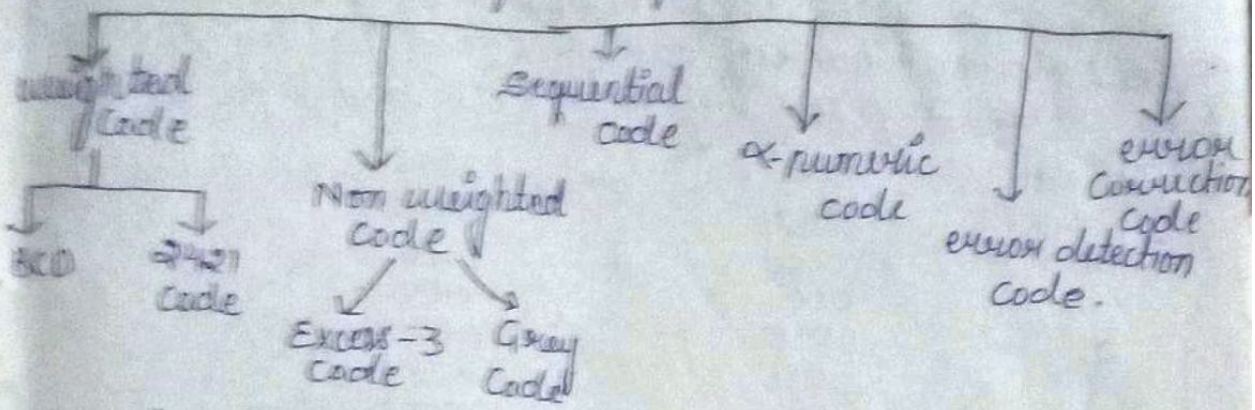
Binary Code:- It is simply "a group of symbols"
It is of six types:-



Advantage of Binary Code:-

1. Binary Codes are suitable for computer application.
2. Binary codes are suitable for Digital Communication
3. Binary Code make the analysis and designing of digital Circuit if we use binary code.
4. Since only 0 and 1 are being used then the implementation becomes easy.

Weighted Code:-

1. It obeys the positional weighted principle.
2. Each position of number represent a specific weight
3. Several Systems of Code are used to represent the decimal digit.
4. In these Codes, each decimal digit is represented by a group of 4 bit.

Ex

2	4	
8+4+2+1	8+4+2+1	
0 0 1 0	0 1 0 0	

5. **Binary Coded Decimal (BCD):-** In this code, each decimal digit is represented by 4 bit binary number.

Ex $(17)_{10} = (0001\ 0111)_2$

$(146)_{10} = (0001\ 0010\ 0110)_2$

$(01001001)_2 = (\underline{0100}\ \underline{1001}) = (49)_{10}$

Compare Binary & BCD:-

10 → 0001 0000 (BCD)

10 → 1010 (Binary)

Binary Coded Decimal is less efficient than binary coz it is using more no. of Bit

⑥ 2421 Code :-

① It is 4 bit Code
② It is a weighted Code. [weight of bit position are 2421]
③ Some decimal number can be coded in two ways
④ It is self complementing code. [LSB least significant bit]

Decimal	Set-1	Set2
	2421	2421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	1100
7	0111	1101
8	1000	1110
9	1001	1111

9 is a complement of 0
8 is a complement of 1
7 " " " " 2
6 " " " " 3
5 " " " " 4

Non Weighted Code :-

In this type of Binary Code, the position weight are not assigned.

⑦ Excess-3 Code :- Excess-3 Code is a non-weight and self complementary code.

Deimal → 8421 $\xrightarrow[3]{\text{add}}$ Excess-3

$$\begin{array}{rl} 5 & \longrightarrow 0101 \\ +3 & \longrightarrow 0011 \\ \hline & 1000 \longrightarrow 8 \end{array}$$

Que find Excess 3 Code of 24

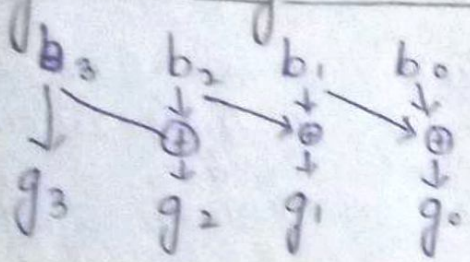
$$\begin{array}{rr} 0010 & 0100 \\ +\ 0011 & 0011 \\ \hline 0101 & 0111 \end{array}$$

∴ Excess-3 Code of 24 is $(01010111)_2$

(5) Grey Code :- This is also known as minimum error code and cyclic permutation code.

* It is unit distant code
* In this two consecutive values changes by only a single bit. Binary code is changed to grey equivalent to lessen the switching operations.

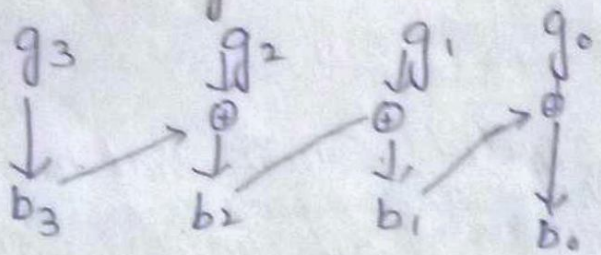
Binary to Gray Conversion :-



A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Ex 100101
110111 ⟶ grey code

Gray to Binary Conversion :-



Ex
1 1 0 1 1 1
1 0 0 1 0 1 ⟶ Binary code.

Logic Gates :-

Basic digital electronic subject that has one or more input and single output.

① Basic Gates
AND
OR
NOT

② Universal Gate
NAND
NOR

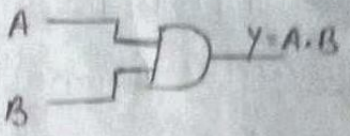
③ Special gate
Ex-OR
Ex-NOR

(a) **AND Gate** :- Its a digital circuit that has two or more input and produce a single output, which is logical and of all closed input. It is optional to represent the logical AND with symbol(.).

Truth table :-

$2^n = 2^2 = 4$
n = input

A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1



(b) **OR Gate** :- It is a digital circuit that has two or more input and produce a single input, which is logical OR of all closed input. It is optional to represent the logical OR with (+).

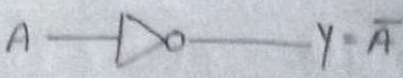
Truth table :-

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

(c) **NOT Gate** :- It is a digital circuit that has single input and produce a single output with the symbol (—).

Truth table :-

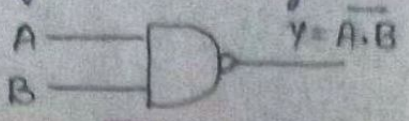
A	Y = $\bar{A}$
0	1
1	0



(d) **NAND Gate** :- It is a digital circuit that has two or more input and produces a output which is inversion of logical AND of all these input.

Truth table :-

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



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

NOR Gate :- It's a digital circuit that has two or more input and produces an output which is inversion of logical OR.

Truth table

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

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

XOR Gate :- It is function is same as OR Gate except some cases when input having even number of 1's.

Truth table

The o/p of Ex-OR is 1 when odd no. of 1's prent at the input also called as odd function.

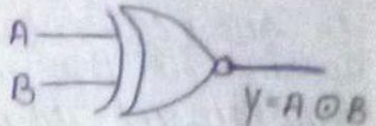
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Ex-NOR Gate :- It's function is same as NOR Gate except some cases when input having odd number of 1's

☞ Output of Ex-NOR is 1 when even number of 1's present at input. Also called as even fnt.

Truth table :-

A	B	$Y = A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1



Boolean theorem of several Variable :-

Commulative Law: $B \cdot C = C \cdot B$, $B + C = C + B$

Associative Law: $(B \cdot C) D = B(C \cdot D)$, $(B+C)+D = B+(C+D)$

Distributive Law: $(B \cdot C) + (B \cdot D) = B(C+D)$, $(B+C)(B+D) = B+(C \cdot D)$

Coverins: $B \cdot (B+C) = B$, $B + (B \cdot C) = B$

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

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

De morgans

$\overline{B_0 B_1 B_2} = \bar{B}_0 + \bar{B}_1 + \bar{B}_2$

$\overline{B_0 + B_1 + B_2} = \bar{B}_0 \cdot \bar{B}_1 \cdot \bar{B}_2$

AND Law	OR Law	EXOR	XNOR
$A \cdot 0 = 0$	$A + 0 = A$	$A \oplus A = 0$	$A \odot A = 1$
$A \cdot 1 = A$	$A + 1 = 1$	$A \oplus \bar{A} = 1$	$A \odot \bar{A} = 0$
$A \cdot A = A$	$A + A = A$	$A \odot 0 = A$	$A \odot 0 = \bar{A}$
$A \cdot \bar{A} = 0$	$A + \bar{A} = 1$	$A \oplus 1 = A$	$A \odot 1 = A$

Standard / Canonical form

- SOP (Sum of product)
- POS (product of sum)

SOP :-

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

Here

$0 \longrightarrow \bar{A}$

$1 \longrightarrow A$

In Canonical / standard SOP from each min terms is having all the variables in normal form and complement form.

Minimal SOP form, each minterms doesnot have all the variable in normal and complement form.

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

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

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

$[Y = B] \rightarrow$ minimal SOP

Que $= Y = \Sigma m(0, 2, 3)$

$Y = m_0 + m_2 + m_3$

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

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

$= \bar{B} + AB$

Product of Sum (POS):-

Here $0 \rightarrow A$
$1 \rightarrow \bar{A}$

Ex

A	B	C	Y	Max term
0	0	0	0	m_0
0	0	1	0	m_1
0	1	0	1	m_2
0	1	1	0	m_3
1	0	0	1	m_4
1	0	1	1	m_5
1	1	0	1	m_6
1	1	1	1	m_7

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

$Y = M_0 * M_1 * M_3$

$Y = \pi(m_0 m_1 m_3)$

$Y = \pi(0,1,3)$

Que In a minimal SOP form no. of minterms in logical Expression $Y(A,B,C) = A+\bar{B}C$

we know $A+\bar{A}=1$

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

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

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

$\therefore$ 5 minterms.

Que $f(a,b,c,d) = \Sigma m(3,7,11,12,13,14,15)$

$f = m_3 + m_7 + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$

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

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

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

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

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

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

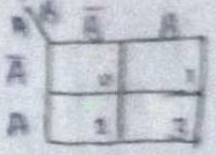
Karnaugh map

* The k-map is graphical representation that provides a systematic method for simplifying the boolean expression.
* for a boolean expression consisting of n variable number of cell required in kmap is equal to 2^n cells.

(a) Two variable k-map :- is drawn for boolean Expression consisting two variable.

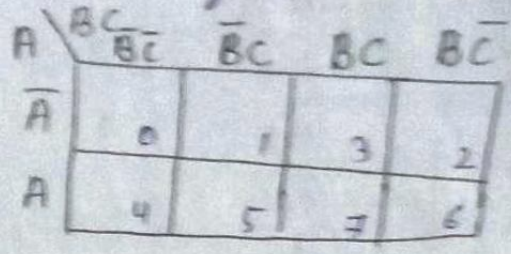
- Number of cells present in 2 variable is $2^2 = 4$ cell $n=2$

eg 2 variable kmap



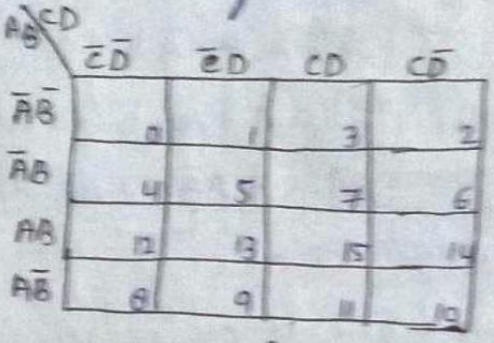
Three variable k-map :-

for three variables No of cells $= 2^3 = 8$ cells



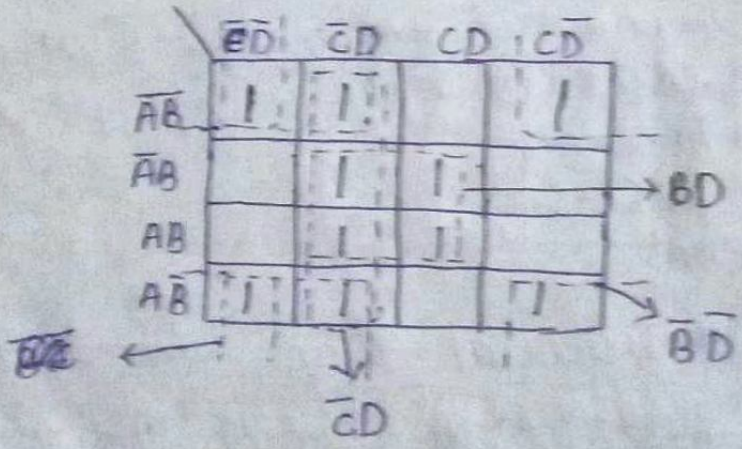
four variable kmap :-

for four variable No of cells $= 2^4 = 16$ cells



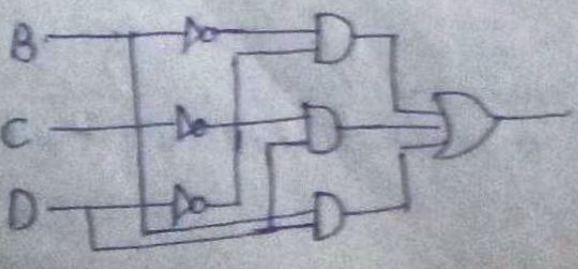
Que Minimize the following boolean function.

$f(A,B,C,D) = \Sigma(0,1,2,5,7,8,9,10,13,15)$

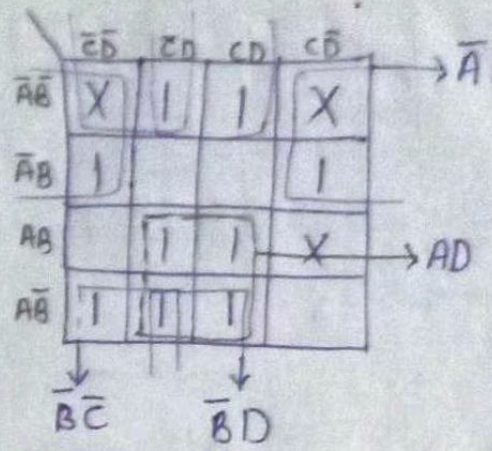


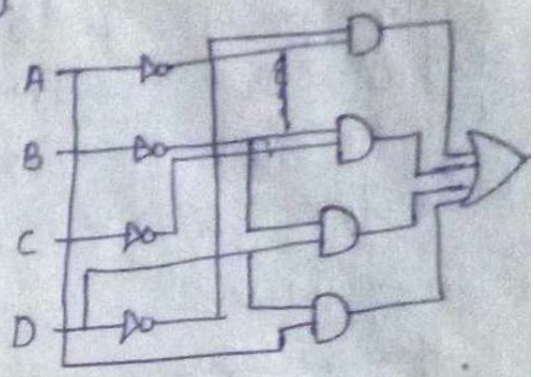
$\therefore f(ABCD) = \bar{B}\bar{D} + \bar{C}D + BD$

Logic Circuit



$f(A,B,C,D) = \Sigma m(1,3,4,6,8,9,11,13,15) + d(0,2,14)$

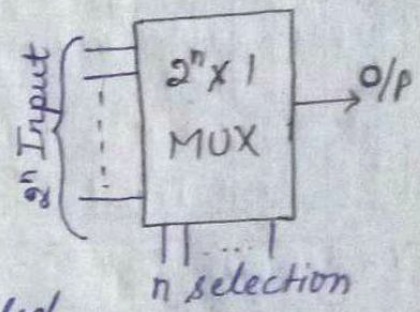




$Y = \bar{A}\bar{D} + \bar{B}\bar{C} + \bar{B}D + AD$

Multiplexer

Multiplexer means many into one.
In digital data, streams are combine into 1 single shared medium by which multiple digital data stream are combined into one signal over all shared medium.
A digital circuit that performs multiplexing of digital signal is called a multiplexer (mux) also called as data sectors.

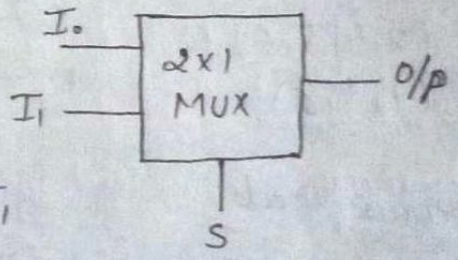


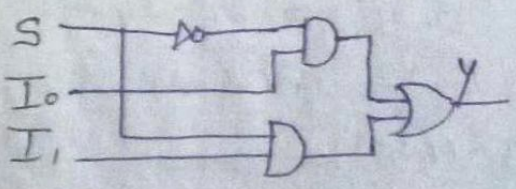
There are four types of multiplexer :-
① 2x1 Mux ② 4x1 mux ③ 8x1 MUX ④ 16x1 MUX

① 2x1 MUX :-
2x1 MUX have 2 input fields and one output field with one selection line.
Truth table :-

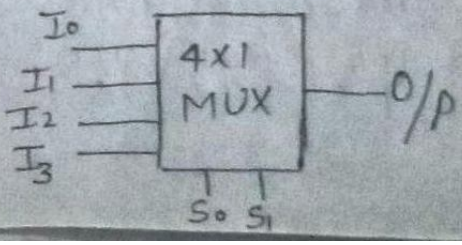
S	Y
0	I_0
1	I_1

$Y = \bar{S}I_0 + SI_1$



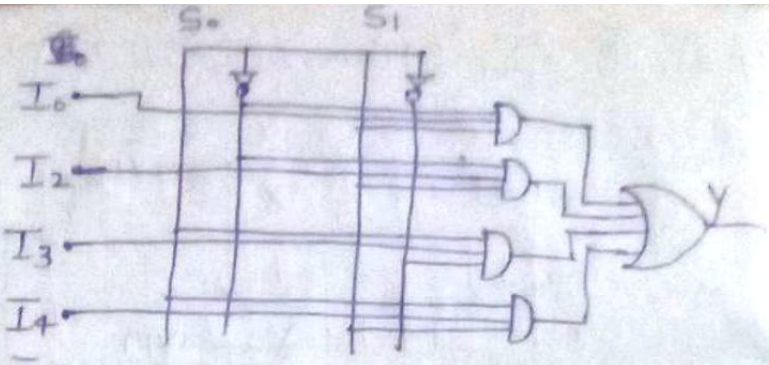


② 4x1 MUX :-
4x1 MUX have 4 input fields and one output field with two selection line.



Truth table:-

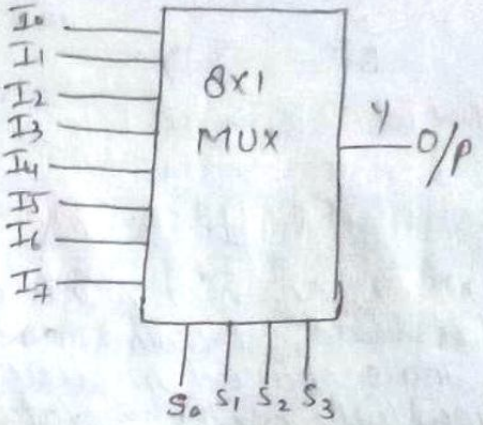
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



$Y = \bar{S}_0\bar{S}_1I_0 + \bar{S}_0S_1I_1 + S_0\bar{S}_1I_2 + S_0S_1I_3$

③ 8×1 MUX:-

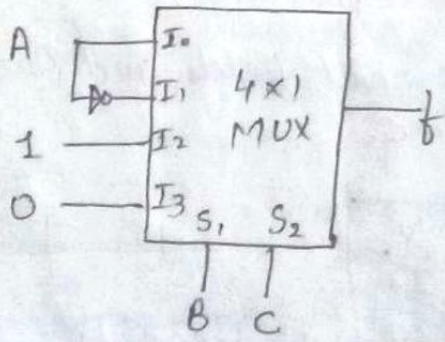
S_0	S_1	S_2	S_3	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7



$Y = \overline{S_0S_1S_2S_3}I_0 + \overline{S_0}S_1S_2S_3I_1 + \bar{S}_0\bar{S}_1S_2\bar{S}_3I_2 + \bar{S}_0\bar{S}_1S_2S_3I_3 + \bar{S}_0S_1\bar{S}_2\bar{S}_3I_4 + \bar{S}_0S_1\bar{S}_2S_3I_5 + \bar{S}_0S_1S_2\bar{S}_3I_6 + \bar{S}_0S_1S_2S_3I_7$

Que:- A 4×1 MUX is used to implement A 3 input boolean function as shown in figure the f(ABC) is implemented as.

① $\Sigma(1,2,4,6)$
② $\Sigma(1,2,6)$
③ $\Sigma(2,4,5,6)$
④ $\Sigma(1,5,6)$



We know that

$Y = \bar{S}_1\bar{S}_2I_0 + \bar{S}_1S_2I_1 + S_1\bar{S}_2I_3 + S_1S_2I_4$

$= \bar{B}\bar{C}A + \bar{B}C\bar{A} + B\bar{C}\cdot 1 + BC\cdot 0$

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

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

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

$= 100 \quad 001 \quad 110 \quad 010$

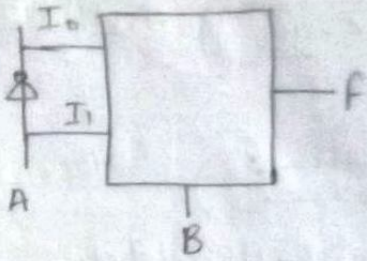
$4 \quad 1 \quad 6 \quad 2$

Que:- Consider the following circuit which use 2×1 MUX shown in figure the boolean expression of function in term of A & B.

$Y = \bar{S}I_0 + SI_1$

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

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

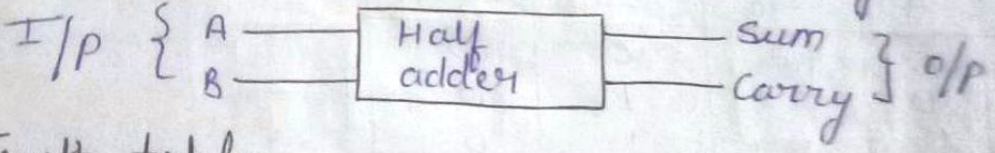


Adder:-

A combinational logic circuit that performs the addition of bits is known as adder.
there are two forms of adder.

* Half adder
* full adder

① **Half adder:-** A combinational logic circuit that performs the addition of two single bits is called half adder.
Half adder contains two inputs and produce Two output. Inputs are called Augend and Added bit and outputs are called Sum and Carry.

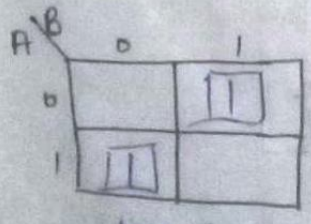


Truth table

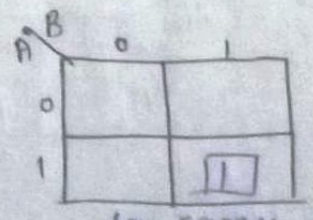
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-map simplification:-

Logic dia



for sum



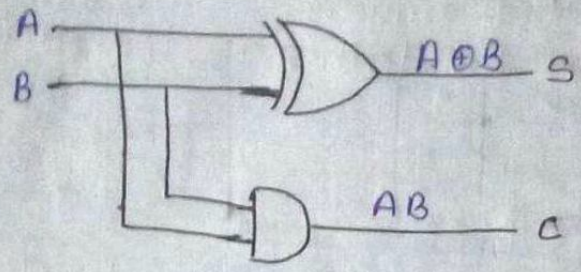
for carry

$S = A'B + B'A$

$S = A \oplus B$

$C = AB$

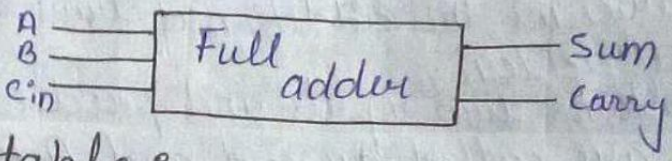
Logic diagram :-



half adder

full adder :-

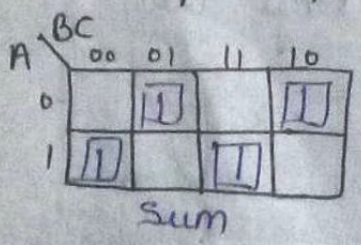
It is a arithmatic combinational logic circuit that performs addition of three single bits. It contains 3 inputs (A, B, C_{in}) and produce two output (Sum and Cout)



Truth table :-

A	B	C_{in}	Sum	Carry
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

K-map simplification



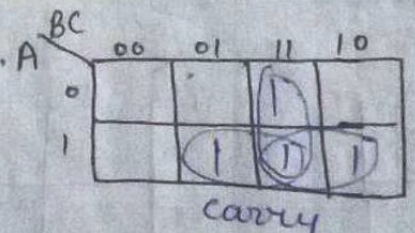
Sum

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

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

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

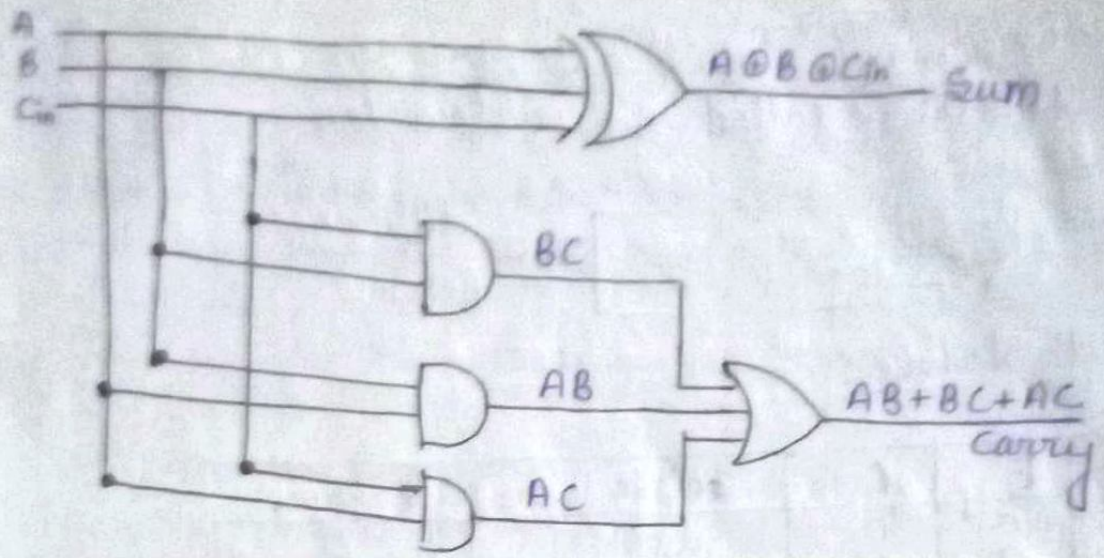
$= A \oplus B \oplus C$



carry

$C = BC + AB + AC$

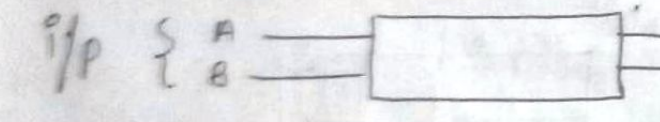
Logic diagram :-



full adder

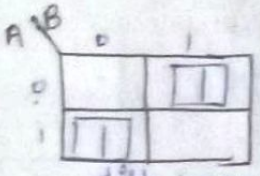
Subtractor :-

A Combinational Circuit that subtract two bits and produces their difference is called half subtractor.
It has an output to specify if 1 has been borrow

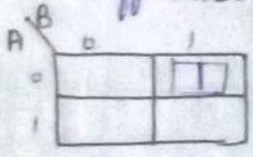


Truth table :-

A	B	difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

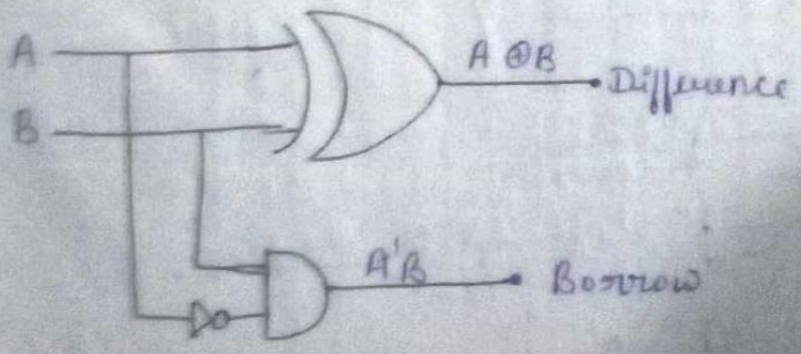


D = A'B + AB'
D = A ⊕ B



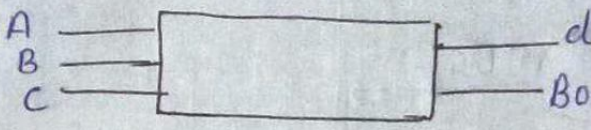
B = A'B

Logic diagram :-



full subtractor :-

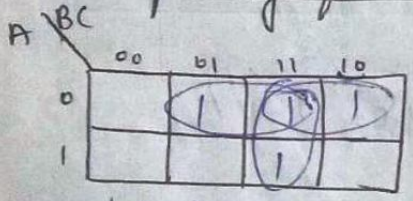
A combinational circuit that perform subtraction of three bits is called full subtractor.



Truth table :-

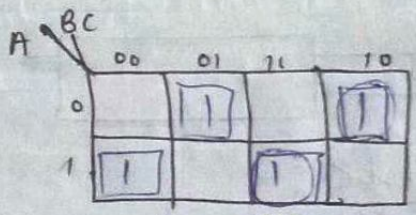
A	B	C	difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-map signification :-



for Borrow

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



for difference

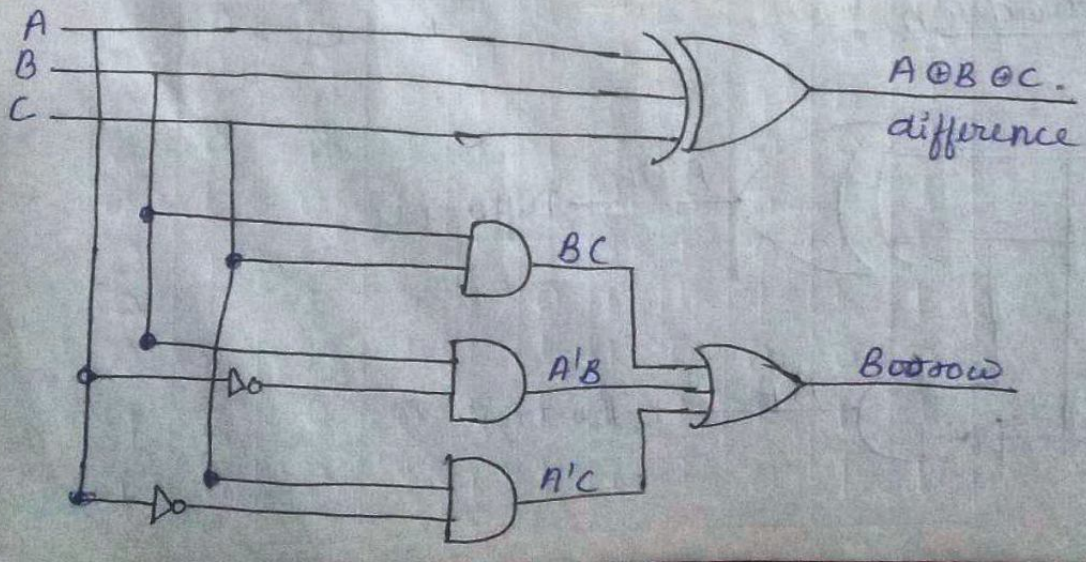
$D = A'B'C + AB'C' + ABC + A'BC'$

$D = B'(A'C + AC') + B(AC + A'C')$

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

$D = A \oplus B \oplus C$

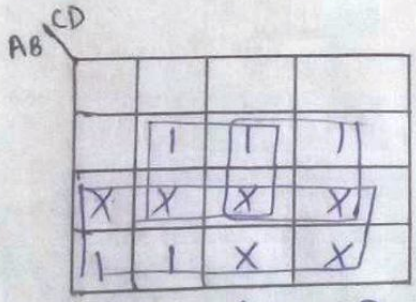
Logic diagram :-



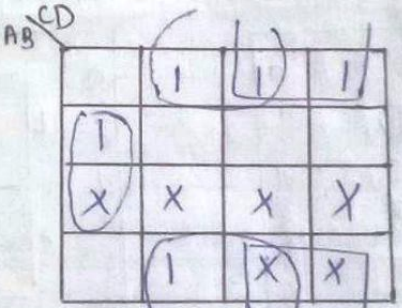
BCD to Excess 3 Converter:-

A combinational circuit that converts BCD code to Excess-3 code.
A BCD digit can be converted to Excess-3 code by simply adding 3 to it. Since we have only 10 digits (0 to 9) in decimal, we don't care about the rest and mark them (X).

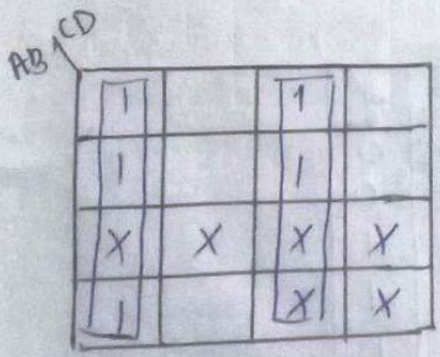
BCD (8421)				EXCESS-3			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



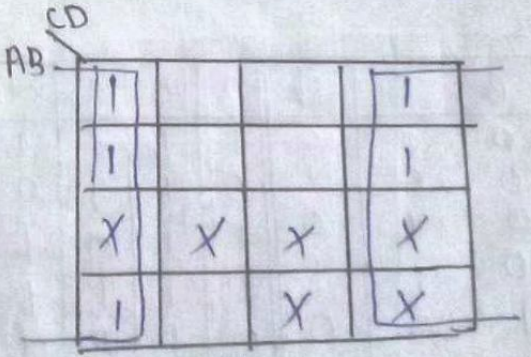
$W = A + BC + BD$



$X = B'C + B'D + BC'D'$

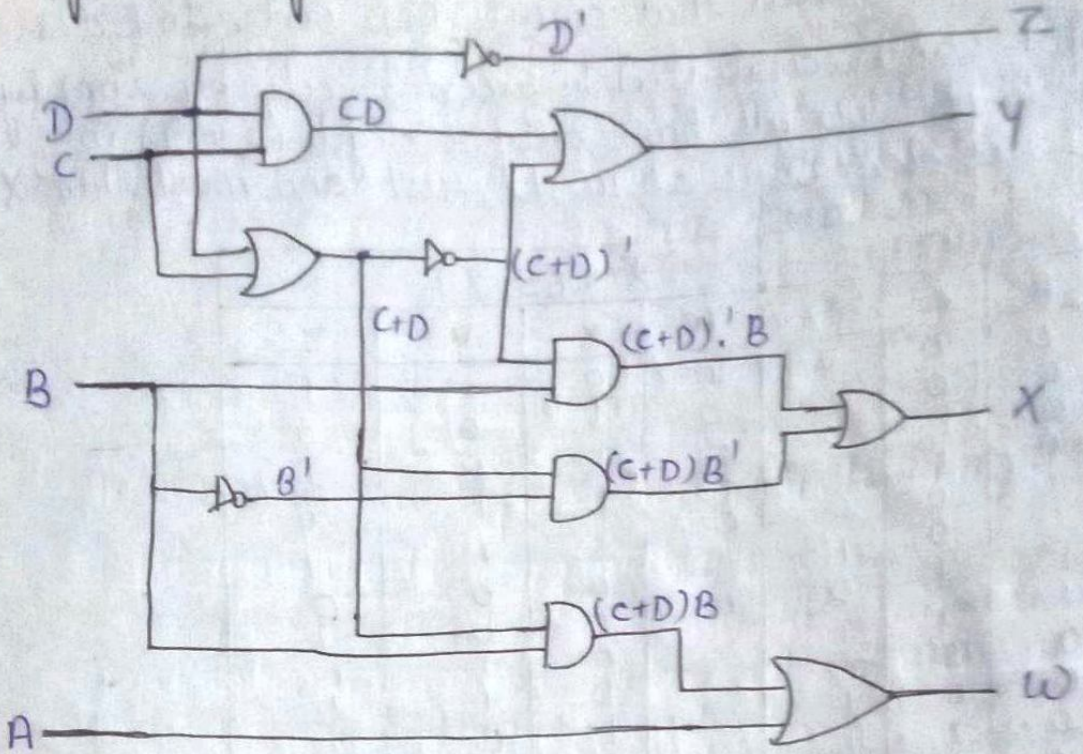


$Y = CD + C'D'$

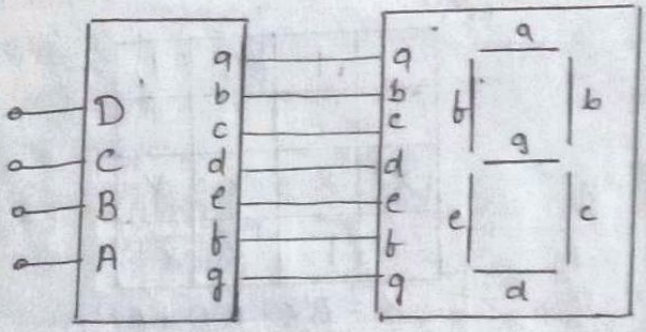


$Z = D'$

Logical diagram:-



BCD to 7 segment decoder :-

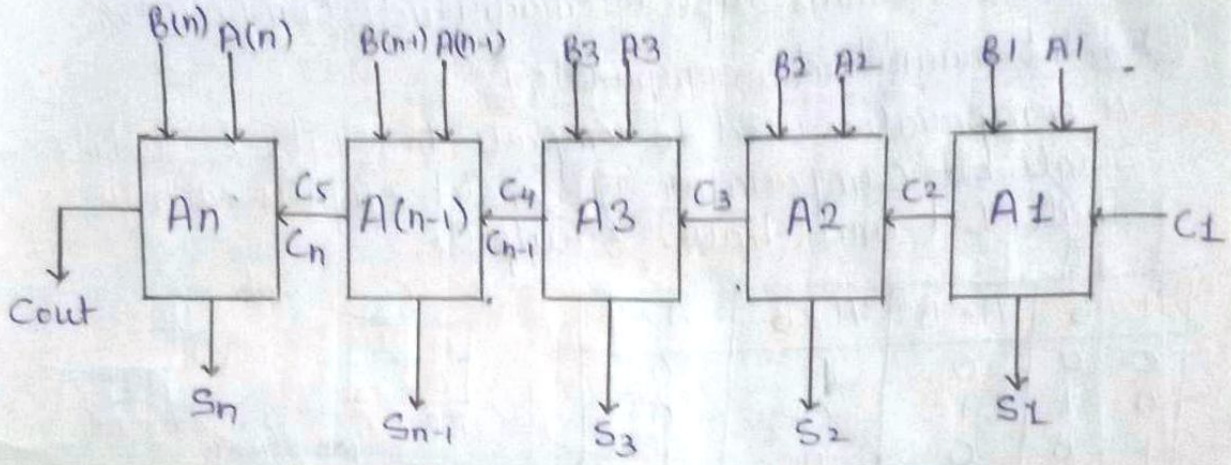


A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Parallel Adder :-

A single full adder performs the addition of two one-bit numbers and an input carry. But a parallel adder is a digital circuit capable of finding the arithmetic sum of two binary number greater than one bit.

It contains of full adders connected in a chain where the output carry from each full adder is connected to carry input of the next higher order full adder in the chain.

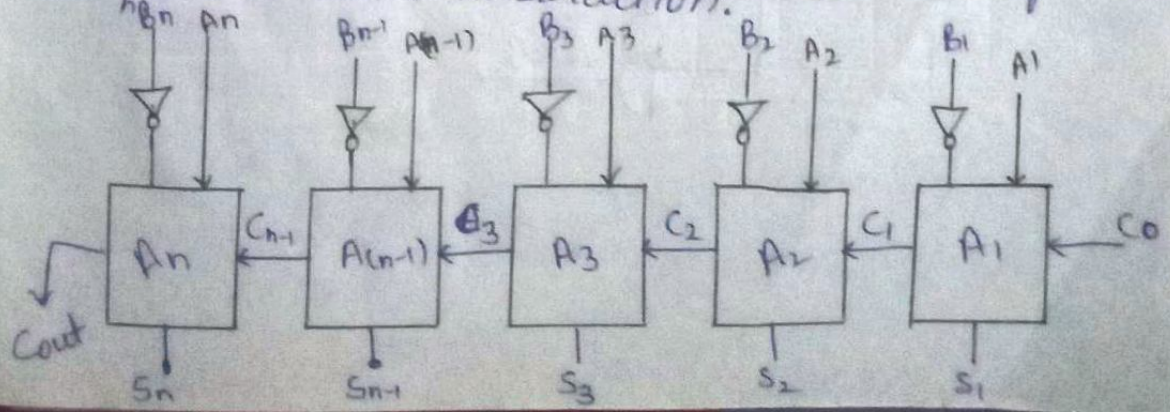


Parallel Subtractor :-

A parallel subtractor is a digital circuit capable of finding arithmetic difference of two binary numbers that is greater than one bit in length.

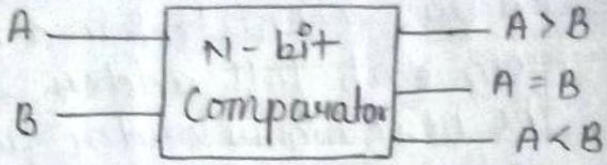
The parallel subtractor can be designed in several ways including half and full subtractors.

The below diagram (binary subtractor) is formed by comination of all full adders with subtrahend complement input. We are using adder because the addition of minuend with 2's complement of subtrahend is equal to their subtraction.



Magnitude Comparator :-

A magnitude comparator is a combinational ckt that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary No.

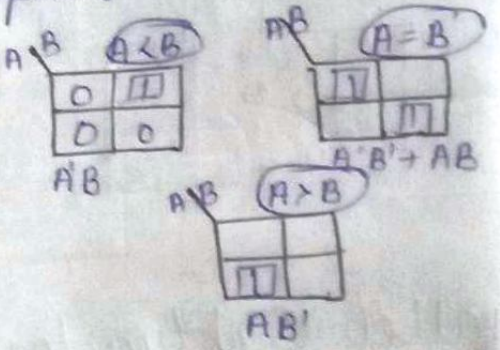


There are various types of magnitude Comparator

① 1 bit magnitude comparator :-

A comparator used to compare two bits is called single bit comparator. It consist 2 inputs each for single bit and have 3 outputs.

A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

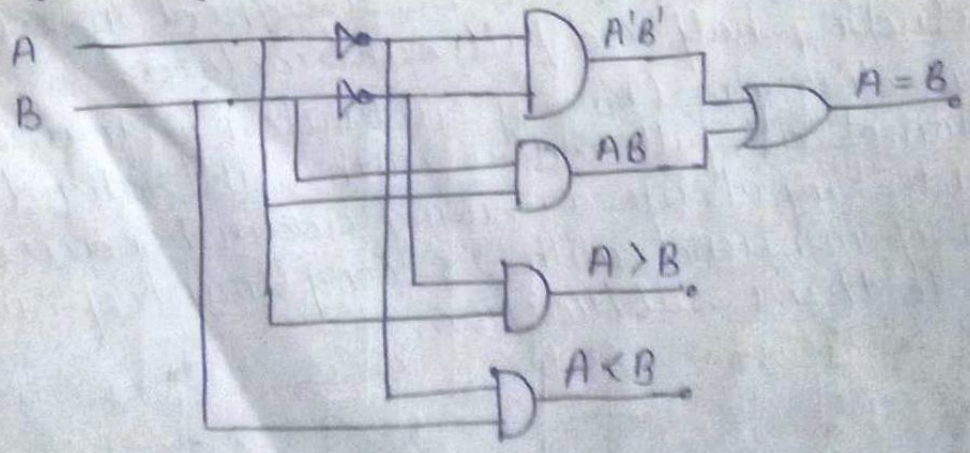


$A > B : AB'$

$A = B : A'B' + AB$

$A < B : A'B$

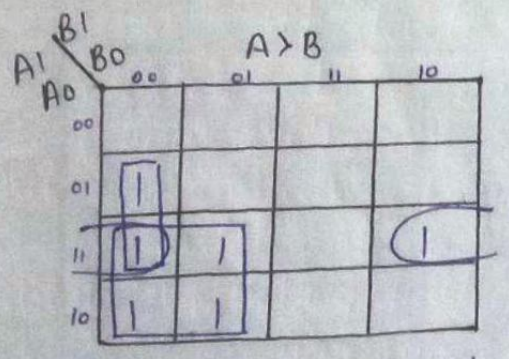
Logic diagram :-



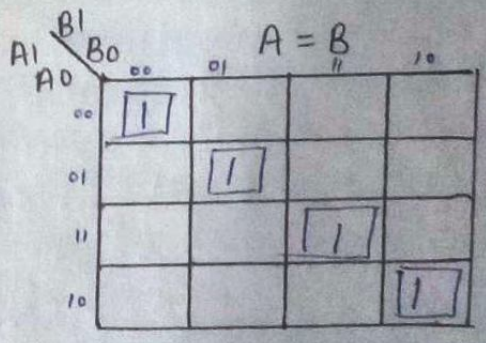
2-bit Magnitude Comparator :-

A Comparator used to compare two binary numbers such each of two bits is called 2-bit magnitude Compa. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binaries.

A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

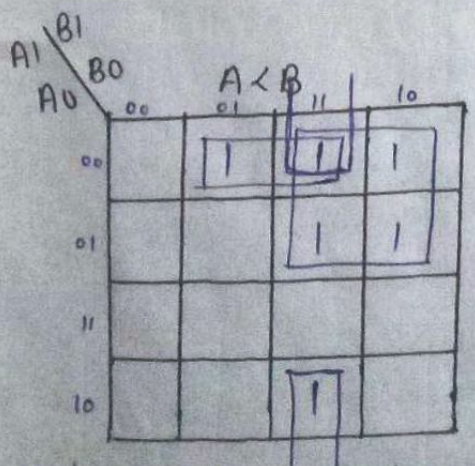


$A_1B_1' + A_0B_1'B_0 + A_1A_0B_0'$



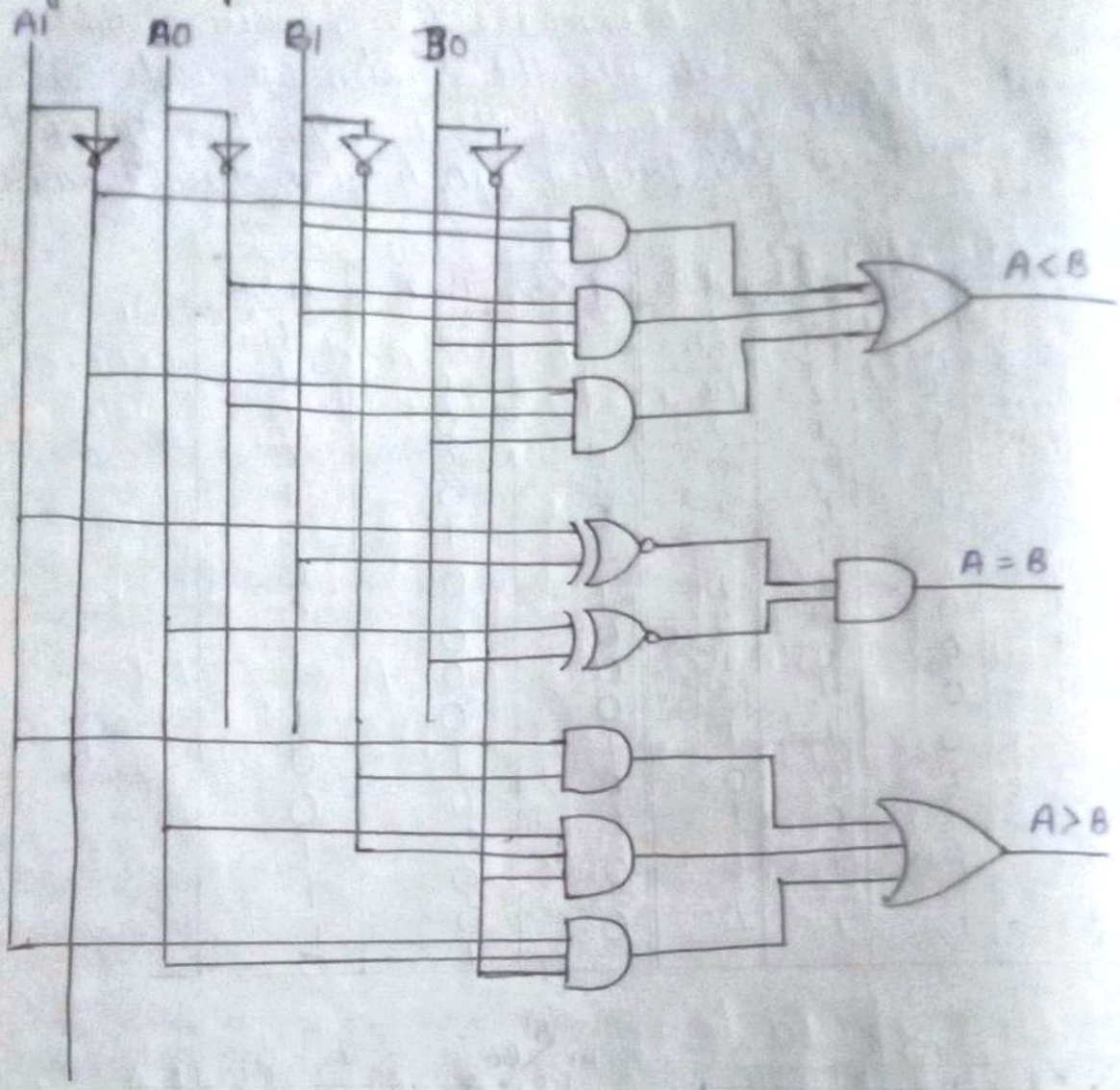
$A_1'A_0'B_1'B_0' + A_1A_0B_1B_0 + A_1A_0'B_1B_0'$

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



$A_1'B_1 + A_0'B_1B_0 + A_1'A_0'B_0$

Logic diagram:-

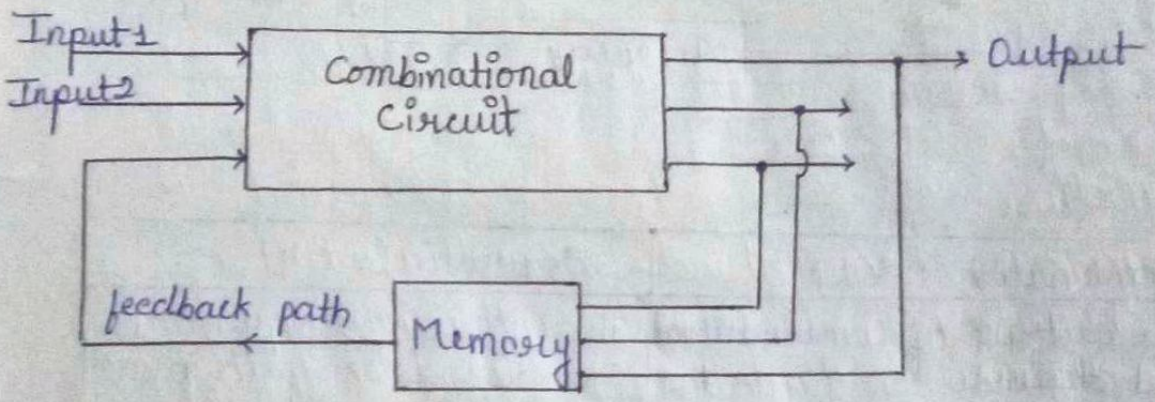


Sequential Circuits :-

A sequential circuit is a combinational logic circuit that consists of inputs variable, logic gates and output variable.
Sequential circuit produces an output based on current input and previous input variables.
Sequential circuit include memory elements that are capable of storing binary information. The binary information defines the state of the sequential circuit at that time.

Sequential Ckt = Combinational ckt + memory element

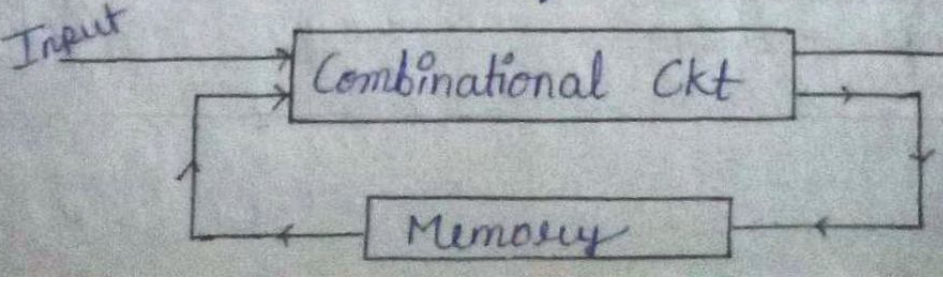
Block diagram :-



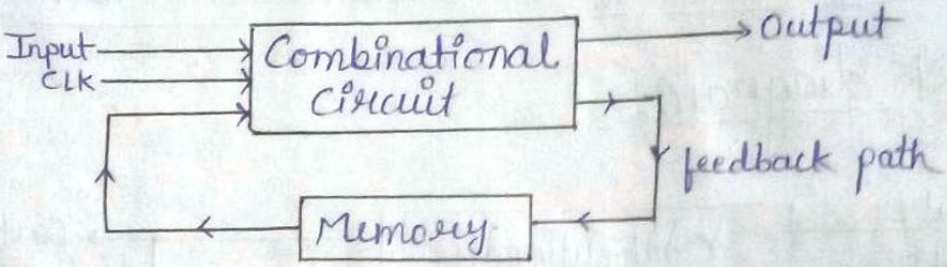
Types of sequential Circuit :-

There are two types of sequential circuit.

Asynchronous :- These ~~signals~~ circuits do not use a clock signal but uses the pulse of the inputs. These circuits are faster than synchronous Sequential circuits because there is clock pulse and change their state immediately when there is a change in input signal.
We use asynchronous sequential ckt when speed of operation is important and independent of internal clock pulse.



Synchronous:- These ckt uses clock signal and level inputs (or pulsed). The output pulse is the same duration as the clock pulse for the clocked sequential ckt. Since they wait for the next clock pulse to arrive to perform the next operation, so these ckt are bit slower compare to asynchronous.
Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.
Synchronous sequential ckt is used in flip-flop, Register, RAM, Counters etc.



Combinational Ckt	Sequential Ckt
The output of combinational ckt depends only on the present input.	The output of sequential ckt depends on both present and previous output.
feedback path is not present in combinational ckt	feedback path is present in the sequential ckt.
In combinational ckt, memory element are not required.	In the sequential ckt memory elements play an important role and required.
The clock signal is not required for combinational circuits.	The clock signal is required for sequential ckt.
The combinational ckt is simple to disign	It is not simple to design a sequential ckt

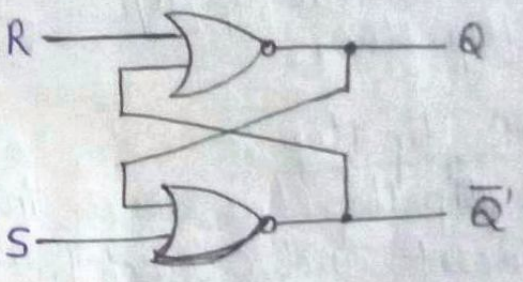
Latches :-

The basic storage element is called latch as its name suggest latch 0 and 1.
The output of latch depends upon its current input and on its previous output as it changes. Latches are level-sensitive device.

SR (Set-Reset) Latch :-

SR latch is a circuit with two cross-coupled NOR gate or 2 cross-compled NAND gate.
SR latch has 2 input S for set and R for RESET.
Q and Q' two outputs are present in SR latch.

Q	$\overline{Q}$	STATE
1	0	Set
0	1	Reset



Case-1:
S=0, R=0, there is no effect on the output state i.e hold state.

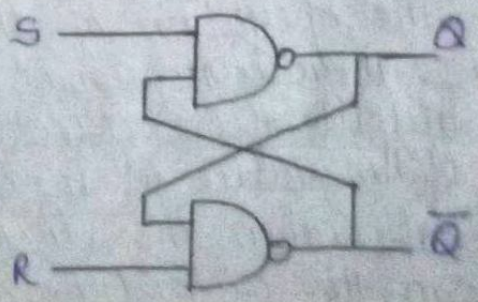
Case-2:
S=1, R=0, this will always set Q=1, where it will remain even after SET return to 0.

Case-3:
S=0, R=1, this will always Reset Q=0, where it will remain even after RESET return to 0

Case-4:
S=1, R=1, this condition tries to SET and RESET the latch at the same time and produces Q=Q'=0, the resulting output state is erratic and unpredictable.

SR Latch (NAND Gate):-

S	R	Qn
0	0	Invalid
0	1	1
1	0	0
1	1	ON(Memory



Flip-flops :-

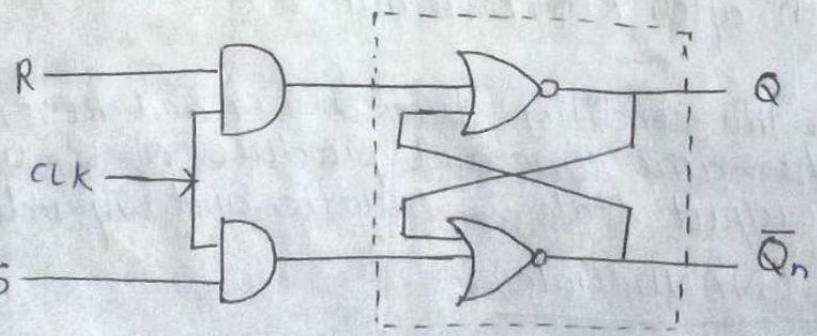
The memory elements used in sequential ckt are known as flip flops. The flip flops are binary cell that can store one bit of information. A flip flop ckt has two outputs, one for the normal value and other for the complement value of bit stored in it.

There are various types of FF that we need to stu

- SR flip flop
- D flip flop
- JK flip flop
- Master slave flip flop
- T flip flop.

(1) SR flip flop :-

SR flip flop has three inputs, labelled, S (set), R (Reset) and C [clock]. It has an output Q and sometimes the flip flop has a complemented output. There is an arrowhead-shaped symbol infront of the letter C to designate a dynamic input. The dynamic indicator symbol denotes the fact that the flip flop responds to a positive transition (from 0 to 1) of input clock pulse.



The operation of the SR flip flop is as follows:

- If there is no signal at the clock input C, the output of the circuit cannot change irrespective of the value at inputs S and R.
- Only when the clock signals changes from 0 to 1 can the output be affected according the values in inputs S and R.

Truth table :-

S	R	Q_n
0	0	Q_n
0	1	0 (Reset)
1	0	1 (Set)
1	1	Invalid

Characteristics table :-

Clk	S	R	Q_n	Q_{n+1}	
↑	0	0	0	0	Q_n
↑	0	0	1	1	
↑	0	1	0	0	0
↑	0	1	1	0	
↑	1	0	0	1	1
↑	1	0	1	1	
↑	1	1	0	x	Invalid
↑	1	1	1	x	

k-mapping :-

S \ $R\bar{Q}_n$	$\bar{R}\bar{Q}_n$	$\bar{R}Q_n$	RQ_n	$R\bar{Q}_n$
$\bar{S}$		1		
S	1	1	X	X

Boolean Expression :- $\boxed{\bar{R}Q_n + S}$

Characteristic Equation :-

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

Excitation table :-

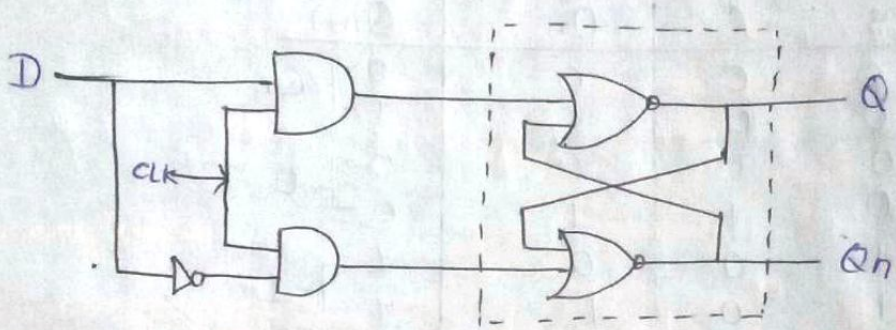
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

{X = Either 1 or 0}

D - flip flop :-

This is also known as data flip flop or delay ff. It is a digital circuit used to delay the change of state of its output signal (Q) untill the next rising edge of a clock timing input signal occurs.

The output follows the input only on the active stage of the clock pulse.



Truth table :-

clk	D	
↓	X	Qn
↑	0	0 (Reset)
↑	1	1 (Set)

K-mapping :-

Characteristic Equation ⇒ $\boxed{Q_{n+1} = D}$

Characteristic table :-

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

Excitation table :-

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

Jk flip-flop :-

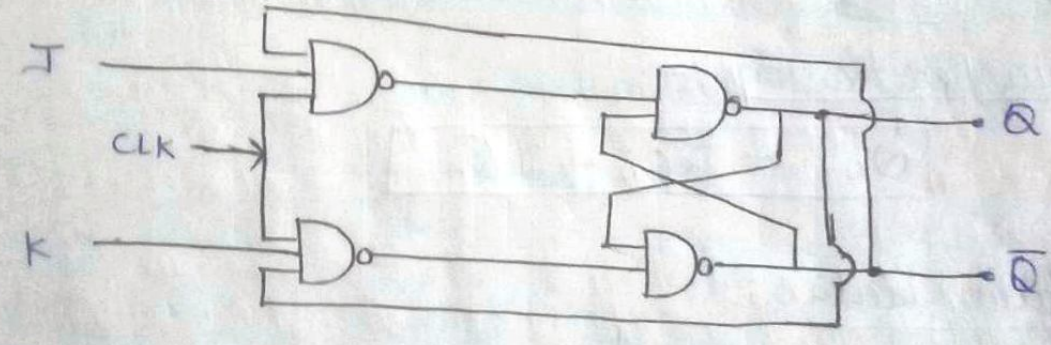
The J k flip flop is similar to the SR flip flop but there is no change in state when the J and k inputs are both Low.

The SR flip-flop or set Reset flip-flop has lots of advantages, But it has the following switching problem

* When S and R inputs are set 0, this condition is always avoided.
* When S or R input changes their state while the enable input is 1, the incorrect latching action occurs.

The Jk flip-flop removes these two drawbacks of SR flip-flop.

The difference b/w Jk flip flop and SR flop flop is that when both inputs of SR flip flop is set to 1, the circuit produces an invalid state as output, but in case of Jk flip flop there is no invalid state even if both the J and k are set to be 1.



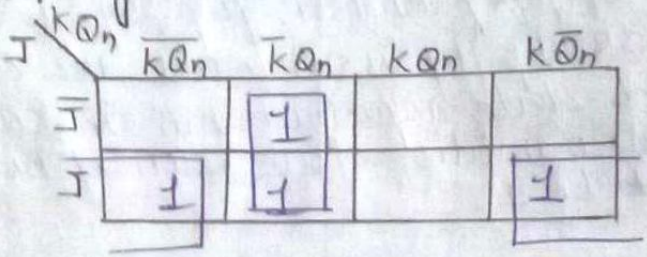
Truth table :-

Clk	J	K	Q_{n+1}
↓	X	X	Q_n
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	Q_n (Toggle state)

Characteristics Table :-

J	k	Q_n	Q_{n+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	1
1	1	1	0

k-mapping :-



Characteristic Eqn :-

$$Q_{n+1} = \bar{k}Q_n + J\bar{Q}_n$$

Excitation table :-

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

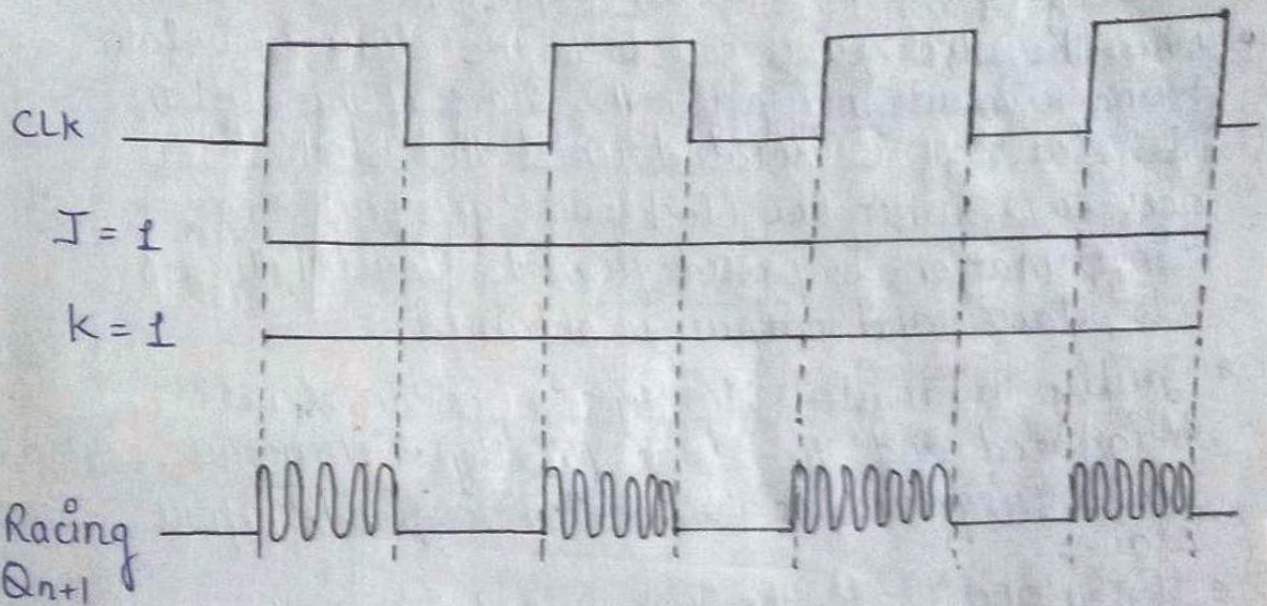
Race around condition in JK flip flop :-

Race around condition occurs in JK flip-flop when both its inputs J=k=1 and also the clock is equal to 1 for a long period.

- For JK flip flop, if J=k=1 and clk = 1 for a long period of time, then output Q will toggle

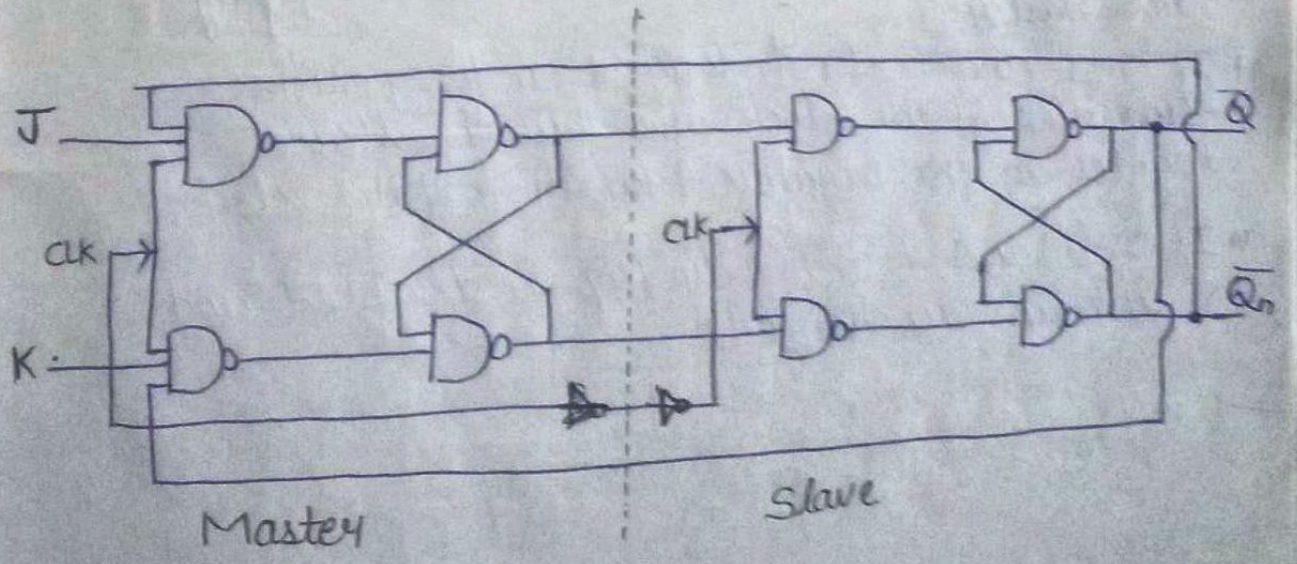
as long as clk remains high which makes the output unstable or uncertain.

- This is called race around condition in JK flip flop
- We can overcome this problem by making the clock is equal to 1 for very less duration
- The circuit used to overcome race around conditions is called the Master Slave JK flip flop.



1) T/2 > delay
2) Edge triggering

Master slave JK flip flop :-



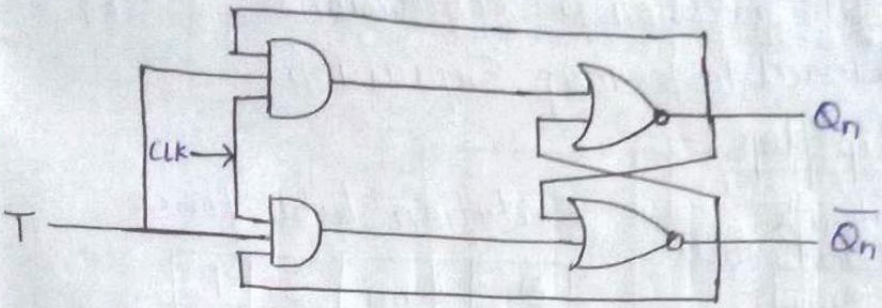
- The overall circuit acts as a single flip flop and store one bit.
- At any instance only one flip flop is triggered
- The problem of race arround condition is not present in master slave.
- The function table of Master Slave JK flip flop and JK flip flop are the same.
- It stories one bit of data.

Working of master slave flip flop :-

- when the clock pulse goes to 1, the slave is isolated. J and K inputs may affect the state of the system. The slave ff is isolated until the clock pulse goes to 0. when the Clock pulse goes back to 0, information is passed from the master flip flop to slave and output is obtained.
- Firstly the master flip flop is positive level triggered and the slave flip flop is negative level triggered, so the master responds before slave.
- If J=0 and K=1, the high Q' output of the master goes to the K input of the slave and the clock forces the slave to reset, thus the slave copies the master.
- If J=1 and K=0, the high Q output of the master goes to the J input of the slave and the negative transition of the clock sets the slave, copying the master.
- If J=1 and K=1, it toggles on the positive transition of the clock and thus the slave toggles on the negative transition of the clock.
- If J=0 and K=0, the flip flop is disabled and Q remains unchanged.

T flip flop (Toggle flip flop) :-

T flip flop is the simplified version of Jk flip flop. It is obtained by connecting the same input 'T' to both inputs of Jk flip flop. It operates with only positive clock transition or nigative clock transition.



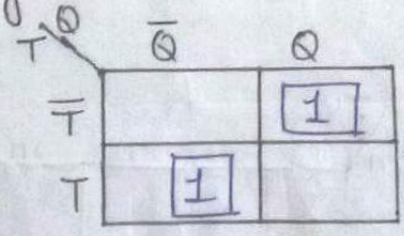
Function Table :-

Clk	T	Q_{n+1}
↑	0	Q_n
↑	1	$\overline{Q}_n$

Characteristics Table :-

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

k-mapping :-



Characteristic Eqn ⇒ $Q_{n+1} = \overline{T}Q + T\overline{Q}$

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

Excitation Table :-

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

flip-flop Conversion :-

The following steps to convert the FF

* Consider Truth table for destination FF and Extend it as exitation table of source FF
* Draw k map for source FF input variables. It will provide Expression for conversion
* Draw the ckt acc to k-map Expression.

① SR to JK flip flop :-

Truth table of JK (use characteristic table)

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

Exitation table for SR

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

Conversion table

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

k-map

S

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

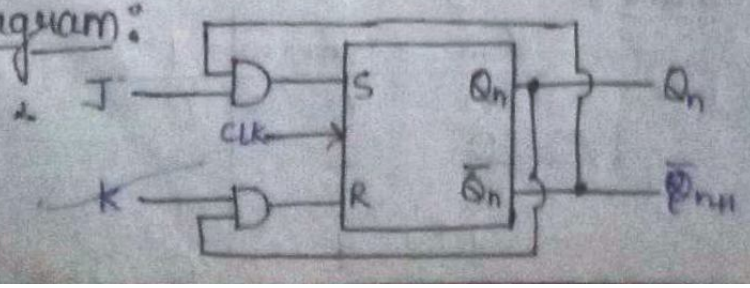
$S = \overline{Q_n} J$

R

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

$R = Q_n K$

Logic diagram:



⑤ SR to D flip flop :-

Truth table of D

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

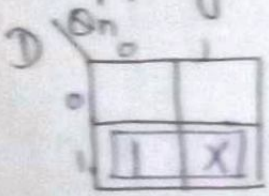
Excitation table of SR

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

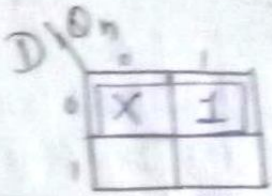
Conversion table

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

K mapping :-

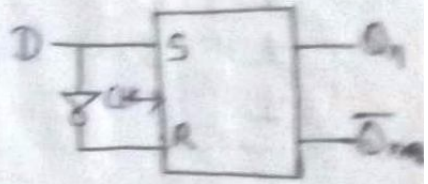


$S = D$



$R = \bar{D}$

Logic diagram



③ SR to T flip flop :-

Truth table of T

T	Q_n	Q_{n+1}
0	X	Q_n
1	X	$\bar{Q}_n$

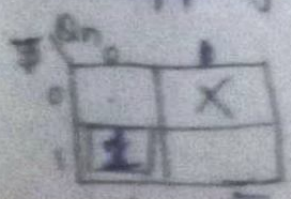
Excitation table of SR

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

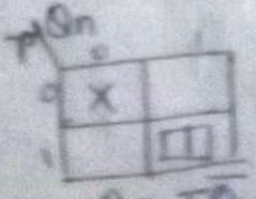
Conversion table

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

K-mapping

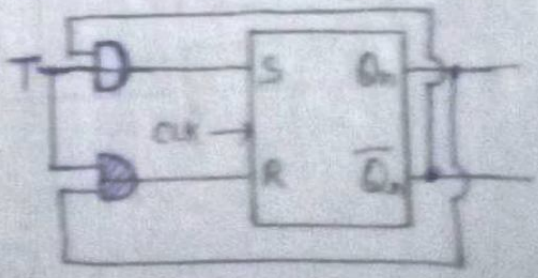


$S = T\bar{Q}_n$



$R = TQ_n$

Logic diagram



(4) JK to SR flip flop :-

Truth table of SR

S	R	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	Invalid

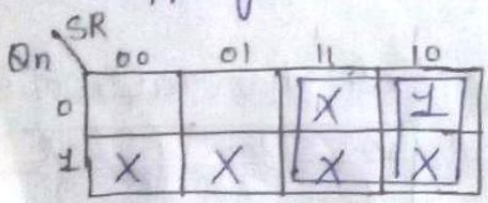
Excitation table of

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

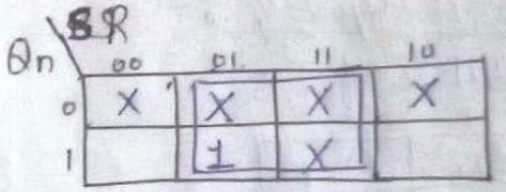
Conversion table

S	R	Qn	Qn+1	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	Invali	Invalid	Dont Care	
1	1	Invali	Invalid	Dont Care	

K-mapping

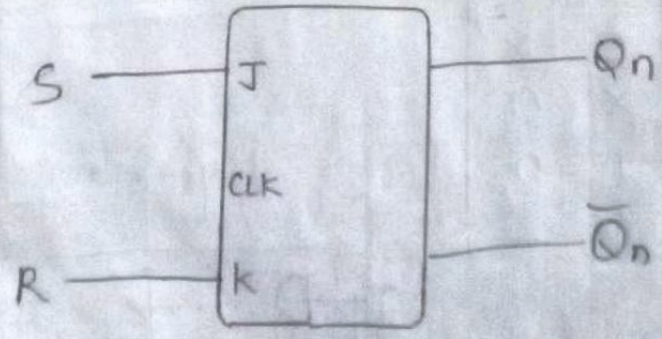


J = S



K = R

Logical Circuit :-



⑤ JK to T flip flop :-

Characteristic table of T

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

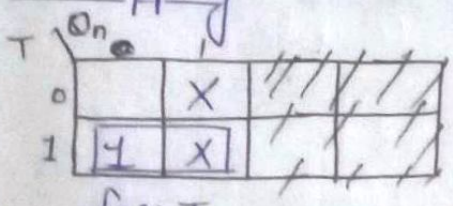
Excitation table of JK

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

Conversion table

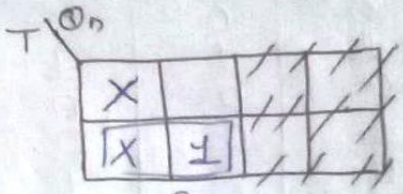
T	Q_n	Q_{n+1}	J	k
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

k-mapping



for J

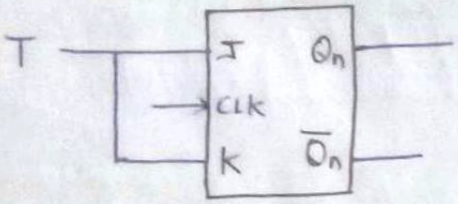
$J = T$



for k

$k = T$

Logic diagram :-

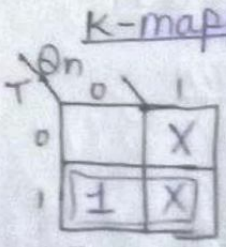


⑥ JK to D flip flop :-

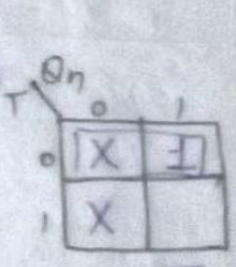
Conversion table

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

k-mapping

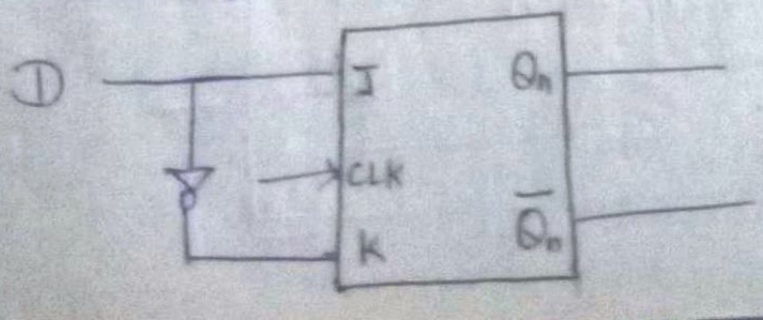


$J = D$



$k = \overline{D}$

Logic circuit :-



⑦ D flip flop to SR flip flop :-

Characteristic table of SR

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

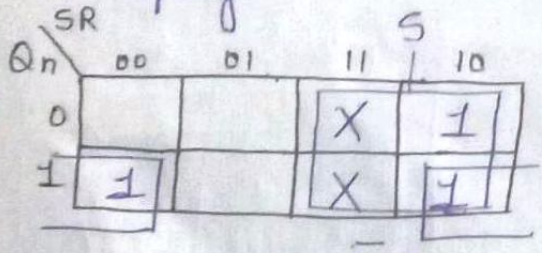
Excitation table

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

Conversion table

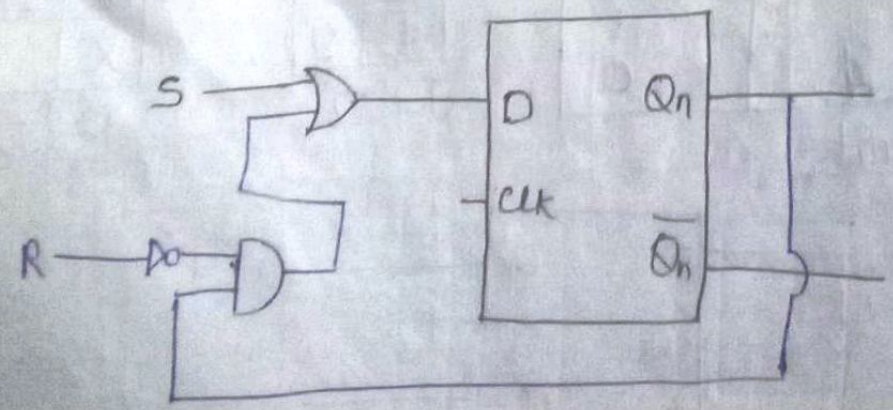
S	R	Qn	Qn+1	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	X	X
1	1	1	X	X

k-mapping :-



$D = S + \bar{R}Q_n$

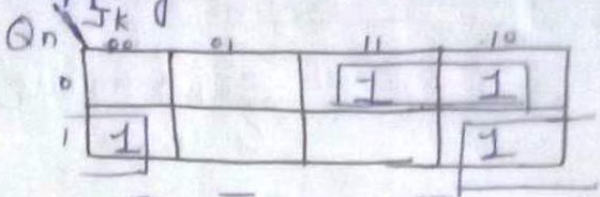
logic diagram



⑧ D to JK :-

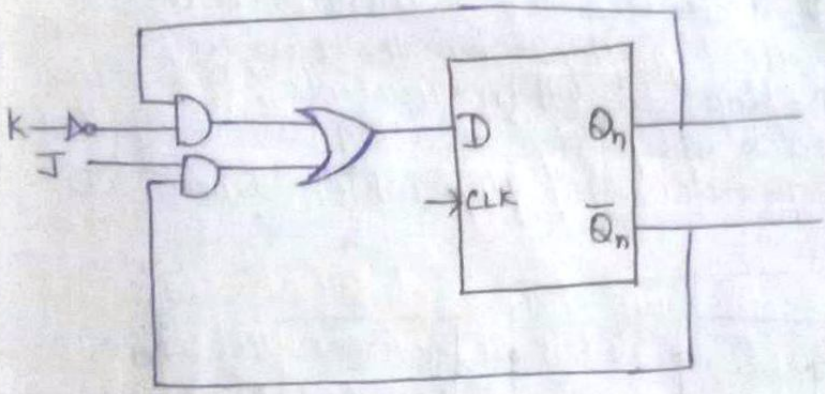
J	K	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

K-mapping :-



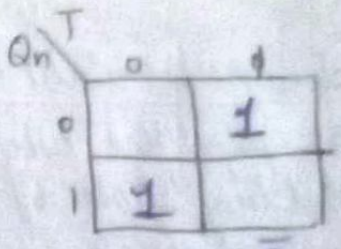
$D = \bar{K}Q_n + J\bar{Q}_n$

Logical Circuit :-



⑨ D to T flip-flop :-

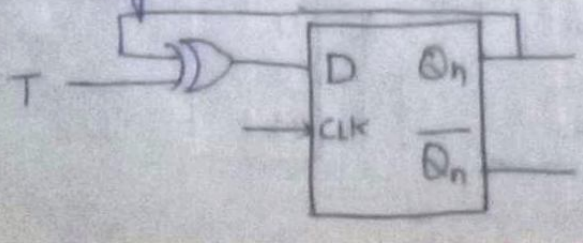
T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0



$D = T \oplus \bar{Q}_n + \bar{T} Q_n$

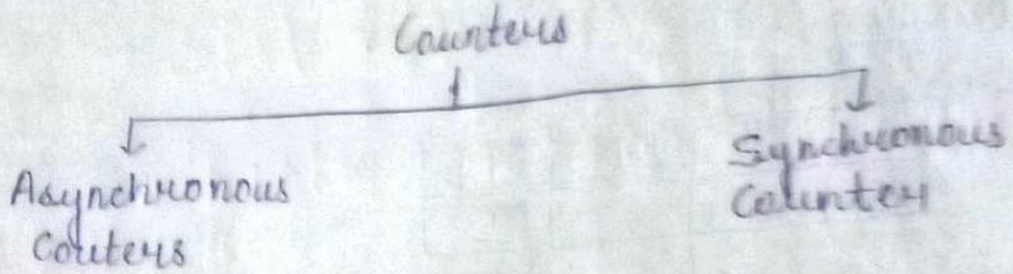
$D = T \oplus Q_n$

logic diagram :-



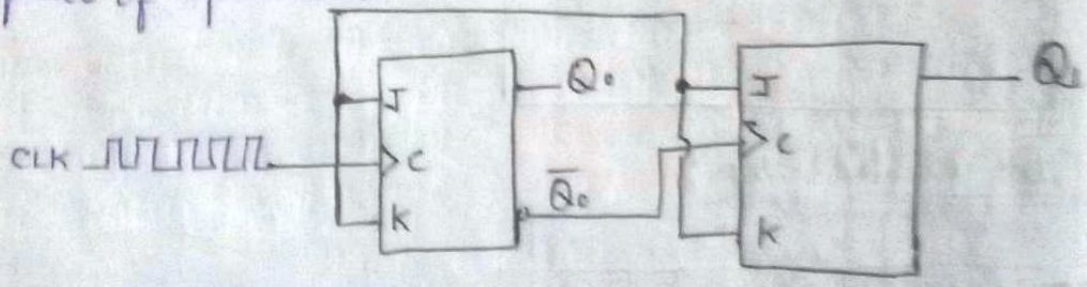
Counters:-

A counter is a sequential ckt consisting a set of flip-flops connected in a suitable manner to count the sequence of the input pulses presented to it in digital form.
If the pulse is generated by an event then it is called event counter.
The n-number of flip-flop can count upto 2^n No. of pulses.



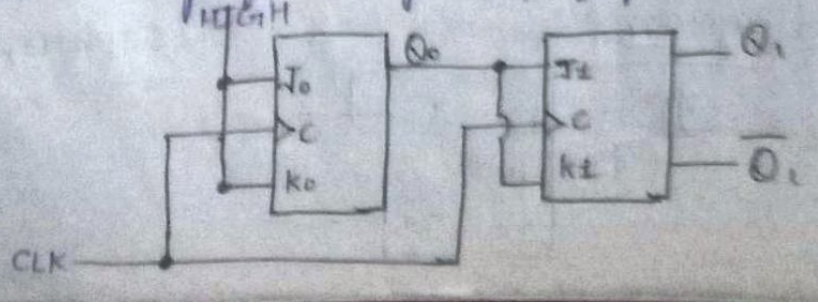
① Asynchronous counter:-

This counter is also known as Ripple counter. The first flip flop is driven by external clock while the successive flip-flops by the output of preceding flip-flop. In other words, In this ckt each flip-flop is triggered by the output from the previous flip-flop which limit its speed of operation.



② Synchronous Counter:-

All the flip-flop are simultaneously driven by common clock. Setting time of counter is equal to the propagation delay of a single flip-flop.



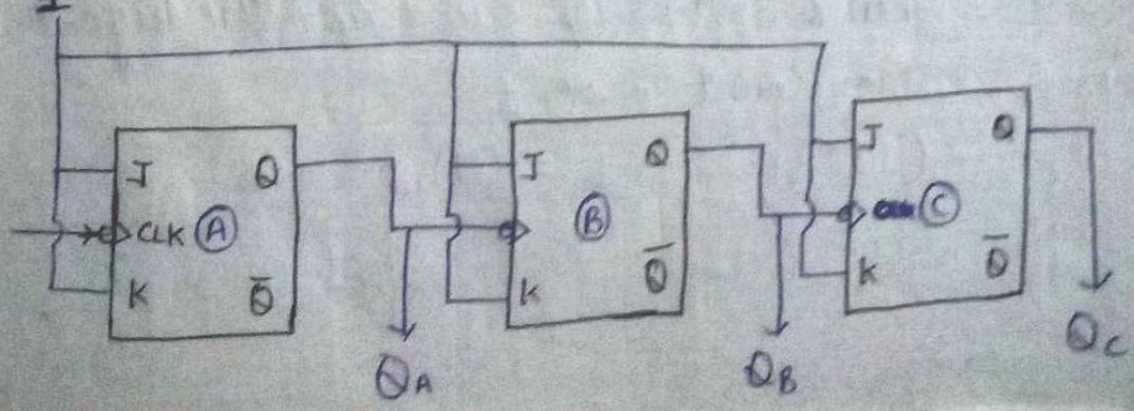
Asynchronous Counter	Synchronous Counter
The flip-flop are triggered either by clock pulse or by the output generated by preceding flip-flop.	All the flip-flops are triggered by a common clock pulse
Flip-flop are not clocked simultaneously	Flip flops are clocked simultaneously
Circuit is simple for more no. of states.	Circuit become complicated as no. of state increases.
Speed is slow as the clock is propagated through no. of stages.	Speed is high as the clock is given at the same time
Decoding error is present	Decoding error is not present.

Ripple Counter (Asynchronous Counter):-

In this counter, clock pulse is applied to the first flip-flop i.e the least significant bit stage of the counter and the successive flip-flop is triggered by the output of the previous flip-flop and thus the counter has a cummulative settling time.

The first stage of the counter switches first, on the application of a clock pulse. This change in state of Ist FF act as clock input to IInd FF similary change in state of IInd FF act as clock input to III FF.

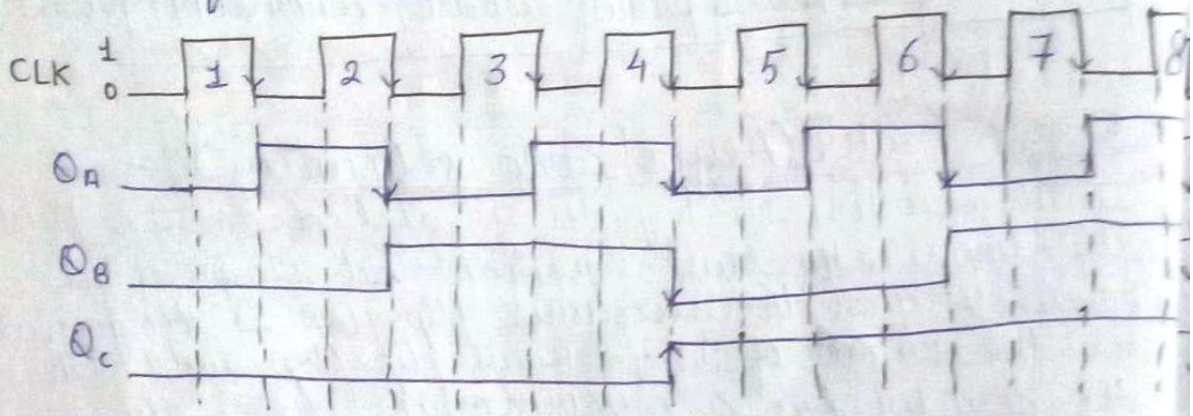
The successive stages changes their state in turn causing a ripple through effect of the count pulse. As triggers move through the FF like a ripple, it is called a ripple counter.



The flip-flop is negative edge triggered i.e FF will change its state only when clock or output of previous flip-flop changes from '1' to 0.

Q_C	Q_B	Q_C	CLK
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

The truth table of 3-bit FF can also be represented in waveform as



The counter in the diagram has 8 different states. Thus it is a MOD-8 ripple counter. The MOD number (or modulus) of a counter is the total number of states it sequences through in each complete cycle.

$$\text{MOD number} = 2^n$$

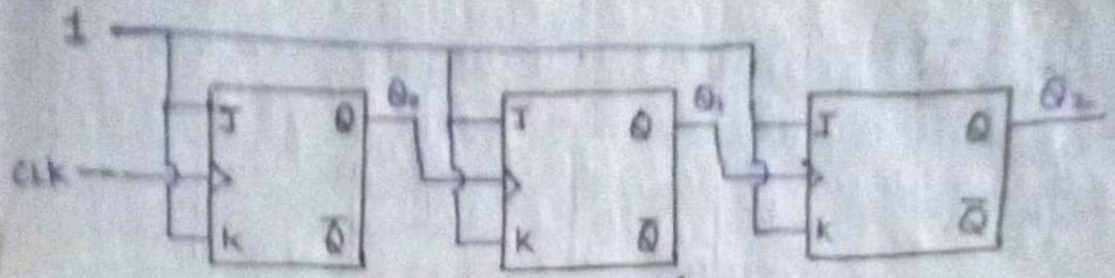
where n is number of flip-flop

The maximum binary number counted by the Counter is $2^n - 1$. Thus a 3-flip-flop can count as high as 7.

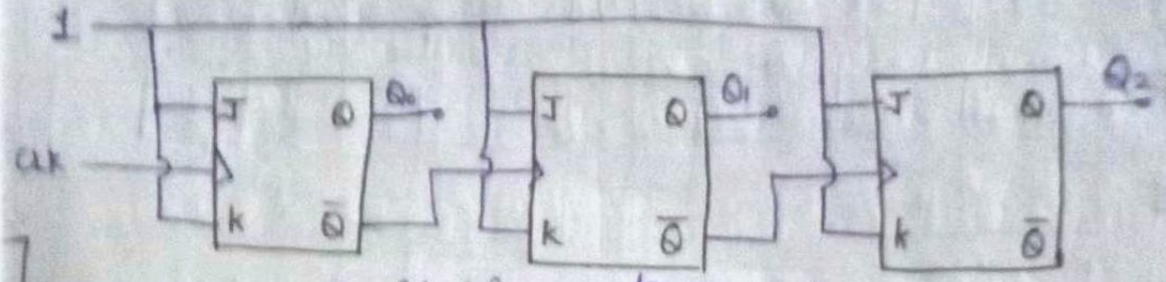
$$\text{Max Count} = 2^n - 1$$

$$(111)_2 = 2^3 - 1 = (7)_{10}$$

Up Counter and Down Counter for positive Edge clock :-

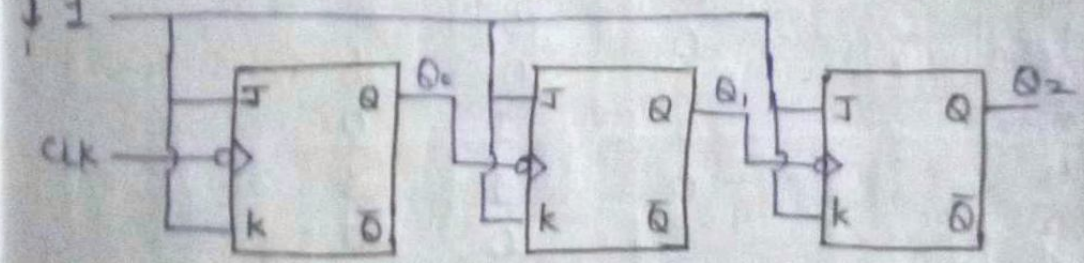


3-bit binary ~~down~~ counter

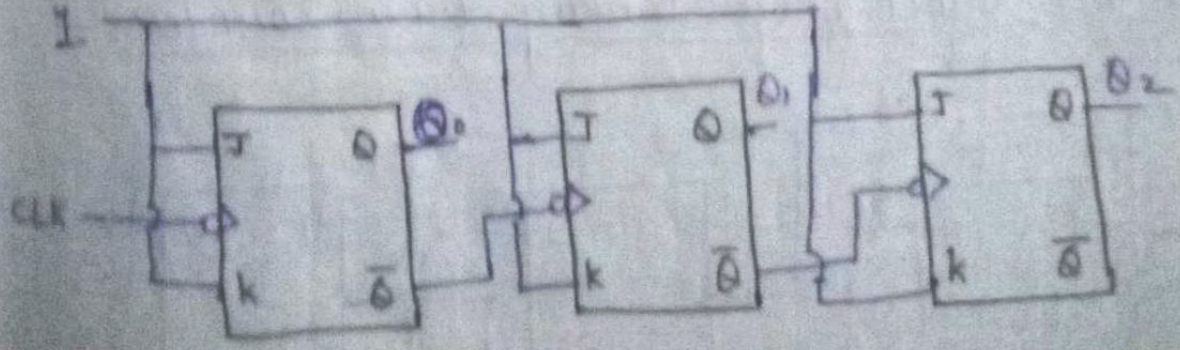


3-bit binary down counter

Up Counter and down counter for negative Edge clock :-

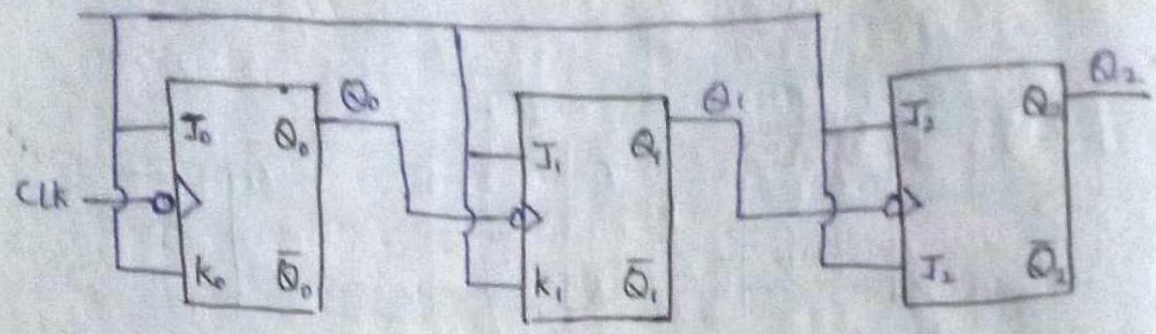


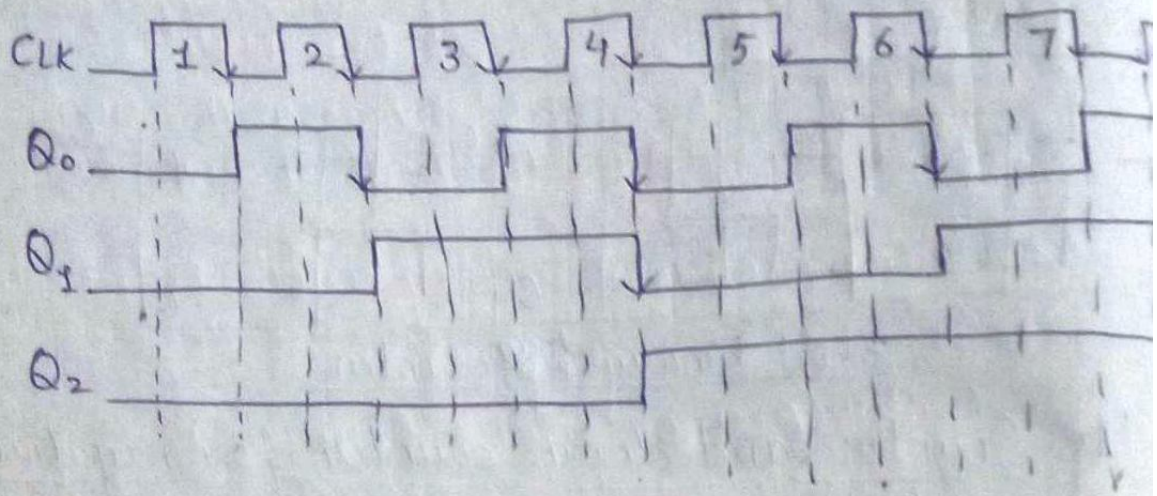
3-bit binary up counter



3-bit binary down counter

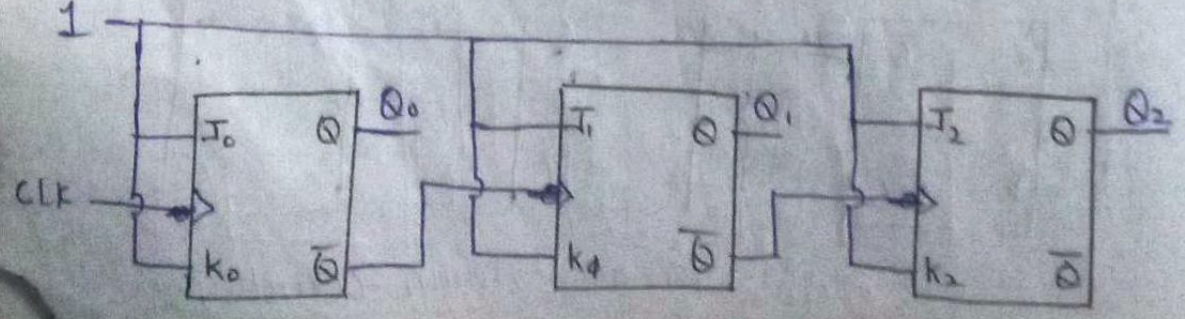
3-bit asynchronous Binary Up Counter :-





No. of pulse	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
8			

3-bit asynchronous Binary down counter :-



Decimal	Q_C	Q_B	Q_A
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0

MOD-6 Asynchronous Counters :-

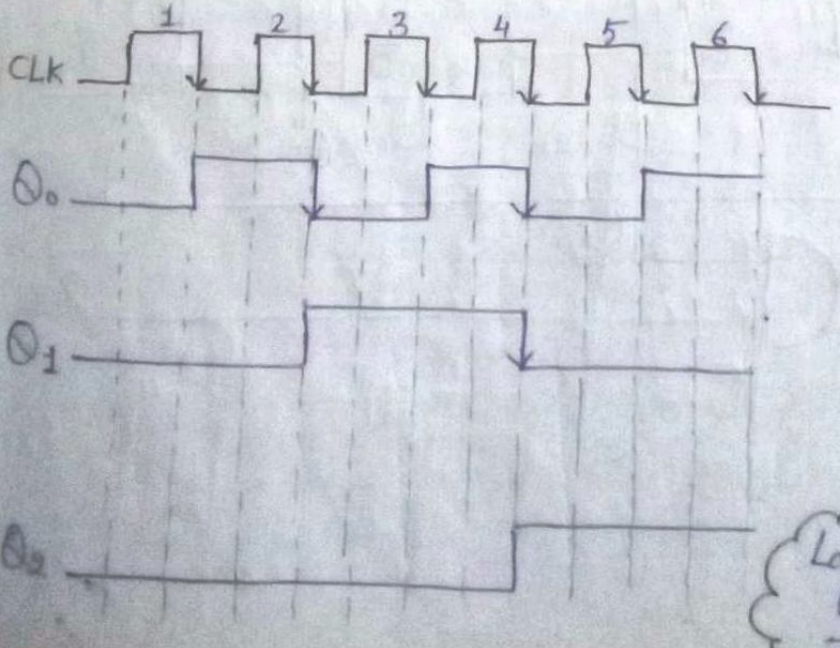
For constructing a Mod-6 counter we require three flip-flop (2^3). MOD-6 counter will count 6 states from 000 to 101.

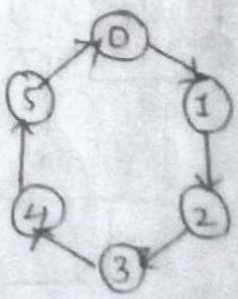
Truth table :-

Q_C	Q_B	Q_A	Reset logic
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	0
1	1	1	0

Present state	Next state
000	001
001	010
010	011
011	100
100	101
101	000

Reset the state to 000_2 when 110_2 is detected.





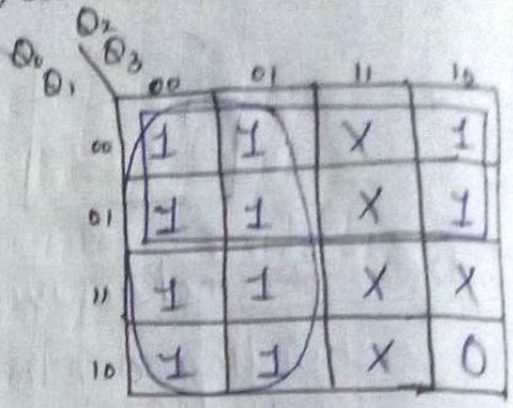
$$\text{Clear} = Q_c' + Q_b'$$
$$= \overline{Q_c \cdot Q_b}$$

Logic diagram as next page with 3 flip flop

BCD Ripple Counter:-

This is a also called decade counter. A binary code decimal (BCD) is a serial digital counter that counts ten digits. And it resets for every new clock input. As it can go through 10 unique combination of output. it is also called as "Decade counter". A BCD counter can count 000 to 1001

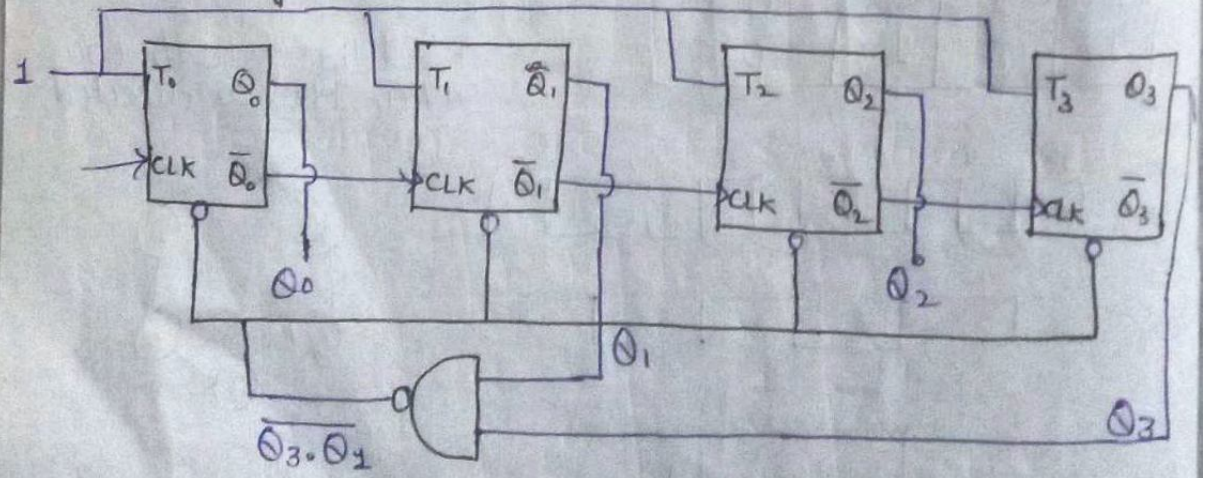
	Q_3	Q_2	Q_1	Q_0	Cor
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	X
12	1	1	0	0	X
13	1	1	0	1	X
14	1	1	1	0	X
15	1	1	1	1	X



$Cor = \overline{Q_3} + \overline{Q_1}$

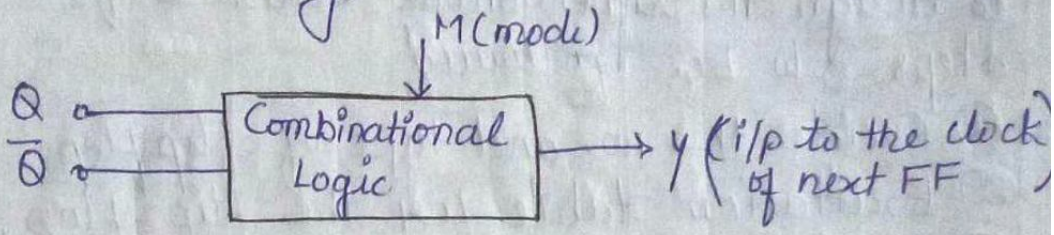
$= \overline{Q_3 \cdot Q_1}$

Logic diagram:-



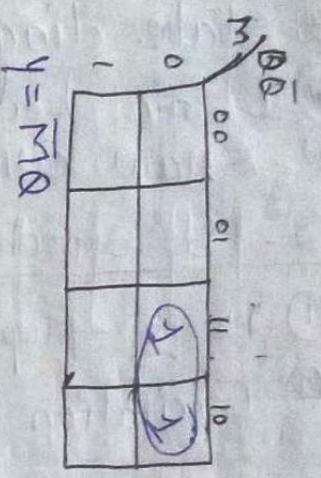
3-bit up/down Counter :-

- Both Up and Down Counters are combined.
- A mode Control input (M) is used.
- M = 0 (up Counting)
  M = 1 (Down Counting)

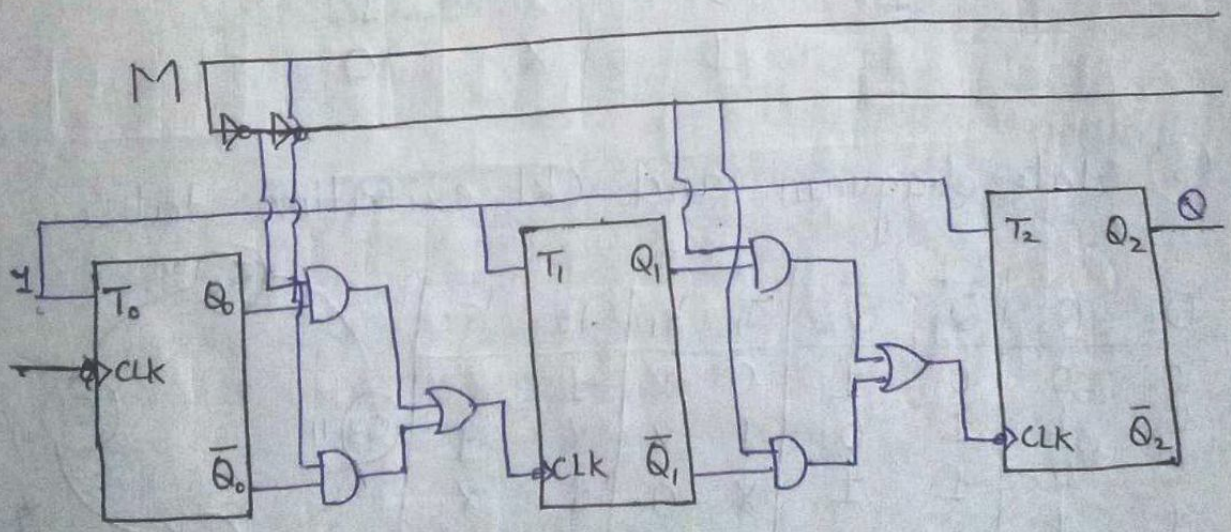


M	Q	$\bar{Q}$	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$\rightarrow M\bar{Q}$



$\therefore Y = \bar{M}Q + M\bar{Q}$



3-bit up-down Counter

Synchronous Counter

Synchronous means at same time. In synchronous counters, clock pulse is applied to all the flip-flops at the same time.

To design the synchronous counters following steps to be followed

1. Decide the no. and type of flip flop.
2. Excitation table for the flip-flop.
3. State diagram and circuit exitation table
4. Obtain boolean expression.
5. Draw circuit diagram.

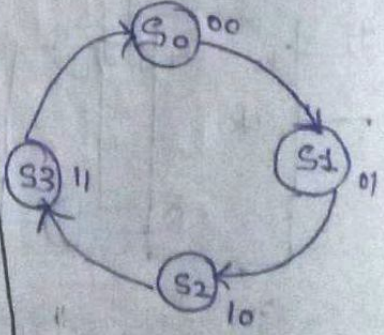
2-bit Synchronous up counter :-

1. JK flip-flop (2 flip-flop)
2. Excitation table of JK FF

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

3. State diagram and Ckt exitation table

Q_1	Q_2	$\bar{Q}_1$	$\bar{Q}_2$	J_1	K_1	J_2	K_2
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	1	1	X	0	1	X
1	1	0	0	X	1	X	1



④

Q_1 \ Q_2	0	1
0		1
1	X	X

$J_1 = Q_2$

Q_1 \ Q_2	0	1
0	1	X
1	1	X

$J_2 = 1$

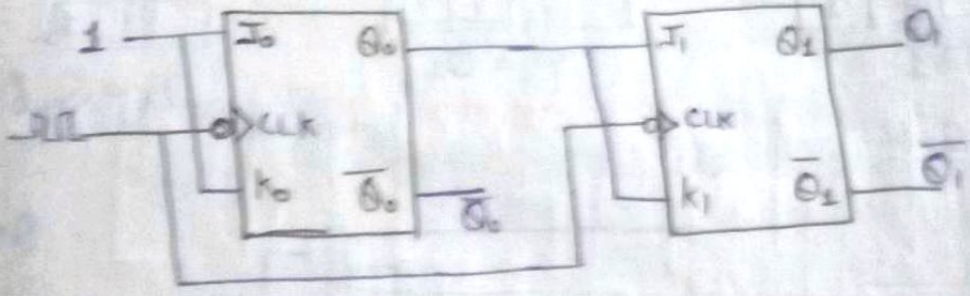
Q_1 \ Q_2	0	1
0	X	X
1	0	1

$K_1 = Q_2$

Q_1 \ Q_2	0	1
0	X	1
1	X	1

$K_2 = 1$

⑤ Logic diagram



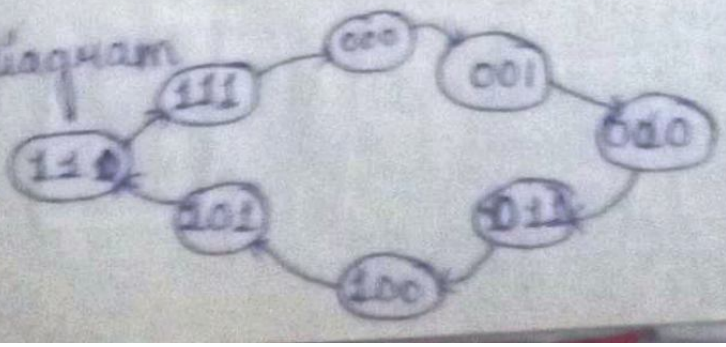
3-bit Synchronous Up Counter :-

① Determine number of flip-flop and decide type of flip-flop.
No of flip-flop = 3

② Excitation table of T flip-flop

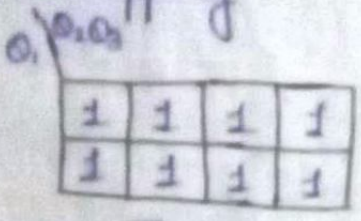
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

③ State diagram

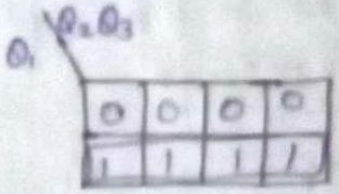


Present state			Next state			Flip flop		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

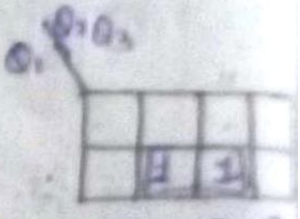
④ k-mapping



$T_1 = 1$

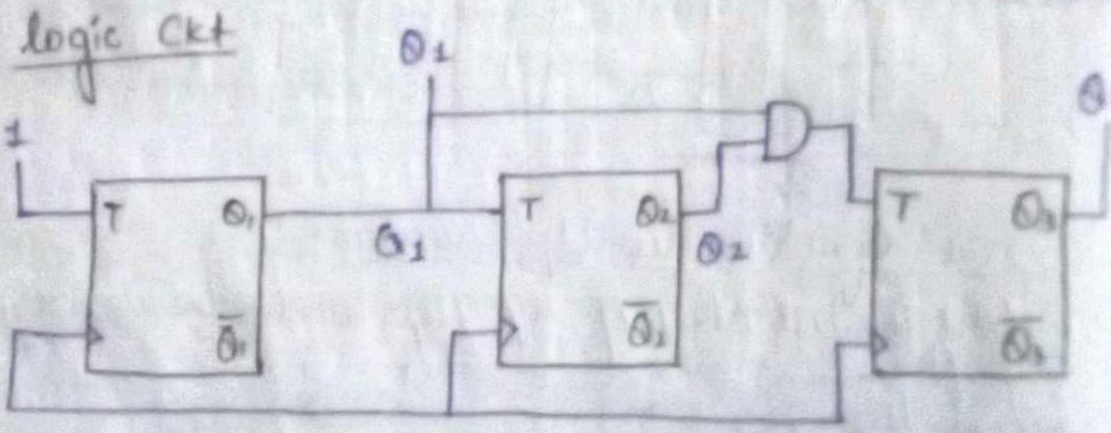


$T_2 = Q_1$



$T_3 = Q_1 Q_2$

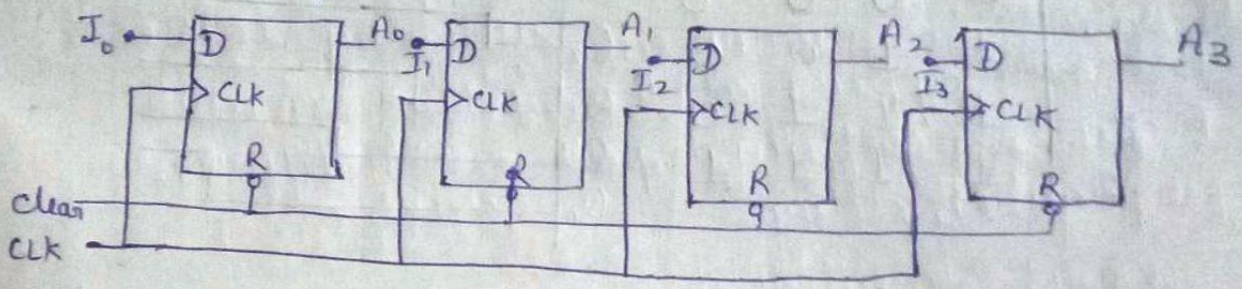
⑤ Logic Ckt



Pkg@084

Register :-

The flip-flop are essential component in clocked sequential ckt.
To increase the storage capacity, we have to use group of flip-flop. The group of flip-flop is known as storage or register.
n-bit register consists of group of n-flip flop capable of storing n bit of binary information.
All the flip-flops are connected in series to store multiple bit of data.



(register Constructed with four D type ff.

* 'Clock' triggers all flip-flop on the positive edge of pulse.
* 'Clear' is used for clearing the register to all 0 prior to its clocked operation

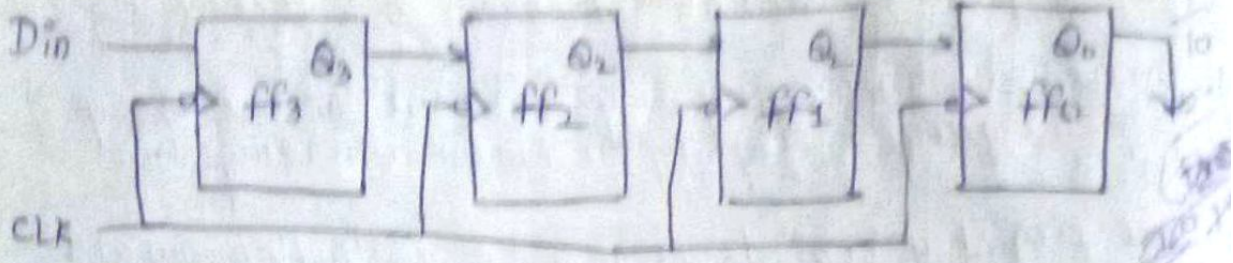
D_{1011}

1	0	1	1

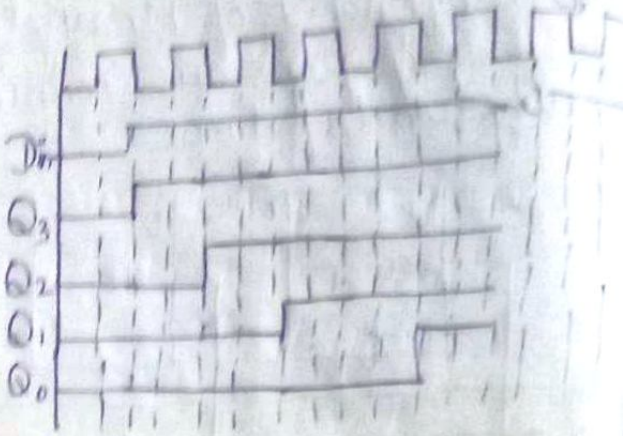
Shift Register :-

Shift register is a group of ff used to store multiple multiple bit of data. The bits stored in such registers can be made to move

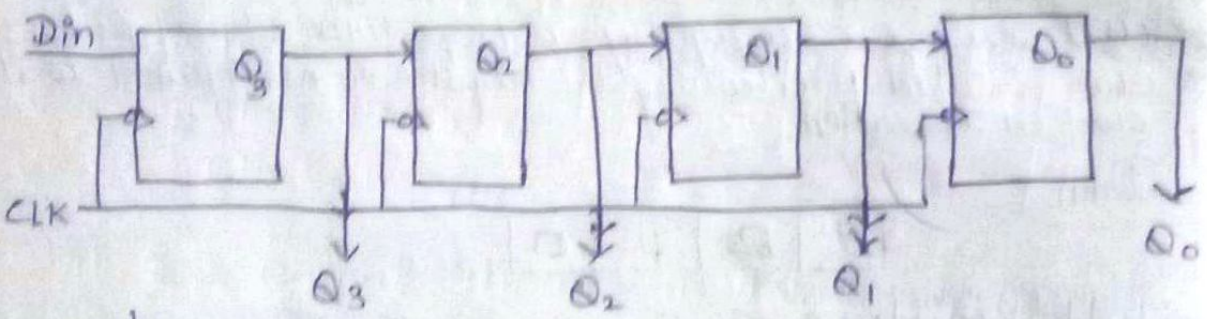
Serial Input Serial Out (SISO):-



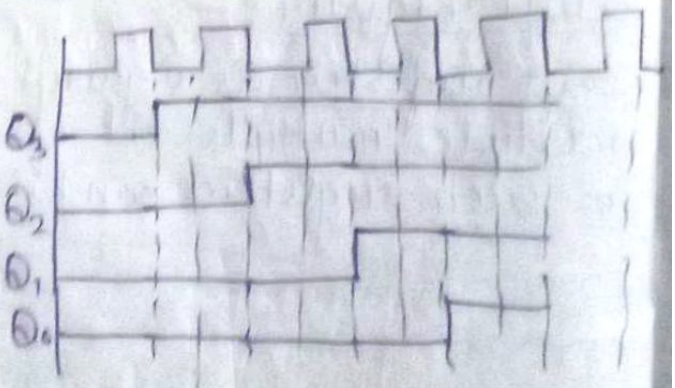
clk	Q_3	Q_2	Q_1	Q_0
initially	0	0	0	0
↓	1	0	0	0
↓	1	1	0	0
↓	1	1	1	0
↓	1	1	1	1



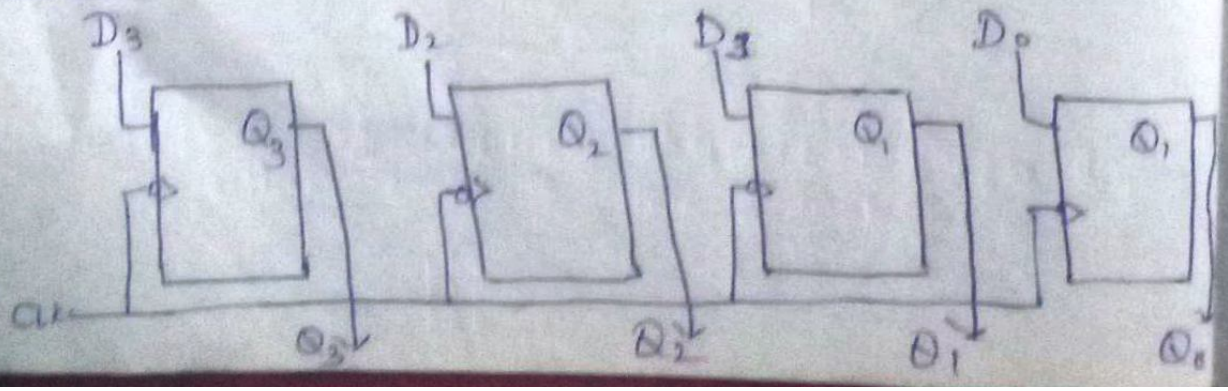
SIPO



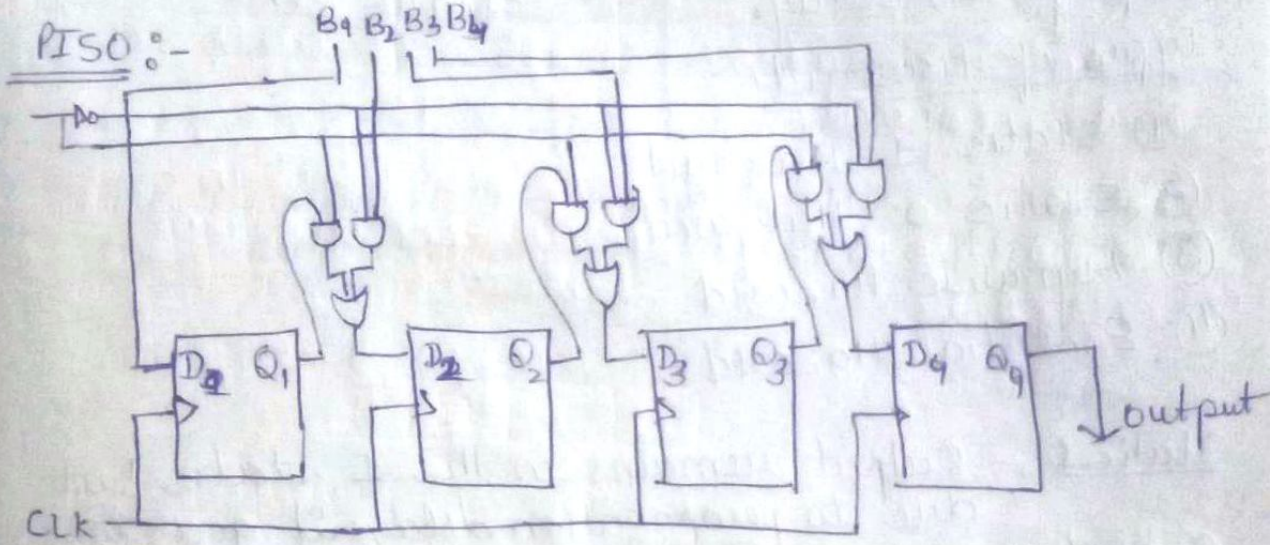
clk	Q_3	Q_2	Q_1	Q_0
initial	0	0	0	0
↑	1	0	0	0
↑	1	1	0	0
↑	1	1	1	0
↑	1	1	1	1



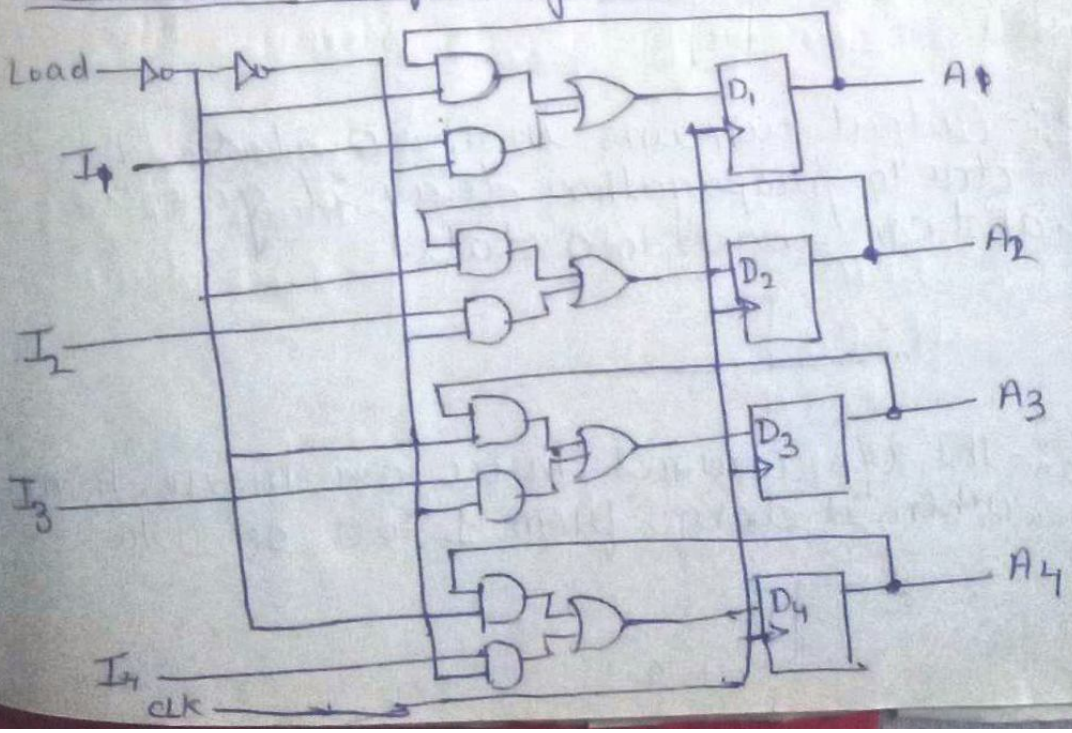
PIPO



PISO :-



Universal Shift register :-



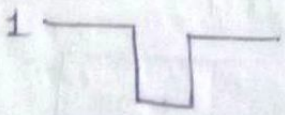
① when load = 1
The data Input is reach the Input of flip flop
② when load = 0
The data output of each flip flop feedback to D Input

Hazards occurs in combinational ckt when they may cause a temporary false-o/p value when this condition occurs in asynchronous sequential ckt, it may result in a transition to a wrong stable state

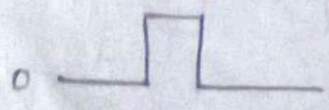
Hazards :- Hazards are unwanted switching transients that may appear at the o/p of a circuit because of different path ~~circuit~~, different propagation delay. exhibits.

Types of Hazard
① Static 1 Hazard
② Static 0 Hazard
③ Dynamic Hazard
④ Essential Hazard

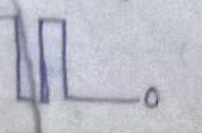
Static 1 :- Output remains in the 1 state but due to propagation delay it goes to 0 and come back to 1 state



static 0 :- Output remains in the 0 state but due to propagation delay it goes to 1 state and come back to 0 state



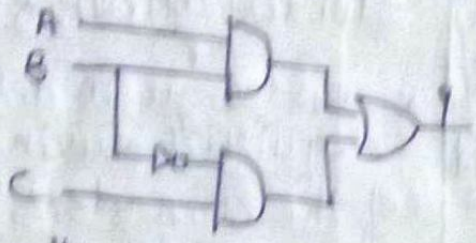
Dynamic :- The o/p changes three or more times when it change from 1 to 0 or 0 to 1



Essential Hazard :

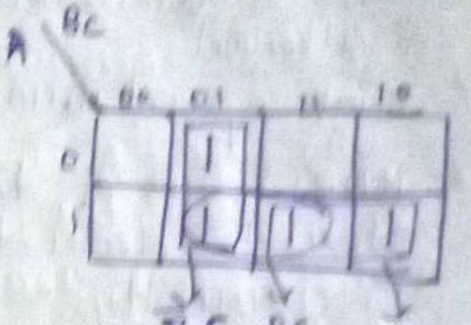
This occurs in asynchronous ckt, due to the unequal delays along 2 or more path.

Example



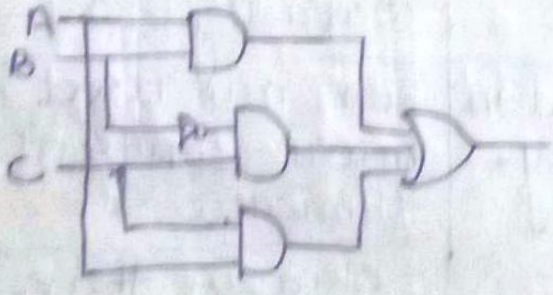
$Y = x_1 x_2 + \bar{x}_1 x_3$

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



for removing hazard we need 1 more pair

∴ hazard free logic of above ckt



RAM	ROM
* RAM is volatile memory.	ROM are non-volatile.
* Data stored in RAM can be retrieved & altered	Data in ROM can only be read.
* High speed memory	slower than RAM.
* RAM is used to store the data that is currently processing	* It store instructions required during bootstrap of the computer.
CPU can access the data stored in RAM	CPU cannot access the data in ROM
Large size with higher capacity	Small size with less capacity.
RAM are used in primary memory, CPU cache.	ROM are used in micro-controllers, firmware
costlier.	cheaper than RAM