UNIT- III

G. Suman

①

llabus :- Digital Electronics

Logic gates, Simple Combinational Circuits – Half and ull Adders, BCD Adder, Latches and Flip Flops (S-R, J-K, T and D), Shift Registers and Counters.

Logic Gates :-

Digital Logic gate is an electronic component which results an output after implementing logic on its input signal. These serve as basic building blocks of any digital system irrespective of its Complexity.

Digital Logic Gates are catagorized into

1) AND Gate 2) OR Gate 3) NOT Gate 4) NAND Gate 5) NOR Gate 6) Ex-OR Gate 7) Ex-NOR Gate.

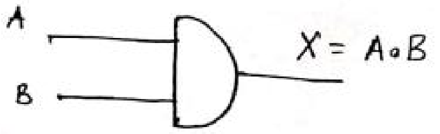
Among these first three (AND, OR, NOT) Gates are called as basic logic gates and also termed as AOI Logic

The Next two gates i.e, NAND and NOR gates are called as Universal Logic gates since by using these two gates we can realize any logic gates.

The Ex-OR and Ex-NOR gates are other special logic gates which can be used as Odd function and Even function respectively.

AND Gate :-

In AND Gate, when two inputs are active high, then the output is Active high, otherwise output is active Low. The IC Number for AND gate is IC7408. The Logic symbol and truth table for AND gate is shown below.

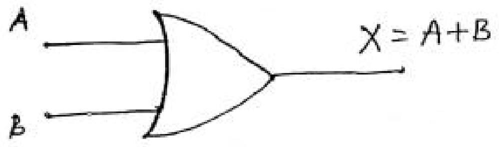


Symbol.

A	B	X = A.B
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate :-

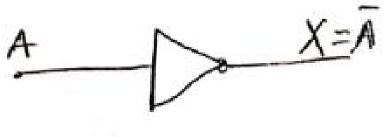
In OR gate, when two inputs are active low then the Output is active low, otherwise output is active high. The IC number for OR gate is IC7432. The Logic symbol and truth table for OR gate is as shown below.



A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate :-

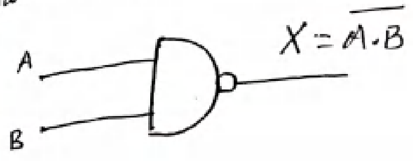
In NOT gate, when input is high then output is low and vice versa. The IC number for NOT gate is IC7404. The Logic symbol and truth table for NOT gate is as shown below.



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

ND

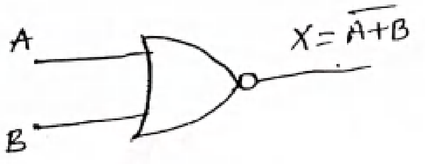
The inverse operation of AND gate is called NAND. "AND + NOT = NAND". In NAND gate, when two inputs are high then output is active low. Otherwise, output is high. The IC number for NAND gate is IC 7400. Logic symbol and truth table for the NAND gate is as shown below.



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

5. NOR Gate :-

The inverse operation of OR gate is called NOR Gate. "OR + NOT = NOR". In NOR Gate, When two inputs are active low, output is active high. Otherwise, the output is active low. The IC Number for NOR gate is IC 7402. The Logic symbol and truth table for the NOR gate is as shown below.

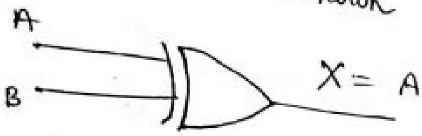


A	B	$X = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

6. Ex-OR gate :-

This gate can be used for representing Odd Function. In Ex-OR gate, when two inputs are different then output is active high, other wise output is active low. This gate also can be used to represent "Not equal to Compare" Function. The Logic Symbol and truth table for Ex-OR

~ shown below.



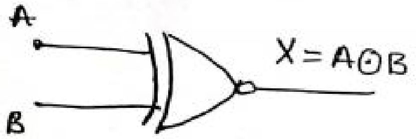
A	B	X = A⊗B
0	0	0
0	1	1
1	0	1
1	1	0

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

The IC number for Ex-OR gate is IC7486.

7. Ex-NOR Gate :-

This gate can be used to represent the Even function. In Ex-NOR gate, when two inputs are same output is active high otherwise output is active Low. This gate can also be used to represent " Equal to Compare " function. The IC number for Ex-Nor gate is IC 7486. The Logic symbol and truth table for the Ex-Nor gate is as shown below.



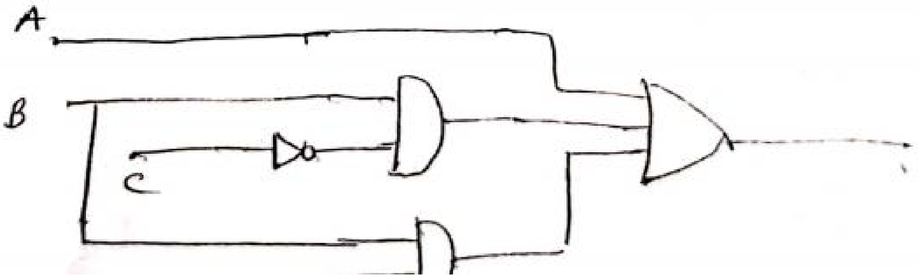
A	B	X=A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

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

* Some Boolean Expressions :-

All Logic gates can be represented with mathematical expressions. Those expressions are called Boolean expressions.

For Example :- Let us consider $A + B.\bar{C} + BD$.



The expressions can be minimized by using some ~~Boolean~~ Boolean rules and Karnaugh Maps.

Boolean Rules are :-

i) Commutative law :-

$A \cdot B = B \cdot A$

$A + B = B + A$

ii) Associative Law :

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

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

iii) Distributive Law :-

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

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

iv) AND Law :-

$A \cdot 0 = 0$

$A \cdot 1 = A$

$A \cdot A = A$

$A \cdot \overline{A} = 0$

v) OR Law :-

$A + 0 = A$

$A + 1 = 1$

$A + A = A$

$A + \overline{A} = 1$

vi) Inversion Law :-

$\overline{\overline{A}} = A$

vii) Demorgan's Law :-

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

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

Various Karnaugh's Maps are :-

For single variable :-

$\overline{A}$	0
A	1

For Two variable (A, B)

	$\overline{B}$	B
$\overline{A}$	0	1
A	2	3

For three variable (A, B, C) :-

	$\overline{B}\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	0	1	3	2
				6

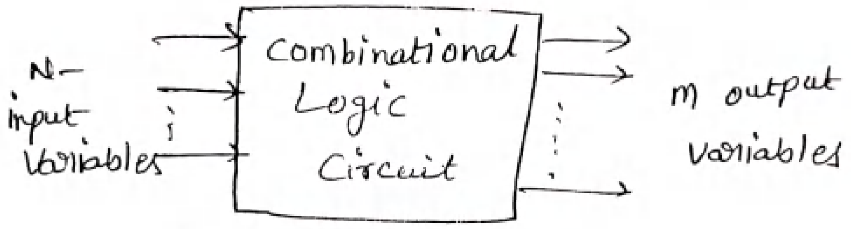
For Four Variabe :-
(A, B, C, D)

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	3	2
$\overline{A}B$	4	5	7	6
AB	12	13	15	14
$A\overline{B}$	8	9	11	10

Combinational Circuits :-

When Logic gates are connected together to produce specified output for specified combinations of input variables, with no storage involved, the resulting circuit is called " Combinational Logic ".

It contains input variables, Logic gates and output variables. Outputs depends on present input combinations only.



Design procedure for Combinational Circuit :-

1) Definition of Statement

2) Identifying the input and o/p variables

3) Representation of i/p and o/p variable with symbols.

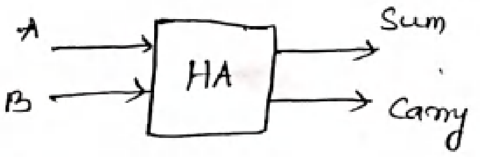
4) Construct the truth table for the given input statement.

5) Simplify the boolean function upto Least equation.

6) Draw the Combinational Circuit.

Half - Adder :-

Half Adder adds the two input variables at a time. The input variables for half adder is A, B. Outputs are Sum(S), Carry (C).



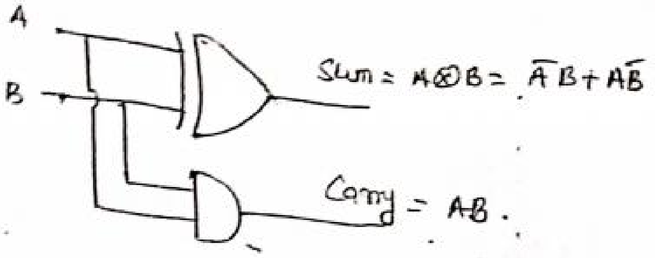
Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum :-

A \ B	$\bar{B}$	B
$\bar{A}$	$_0$	1_1
A	1_2	$_3$

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

	$\bar{B}$	B
$\bar{A}$	$_0$	$_1$
A	$_2$	1_3



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

$Carry = AB$.

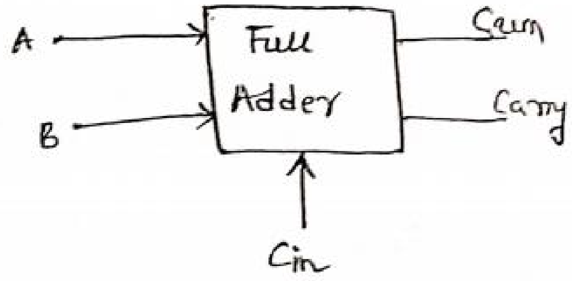
ll Adder :-

Full Adder adds three bits at a time and produces a sum and carry.

* Inputs are A, B & C_{in}

* Outputs are Sum(S) & Carry (C).

input			output	
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



Sum :-

	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$
$\bar{A}$	$_0$	1_1	$_3$	1_2
A	1_4	$_5$	1_7	$_6$

$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

$= \bar{A}(\bar{B}C_{in} + B\bar{C}_{in}) + A(\bar{B}\bar{C}_{in} + BC_{in})$

$= \bar{A}(B \otimes C_{in}) + A(B \odot C_{in})$

$= \bar{A}(B \otimes C_{in}) + A(\overline{B \otimes C_{in}})$

Let $B \otimes C_{in} = X$

$= \bar{A}X + A\bar{X}$

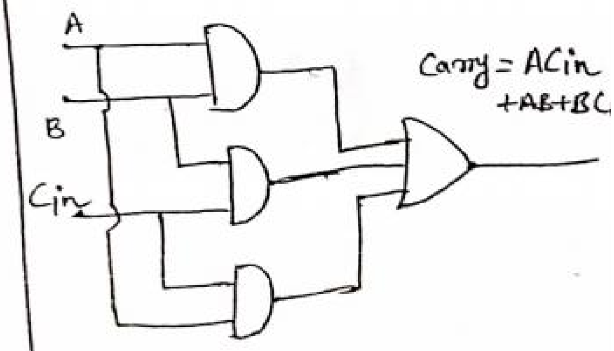
$= A \otimes X = A \otimes B \otimes C$

$\therefore Sum = A \otimes B \otimes C$

Carry :-

	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$
$\bar{A}$	$_0$	$_1$	1_3	$_2$
A	$_4$	1_5	1_7	1_6

$Carry = AC_{in} + AB + BC_{in}$

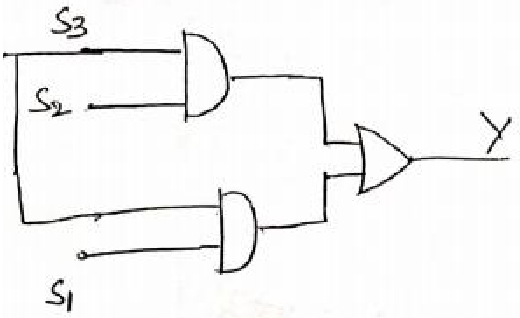


greater (or) less than 9.

y :-

	$\bar{S_1}\bar{S_0}$	$\bar{S_1}S_0$	S_1S_0	$S_1\bar{S_0}$
$\bar{S_3}\bar{S_2}$	0	1	3	2
$\bar{S_3}S_2$	4	5	7	6
S_3S_2	1 12	1 13	1 15	1 14
$S_3\bar{S_2}$	8	9	1 11	1 10

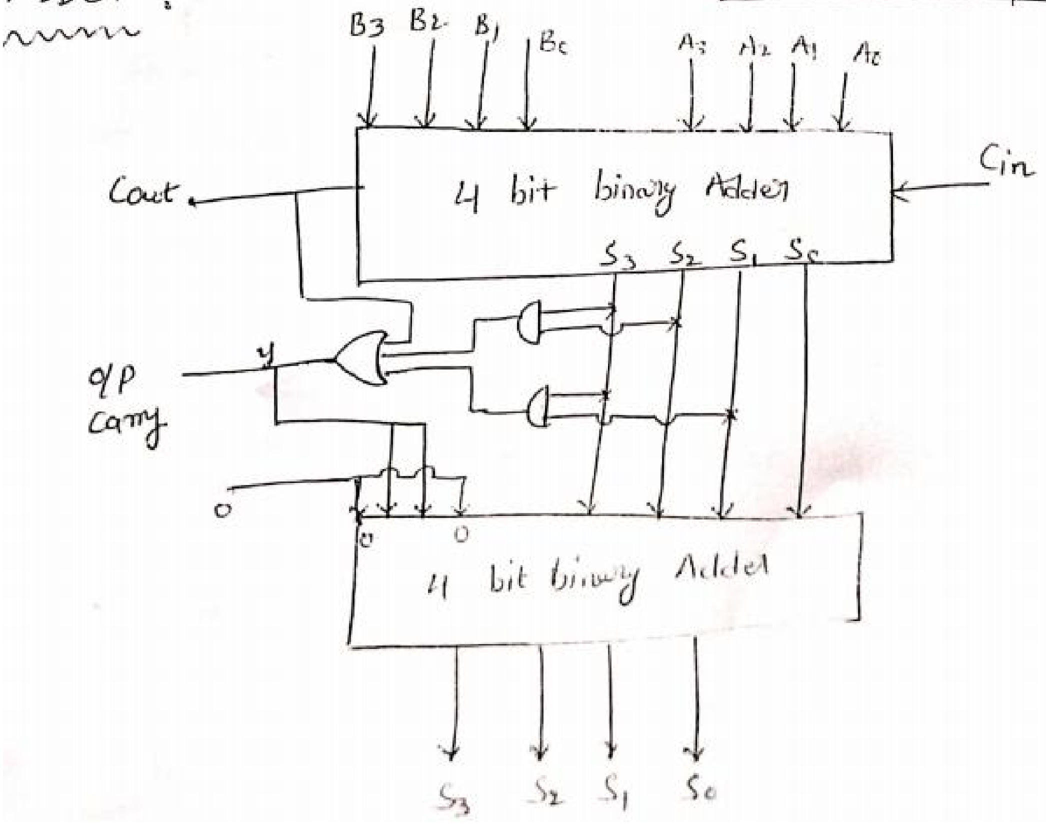
$$y = S_3S_2 + S_3S_1$$



Inputs				Output
S_3	S_2	S_1	S_0	y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

- invalid states

Block Diagram of BCD ADDER :-



output 'y' becomes 1 when sum > 9. that y will adds (6) to the BCD sum to correct it.

When sum greater than 9, it adds 0110 in 2nd binary adder.

Ex:- 6 + 8

6 → 0110

8 → 1001

1110 (14) Invalid

+6 0110

0001 0100

1 4

2:- When Cout = 1 and sum lessthan 9. Then also becomes 1 so, the 2nd binary number adds 0110 to the result :-

Ex:- 8 + 9

8 → 1000

9 → 1001

carry 1 0001

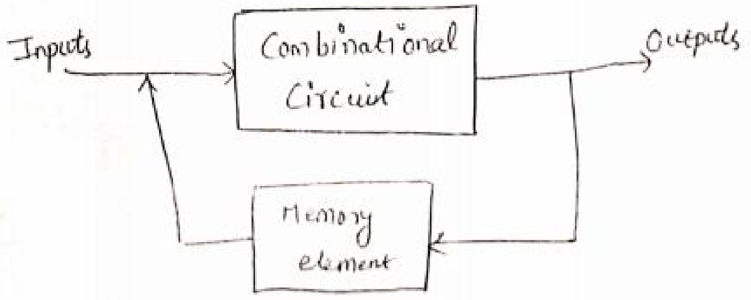
0110

1 0111

1 7

Sequential Circuits :-

* The Combination of Combinational Circuit and memory element is called " Sequential Circuits".
* It consists of Logic gates and memory elements.
* In sequential circuits, Output depends on present inputs and past outputs. The past outputs are provided by memory element connected in feed back path.



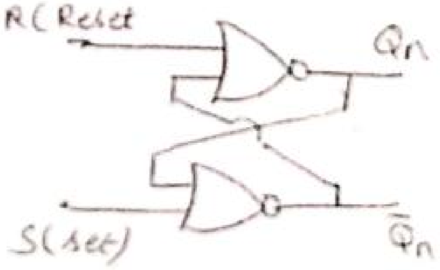
Difference between sequential and Combinational Circuit :-

Combinational Circuit	Sequential Circuit
* Output depends on present inputs only	* Output depends on present i/p and past outputs.
* No memory element	* has memory element.
* No feedback	* Feedback is present.
* Not Complex circuit	* Complex Circuit
* Easy to design	* Difficult to design

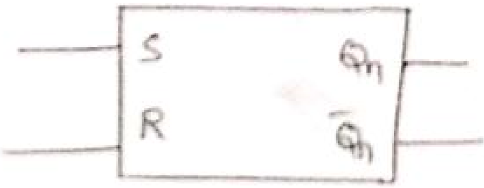
Latches & Flip Flops :-

Latch is a device which stores one bit of memory. Latch does not have any clock signal. The Latches are ~~explain~~ formed by using either NOR gates (or) NAND gates. The Latches are we have S-R Latch, J-K Latch .. etc. Let us see the operation of S-R Latch using NOR gates.

S-R Latch :-

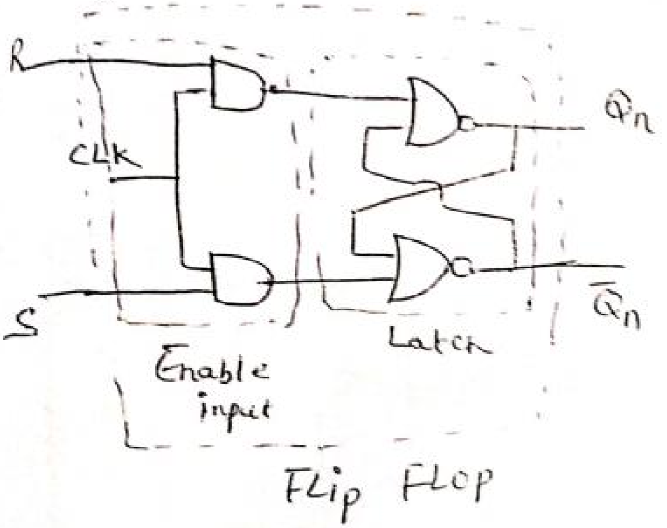


S	R	Q_n	Q_{n+1}	State
0	0	0	0	NC change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	x	Invalid
1	1	1	x	



S	R	Q_{n+1}
0	0	NC
0	1	Reset
1	0	Set
1	1	invalid

give clock signal to a Latch then that is d Flip flop. The enable input can be given using following arrangement. For Nor gate Latch use AND ~~and~~ gate arrangement and for Nand gate Latch use NAND gate arrangement. Let us see the ~~No~~ S-R ~~Latch~~ Flip flop with ~~and~~ AND gate enable input arrangement.



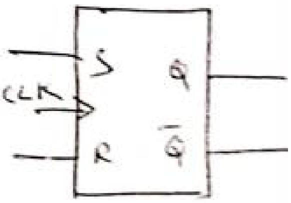
CLK	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	No Change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	X	invalid
↑	1	1	1	x	

Q_{n+1} :-

	$\overline{R}\overline{Q_n}$	$\overline{R}Q_n$	RQ_n	$R\overline{Q_n}$
$\overline{S}$	$_0$	1	$_3$	$_2$
S	1 $_4$	1 $_5$	X $_7$	X $_6$

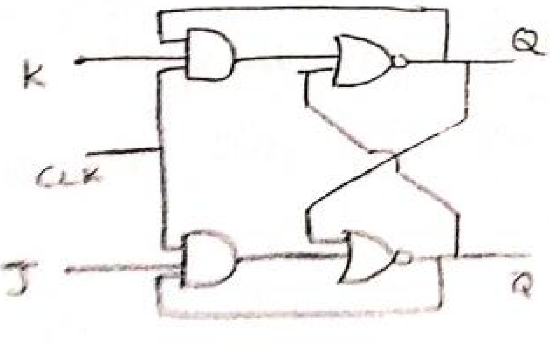
$Q_{n+1} = S + \overline{R}Q_n$

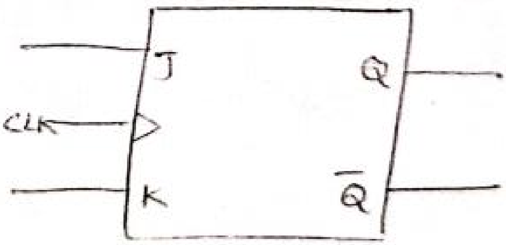
CLK	S	R	Q_{n+1}
↑	0	0	NC
↑	0	1	Reset
↑	1	0	Set
↑	1	1	invalic



J-K Flip Flop :-

If output is fed back as input to the S-R Flip Flop, then that is called J-k flip flop. J-k flip is to eliminate invalid state.





CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0 ~~Reset~~
↑	1	0	1
↑	1	1	$\bar{Q}_n$

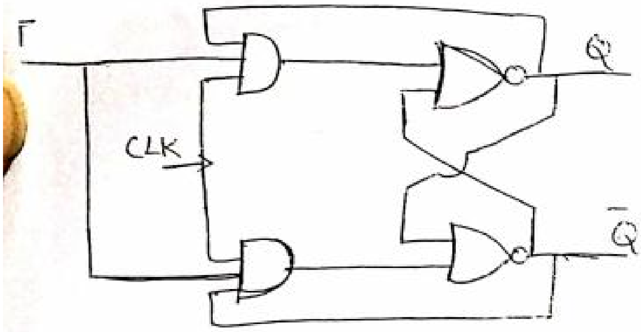
CLK	J	K	Q_n	Q_{n+1}	State
↑	0	0	0	0	No Change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	1	Toggle
↑	1	1	1	0	

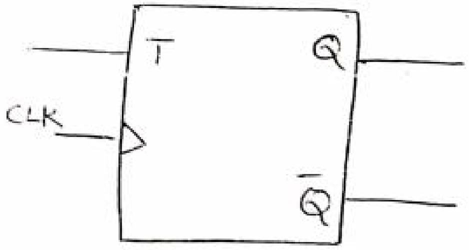
Q_{n+1} :-

	$\bar{K}\bar{Q}_n$	$\bar{K}Q_n$	KQ_n	$K\bar{Q}_n$
$\bar{J}$	$_0$	1_1	$_3$	$_2$
J	1_4	1_5	$_7$	1_6

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

T- FLip FLOP (T- Flip Flop) :-





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

=

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

Q_{n+1} :-

	$\bar{Q}_n$	Q_n
$\bar{T}$	$_0$	1_1
T	1_2	$_3$

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

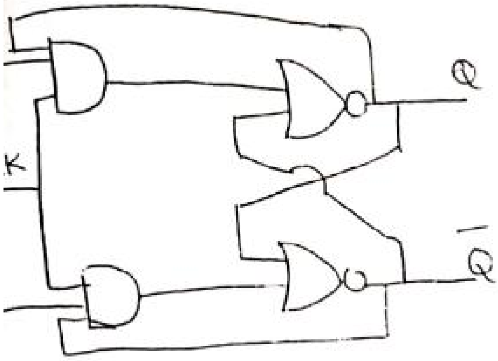
* T- Flip Flop can be obtained from J-K flip flop by joining J & K inputs.

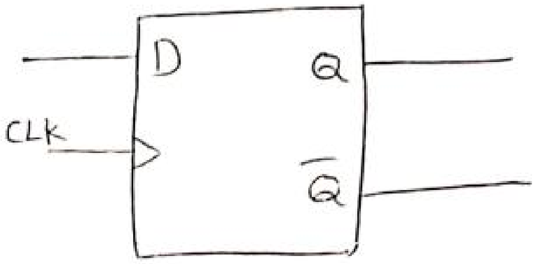
* If $T=0$, No change in output
$T=1$, Output toggles ($\bar{Q}_n$).

Flip Flop :-

Flip Flop can be obtained from J-k Flip Flop by
ering not gate between J & k inputs.

n D Flip Flop if D = 0 , output is reset
D = 1 , output is set



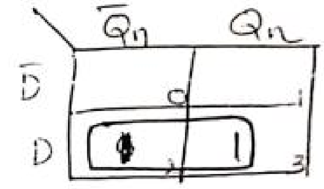


	Qn	Qn+1	
0	0	0	} Reset
0	1	0	
1	0	1	} → Set
1	1	1	

⟹

D	Qn+1
0	0
1	1

Qn+1 :-



$Q_{n+1} = D$

ounters :-

Counter is the one of the application of flip flops.
ounters are used to count the number of events by producing different number of states.

Counters are two types :
1) Synchronous Counters
2) Asynchronous Counters

In Synchronous Counters clock input is given to all flip flops at a time.

In Asynchronous counters clock input is given to first flip flop only. Asynchronous counters ~~als~~ are also called as Ripple Counter.

Differe

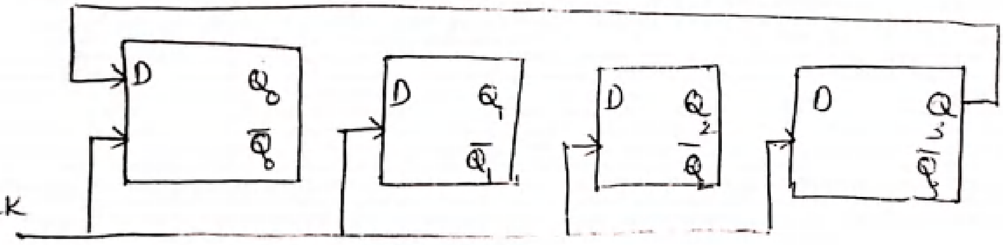
Asynchronus Counter	Synchronus counter
* Clock input is connected to first flip flop only	* Clock is connected to all flip flops.
* Speed is very low	* Speed of operation is high
* Propogation delay exist	* No propogation delay
* It may be unstable	* It may be stable
* Difficult to design	* Easy to design

Ring Counter :-

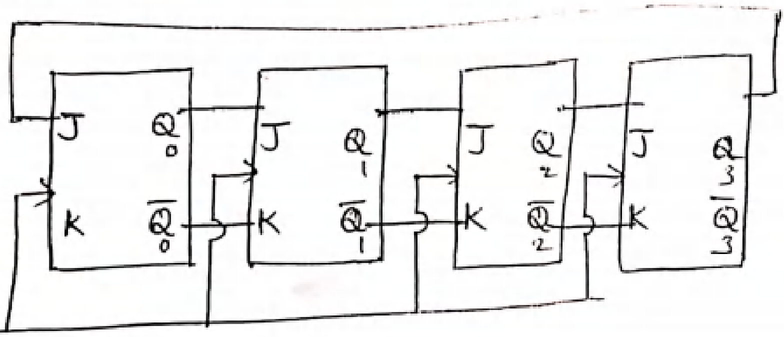
* Ring Counters are used in digital counters to control the execution of instructions in a proper sequence at the appropriate time.

* A ring counter can be designed using by D (or) J-K FlipFlops.

using D- Flip Flop :-



using J-K Flip Flops :-

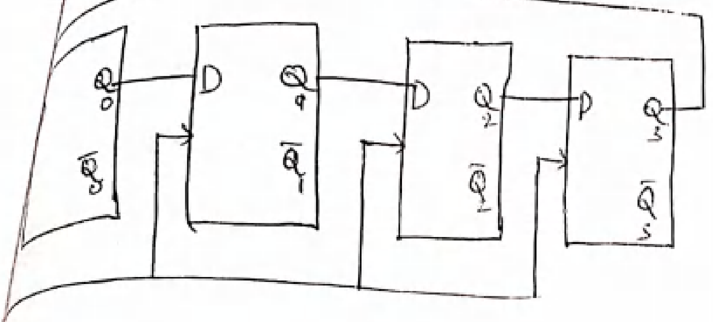


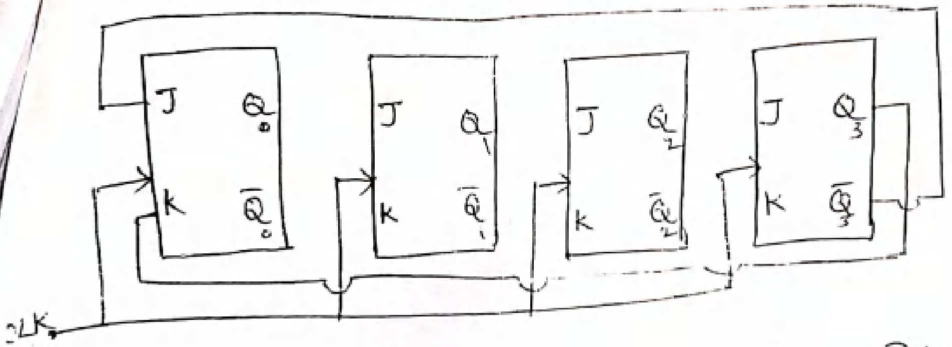
CLK	Q_0	Q_1	Q_2	Q_3
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Another name For Johnson Counter is ...ted Ring Counter.

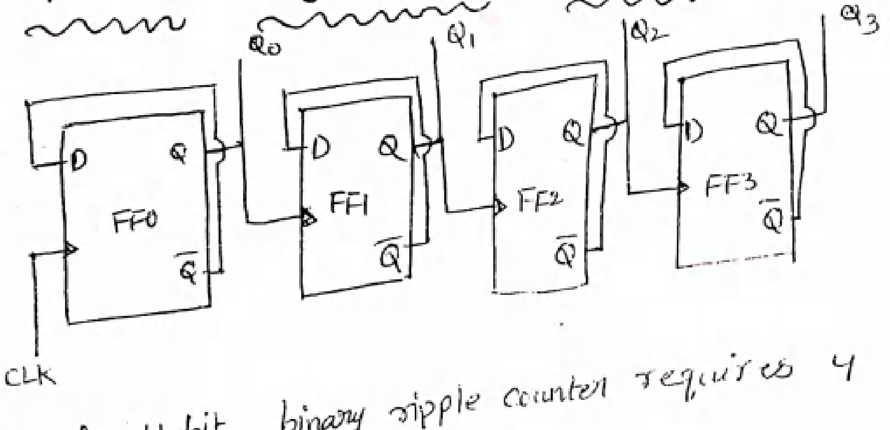
A Johnson counter can be designed using D- Flip Flops (or) J-k Flip Flops.

CLK	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0
9	1	0	0	0





4 - bit Asynchronous Counter (or) Ripple Counter :-



CLK	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	[illegible]

- * A 4-bit binary ripple counter requires 4 D FlipFlops.
- * First Flip Flop generates LSB and Last Flip Flop generates HSB.
- * In Ripple Counter, output of first counter acts as clock pulse to the next Flip Flop.

Shift Registers :-

The group of flip flops can be used to store a word, which is called "register". A flip flop can store 1-bit information. So, an n-bit register has a group of n flip-flops and is capable of storing any binary information.

The binary data in a register can be moved from stage to stage within the register (or) into (or) out of the register upon application of clock pulse. This type of movement (or) shifting of data in registers are called "Shift Registers".

According to data movement in a register, the various types of shift registers are.

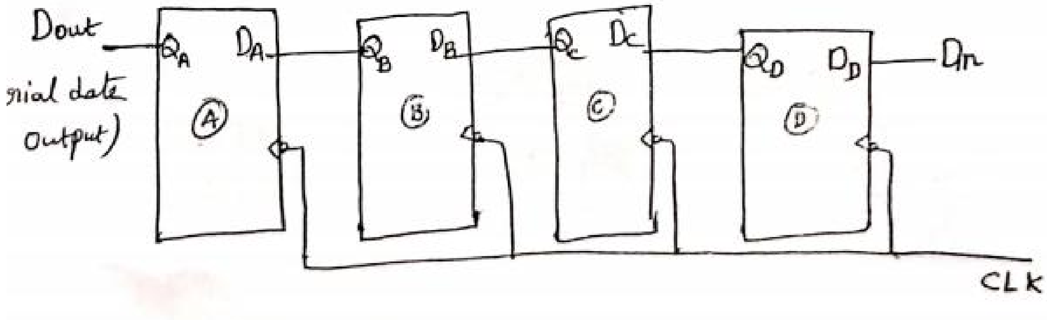
1) Serial in Serial Out Register.
2) Serial in parallel out Register.
3) Parallel in Serial Out Register.
4) Parallel in Parallel out Register.

1) Serial In Serial out Shift Register :-

The serial in and serial out data transmission may be done in two ways : i) Shift Left.
ii) Shift Right.

Shift Left Mode :-

The diagram for serial-in Serial-out shift register.

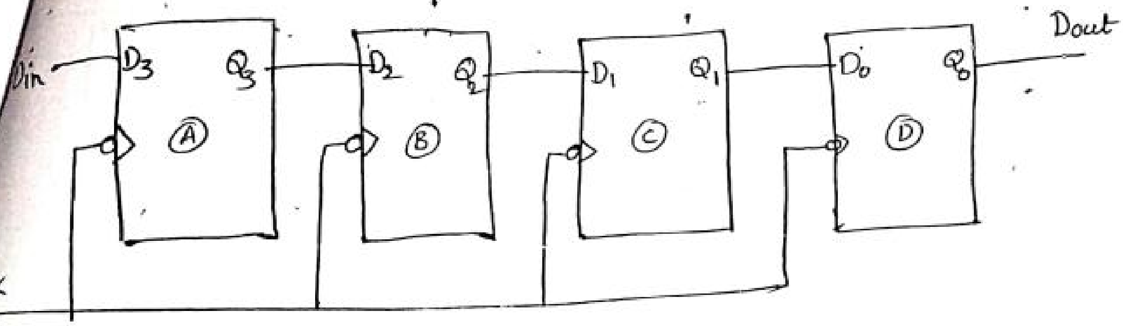


...e
can store
group of
...ary
...ed from
of
...vement

...explained D-flip flops. All shift operations can be

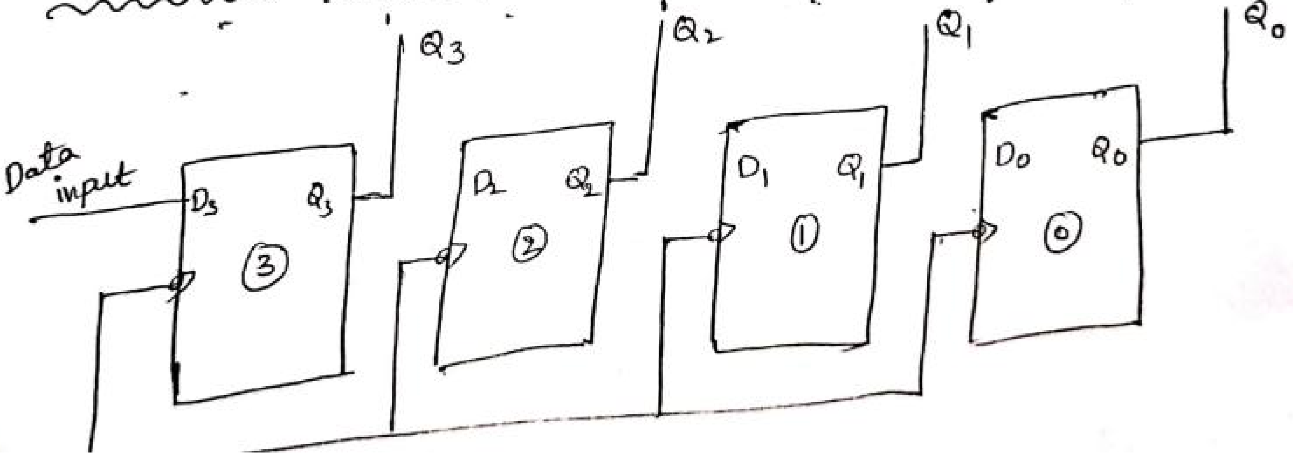
CLK	Q3	Q2	Q1	Q0	Din
initialy	0	0	0	0	1
↑	0	0	0	1	1
↑	0	0	1	1	1
↑	0	1	1	1	1
↑	1	1	1	1	1

Shift Right Mode :-



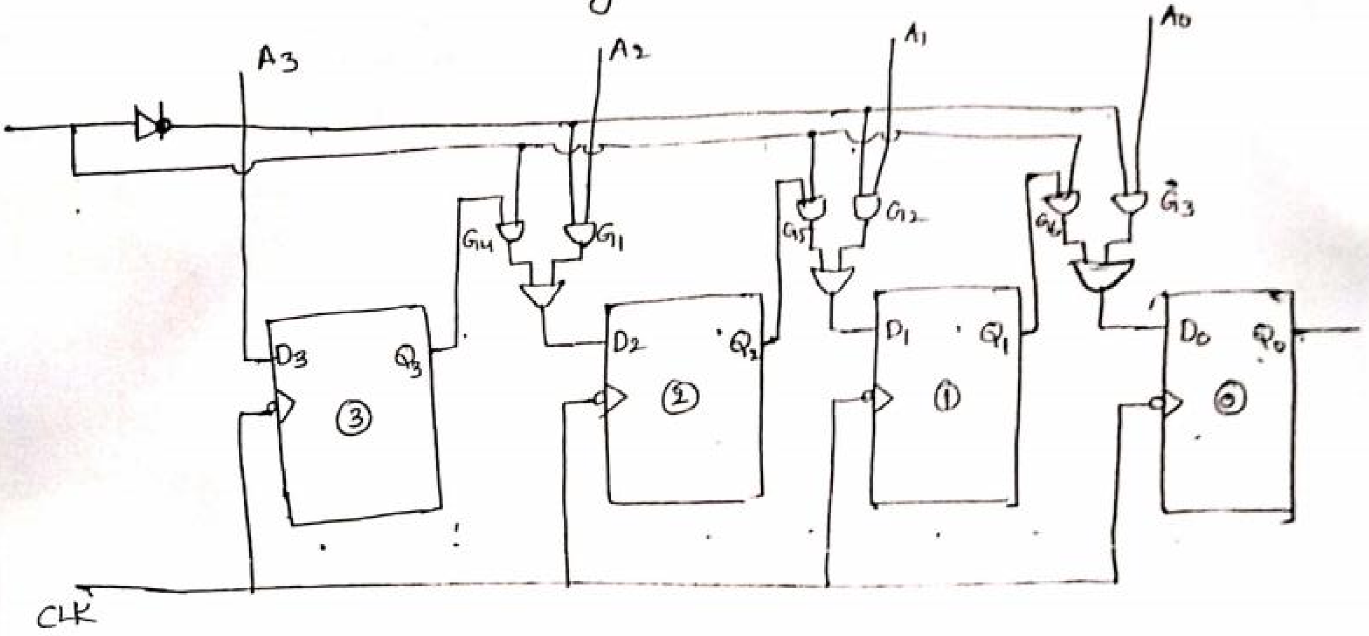
CLK	Din	Q3	Q2	Q1	Q0
initially	1	0	0	0	0
↑	1	1	0	0	0
↑	1	1	1	0	0
↑	1	1	1	1	0
↑	1	1	1	1	1

Serial in Parallel Out (SIPO) Shift Register :-



Parallel In Serial Out (PISO) Shift Register :-

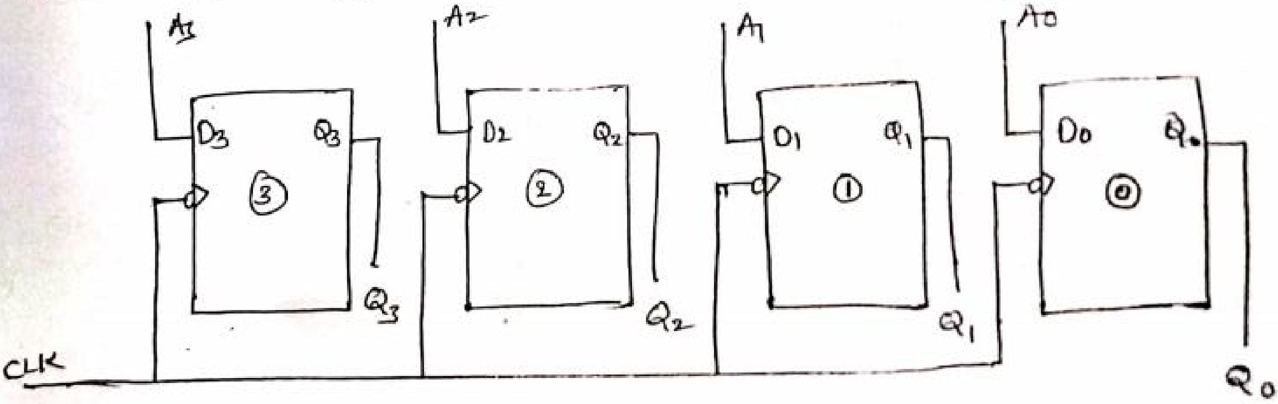
Here There are Four input Lines, A_3, A_2, A_1, A_0 for entering data in parallel into the Register. ~~SH~~ Shift/$\overline{\text{Load}}$ is the control input which allows the data and ~~SH~~ parallely and shift the data serially.



When SHIFF/$\overline{\text{LOAD}}$ is Low parallel input A_3 A_2 A_1 A_0 is applied to flip flops.

When shift/$\overline{\text{Load}}$ is high serial data shift takes place in the Flip Flops.

Parallel In Parallel Out (PIPO) Shift Register :-



In this type of shift registor, input will given to all flip flops at a time i.e, parallel and output also recive parallel (i.e all outputs at a time.)