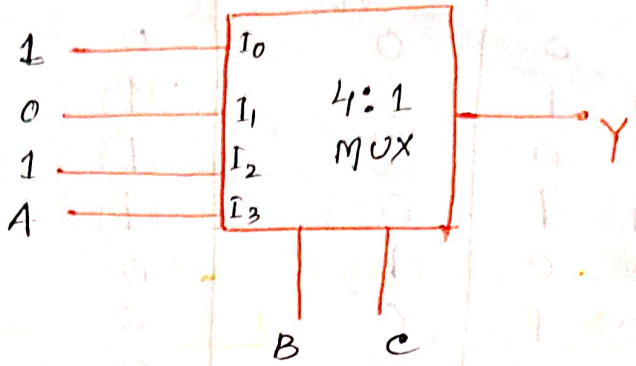
Boolean Function implementation using MUX

Q. $F(A,B,C) = \Sigma m(0,2,4,6,7)$ Implement using 4:1 MUX

I/P, selection line.

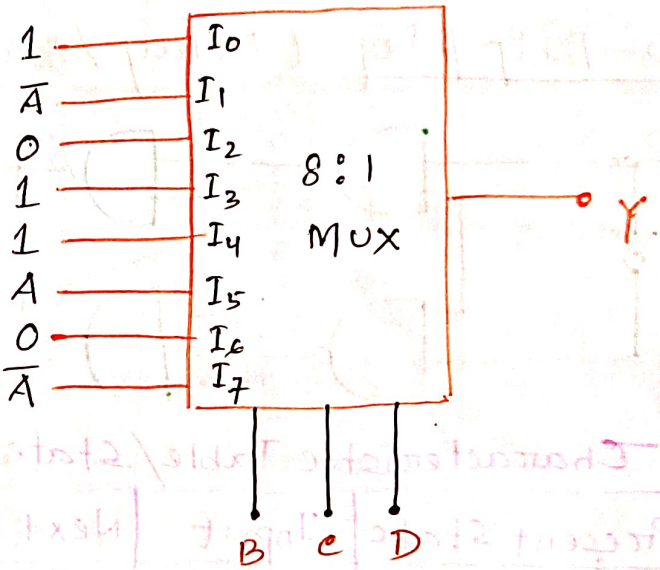
Method - 1

I/P →	I_0	I_1	I_2	I_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	1	0	1	A



Q. $F(A,B,C,D) = \Sigma m(0,1,3,4,7,8,11,12,13)$ using 8:1 MUX

I/P	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	$\bar{A}$	0	1	1	A	0	$\bar{A}$



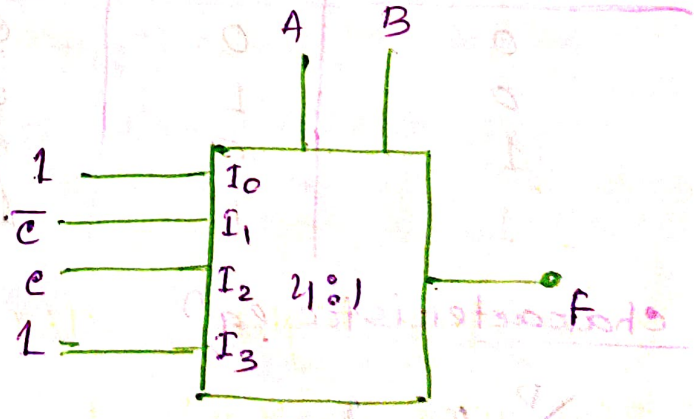
METHOD - II

Q. $F(A,B,C) = \Sigma m(0,1,2,5,6,7)$

selection I/P

Implement using 4:1 MUX :

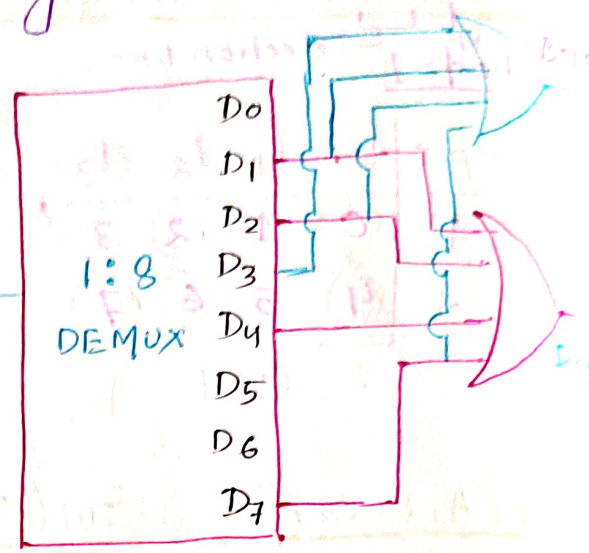
A	B	C	F	
0	0	0	1	$= I_0 = 1$
0	0	1	1	
0	1	0	1	$= I_1 = \bar{C}$
0	1	1	0	
1	0	0	0	$= I_2 = C$
1	0	1	1	
1	1	0	1	$= I_3 = 1$
1	1	1	1	



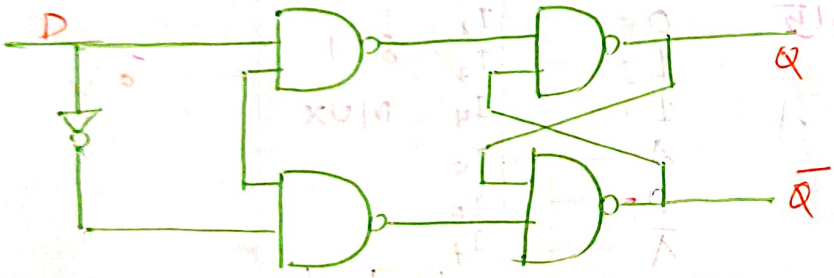
DEMULTIPLEXER IMPLEMENTATION:

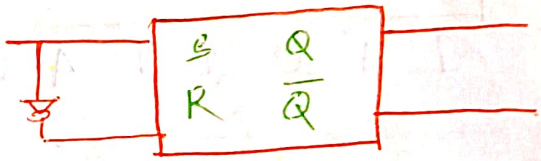
Q. Implement full substractor using demux:

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



D-Flip flop (Delay / Data F/F)





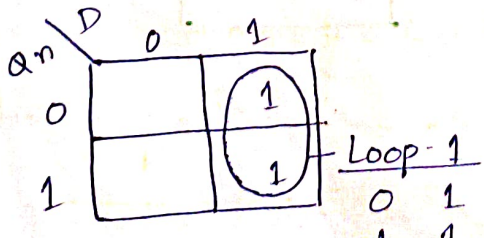
Characteristic Table / State Table

Present State	Input	Next state
Qn	D	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

clk	Input	O/p	State
	D	Qn+1	
↓	X	Qn	← No change
↑	0	0	← Reset
↑	1	1	← set.

Characteristic eqn



Loop 1

0	1
1	1

D.

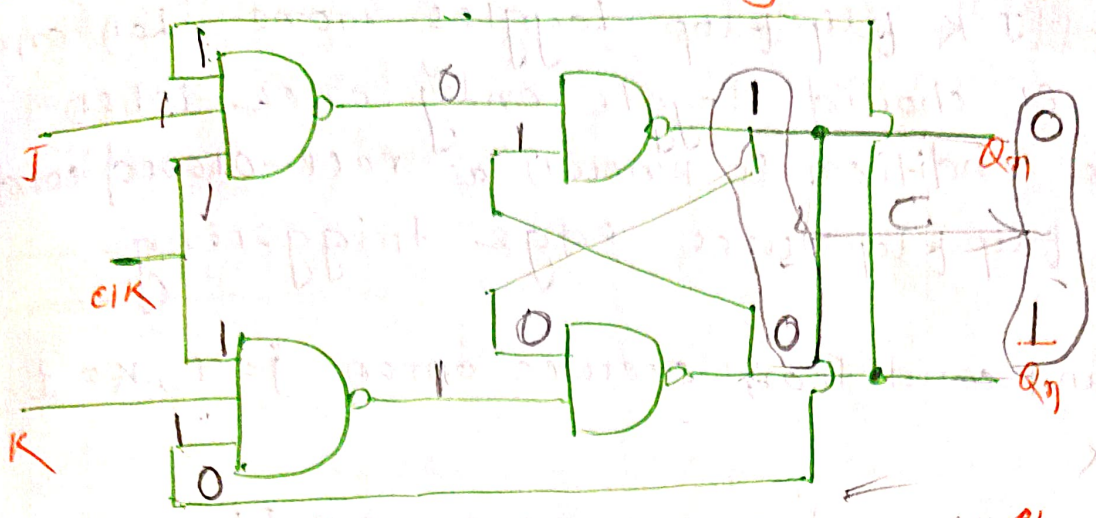
$Q_{n+1} = D$

Excitation Table

Qn	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

- From the characteristic eqn, the future output value (Qn+1) can be predicted if the present input & present output value is given.

J-K Flip flop (Jack – Kilby)



Truth table

CLK	J	K	Qn+1	state
↓	X	X	Qn	No change
↑	0	0	Qn	No change
↑	0	1	0	← Reset
↑	1	0	1	← set
↑	1	1	$\overline{Qn}$	← Toggle

Characteristic Table / State table

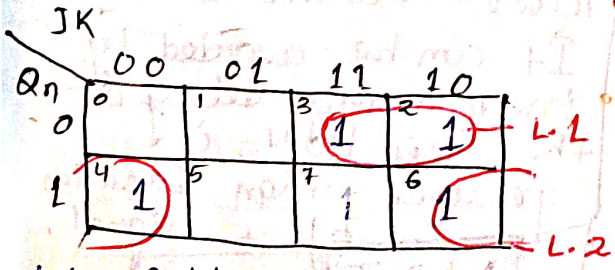
Present state	Input		Next state
Qn	J	K	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

- J-K flip flop is designed to overcome the invalid condition present in S-R flip flop when both S & R = 1.
- so to avoid invalid condition in J-K flip flop feedback structure is used.
- In J-K flip flop, when both the inputs are 1 the output toggles / complements the present output values.

Excitation table

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

characteristic eqn



L-1 $\quad 0\ 11 \quad 01\ 0 \quad \frac{}{\overline{Qn}J}$; L-2 $\quad 100 \quad 110 \quad \frac{}{Qn\,\overline{K}}$

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

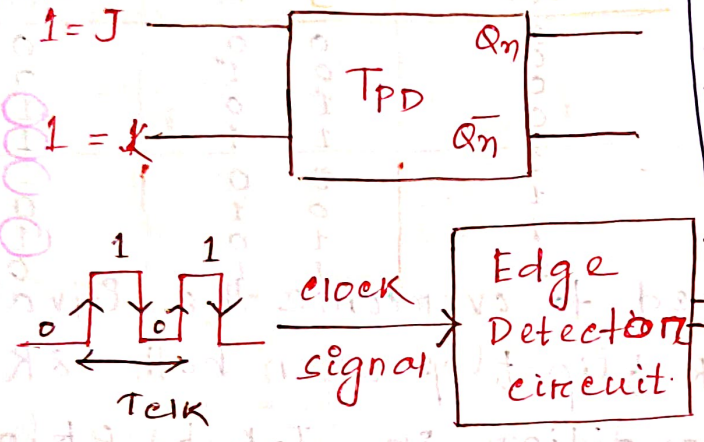
Race-around condition

- The output of J-K flip flop toggles more than once time where as it should toggle only once when J = 1, K = 1, this condition is known as race-around cond^n
- Generally the flip flop uses edge triggering clock pulse.
- The race-around condition occurs when J = 1, K = 1 and $T_{PW} > T_{CLK}$

T_{pd} → Propagation delay time of J-K F/F.

T_{PW} → Edge trigger pulse duration (clock pulse width)

Method-1



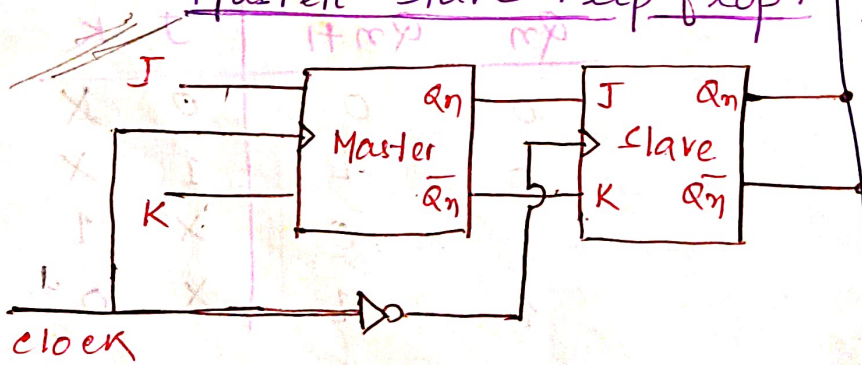
J=K=1, o/p Q=0, after the propagation delay Δt of flip flop, the o/p of JK flip flop changes from 0 to 1. The o/p of JK connected to input hence after next delay Δt, the o/p changes from 1 to 0. This process will continue till the end of the applied clock signal. Thus the o/p of JK flip flop is uncertain. This condition called race-around cond^n. It can be avoided by increasing delay of the flip flops. i.e greater than duration of clock signal.

$$T_{PW} < T_{PD} < T_{CLK}$$

$$\Delta t > T$$

Method - II

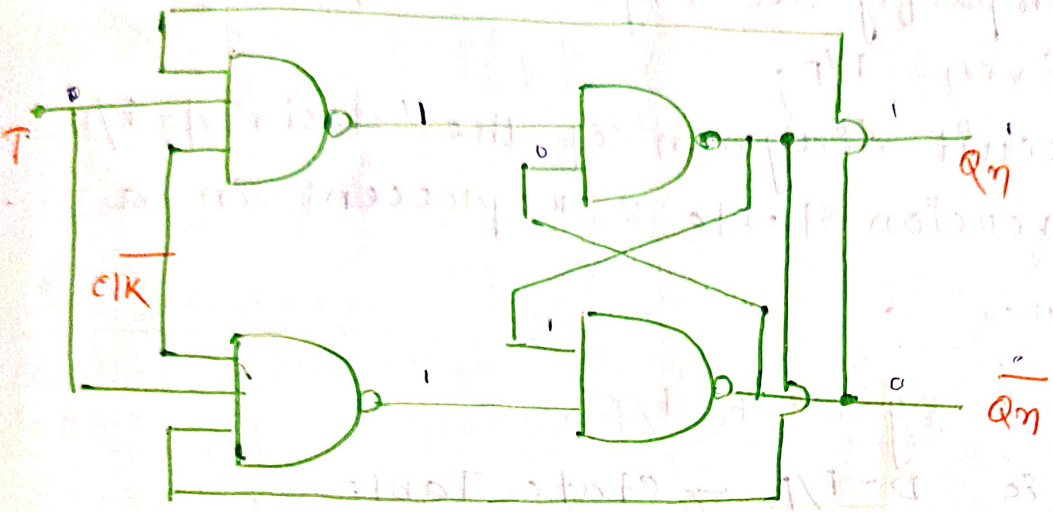
Master - slave Flip flop.

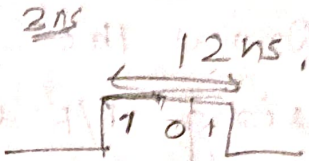


- The overall circuit acts as a single J-K flip flop which stores only one bit of data.
- At the +ve edge of clock pulse master flip flop

is active & -ve edge of clock pulse slave is active. At any instant only one F/F is triggered.
- By this master-slave arrangement the race-around condition is avoided.

T-Flip flop : (Toggle F/F)





Truth table

CLK	T	Q_{n+1}	State:
↓	X	Q_n	No change.
↑	0	Q_n	No change.
↑	1	$\overline{Q_n}$	← Toggle.

characteristic table / State Table

Present o/p	Input	Next o/p.
Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

characteristic eqn

Q_n \ T	0	1
0		1
1	1	

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

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

Excitation table.

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

Flip flop conversion:

- Write down the state table of the required F/F and excitation table of the given F/F.
- combine the state table & the excitation table to get conversion table.
- Using K-map simplify the expression for excitation input of the given F/F.
- Draw the circuit diagram of the desired F/F using F/F conversion table & represent in a clock diagram.

Q. Realise D-F/F using S-R F/F.

Required F/F is D-F/F → state Table

Given F/F → S-R F/F → Excitation Table.

Step-I

D-F/F state Table.

D	Qn	Qn+1
0	0	0
0	1	0
1	0	1
1	1	1

Excitation Table S-R F/F

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

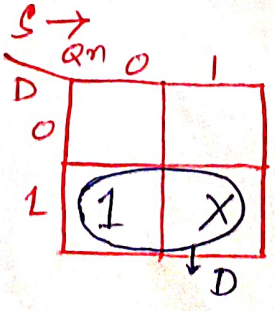
Step-II

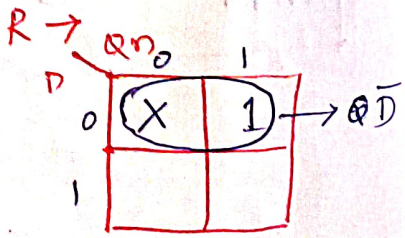
conversion table

D	Qn	Qn+1	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Step-III

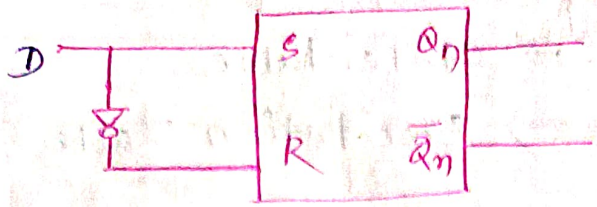
- Qn+1 column is the predicted value or next state value so during K-Map Qn+1 is not taken into consideration





$S = D$

$R = \bar{D}$



Above is the block diagram representation of D F/F using S/R F/F.

Q: Realize D-F/F using J-K flip flop.

Required flip flop = D F/F → state Table

Given flip flop = J-K F/F → Excitation Table.

D-state table

Present state	Input	Next state
Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

KMap

Qn+1 column is the predicted value or next state value. So during K-map, Qn+1 is not taken into consideration.

conversion table

D	Qn	Qn+1	J	K
0	0 →	0	0	X
1	0	1	1	X
0	1	0	X	1
1	1	1	X	0

For J

D \ Qn	0	1
0		X
1	1	X

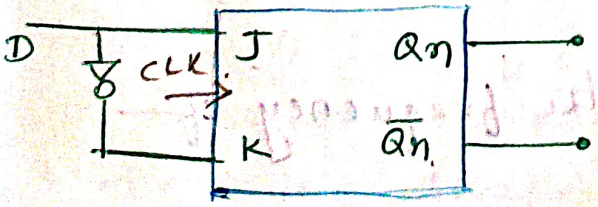
D

For K

D \ Qn	0	1
0	X	1
1	X	

$\overline{D}$

$J = D$, $K = \overline{D}$



Above in the block diagram representation of D F/F using J-K F/F.

Q. Realise T-F/F using S-R Flipflop:

Required F/F = T F/F. → state table

Given F/F = S-R F/F → Excitation table.

State table

Present output	I/p.	Next output
Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

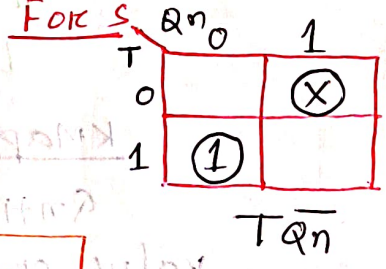
Q	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Conversion Table

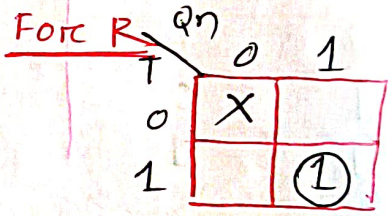
T	Qn	Qn+1	S	R
0	0	0	0	X
1	0	1	1	0
0	1	1	X	0
1	1	0	0	1

Kmap

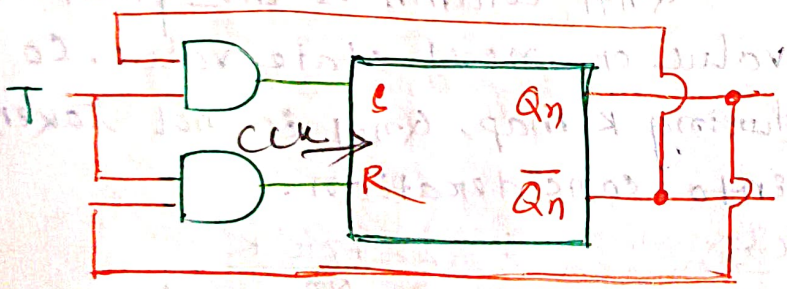
Qn+1 column is the predicted value or next state value. So during K-map Qn+1 is not taken into consideration.



TQn̄



TQn



Above is the block diagram representation of T F/F using S-R F/F.

Application of Flip flop:

- The flip flop when cascaded in perticulare fashion can be used as a counter.
- Flip flops are used to reduce the frequency of input signal.
- It is used as shift registers.

Flip flop conversion Table

To obtain	S	R	J	K	D	T
S-R	—	—	S	R	$S+\bar{R}Q_n$	$S\bar{Q}_n+RQ_n$
J-K	$J\bar{Q}_n$	$\bar{K}Q_n$	—	—	$J\bar{Q}_n+\bar{K}Q_n$	$J\bar{Q}_n+\bar{K}Q_n$
D	D	$\bar{D}$	D	$\bar{D}$	—	$D\oplus Q_n$
T	$T\bar{Q}_n$	TQ_n	T	T	$T\oplus Q_n$	—

Sequential Network Design :

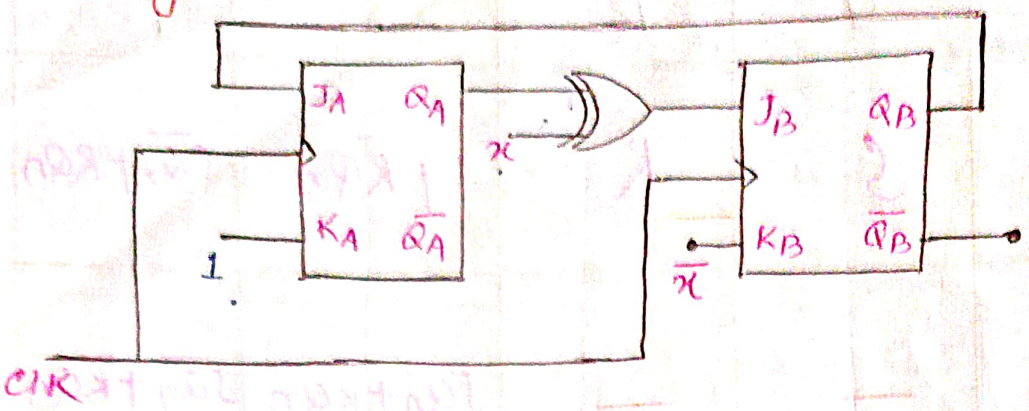
- circuit given ⟶ Draw state Diagram
- state diagram given ⟶ circuit / Block diagram
- Truth table given ⟶ circuit / Block diagram

From circuit draw state diagram :

Steps to find out the state digram from circuit diagram

- find out the logic expression from the given circuit diagram.
- Draw the state table which contain present state, input, F/F input, output & finally the next state. It is used to determine from the characteristic equation
- Draw the state diagram by looking at the state table from present state to next state.

Q. Draw the state diagram from the given circuit diagram.



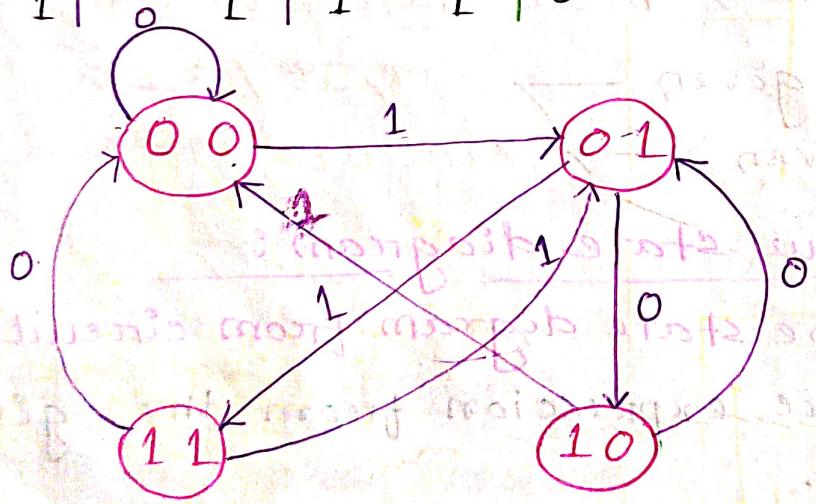
$J_A = Q_B$

$K_A = 1$

$J_B = Q_A \oplus x$

$K_B = \overline{x}$

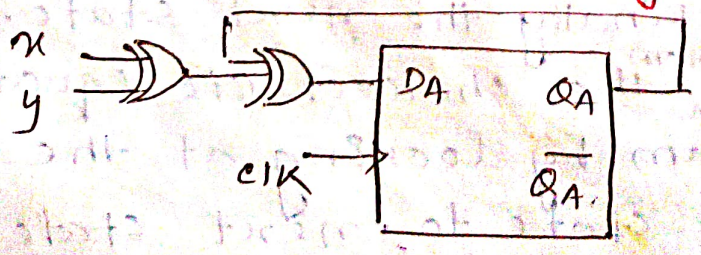
Present state		Input	F/F input				Next state	
Q_A	Q_B	x	J_A	K_A	J_B	K_B	Q_{An+1}	Q_{Bn+1}
0	0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	0	1
0	1	0	1	1	0	1	1	0
0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	0	1
1	0	1	0	1	0	0	0	0
1	1	0	1	1	1	1	0	0
1	1	1	1	1	0	0	0	1



$Q_{An+1} = J\overline{Q}_{An} + Q_{An}\overline{K}$

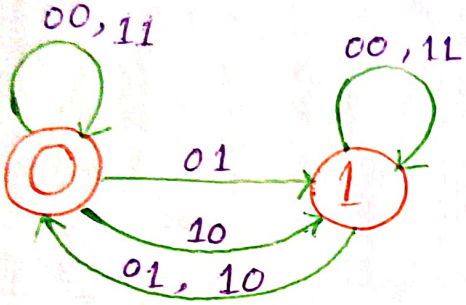
$Q_{Bn+1} = J\overline{Q}_{Bn} + Q_{Bn}\overline{K}$

Q. Draw the state diagram from the given circuit diagram

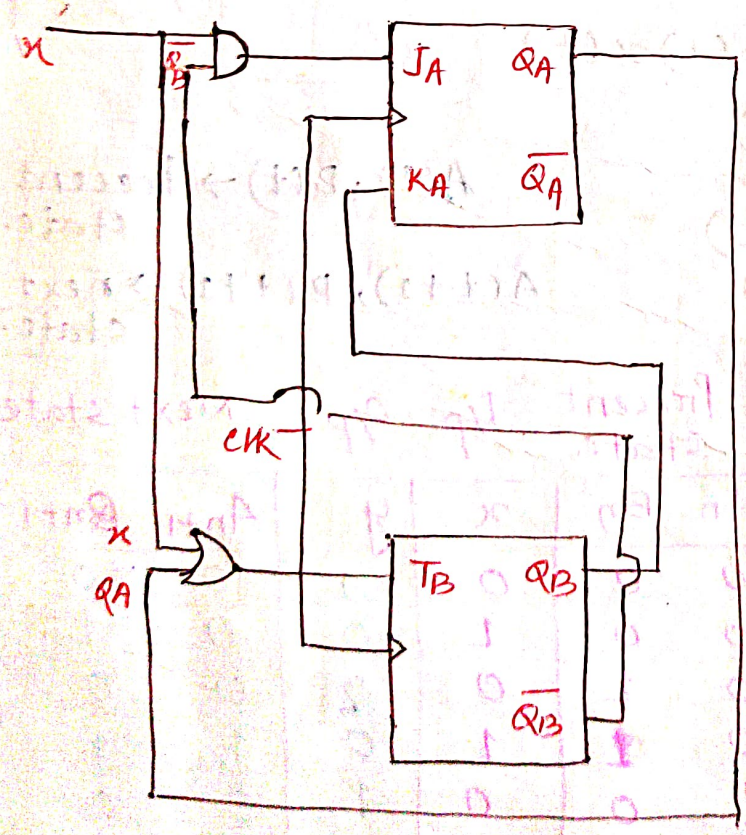


$D_A = x \oplus y \oplus Q_A$

Present state	Input		F/F input	Next state
Q_A	x	y	D_1	Q_{An+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Q. Draw the state diagram from the given circuit diagram

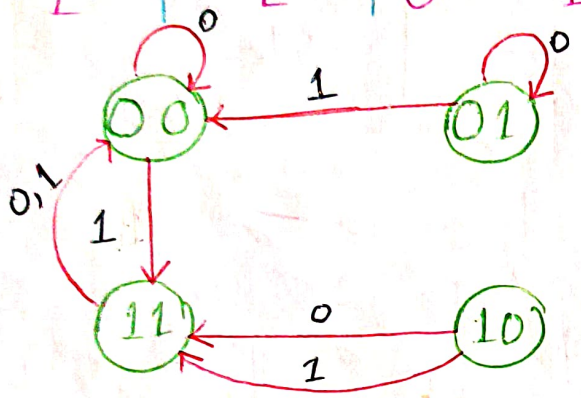


$T_B = x + Q_A$

$J_A = \bar{Q}_B \cdot x$

$K_A = Q_B$

Present state		I/P	F/F Input			Next state	
Q_A	Q_B	x	J_A	K_A	T_B	Q_{An+1}	Q_{Bn+1}
0	0	0	0	0	0	0	0
0	0	1	1	0	1	1	1
0	1	0	0	1	0	0	1
0	1	1	0	1	1	0	0
1	0	0	0	0	1	1	1
1	0	1	1	0	1	1	1
1	1	0	0	1	1	0	0
1	1	1	0	1	1	0	0



State diagram from given Equation :

Q- Draw the state diagram from the following eqn,

$$A(t+1) = A(t)x(t) + B(t)x(t)$$

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

$$y = [A(t) + B(t)]\,\bar{x}(t)$$

A(t), B(t) → Present state.

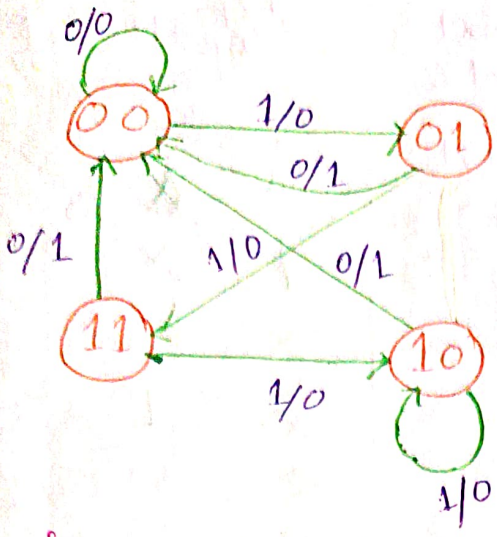
A(t+1), B(t+1) → next state.

$$A_{n+1} = A_n x + B_n x$$

$$B_{n+1} = \bar{A}_n \cdot x$$

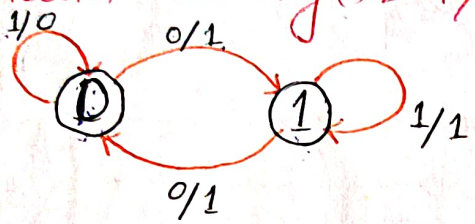
$$y = [A_n + B_n]\,\bar{x}$$

Present state.		I/P	O/P	Next state	
A_n	B_n	x	y	A_{n+1}	B_{n+1}
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	1	0



State diagram to circuit diagram:

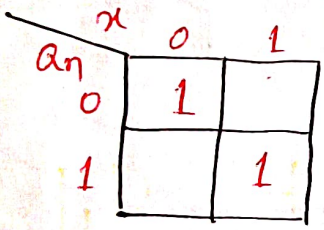
Q. From the circuit diagram from the given state diagram using (i) D F/F (ii) Using J/K F/F.



(i)

Present State	I/P	O/P	Next State	F/F I/P
Q_n	x	y	Q_{n+1}	D
0	1	0	0	0
0	0	1	1	1
1	1	1	1	1
1	0	1	0	0

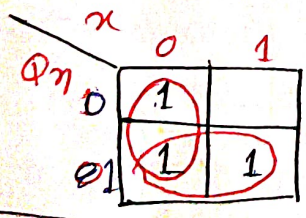
Equation for 'D':



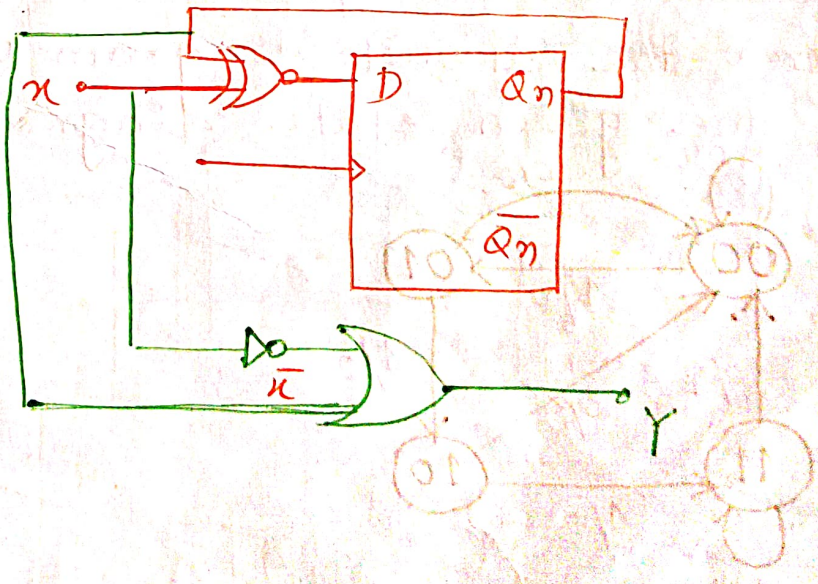
$D = \bar{Q_n}\,\bar{x} + Q_n x$

$\boxed{D = Q_n \odot x}$

y :-



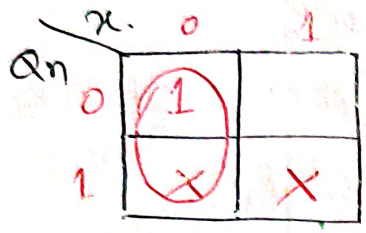
$\boxed{y = Q_n + \bar{x}}$



(ii)

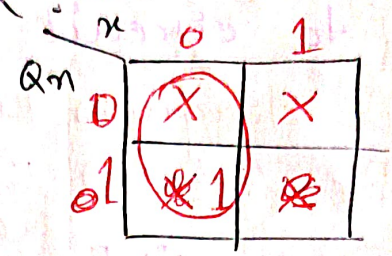
Present state	I/P	O/P	Next state	F/F input	
Q_n	x	y	Q_{n+1}	J	K
0	1	0	0	0	X
0	0	1	1	1	X
1	1	1	1	X	0
1	0	1	0	X	1

J :-

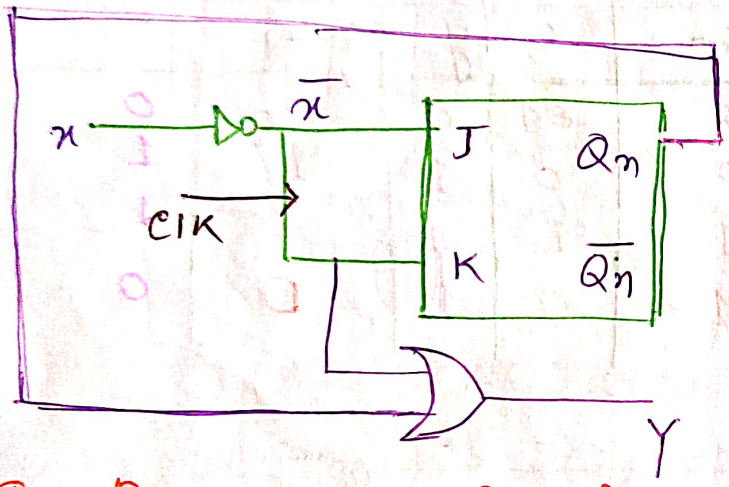


$J = \bar{x}$

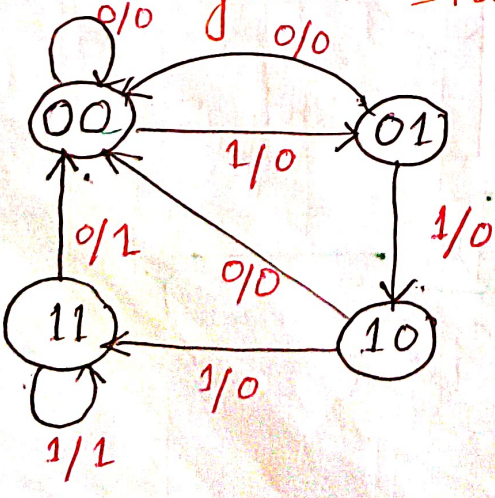
K :-



$K = \bar{x}$

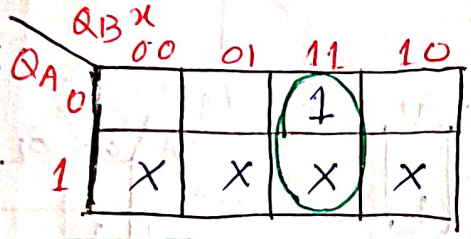


Q. Draw the circuit diagram using J-K F/F from the given state diagram,



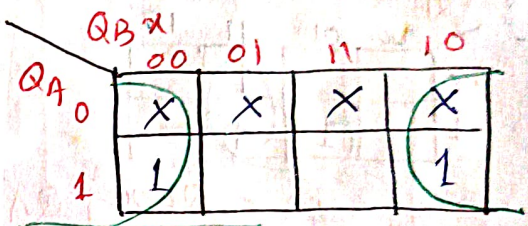
Present state		I/P	O/P	Next state		J_A	K_A	J_B	K_B
Q_A	Q_B	x	y	Q_{An+1}	Q_{Bn+1}	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	1	0	0	1	0	X	1	X
0	1	0	0	0	0	0	X	X	1
0	1	1	0	1	0	1	X	X	1
1	0	0	0	0	0	X	1	0	X
1	0	1	0	1	1	X	0	1	X
1	1	1	1	1	1	X	0	X	0
1	1	0	1	0	0	X	1	X	1

J_A :



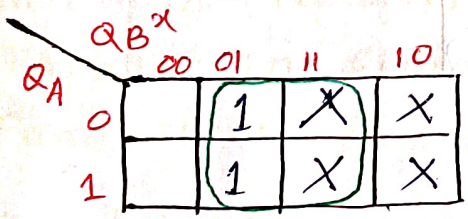
$J_A = Q_B x$

K_A



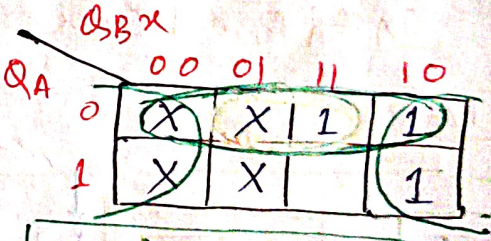
$K_A = \bar{x}$

J_B :



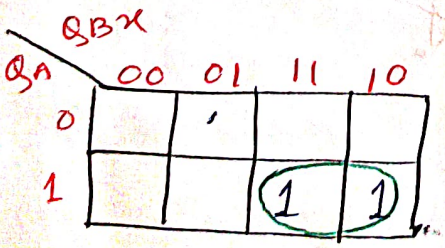
$J_B = x$

K_B

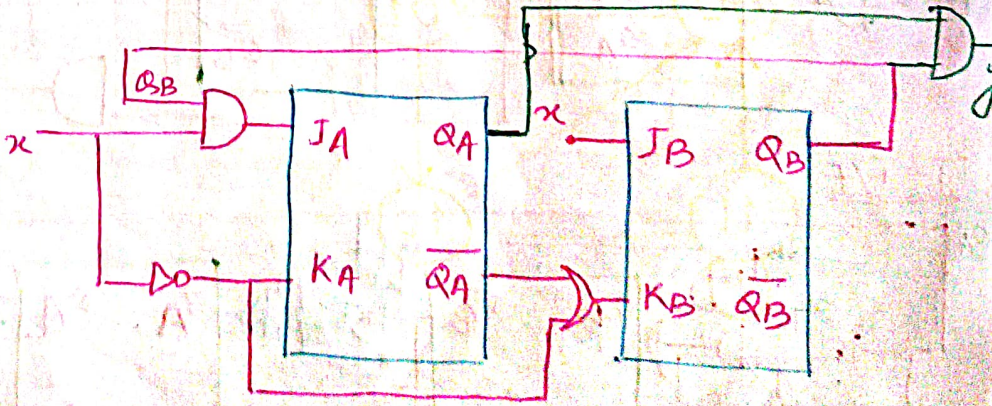


$K_B = \bar{x} + \bar{Q_A}$

y :



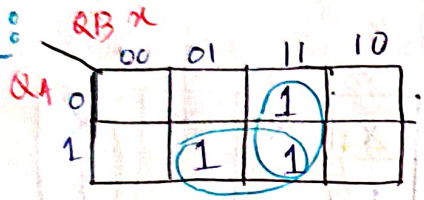
$y = Q_A Q_B$



Q. Draw the circuit diagram using D & T f/f.

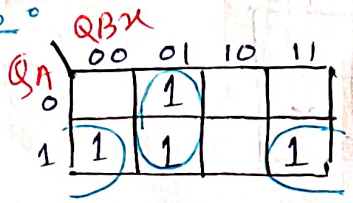
Present State		Input	Output	Next State		F/F input			
Q_A	Q_B	x	y	Q_{An+1}	Q_{Bn+1}	D_A	D_B	T_A	T_B
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	1	0	1
0	1	0	0	0	0	0	0	0	1
0	1	1	0	1	0	1	0	1	1
1	0	0	0	0	0	0	1	1	0
1	0	1	0	1	1	1	1	0	1
1	1	0	1	0	0	0	0	1	1
1	1	1	1	1	1	1	1	0	0

D_A :



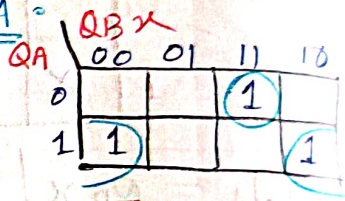
$D_A = Q_B x + Q_A x$

D_B :



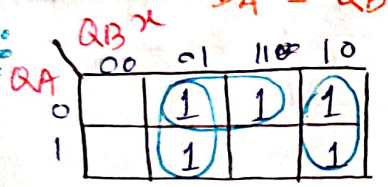
$D_B = \overline{Q_B} x + Q_A$

T_A :



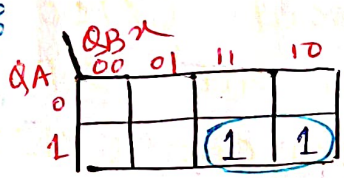
$T_A = \overline{Q_A} Q_B x + Q_A \overline{x}$

T_B :



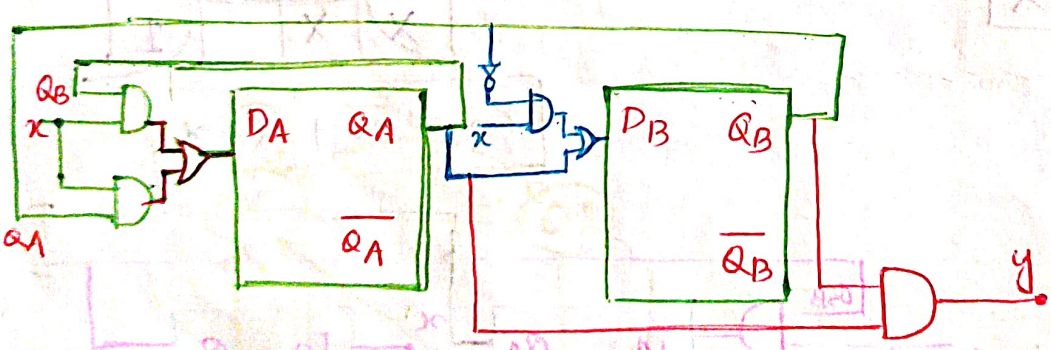
$T_B = \overline{Q_A} x + \overline{Q_B} x + Q_B \overline{x}$

y :

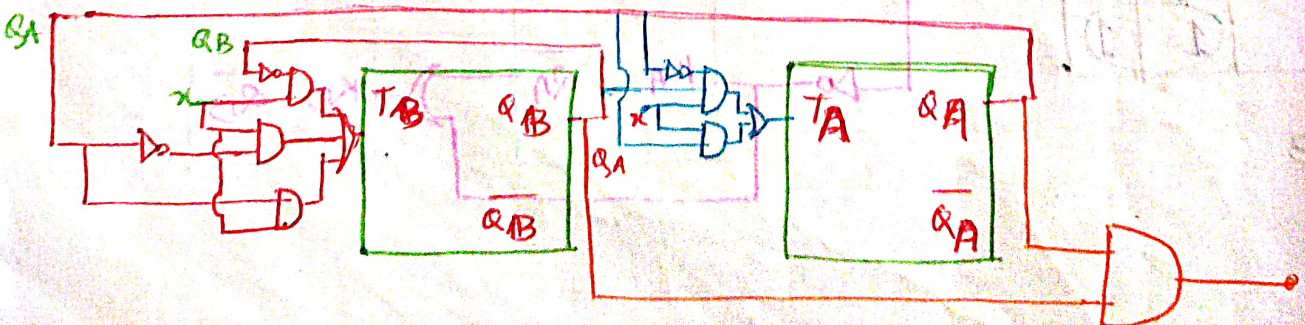


$y = Q_A Q_B$

D-F/F



T-F/F



2. Shift Registers

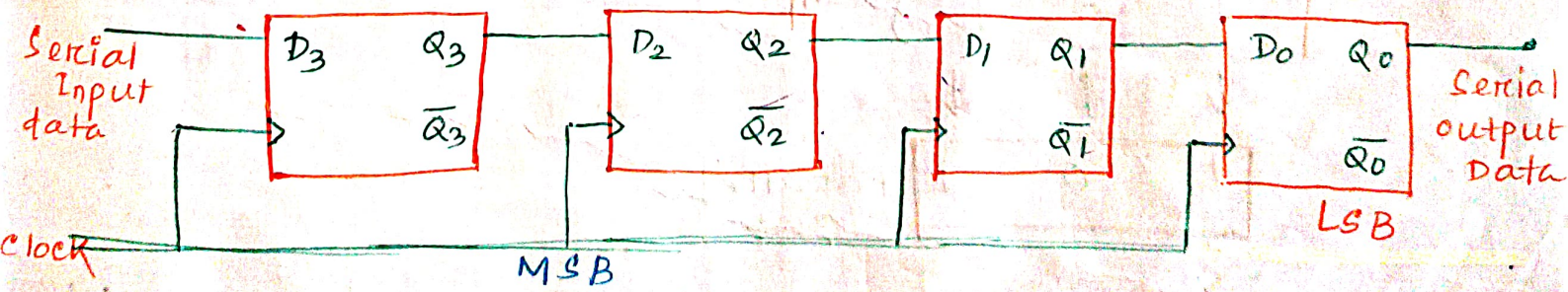
- It consists of a series of connection of D F/F & other flip flop converted to D flip flop.
- Registors are essentially synchronous system.
- Feeding of new data in a register is known as loading / uploading.
- There are two types of loading techniques are there.
  - Serial loading
  - Parallel loading
- Serial data is also known as temporal code & parallel data is also known as spatial code.

Classification of shift Registers

Based on mode of input & output there are four types of shift register.

- Serial In Serial out (SISO) 2n ✓
- Serial In Parallel out (SIPO) n ✓
- Parallel in Serial out (PISO) n+1 ✓
- Parallel in Parallel out (PIPO) 1 ✓

Serial In Serial out (SISO)



CLK	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	1	1	0	1

Serial o/p

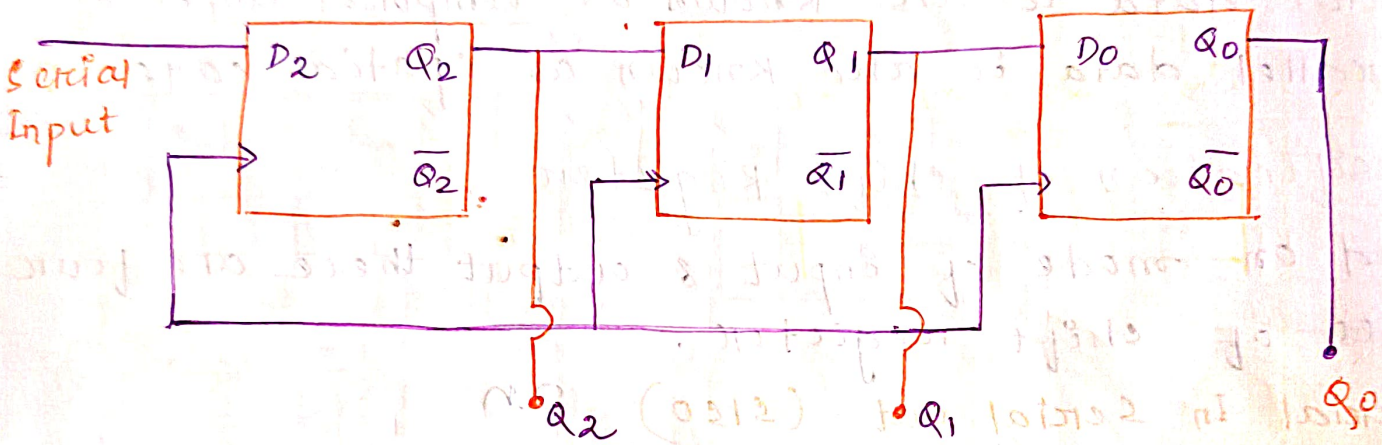
CLK	Q_3	Q_2	Q_1	Q_0
5	0	1	1	0
6	0	0	1	1
7	0	0	0	1

In the above shift register four clock pulses are required to serially input the data & three clock pulses are required to serially output the data.

* For serial data
  - to serial in ⇒ 'n' clock pulse are required
  - to serial out ⇒ 'n-1' clock pulse are required

  n is the number of bit of shift register.

3-bit SIPO Register.



CLK	Q2	Q1	Q0
Initial cond^n 0	0	0	0
1	1	0	0
2	0	1	0
3	1	0	1

Serial Input

1	0	1

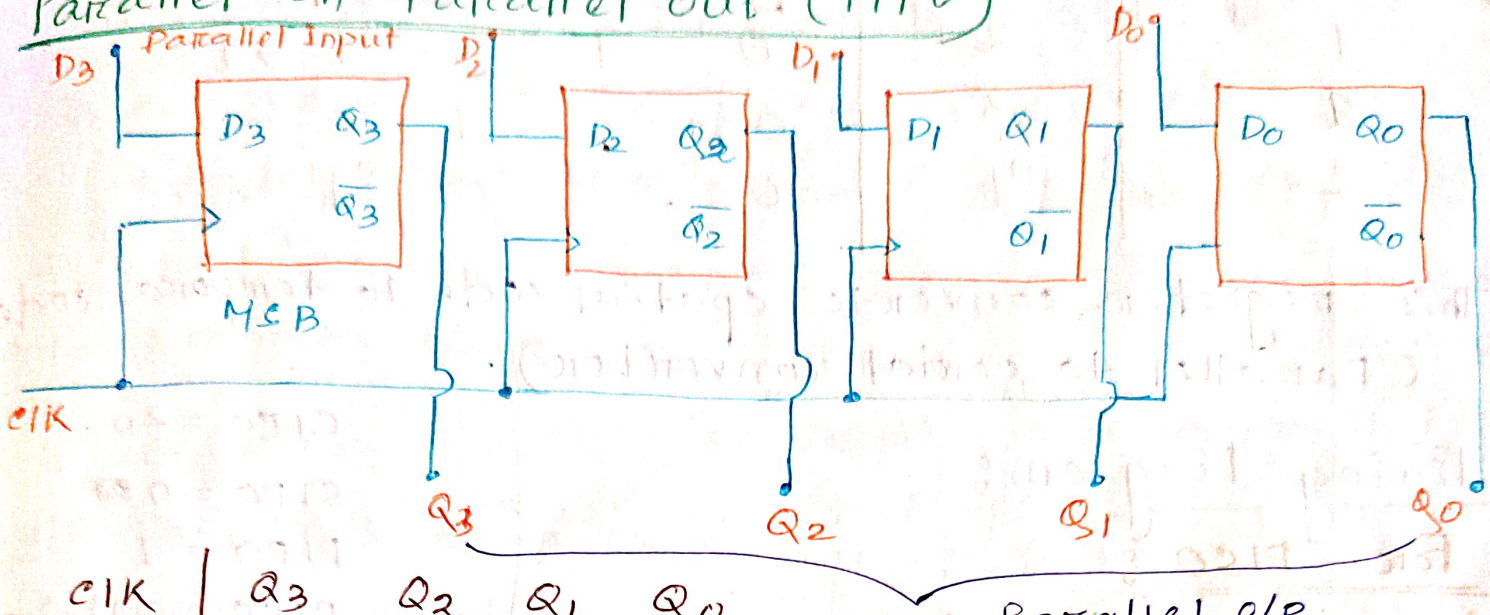
Parallel out.

In serial in parallel out register (for 3 bit register) three clock pulses are required to serially input the data and zero clock pulses are required to parallelly output the data.

- Once the output is available at the output of

4 D F/F at the end of third clock pulse. Parallelly data is retrieved.

Parallel In Parallel out (PIPO)



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

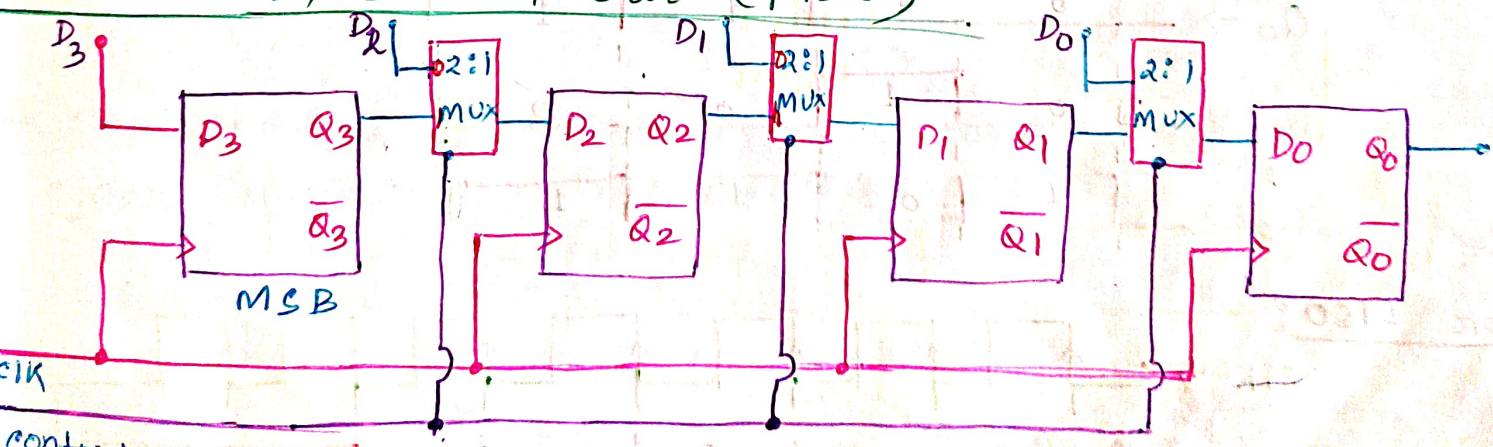
← Parallel input

⇑ Parallel output

Parallel o/p.

- It is the fastest register.
- Here shifting of data is not possible. So it can't be a shift register.

Parallel In Serial out (PISO)



control	Operation
0	→ Parallel Data Input
1	→ Serial Data Output

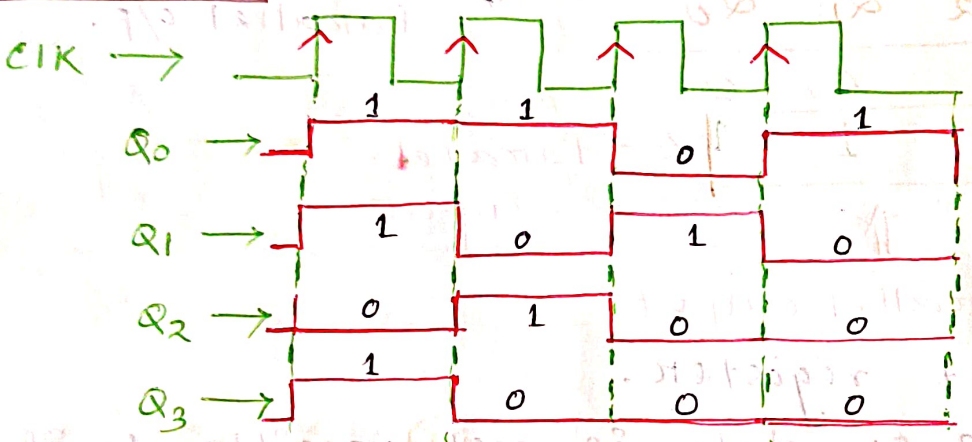
Control Line	Clock	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	0	
0	1	1	0	1	1	← Parallel In.
1	2	0	1	0	1	→ 1
1	3	0	0	1	0	→ 1
1	4	0	0	0	1	→ 0

- This register converts spatial code to temporal code (Parallel to serial converter).

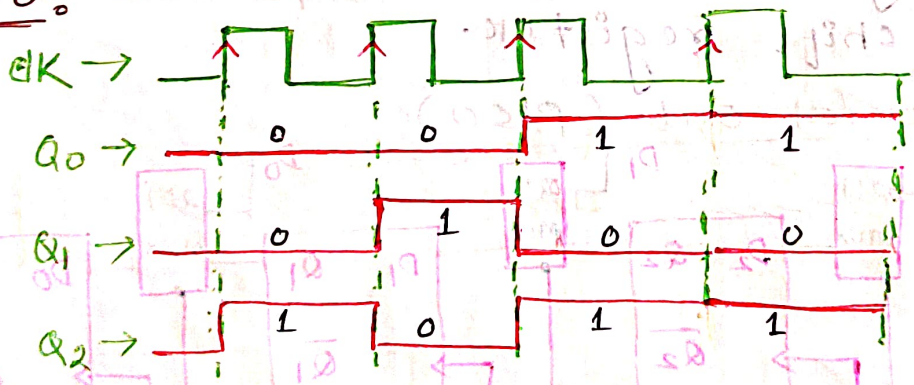
SISO = 2n
SIPO = n
PIPO = 1
PISO = n+1

Timing Diagram:

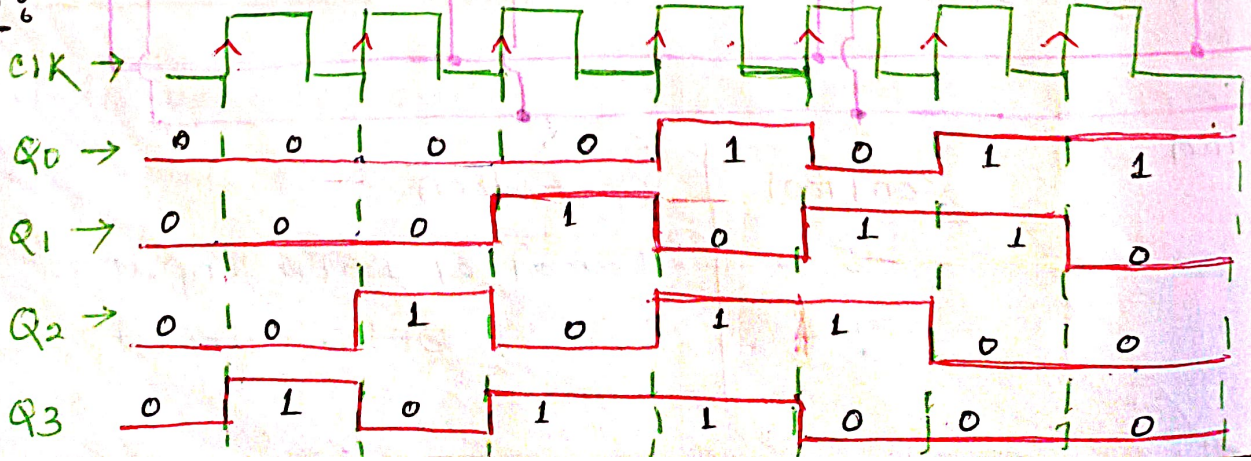
For PISO:



FOR SIPO:



For PISO:



COUNTER

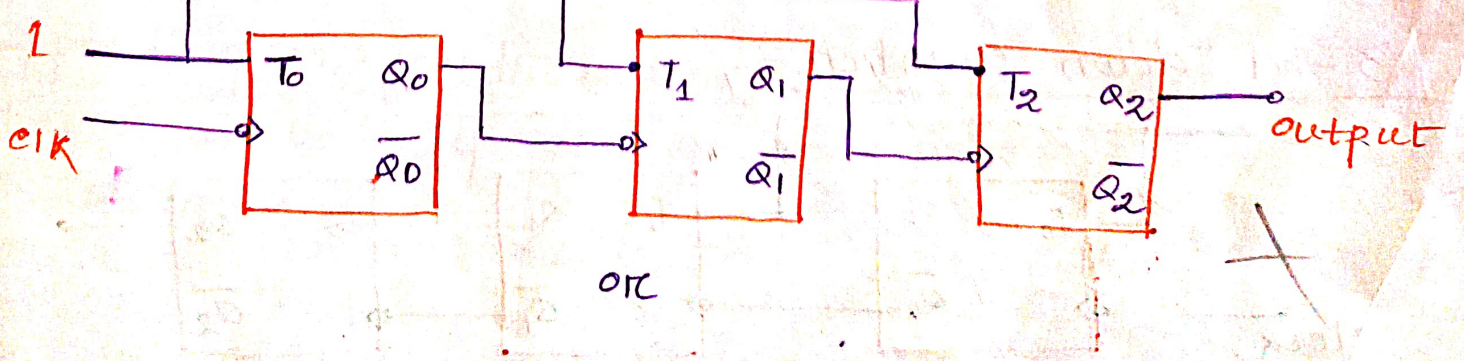
- A counter is a special type of register that goes through a predetermined sequence of states.
- There are two types of counters present.

• Synchronous Counter	• Asychronous counter
- All flip flops are triggered by a common clock pulse.	- The flip flops are triggered either by a clock pulse or the output of adjecent F/F.
- It is faster.	- It is slower.
- It has more complex circuit.	- It has less complex circuit.
- Up, down & Random. counters can be degigned.	- only up & down counters are degigned.
- Decoding errcores are not present.	- Decoding errcores are present.

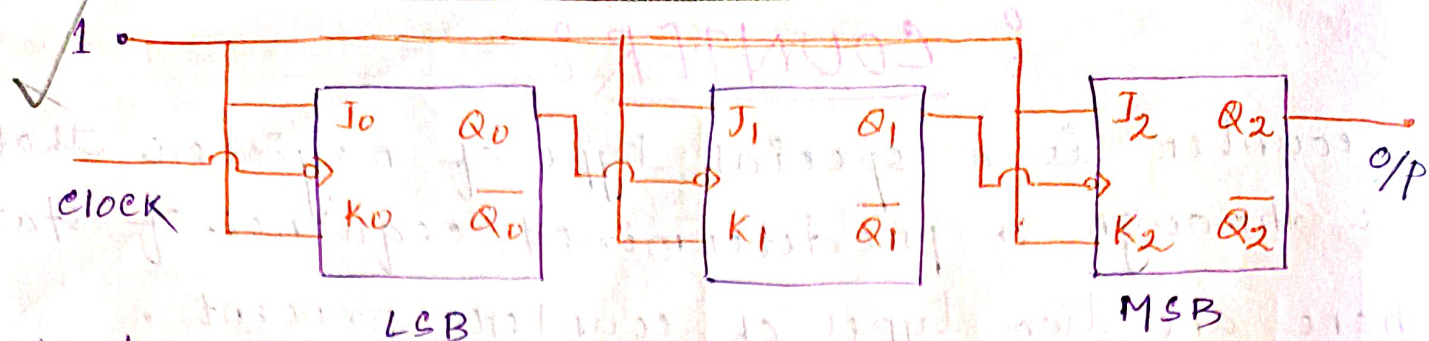
Asynchronous counter (Ripple counter)

- It consist of a sercies connected of T F/F or J-K F/F.
- All the F/F s are operated in only toggle mode.
- The clock pulse is applied to only one F/F. This flip flop represents the LSB of the count sequence
- The output of theach F/F is connected to the clock input of adjecent F/F.

3-bit up counter.



or



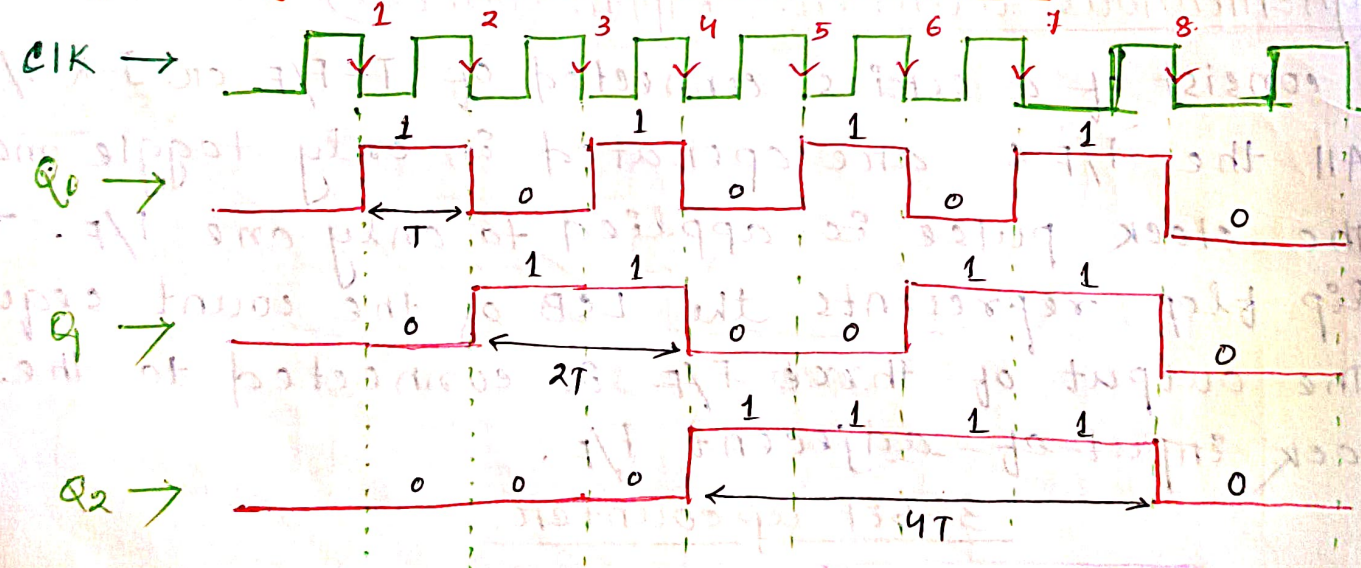
State transition table

clk	Q_2	Q_1	Q_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	1	1	1
	0	0	0

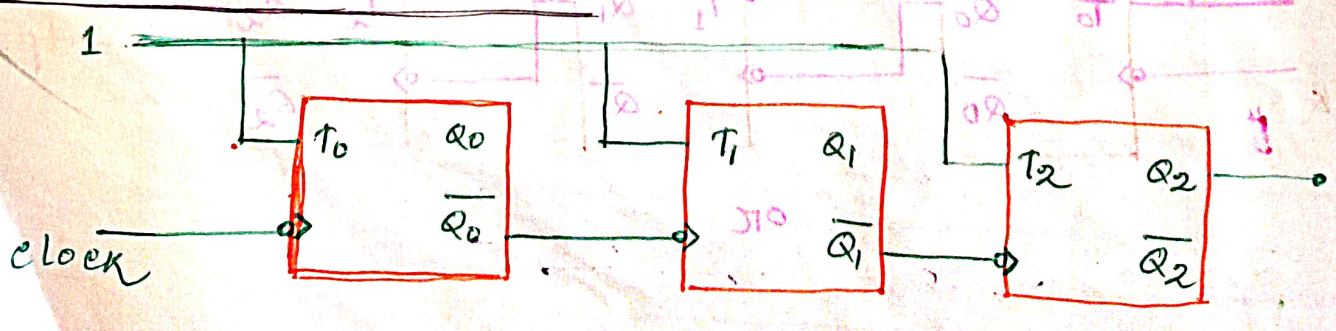
MOD 8 counter

- Q_0 toggles on every clock pulse.
- Q_1 toggles when Q_0 goes from 1 to 0.
- Q_2 toggles when Q_1 goes from 1 to 0.

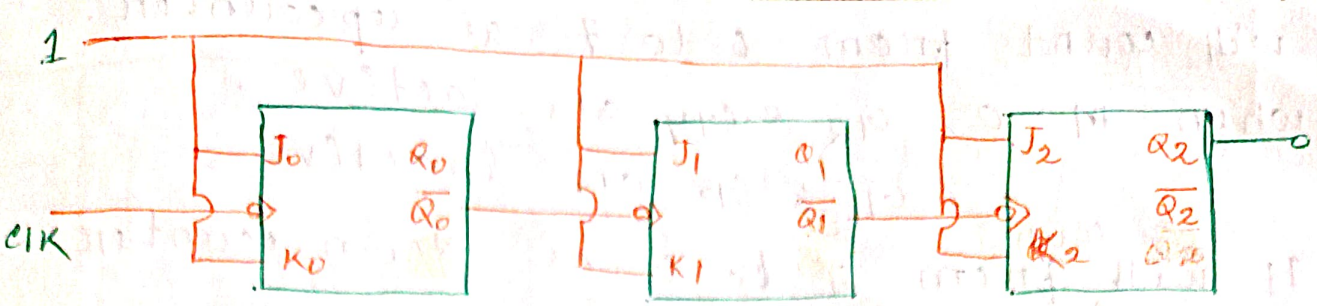
Timing diagram of 3-bit up counter:



3-bit down counter.



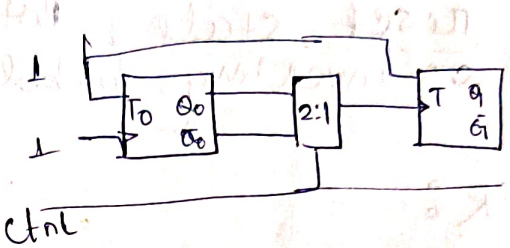
etc



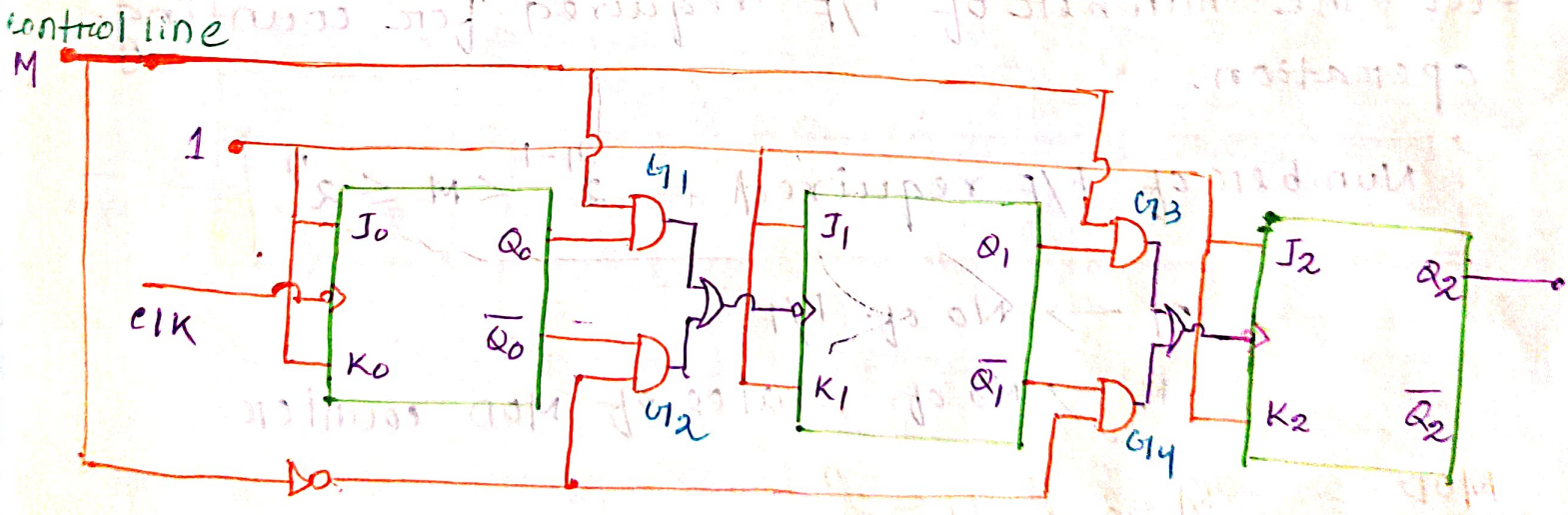
Down counter

clk	Q_2	Q_1	Q_0
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0
8	1	1	1

Timing diagram



3-bit asynchronous Up Down Counter :-



- Depending on control line 'M', the above counter behaves as both up and down counter.

when * M ⇒ 1 → up counter

M = 0 → down counter

* M = 1, G_1 & G_3 becomes active.
G_2 & G_4 becomes inactive.

It will count from 0 to 7 as up counter.

* when M = 0, G12 & G14 are active.
  G11 & G13 are inactive.

It will count from 7 to 0 as down counter.

→ Asynchronous input on a flip-flop have control over the outcomes (Q and $\bar{Q}$) regardless of clock input states. There inputs are called the preset (PRE) and clear (CLR). The preset input drives the F/F to a set state while clear input drives into a reset state. Both are active-low because there are an inverting bubble at I/P

MOD COUNTER:

MOD of a counter means the total number of states through which the counter can progress for counting operation.

- To design a mod counter first we have to find out the number of F/F required for counting operation.

> Number of F/F required = $2^{n-1} < M \leq 2^n$

$n \rightarrow$ No of F/F

$M \rightarrow$ No of states of MOD counter.

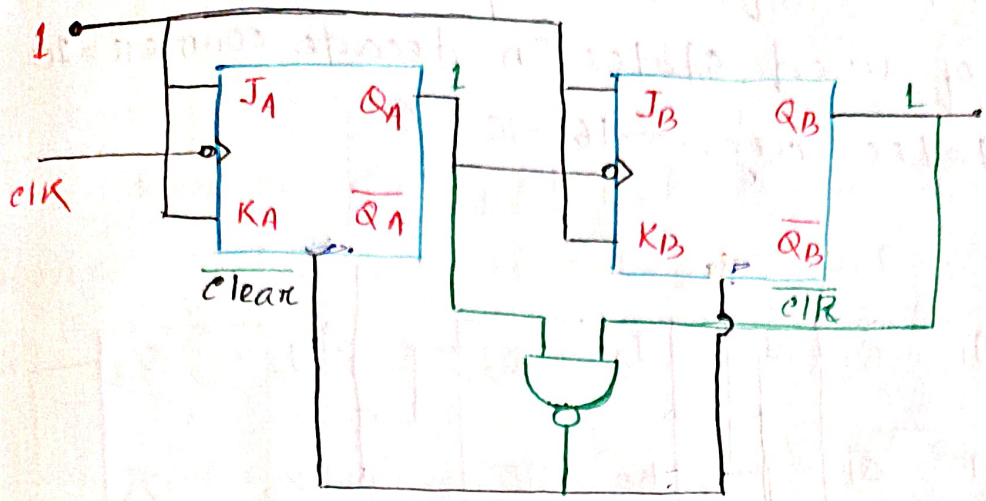
MOD-3 $M = 3$, $M \leq 2^n$

$3 \leq 2^2 \leftarrow$ 2 F/F

$0 \rightarrow 1 \rightarrow 2$

$00 \rightarrow 01 \rightarrow 10 \rightarrow 11$

state



Mod-5 Asynchronous counter :-

No of states = M = 5 ⇒ 0 → 1 → 2 → 3 → 4

No of F/F required =

$M \leq 2^n$

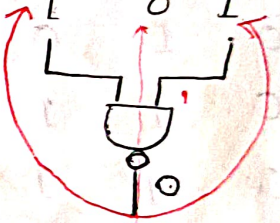
$5 \leq 2^{③}$ ← F/F required

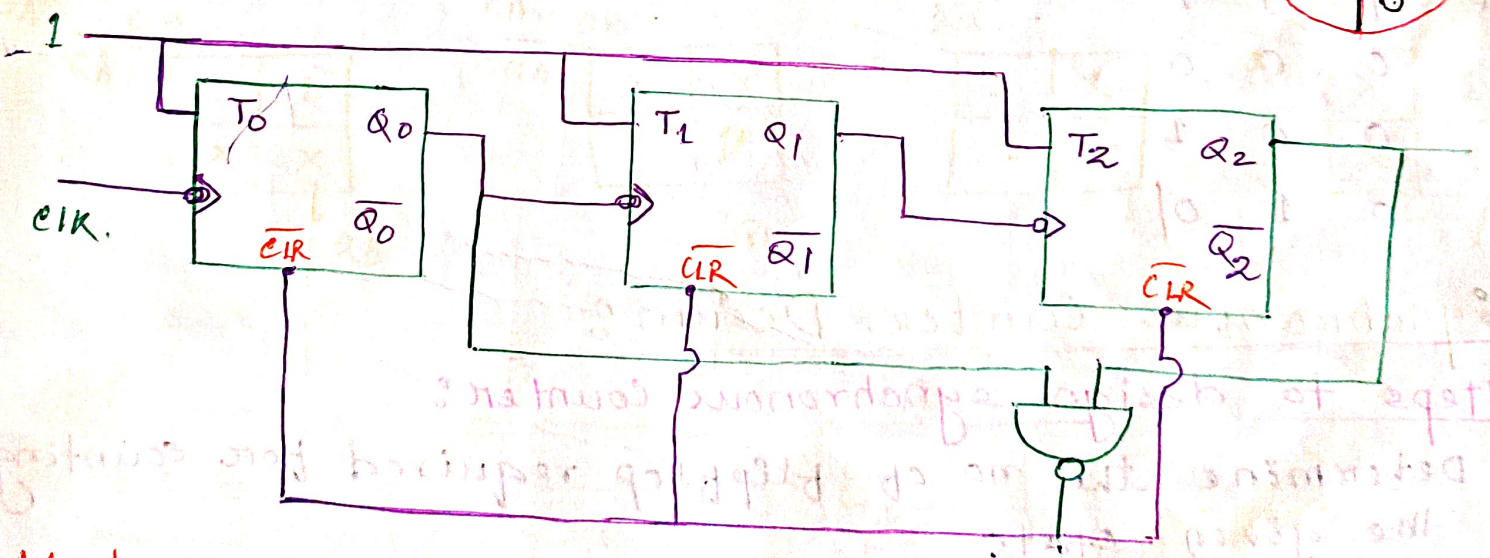
Total states = 8 (0-7)

No of used state = 5 (0 to 4)

No of unused states = 8 − 5 = 3.

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





Mod-10 Asynchronous counter :-

(Decade counter)

Dividing by -10 counter.

0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 0

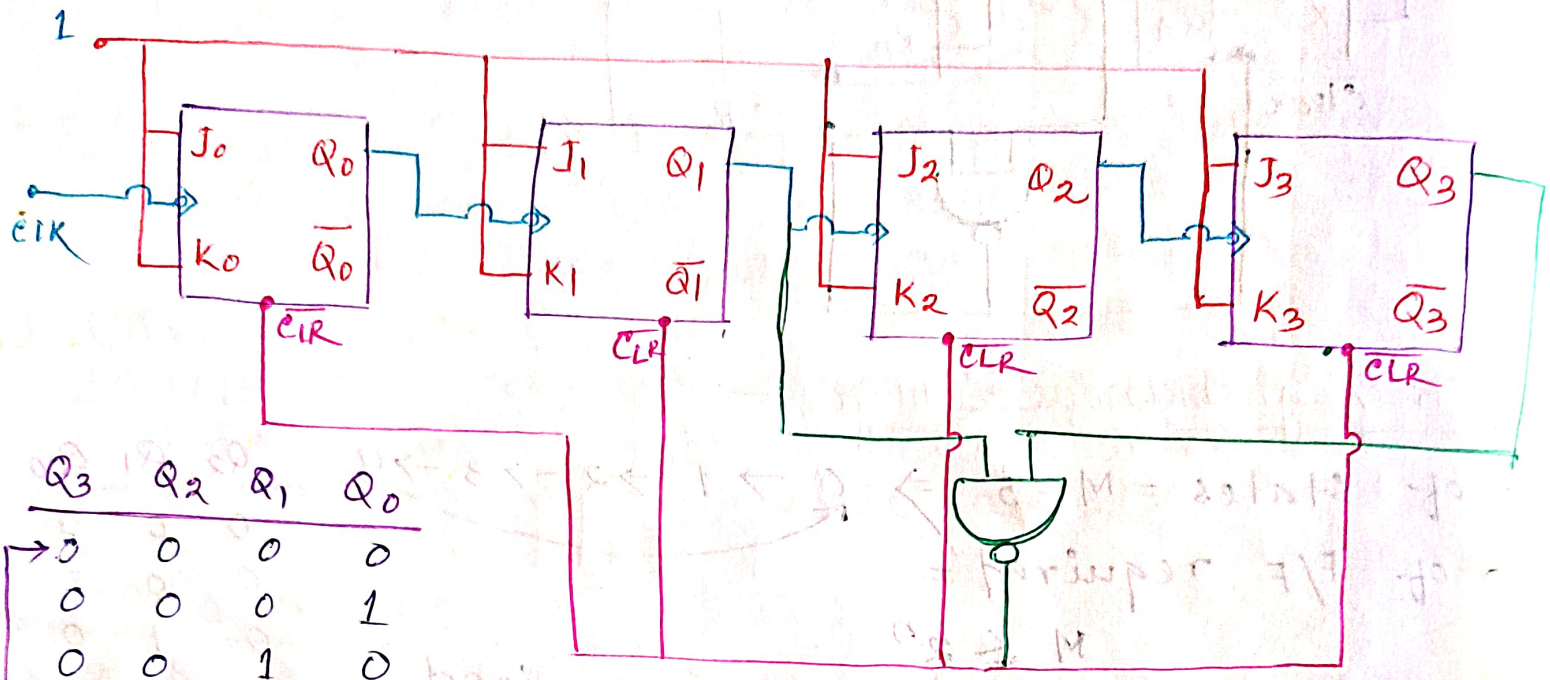
M = 10

$10 \leq 2^{④}$ ← 4-F/F Req.

Using 4 F/F total no of states possible = 16

no of used states in decade counter = 10

Unused states are = 16 − 10 = 6



Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0

Synchronous Counter Design:

Steps to design synchronous counter:

- Determine the no of flipflop required for counting the given state.
- Draw the state diagram according to the counting state.
- write down the excitation table which contains the present state, next state & corresponding flipflop input.
- Use K-map which contain present value Vs F/F input

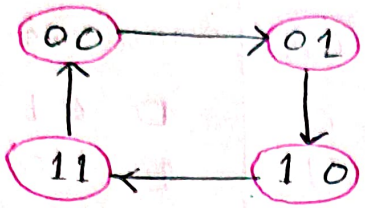
- Draw the logic circuit according to the K-Map expression.

2-bit Synchronous counter:

number of bit of F/F = number of F/F.

(i) Here 2 F/F required for design the counter.

(ii)



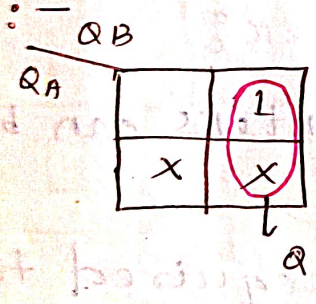
(iii)

Present state		Next state		F/F Input	
Q_A	Q_B	Q_{An+1}	Q_{Bn+1}	D_A	D_B
0	0	0	1	0	1
0	1	1	0	1	0
1	0	1	1	1	1
1	1	0	0	0	0

or.

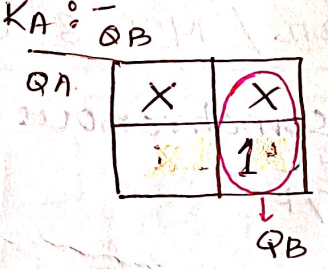
Present state		Next state		F/F Input			
Q_A	Q_B	Q_{An+1}	Q_{Bn+1}	J_A	K_A	J_B	K_B
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	1	1	X	0	1	X
1	1	0	0	X	1	X	1

J_A :-



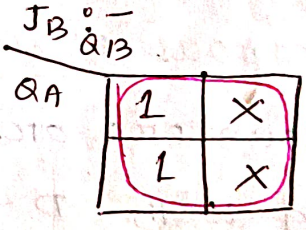
$J_A = Q_B$

K_A :-



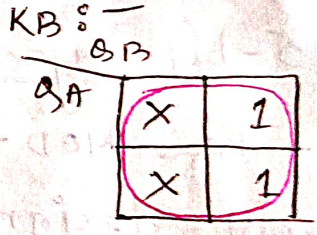
$K_A = Q_B$

J_B :-

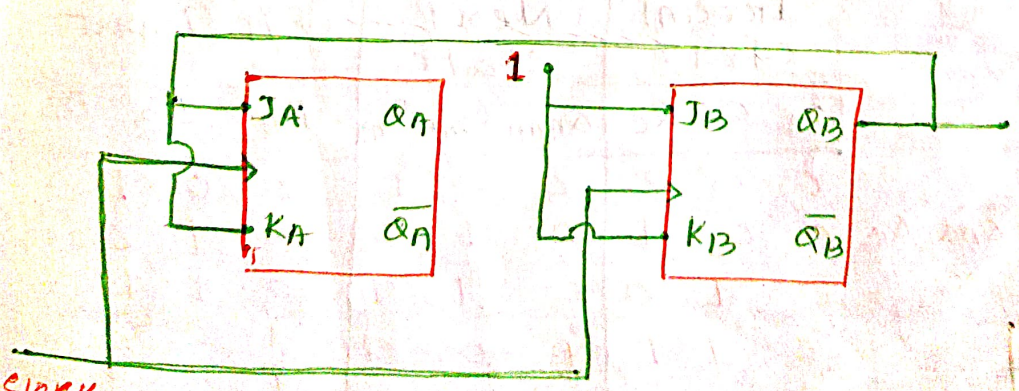


$J_B = 1$

K_B :-



$K_B = 1$



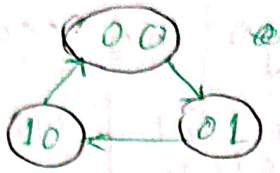
MOD-3 Synchronous Counter:

3 → states $n = 2$ = F/F required.

$3 < 2^n$

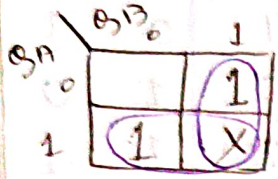
$3 < 4$

0 → 1 → 2

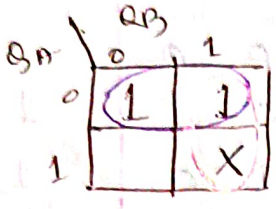


Present state		Next state		F/F Input	
Q_A	Q_B	Q_{An+1}	Q_{Bn+1}	T_A	T_B
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	1	0
1	1	X	X	X	X

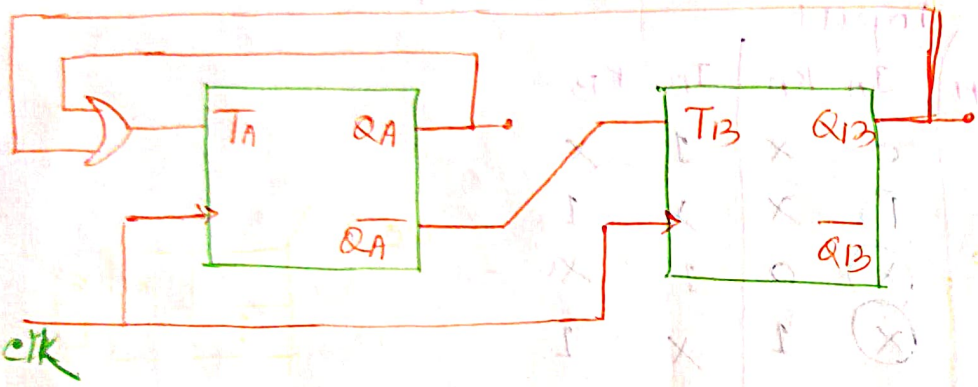
T_A :-



T_B :-



$T_A = Q_A + Q_B$ $T_B = \overline{Q_A}$

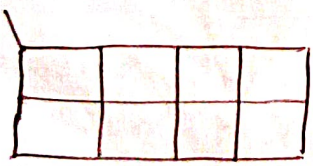


3-bit Synchronous counter / MOD-8 counter:

- MOD - counters or synchronous counters can be best designed with D - F/F.
- 3-bit counter means three F/F are required to design the counter.

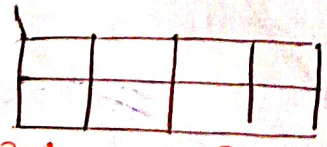
Present state			Next state			F/F Input		
Q_A	Q_B	Q_C	Q_{An+1}	Q_{Bn+1}	Q_{Cn+1}	D_A	D_B	D_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

D_A :-



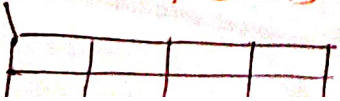
$D_A = Q_A\overline{Q_B} + Q_A\overline{Q_C} + \overline{Q_A}Q_BQ_C$

D_B :-



D_B :- $Q_A \oplus Q_B$

D_C :-



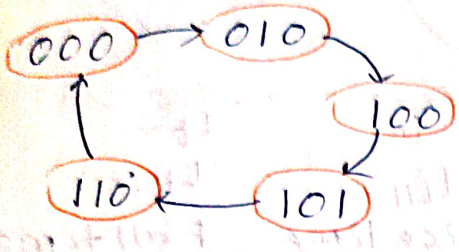
$D_C = \overline{Q_A}$

Skip Counter

Q. Design a skip counter using D F/F or J-K F/F for the given state.

$0 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 0$

Maximum no in the counting sequence = 6. So three F/F are required to design this skip counter.



Present state			Next state			F/F Input		
Q_A	Q_B	Q_C	Q_{An+1}	Q_{Bn+1}	Q_{cn+1}	D_A	D_B	D_C
0	0	0	0	1	0	0	1	0
0	0	1	X	X	X	X	X	X
0	1	0	1	0	0	1	0	0
0	1	1	X	X	X	X	X	X
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	0	0	0	0	0	0
1	1	1	X	X	X	X	X	X

D_A :-

Q_A \ Q_BQ_C	00	01	11	10
0		X	X	1
1	1	1	X	

$D_A = Q_A \overline{Q_B} + \overline{Q_A} Q_B$

$= Q_A \oplus Q_B$

D_B :-

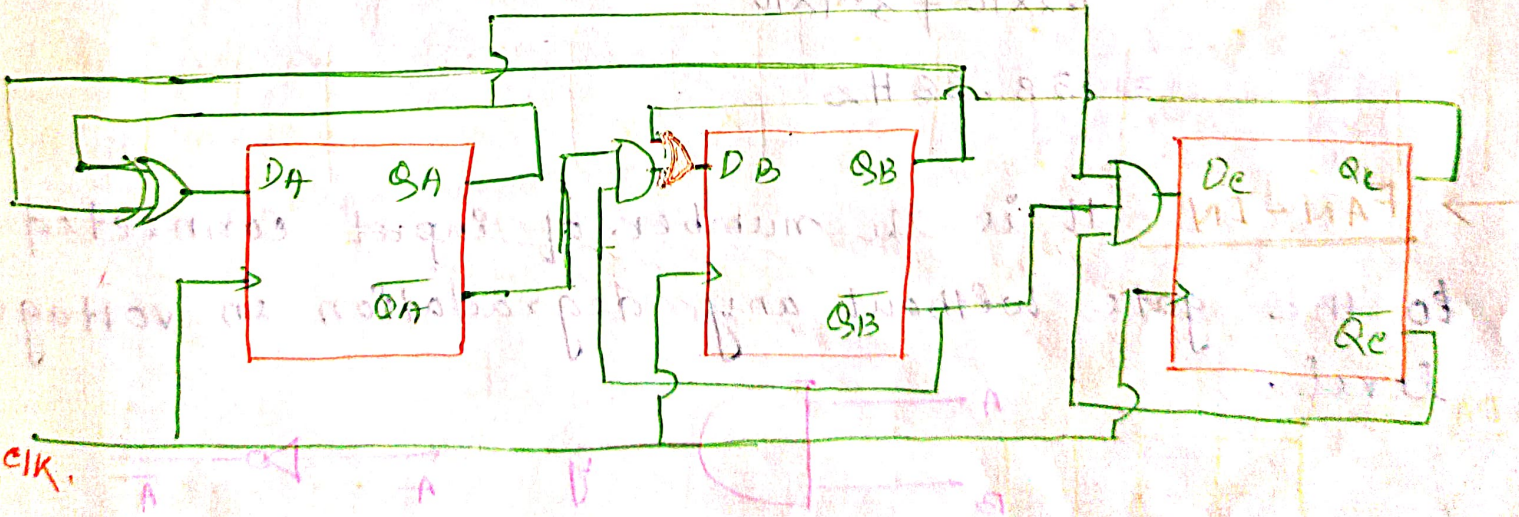
Q_A \ Q_BQ_C	00	01	11	10
0	1	X	X	
1		1	X	

$D_B :- \overline{Q_A} \overline{Q_B} + Q_C$

D_C :

0		X	X	
1	1		1	

$D_C = Q_A \overline{Q_B} \overline{Q_C}$



CMOS (complementary MOSFET)

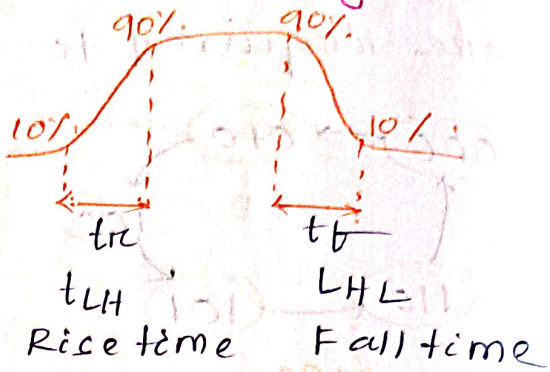
→ Logic Variable ⟶ +ve logic → 1 → High, 0 → Low
-ve logic → 1 → Low, 0 → High

→ Switching time:

$$t_{min} = t_{LH} + t_{HL}$$

t_{min} = minimum switching time

where t_{LH} is time taken to grow from low to high & t_{HL} to grow from high to low



$$f_{max} = \frac{1}{t_{min}}$$

$$f_{max} = \frac{1}{t_{LH} + t_{HL}}$$

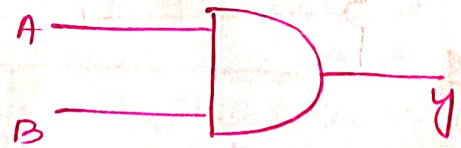
f_{max} = max^m switching frequency.

Q. Find out the max^m switching frequency for any logic circuit having rise time 7.2 msec & falling time 3.9 nsec. Find out f_{max}.

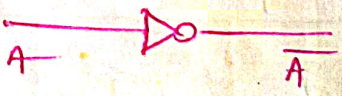
$$f_{max} = \frac{1}{7.2\times10^{-3} + 3.9\times10^{-9}}$$

$$= 138.88\ Hz$$

→ **FAN-IN** It is the number of input connected to the gate without any degradation in voltage level.

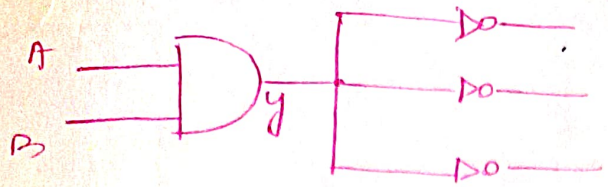


FAN IN = 2

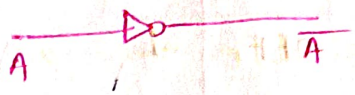


FAN IN = 1

→ **FAN OUT**: It is defined as the maximum number of logic gates that a gate can drive without any degradation of the voltage level.



Here FAN OUT = 3



FAN OUT = 0

Total delay (t_{PN}) = $t_{P_0} + N.t_{P_L}$

t_{PN} → time delay of entire ckt

t_{P_0} → duration of starting gate

t_{P_L} → duration of loaded gate.

5. For the above circuit if the propagation delay of and gate is 2 nanosec & propagation delay of not gate is 1nsec then total delay (t_{PN}) = ?

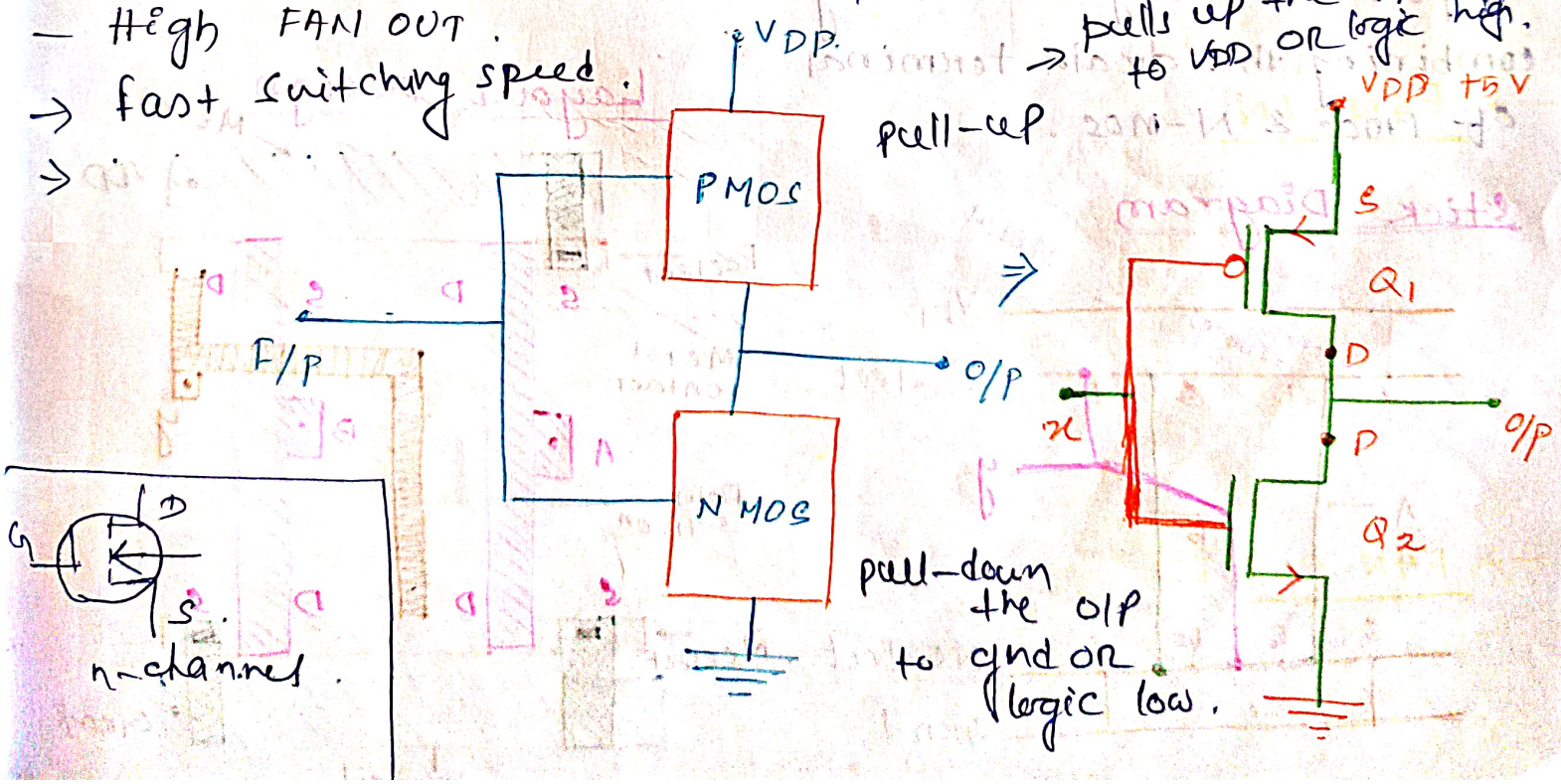
→ $2 + 3\times1 = 5$ nsec = t_{PN}.

Logic gate is a fundamental element of digital ckt that perform specific logical operation. CMOS technology is used for manufacturing IC. It is used to manufacture ~~desig~~ LED, sensors and logic gat

CMOS LAYOUT DESIGN :

Advantages of CMOS

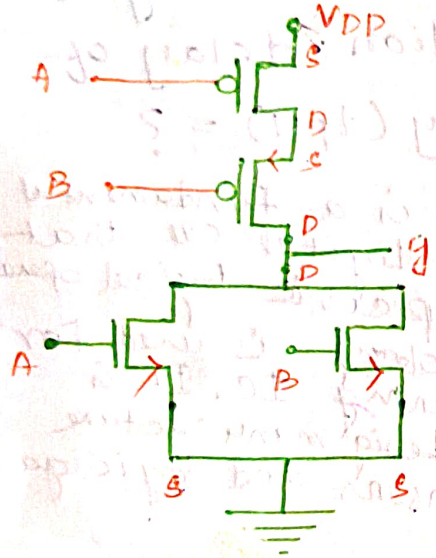
- Lower propagation delay.
- Low power dissipation.
- High FAN OUT.
- → fast switching speed.
- →



I/P x	Q1	Q2	O/P y
0	ON	OFF	1
1	OFF	ON	0

Operation	NMOS	PMOS
OR (+)	Parallel	Series
AND (·)	Series	Parallel

NOR Gate

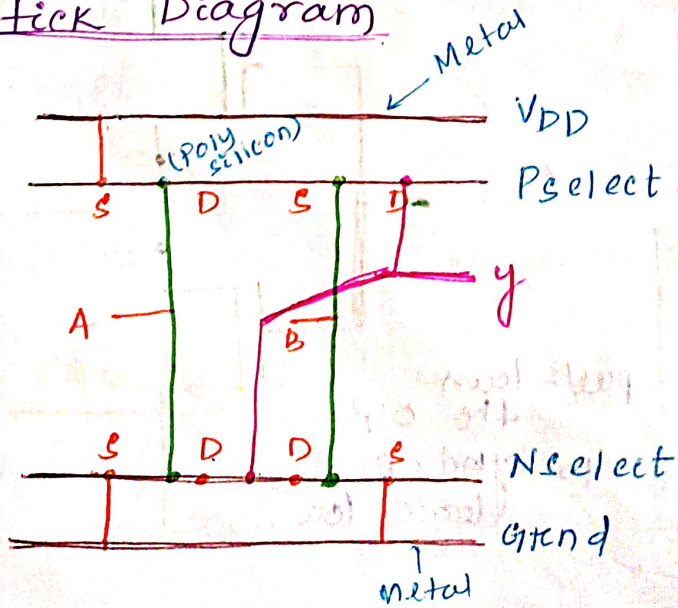


Truth table

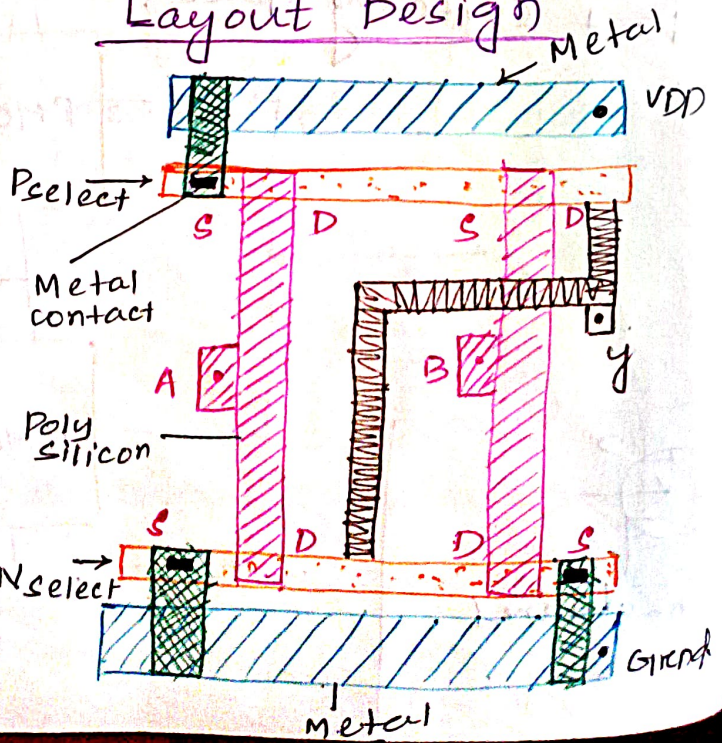
A	B	y
0	0	1
0	1	0
1	0	0
1	1	0

Output is always taken by combining the drain terminal of PMOS & N-MOS.

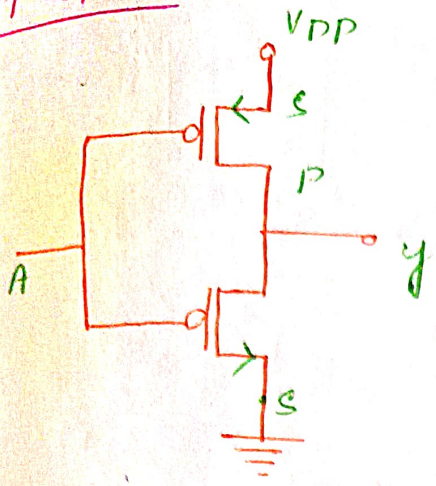
Stick Diagram



Layout Design



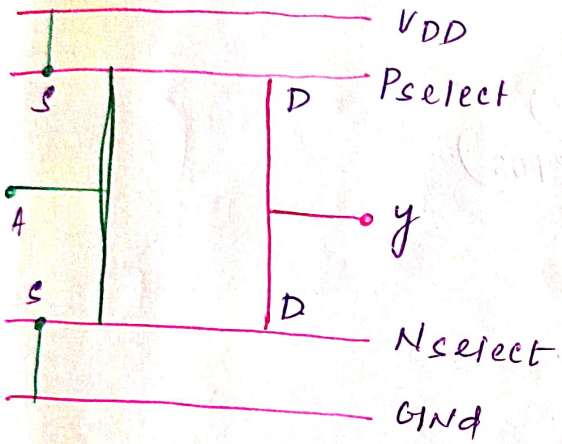
NOT GATE :



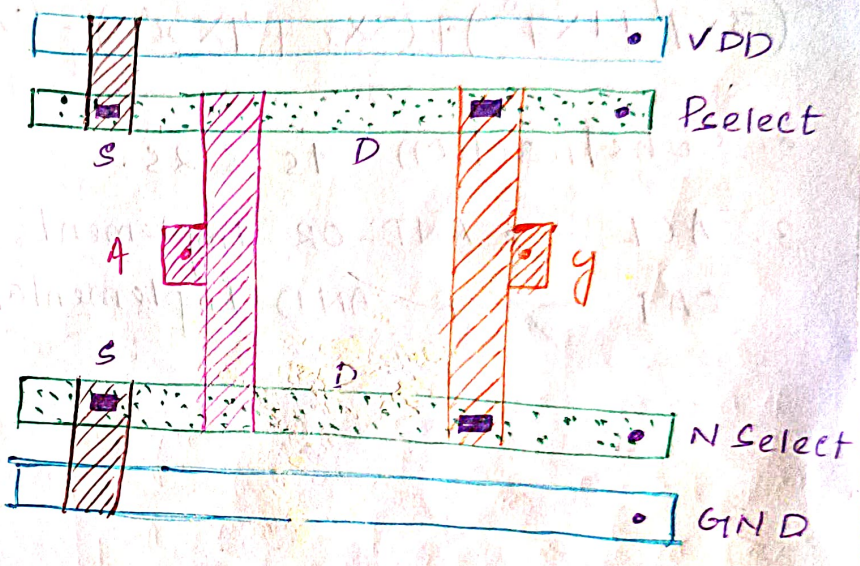
Truth table

I/P	O/P
A	y
0	1
1	0

Stick diagram



Layout Design



For P MOS

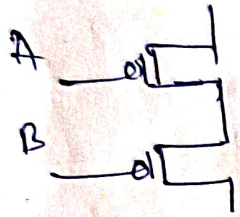
If there is AND gate (·) symbol then the connection type is parallel.

OR gate (+) → series.

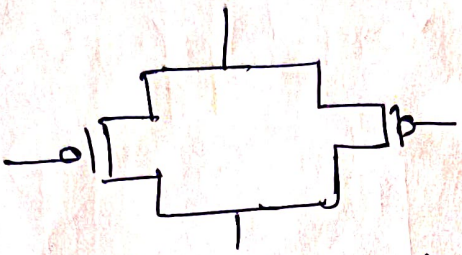
For N MOS

AND gate → series

OR gate → parallel

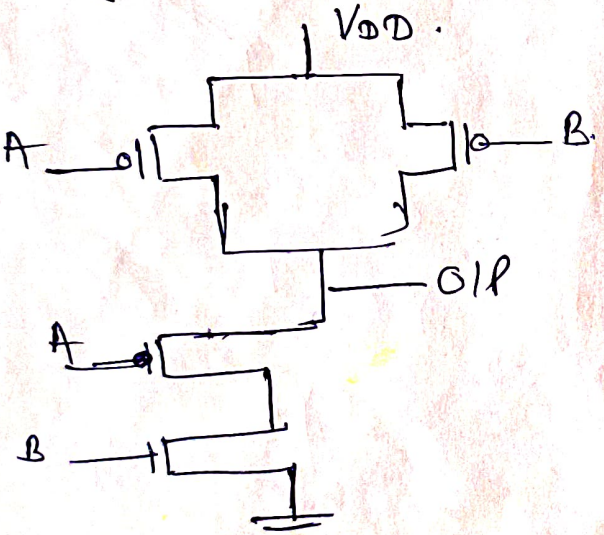


(series)

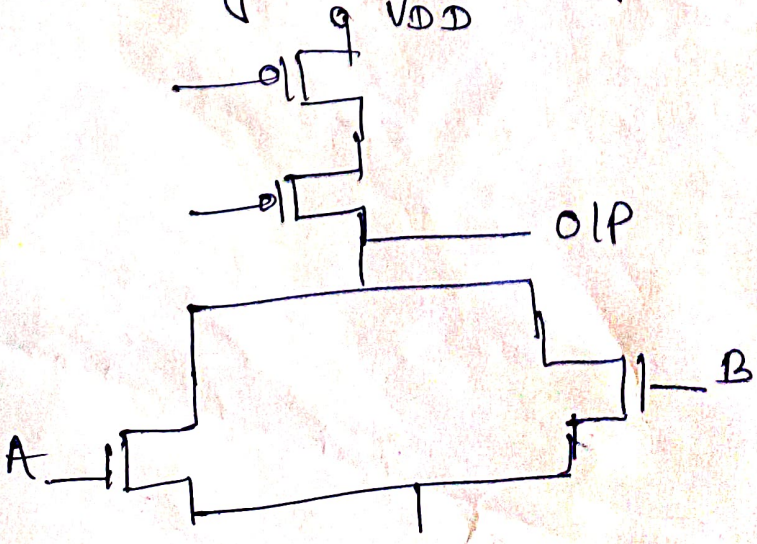


(parallel)

* Design NAND using CMOS. $\overline{A \cdot B}$

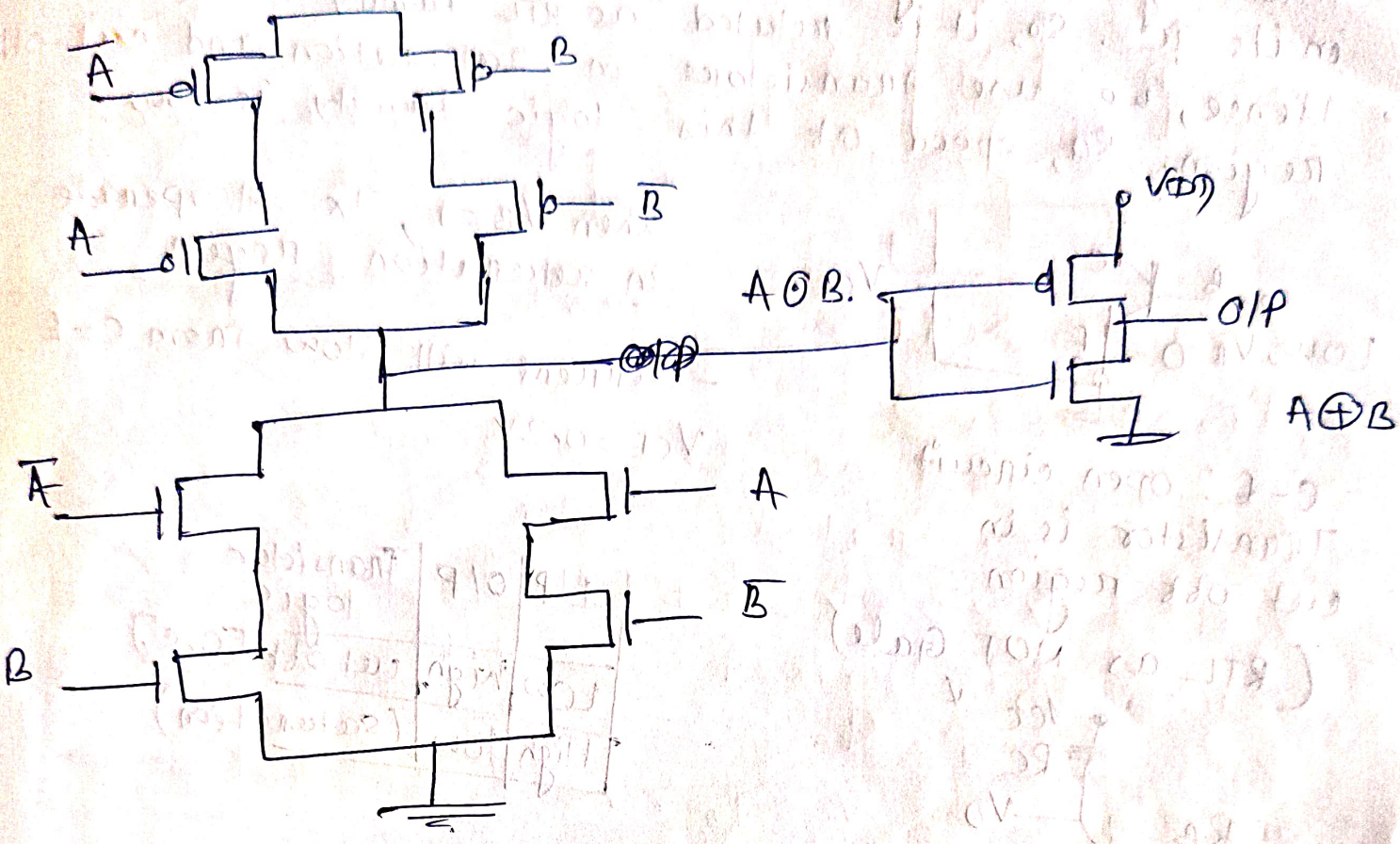


* Design NOR using CMOS. $\overline{A+B}$



Design XOR using CMOS

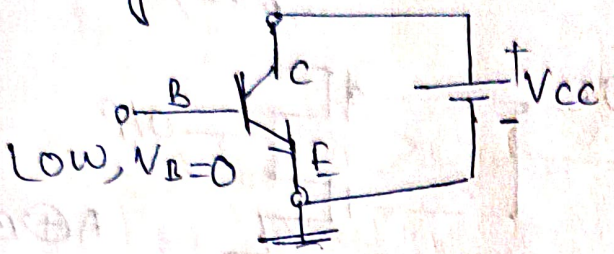
$A\bar{B} + \bar{A}B$



Implement of X-NOR using CMOS.

RTL

- This logic family includes resistors and transistor in its IC. So, it is related as RTL family.
- Hence, we use transistors in saturation and cut-off region. So, speed of this logic family is low.



when $V_B = 1$, Tx will operate in saturation region.

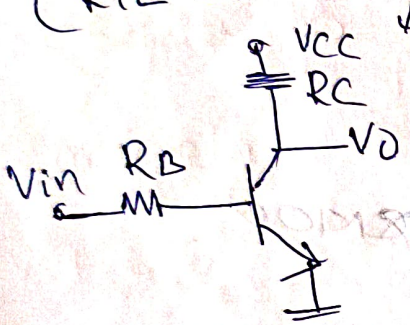
- current will flow from C-E.

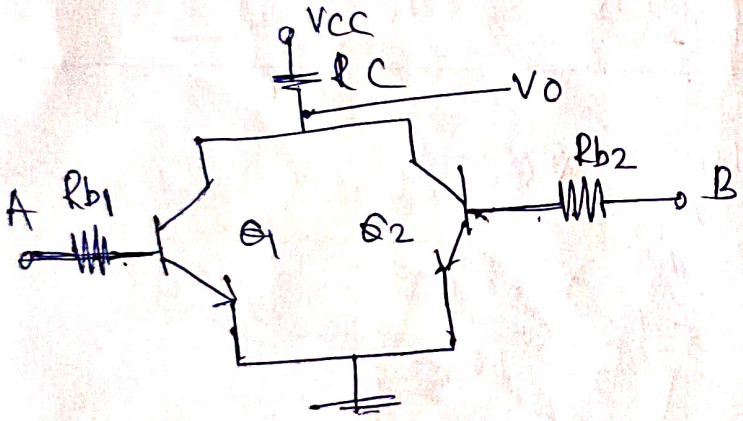
$V_{CE} = 0.2V$

C-E: open circuit
Transistor is in cut-off region.

I/P	O/P	Transistor logic
Low	High	cut off (0.7)
High	low	(saturation)

(RTL as NOT Gate)



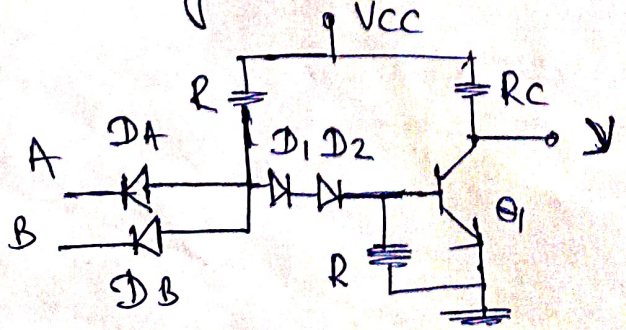


A	B	VO
L	L	H
L	H	L
H	L	L
H	H	L

NOR

DTL (NAND)

RTL Family has low NM, low fan out, slow speed and high power dissipation. So, we don't use RTL



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

when $D_A, B \rightarrow OD$

$D_A, D_B \rightarrow FB$ (potential difference is there)

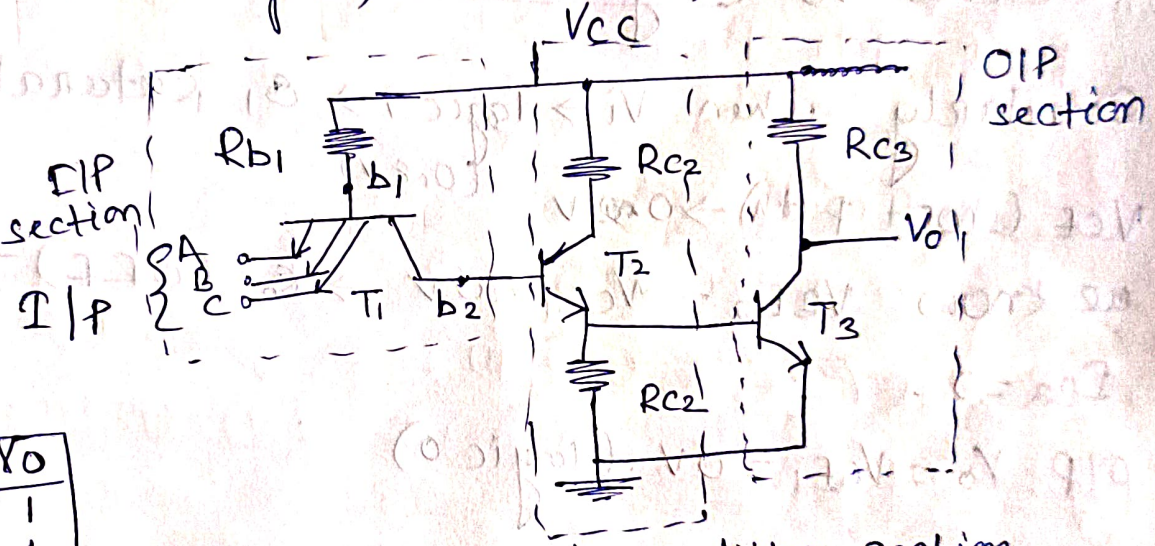
$D_1, D_2 \rightarrow RB$, $Y = V_{cc}$ (1)

Base current minimum

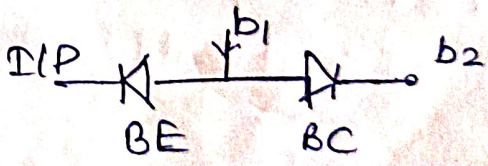
Q_1 = OFF

- when $A=0$, $B=1$, $D_A \rightarrow FB$, $D_B \rightarrow RB$, $D_1, D_2 \rightarrow RB$, $Q_1 \rightarrow$ cut-off, $Y = V_{cc}$ (1)
- when $A=1$, $B=0$, $D_A \rightarrow RB$, $D_B \rightarrow FB$, $D_1, D_2 \rightarrow RB$, $Q_1 \rightarrow$ cut off, $Y = V_{cc}$ (1)
- when $A=B=1$, $D_A, D_B \rightarrow RB$, $D_1, D_2 \rightarrow FB \rightarrow Q_1 \rightarrow ON$. saturation region, $V_{CE} = 0.2V$, OIP low

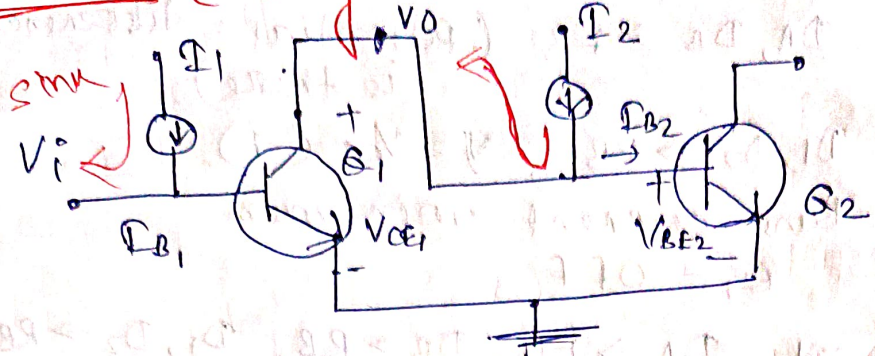
TTL



A	B	C	Yo
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



I^2L (Integrated Injection Logic)



$V_{CE_1} = V_{BE_2}$

$V_i \to 0 \to Q_1 \to$ OFF $\to$ CE terminal open $\to I_{e1} = 0 \to$

$I_{E_1} \to 0 \to I_{B_1} = 0$,

At the same time I_{B_2} is given to Q_2 transistor. Q_2 is in saturation (on) state so, $V_{CE} = 0.2V$ and $V_{BE} > 0.7V \simeq 0.8V$ and that will appear across V_o (logic 1 when I/P logic 0)

Similarly when $V_i \to$ logic 1 $\to Q_1$ saturation (on) $\to$ (0.8V)

V_{CE} (short path) $\to 0.2V$

we know $V_{BE_2} = V_{CE_1} = 0V \to Q_2$ (OFF) $\to I_{C2} = 0, I_{E2} = 0$

$I_{B2} = 0$.

O/P $V_o = V_{CE_1} = 0V$. (logic 0)

Hamming Code

Qn: 0110 (even parity)

P_1	P_2	M_3	P_4	M_5	M_6	M_7	P_8
		0		1	1	0	

$P_1 = XOR(3, 5, 7, 9) = 0 \oplus 1 \oplus 0 = 1$

$P_2 = XOR(3, 6, 7, 10) = 0 \oplus 1 \oplus 0 = 1$

$P_4 = XOR(5, 6, 7, 12) = 1 \oplus 1 \oplus 0 = 0$

P_1	P_2	M_3	P_4	M_5	M_6	M_7	P_8
1	1	0	0	1	1	0	1

Qn:- Rx = 101101010 (odd)

P_1	P_2	M_3	P_4	M_5	M_6	M_7	P_8	M_9	M_{10}
1	0	1	1	0	1	0	1	0	

$C_1 = XOR(1, 3, 5, 7, 9)$

$= 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$

$C_2 = XOR(2, 3, 6, 7, 10) = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 0$

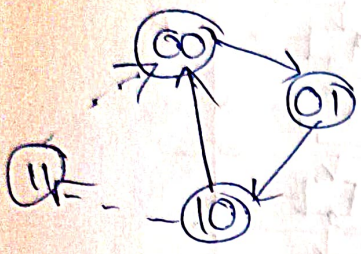
$C_4 = XOR(4, 5, 6, 7, 12) = 1 \oplus 0 \oplus 1 \oplus 0 = 1$

$\Rightarrow 1011 \rightarrow$ 13 (error)

Actual → 101101110

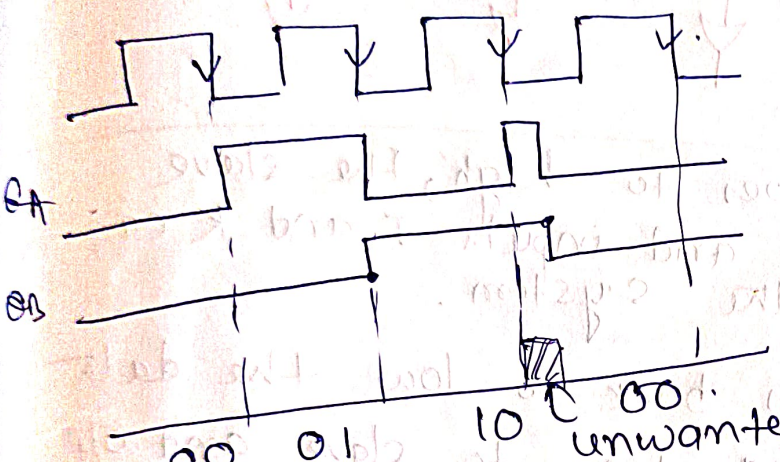
Glitch

It is a short duration pulse or spike that appears in the o/p of a ripple counter with number < 2n.



Q_B	Q_A
0 0	→ 0 1
0 1	→ 1 0
1 0	→ 0 0
1 1	→ x x

Q_C	Q_B	Q_A
0	0	0 → 0 0 1
0	0	1 → 0 1 0
0	1	0 → 0 1 1
0	1	1 → 1 0 0
1	0	0 → 1 0 1
1	0	1 → 0 0 0
1	1	0 → x x x
1	1	1 → x x x.



Fanout of TTL

$I_{OH} = 400\mu A$, $I_{IH} = 40\mu A$, $I_{OL} = 16mA$, $I_{IL} = 1.6mA$

$$\text{fanout high} = \frac{400}{10} \quad \frac{I_{OH}}{I_{IH}} = \frac{400}{40} = 10$$

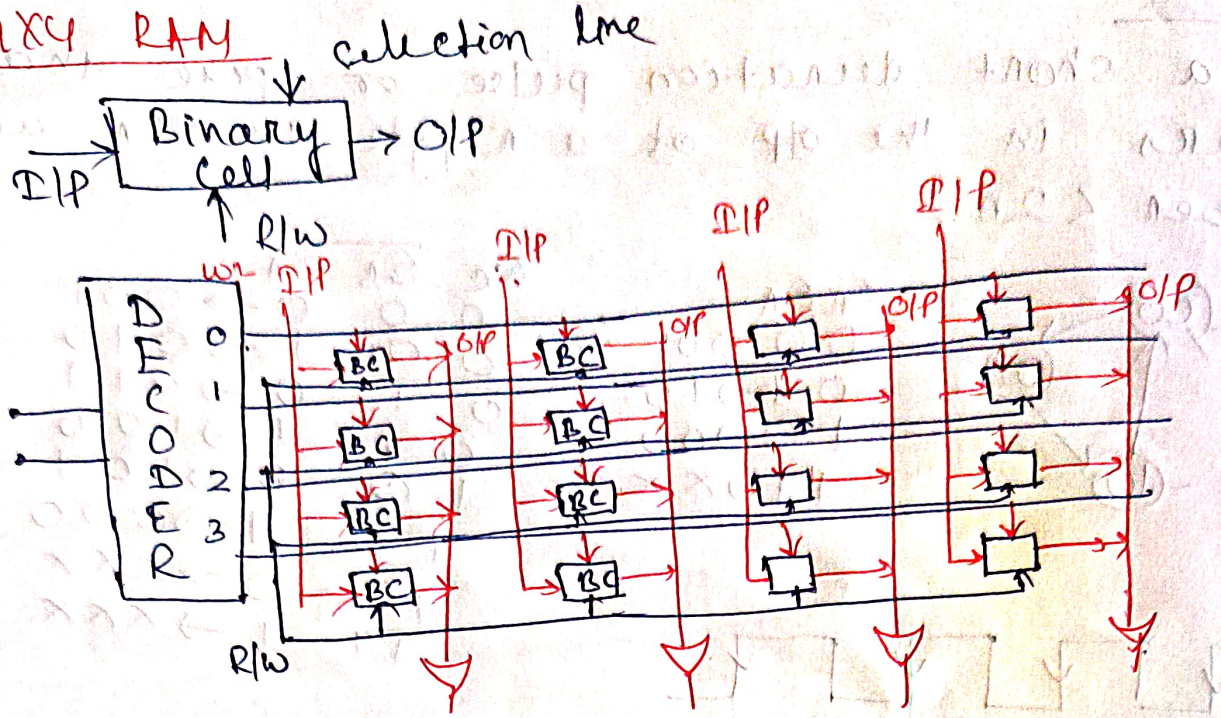
$$\text{fanout low} = \frac{I_{OL}}{I_{IL}} = \frac{16}{1.6} = 10$$

Effective $= f_{min}[10, 10] = 10$.

Floating Point Representation

S	5-Bit Exponent	8-bit Mantissa

4X4 RAM



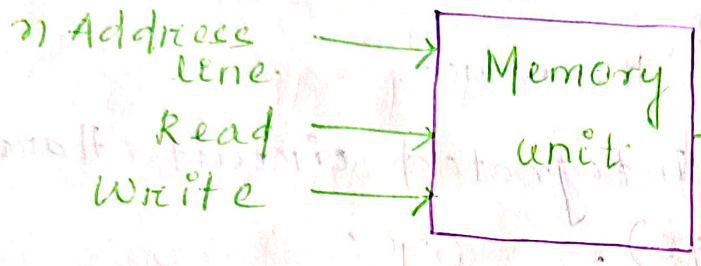
- When the clk pulse goes to high, the slave flip flop becomes inactive and inputs J and K can control the state of the system.
- when the clk pulse goes back to low the data is transferred from Master to slave and o/p is obtained.
- when J=K=0, both JK flip flops remains inactive hence the O/P & remain unchang and this state is Hold state of MS Flip flop.
- when J=0, K=1, then Q=0, $\bar{Q}$=1 of Master when the clk is given slave flip flop it forces the slave to Reset. therefore O/P of Master same with slave, Reset state
- when J=K, K=0, then Q=1, $\bar{Q}$=0, Then the negative transition of the clock signal sets the slave. and it is called set state.

- when J=K=1, the Master flip flop toggles on the the positive transition of the clock pulse and the slave flip flop toggles on the negative transition of the clk pulse. Hence, the problem of Race around condition can be avoided.

Memory

Memory is an element which stores data. The data used in a program as well as the instruction for executing the program are stored in the memory.

Basic memory element :

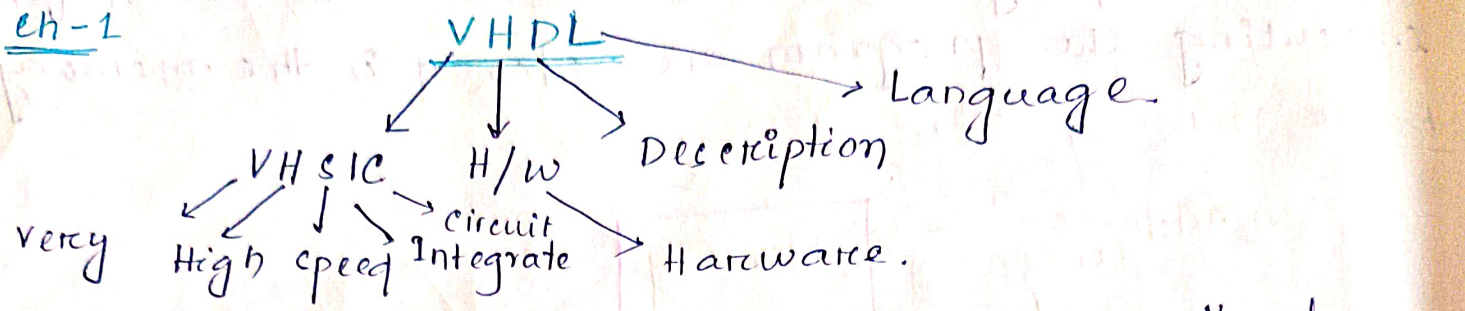


It has 'n' address line to access 2^n location in memory.

MODULE - II

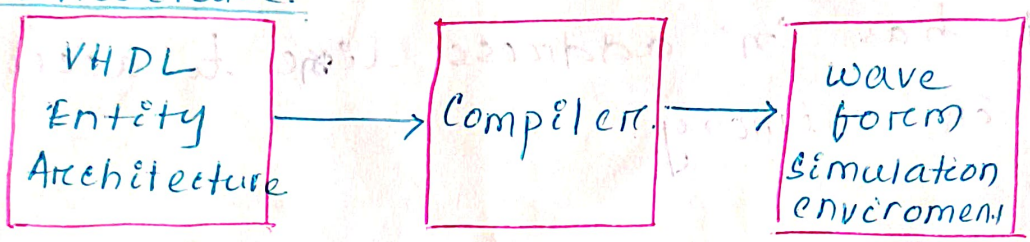
VHDL & Digital circuit design:

Ch - 1



VHDL (Very high speed Integrated circuit Hardware Description Language).

Program structure.



VHDL - Library

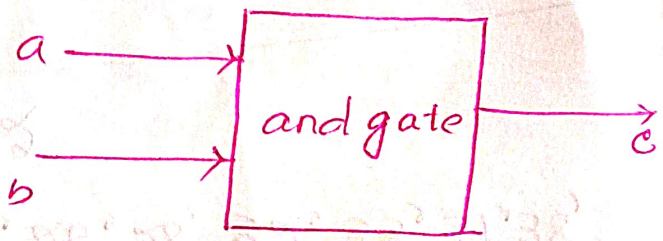
```
Library IEEE;
IEEE.STD_LOGIC_1164.all;
```

Syntax of entity declaration

```
entity entity-name is
    port (port_name: mode_type;
          port_name: mode_type);
end entity-name;
```

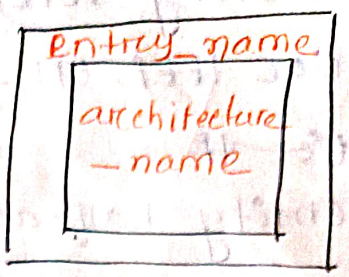
Example

```
entity andgate is
port (a: in bit;
      b: in bit;
      c: out bit);
end andgate;
```



Architecture syntax:

architecture <architure_name>
of <entity name> is
<declaration>
begin
<VHDL statements>
end <architecture_name>;



* Entity name & architecture name always be different.
* Architecture <= Dataflow, Structural, Behavioural.

• **Data flow model:** (Half Adder)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity full adder is
Port (A,B, cin : in STD_LOGIC;
      S, cout : out STD_LOGIC);
end full_adder;
architecture df of full_adder is
begin
Sum <= A xor B xor cin;
cout <= (A and B) or (B and cin) or (A and cin);
end df;
```

Half Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IE
entity half_adder is
  Port (A,B: in std_logic;
        sum, c: out std_logic);
end half_adder;
architecture df of half_add is
begin
  SUM<= (A xor B;
  cout<= A and B;
end df;
```

Design full adder using half adder using structural model.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity full_adder is
  Port (A, B, cin: in STD_LOGIC;
        s, cout: out STD_LOGIC);
end full_adder;

architecture fastr of full_adder is
component half_adder is
port (A, B: in std_logic;
      sum, carry: out std_logic);
end component;
component or_gate
  port (A,B: in std_logic;
```

```
        Y : out std_logic);
end component;
signal P, Q, R : std_logic;
begin
U0 : half_adder port map (A, B, P, Q);
U1 : half_adder port map (Q, Cin, R, S);
U2 : or_gate port map (P, R, Cout);
end fact;
```

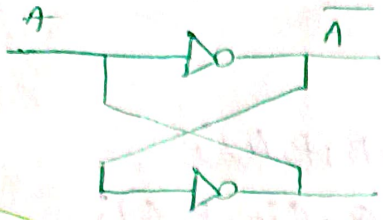
Full Adder truth table.

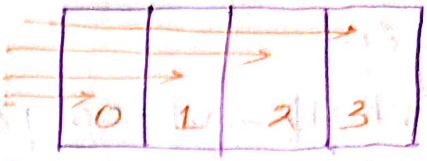
A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Semiconductor Memory

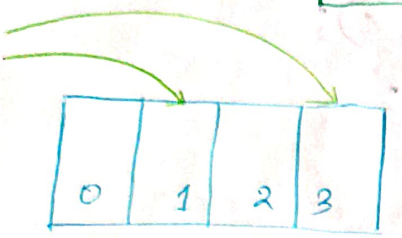
RAM

ROM





Sequential Data Access



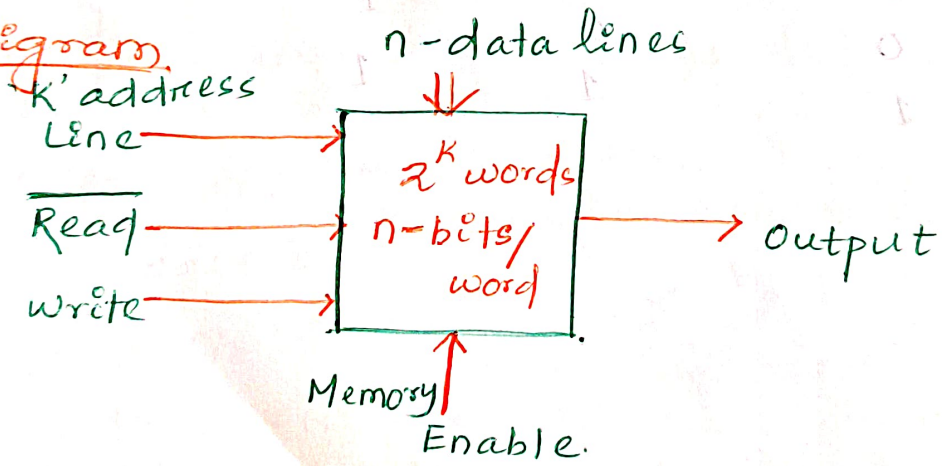
Random Access Memory

- Static (S-RAM)
- Dynamic (D-RAM)

- It is a volatile memory in which both read & write operation can be perform

Volatile memory means data is lost when power is stand off.

Block digram



- The n-data lines transfer data to or from one complete word at a time.
- The binary address is given by 'K'-address lines. An internal decoder having $K \times 2^K$ is used to generate the address of memory space and selects one of the 2^K address value/words.
- Read & write control lines are present to enable the operations.

S-RAM

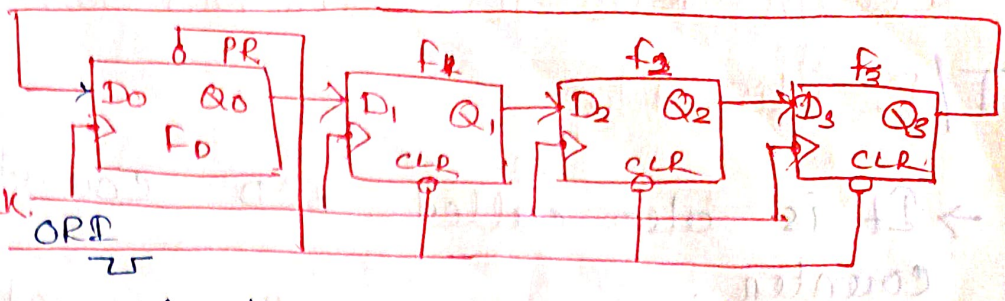
- Data is stored in F/F.
- Periodic recharge is not present.
- Implemented using both MOS and BJT technology.
- For storing single bit data six transistors are required.
- Power dissipation is high.

D-RAM

- Data is stored in form of charge in the capacitor.
- Periodic recharging or refreching is required to restart the charging of capacito
- Implemented using only MOS.
- For storing single bit data one MOS transistor is required.
- Power dissipation is low.

Ring Counter

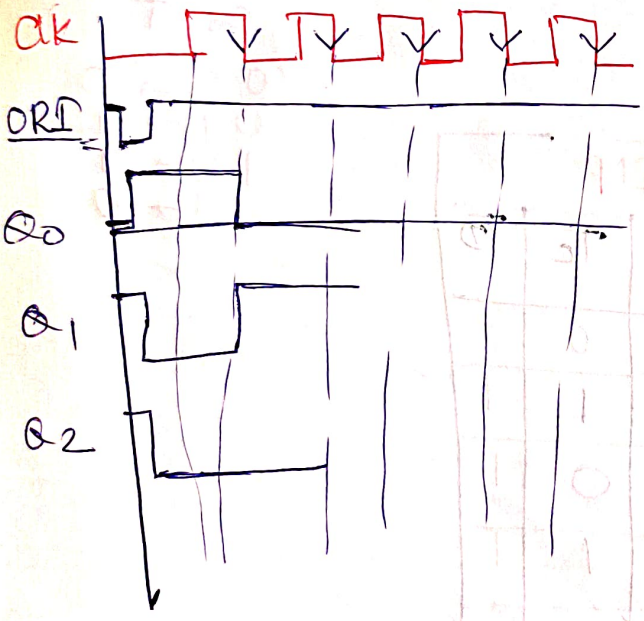
→ It is a typical application of shift register.



2. The only change is the output of last ff is connected to the input of first flip. It is a special type of counter where the no. of states = no. of flip used.
3. In this case 4 FF, so no. of states = 4.

when PR=0, Q=1 (set), CLR=0, Q=0 (Reset)
It does not depend on IIP or clk value. CLR and PR over-ride the ~~input~~ output.

ORI (Over-Ride Input) is a active low signal.



Preseted value

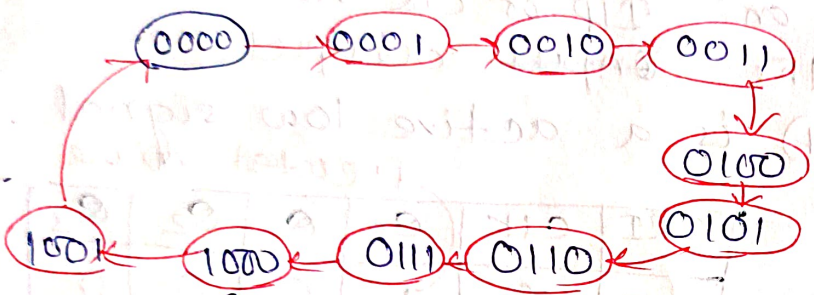
ORI	CLK	Q_0	Q_1	Q_2	Q_3
⌐	×	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	0	0	0	1
1	↓	1	0	0	0

- In this process 1 is circulated around the register as long as clk is given to the circuit.
- for this reason, this counter is called circulating counter or ring counter.

Design MOD-10 Synchronous Counter Using T/D Flip Flop.

→ It is also called BCD counter or Decimal counter.

- No. of states count by counter = 10 (M)
  Range of counting 0 to M-1 = 0 to 9
  Required FF = $M \leq 2^n$ (where n is no. of FF required)
  here $M = 10 \leq 2^n$, $n = 4$



(state diagram)

Unused states are 10, 11, 12, 13, 14, 15

Excitation table of T

0 0 → 0
0 1 → 1
1 0 → 1
1 1 → 0

P.S.				N.S.				FF IP			
Q_A	Q_B	Q_C	Q_D	Q_{An+1}	Q_{Bn+1}	Q_{Cn+1}	Q_{Dn+1}	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	X	X	X	X	X	X	X	X

$T_A = \Sigma m(7, 9) + d(10, 11, 12, 13, 14, 15)$

$T_B = \Sigma m(3, 7) + d(10, 11, 12, 13, 14, 15)$

$T_C = \Sigma m(1, 3, 5, 7) + d(10, 11, 12, 13, 14, 15)$

$T_D = \Sigma m(0, 1, 2, 3, 4, 5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$

$T_A = Q_A Q_D + Q_B Q_C Q_D$ $\quad T_C = \overline{Q_A} Q_D$

$T_B = Q_C Q_D$ $\quad T_D = 1$