Chapter-1

DIGITAL LOGIC CIRCUITS

Chapter-1

Digital Logic Circuits

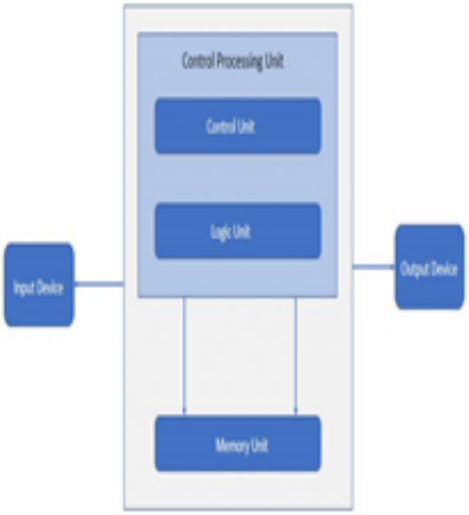
1 What is Computer Architecture

Computer architecture is a specification describing how hardware and software technologies interact to create a computer platform or system

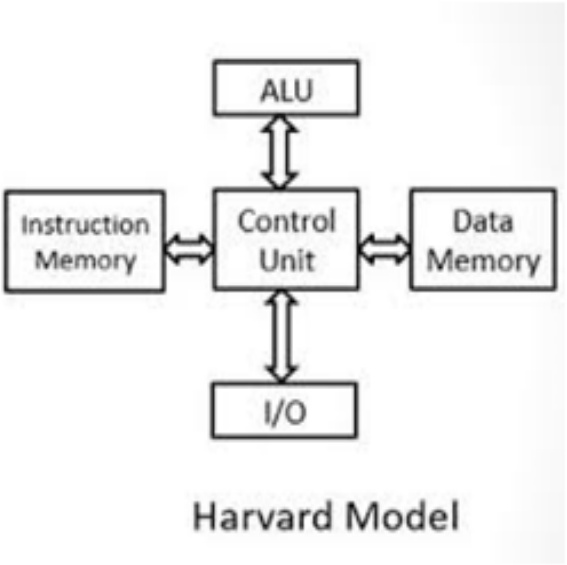
Interconnectivity for a computer's hardware components as well as the mode of data transfer and processing exhibited.

Conceptual design and fundamental operational structure of a computer system

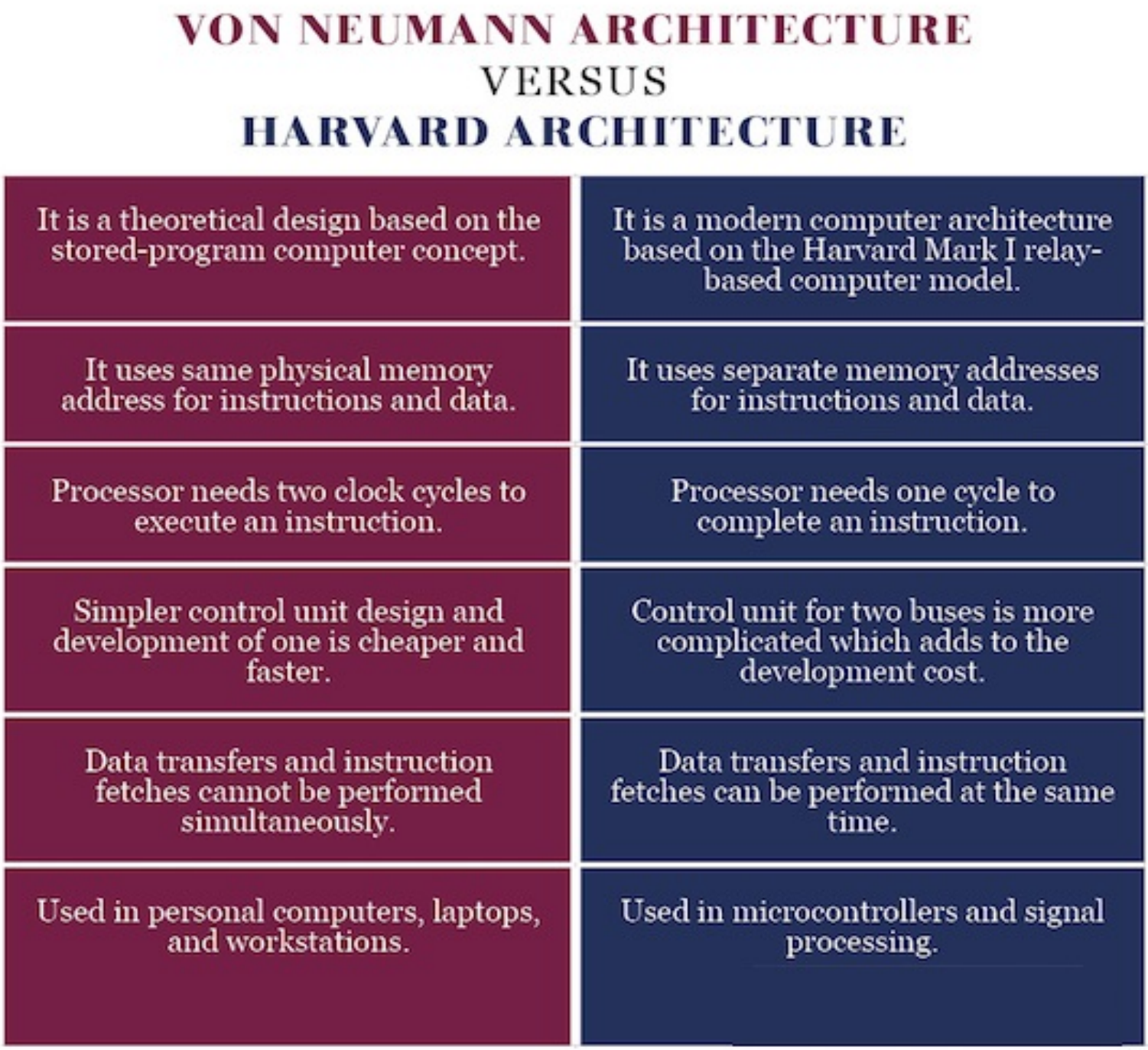
2.Types of computer Architecture



Von-Neumann Architecture



Harvard Architecture



3. Digital logic gates

A logic gate performs a logical operation on one or more logic inputs and produce a single logic output.

➢ Boolean Postulates:

- The fundamental laws of Boolean algebra are called as the postulates of Boolean algebra.
- These postulates for Boolean algebra originate from the three basic logic functions AND, OR and NOT.
- **Properties of 0 and 1:**

I. If $X \neq 0$ then $X = 1$, and If $X \neq 1$ then $X = 0$

II. OR relation (Logical Addition)

a. $0 + 0 = 0$ c. $1 + 0 = 1$

b. $0 + 1 = 1$ d. $1 + 1 = 1$

III. AND relation (Logical Multiplication)

a. $0 . 0 = 0$ c. $1 . 0 = 0$

b. $0 . 1 = 0$ d. $1 . 1 = 1$

IV. Complement Rules

a. $\bar{0} = 1$ b. $\bar{1} = 0$

➢ Principle of Duality Theorem:

- This is very important principle used in Boolean algebra.
- Principle of Duality states that;
  - Changing each OR sign (+) to an AND sign (.)
  - Changing each AND sign (.) to an OR sign (+)
  - Replacing each 0 by 1 and each 1 by 0.
- The derived relation using duality principle is called dual of original expression.
- Example: Take postulate II, related to logical addition:

1) $0 + 0 = 0$ 2) $0 + 1 = 1$ 3) $1 + 0 = 1$ 4) $1 + 1 = 1$

2. Now working according to above relations, + is changed to . and 0's replaced by 1's

a) $1 . 1 = 1$ b) $1 . 0 = 0$ c) $0 . 1 = 0$ d) $0 . 0 = 0$

➢ Boolean Theorems:

- Boolean Theorem can be proved by substituting all possible values of the variable that are 0 and 1.
- This technique of proving theorem is called **Proof by perfect induction.**

Sl No	Theorem	Sl No	Theorem
Properties of 0 and 1		**Associative Law**	
1	$0 + X = X$	12	$X .(Y.Z) = (X.Y).Z$
2	$1 + X = 1$	13	$(X+Y).Z = X+(Y.Z)$
3	$0 . X = 0$	**Distributive Law**	
4	$1 . X = X$	14	$X.(Y+Z) = X.Y + X.Z$
Indempotence Law		15	$X+Y.Z = (X+Y).(X+Z)$
5	$X + X = X$	16	$X + XY = X$
6	$X . X = X$	17	$X(X+Y) = X$
Complementary Law		18	$XY + X\overline{Y} = X$
7	$X + \overline{X} = 1$	19	$(X+Y)(X+\overline{Y}) = X$
8	$X . \overline{X} = 0$	20	$X+\overline{X}Y = X+Y$
Involution Law		21	$X(\overline{X}+Y) = XY$
9	$\overline{\overline{X}} = X$		
Commutative Law			
10	$X + Y = Y + X$		
11	$X . Y = Y . X$		

➢ **Theorem 1: 0 + X = X**

Proof: If X = 0	Proof: If X = 1	Using Truth Table
then LHS = 0 + X = 0 + 0 = 0 = RHS	then LHS = 0 + X = 0 + 1 = 1 = RHS	

0	X	0+X
0	0	0
0	1	1

➢ **Indempotence Law:** "This law states that when a variable is combines with itself using OR or AND operator, the output is the same variable".

➢ **Theorem 5: X + X = X**

Proof: If X = 0	Proof: If X = 1	Using Truth Table
then LHS = X + X = 0 + 0 = 0 = RHS	then LHS = X + X = 1 + 1 = 1 = RHS	

X	X	X+X
0	0	0
1	1	1

- **Complementary Law:** "This law states that when a variable is And ed with its complement is equal to 0 and a variable is OR ed with its complement is equal to 1".

Theorem 7: $X + \overline{X} = 1$

Proof: If X = 0	Proof: If X = 1	Using Truth Table
then LHS $= X + \overline{X}$ $= 0 + 1$ $= 1$ $= RHS$	then LHS $= X + \overline{X}$ $= 1 + 0$ $= 1$ $= RHS$	

X	$\overline{X}$	$X + \overline{X}$
0	1	1
1	0	1

- **Involution Law:** "This law states that when a variable is inverted twice is equal to the original variable".
- **Theorem 9:** $\overline{\overline{X}} = X$

Proof: If X = 0, then $\overline{X} = 1$ Take complement again, then $\overline{\overline{X}} = 0$ i.e. X If X = 1, then $\overline{X} = 0$ Take complement again, then $\overline{\overline{X}} = 1$ i.e. X	Using Truth Table

X	$\overline{X}$	$\overline{\overline{X}}$
0	1	0
1	0	1

- **Commutative Law:** "This law states that the order in which two variable are Or ed or AND ed make no difference".
- **Theorem 10:** $X + Y = Y + X$

Proof: If Y = 0	Proof: If Y = 1	Using Truth Table
then LHS $= X + Y$ $= X + 0$ $= X$ RHS $= Y + X$ $= 0 + X$ $= X$ Therefore LHS = RHS	then LHS $= X + Y$ $= X + 1$ $= 1$ RHS $= Y + X$ $= 1 + X$ $= 1$ Therefore LHS = RHS	

X	Y	X+Y	Y+X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

➢ DeMorgan's Theorem:

- **DeMorgan's First Theorem:**
  - **Statement: "When the OR sum of two variables is inverted, this is same as inverting each variable individually and then AND ing these inverted variables**"
  - This can be written as $\overline{X + Y} = \overline{X}.\ \overline{Y}$
  - We can prove the DeMorgan's First theorem by using Truth Table is

X	Y	$\overline{X}$	$\overline{Y}$	X+Y	$\overline{X + Y}$	$\overline{X}.\ \overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

  - Compare the column $\overline{X + Y}$ and $\overline{X}.\ \overline{Y}$. Both of these are identical. Hence the DeMorgan's first theorem is proved.
- **DeMorgan's Second Theorem:**
  - **Statement: "When the AND product of two variables is inverted, this is same as inverting each variable individually and then OR ing these inverted variables**"
  - This can be written as $\overline{X.Y} = \overline{X}+\overline{Y}$
  - We can prove the DeMorgan's Second theorem by using Truth Table is:

X	Y	$\overline{X}$	$\overline{Y}$	X.Y	$\overline{X.Y}$	$\overline{X}+\overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

  - Compare the column $\overline{X.Y}$ and $\overline{X}+\overline{Y}$. Both of these are identical. Hence the DeMorgan's Second theorem is proved.
- **Application of DeMorgan's Theorem:**
  - It is used in simplification of Boolean expression.
  - DeMorgan's law commonly apply to text searching using Boolean operators AND, OR and NOT.
  - It is useful in the implementation of the basic gates operations with alternative gates.

➢ Minterm:

- Minterm is a product of all the literal (with or without bar) within the logic system.
- **(OR)** A single variable or the logical product of several variables. The variables may or may not be complemented.
- A variable may appear either in its normal form (X) or in its complement form ($\overline{X}$)
- If a variable **value is 0 then its complemented** otherwise it is in its normal form.
- For example, if you have two variables X & Y, there are four possible combination can be formed with AND operation. Each of these four AND operations represents one of the Boolean expressions terms and is called a Minterm or a standard product.

X	Y	Minterm	Designation
0	0	$\overline{X}\,\overline{Y}$	m_0
0	1	$\overline{X}\,Y$	m_1
1	0	$X\,\overline{Y}$	m_2
1	1	$X\,Y$	m_3

- A symbol for each Minterm is also shown in the table and is of the form mj where j denotes the decimal equivalent of the binary number of the Minterm designated.

- For example, the Minterm $\mathbf{X\,Y\,\overline{Z}}$ whose combination is **1 1 0** can be written as $\mathbf{m_6}$ as decimal equivalent of **1 1 0 is 6.**

$$\mathbf{f(X, Y, Z) = \overline{X}\,Y\,\overline{Z} + \overline{X}\,Y\,Z + X\,Y\,\overline{Z} = m_2 + m_3 + m_6}$$

- The above Boolean function is the sum of three product terms. This type of expression is known as Sum of Product (SOP) expression.

$$\mathbf{f(X, Y, Z) = \sum (2, 3, 6)}$$

- Where f is a Boolean function with three variables (X, Y, Z) and it can be read as function f is sum of 2^{nd}, 3^{rd}, and 6^{th} Minterm.
- ***Sum of Product (SOP): A Sum of product expression is a product term or several product terms logically added.***

➢ Maxterm:

- Maxterm is a sum of all the literal (with or without bar) within the logic system.
- **(OR)** A single variable or the logical sum of several variables. The variables may or may not be complemented.
- A variable may appear either in its normal form (X) or in its complement form ($\bar{X}$)
- If a variable **value is 1 then its complemented** otherwise it is in its normal form.
- For example, if you have two variables X & Y, there are four possible combination can be formed with OR operation. Each of these four OR operations represents one of the Boolean expressions terms and is called a Minterm or a standard product.

X	Y	Minterm	Designation
0	0	$X + Y$	M_0
0	1	$X + \bar{Y}$	M_1
1	0	$\bar{X} + Y$	M_2
1	1	$\bar{X} + \bar{Y}$	M_3

- A symbol for each Maxterm is also shown in the table and is of the form Mj where j denotes the decimal equivalent of the binary number of the Maxterm designated.

- For example, the Maxterm $\mathbf{X + Y + \bar{Z}}$ whose combination is **0 0 1** can be written as $\mathbf{M_1}$ as decimal equivalent of **0 0 1 is 1.**

$$\mathbf{f(X, Y, Z) = (X + Y + Z)(X + Y + \bar{Z})(\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z})(\bar{X} + \bar{Y} + \bar{Z}) = M_0.M_1.M_4.M_5.M_7}$$

- The above Boolean function is the product of three sum terms. This type of expression is known as Product of Sum (POS) expression.

$$\mathbf{f(X, Y, Z) = \pi (0, 1, 4, 5, 7)}$$

- Where f is a Boolean function with three variables (X, Y, Z) and it can be read as function f is product of 0^{th}, 1^{st}, 4^{th}, 5^{th} and 7^{th} Maxterm.
- ***Product of Sum (POS): A product of sum expression is a sum term or several sum terms logically multiplied.***

➢ Karnaugh Map:

- A graphical display of the fundamental products in a truth table.
- Fundamental Product: The logical product of variables and complements that produces a high output for a given input condition.
- The map method provides simple procedure for minimizing the Boolean function.
- The map method was first proposed by E.W. Veitch in 1952 known as "**Veitch Diagram**".
- In 1953, Maurice Karnaugh proposed "**Karnaugh Map**" also known as "**K-Map**".

➢ Construction of K-Map:

- The K-Map is a pictorial representation of a truth table made up of squares.
- Each square represents a Minterm or Maxterm.
- A K-Map for **n variables** is made up of **2^n squares**.

➢ Single Variable K-Map:

- A one variable K-Map is shown in the following figure.
- The one variable Boolean expression is of the form f(A).
- There are two Minterms (A and $\overline{A}$) for one variable.
  - Hence the map consists of 2 squares (i.e. **2^n square, 2^1 = 2 square)**

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

Minterm

A	0	1
	0	1

Basic Labelling

A		
	m_0	m_1

  - In one variable K-map:
    - One square represents one Minterm.
    - Two adjacent squares represents a function which is always true i.e. f (A) = 1.

➢ Two Variable K-Map:

- The two variable Boolean expressions are of the form f (A, B).
- There are four Minterms ($\overline{A}\,\overline{B}$, $\overline{A}B$, $A\overline{B}$ and AB for two variable.
- Hence the map consists of 4 squares (i.e. **2^n square, 2^2 = 4 square)**

	$\overline{B}$	B
$\overline{A}$	$\overline{A}\,\overline{B}$	$\overline{A}B$
A	$A\overline{B}$	AB

Minterm

	0	1
0	00	01
1	10	11

Basic Labelling

$\overline{A}$	m_0	m_1
A	m_2	m_3

➢ Three Variable K-Map:

- The three variable Boolean expressions are of the form f (A, B, C).
- There are eight Minterms ($\bar{A}\bar{B}\bar{C}$, $\bar{A}\bar{B}C$, $\bar{A}B\bar{C}$, $\bar{A}BC$, $A\bar{B}\bar{C}$, $A\bar{B}C$, $\bar{A}B\bar{C}$, and ABC) for three variable.
- Hence the map consists of 8 squares (i.e. **2^n square, 2^3 = 8 square)**

		$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
A	$\bar{A}$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}BC$	$\bar{A}B\bar{C}$
	A	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$\bar{A}B\bar{C}$

Minterms

		00	01	11	10
A	0	000	001	011	010
	1	100	101	111	110

Basic Labelling

- **Note**: The ordering of variable i.e. 00, 01, 11 & 10 is in gray (reflected binary code) one should not use straight binary code i.e. 00, 01, 10, 11. The straight binary code was used in Veitch diagram but Mr. Karnaugh modified the veitch diagram and use reflected binary code.

➢ Four Variable K-Map:

- The four variable Boolean expressions are of the form f (A, B, C, D).
- There are sixteen Minterms for four variables.

- Hence the map consists of 8 squares (i.e. **2^n square, 2^4 = 16 square).**
- The rows and columns are numbered in a reflected code system.

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

Minterms

	CD				
		00	01	11	10
AB	00	0000	0001	0011	0010
	01	0100	0101	0111	0110
	11	1100	1101	1111	1110
	10	1000	1001	1011	1010

Basic Labelling

LOGIC GATES

GATE: A gate is simply an electronic circuit which operates on one or more input signals & always produces an output signal

Gates are classified into 2 types:

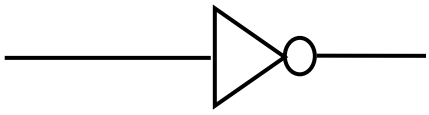
1. Basic Gates: There are 3 basic logic gates:
- NOT gate (inverter)
- OR gate
- AND gate

2. Derived gates: There are 4 derived gates:
- NOR gate
- NAND gate
- XOR gate (Exclusive OR gate)
- XNOR gate (Exclusive NOR gate)

BASIC GATES

1. Inverter (NOT Gate): An inverter is a gate with only 1 input signal & one output signal; the output states is always the opposite of the input state

SYMBOL:

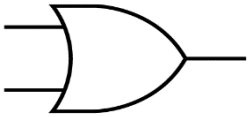


TRUTH TABLE:

X	$\overline{Y}$
0	1
1	0

2. OR Gate: The OR gate has 2 or more input signals but only one output signal. If any of the input signals 1 (high), the output signals is 1 (high)

SYMBOL:

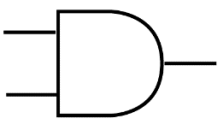


TRUTH TABLE:

X	Y	F= X+Y
0	0	0
0	1	1
1	0	1
1	1	1

3. AND Gate: The AND gate can have 2 or more input signals & produce one output signal. When all the inputs are high then the output is high. Otherwise, the output is low

SYMBOL:



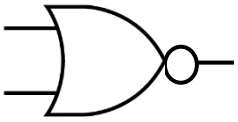
TRUTH TABLE:

X	Y	F= X.Y
0	0	0
0	1	0
1	0	0
1	1	1

DERIVED GATES

1. NOR Gate: The NOR gate has 2 or more input signals but only one output signal. If all the inputs are 0 (low), then the output signal is 1 (high)

SYMBOL:

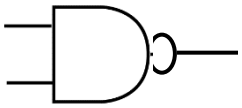


TRUTH TABLE:

X	Y	F= X+Y
0	0	1
0	1	0
1	0	0
1	1	0

2. NAND Gate: The NAND gate has 2 or more input signals but only one output signal. If all of the inputs are 1 (high), then the output produced is 0 (low)

SYMBOL:



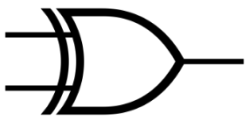
TRUTH TABLE:

X	Y	F= $\overline{XY}$
0	0	1
0	1	1
1	0	1
1	1	0

3. XOR Gate or Exclusive-OR Gate:

- Accepts 2 or more inputs & produces single output
- The output is 0 if there are even numbers 1's in the inputs
- The output is 1 if there are odd number of 1's in the inputs

SYMBOL:



TRUTH TABLE:

No. of 1's Even/Odd	X	Y	Z=A⊕B
Even	0	0	0
Odd	0	1	1
Odd	1	0	1
Even	1	1	0

4. XNOR Gate or Exclusive NOR Gate: An XOR gate is followed by a NOT gate (inverter) becomes XNOR gate. Thus, The XNOR gate is logically equivalent to an inverted XOR gate. Thus XNOR produces 1 (high) as output when the input combination has even number of 1's or when all the inputs are 0's

TRUTH TABLE:

No. of 1's Even/Odd	X	Y	F
Even	0	0	1
Odd	0	1	0
Odd	1	0	0
Even	1	1	1

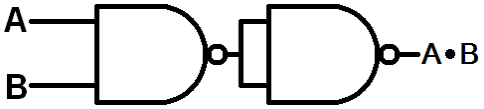
UNIVERSAL GATE

Universal Gate is a gate using which all the basic gates can be designed. NAND & NOR gates are called as the Universal Gates

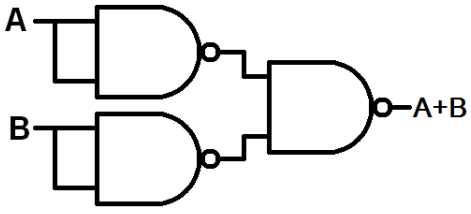
I. Releasing NOT, AND, OR Gate using NAND Gate:

1. NAND TO NOT:

2. NAND TO AND:

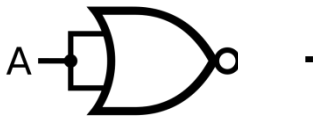


3. NAND TO OR:

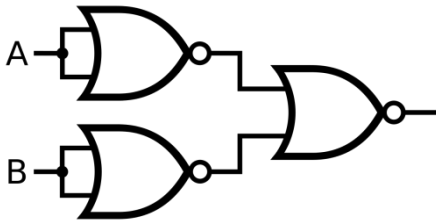


II. Releasing NOT, AND, OR Gate using NOR Gate:

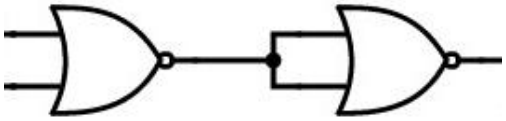
1. NOR TO NOT:



2. NOR TO AND:



3. NOR TO OR:



Combinational circuit

Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer.

Some of the characteristics of combinational circuits are following –

The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.

The combinational circuit do not use any memory.

The previous state of input does not have any effect on the present state of the circuit.

A combinational circuit can have an n number of inputs and m number of outputs.

Block diagram

Half Adder

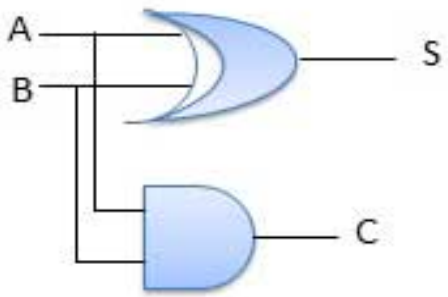
Half adder is a combinational logic circuit with two inputs and two outputs.
The half adder circuit is designed to add two single bit binary number A and B.
It is the basic building block for addition of two single bit numbers.
This circuit has two outputs carry and sum.

Block diagram

Truth table

Inputs		Output
A	B	S C
0	0	0 0
0	1	1 0
1	0	1 0
1	1	0 1

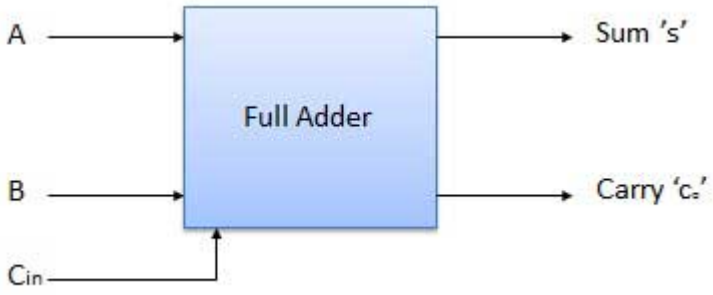
Circuit diagram



Full Adder

Full adder is developed to overcome the drawback of Half Adder circuit.
It can add two one-bit numbers A and B, and carry c.
The full adder is a three input and two output combinational circuit.

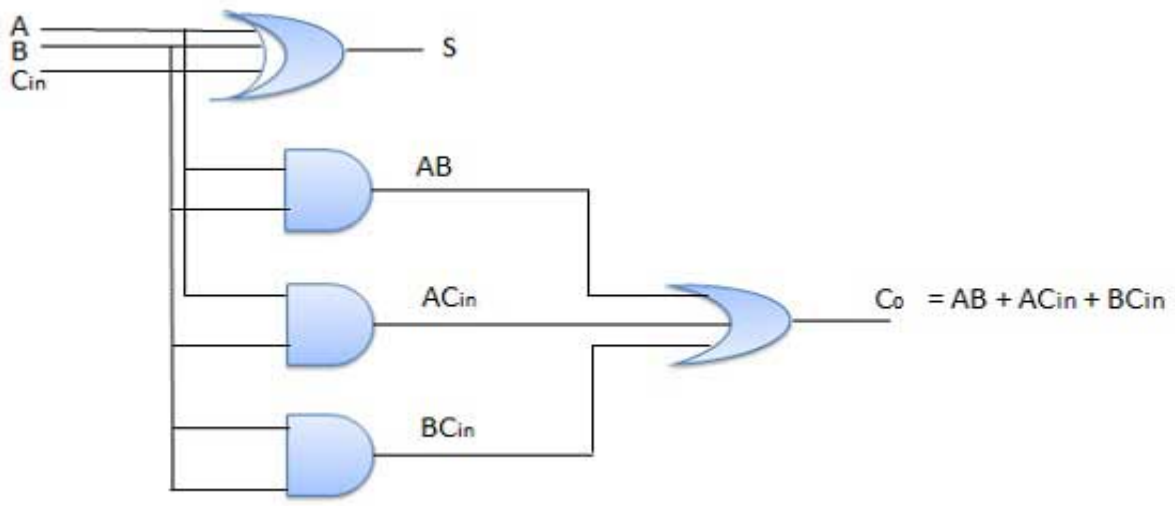
Block diagram



Truth table

Inputs			Output	
A	B	Cin	S	Co
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

Circuit diagram



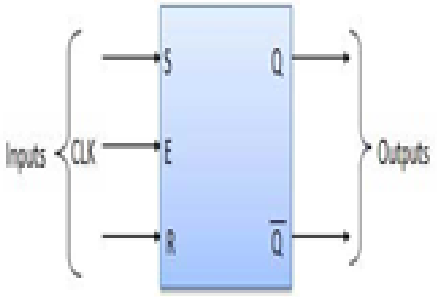
Flip Flop

Flip flop is a sequential circuit which generally samples its inputs and changes its outputs only at particular instants of time and not continuously. Flip flop is said to be edge sensitive or edge triggered rather than being level triggered like latches.

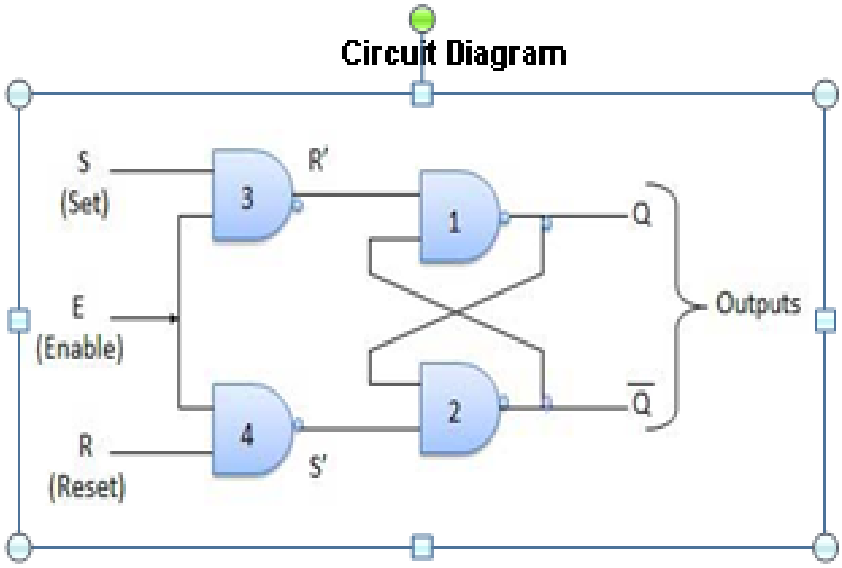
S-R Flip Flop

It is basically S-R latch using NAND gates with an additional **enable** input. It is also called as level triggered SR-FF. For this, circuit in output will take place if and only if the enable input (E) is made active. In short this circuit will operate as an S-R latch if E = 1 but there is no change in the output if E = 0.

Block Diagram



Circuit Diagram



Truth Table

Inputs			Outputs		Comments
E	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	0	Q_n	$\overline{Q}_n$	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	x	x	Indeterminate

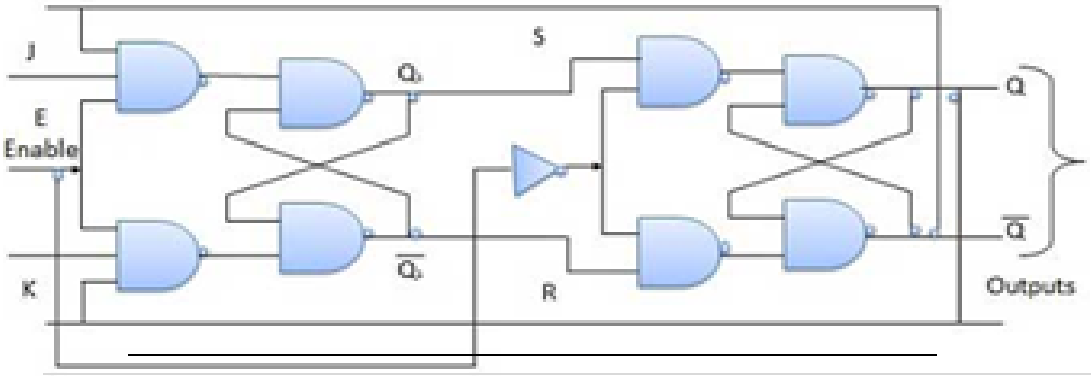
Operation

S.N.	Condition	Operation
1	**S = R = 0 : No change**	If S = R = 0 then output of NAND gates 3 and 4 are forced to become 1. Hence R' and S' both will be equal to 1. Since S' and R' are the input of the basic S-R latch using NAND gates, there will be no change in the state of outputs.
2	**S = 0, R = 1, E = 1**	Since S = 0, output of NAND-3 i.e. R' = 1 and E = 1 the output of NAND-4 i.e. S' = 0. Hence Q_{n+1} = 0 and Q_{n+1} bar = 1. This is reset condition.
3	**S = 1, R = 0, E = 1**	Output of NAND-3 i.e. R' = 0 and output of NAND-4 i.e. S' = 1. Hence output of S-R NAND latch is Q_{n+1} = 1 and Q_{n+1} bar = 0. This is the reset condition.
4	**S = 1, R = 1, E = 1**	As S = 1, R = 1 and E = 1, the output of NAND gates 3 and 4 both are 0 i.e. S' = R' = 0. Hence the **Race** condition will occur in the basic NAND latch.

Master Slave JK Flip Flop

Master slave JK FF is a cascade of two S-R FF with feedback from the output of second to input of first. Master is a positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level. Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and master is inactive.

Circuit Diagram



Truth Table

Inputs			Outputs		Comments
E	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	0	Q_n	$\overline{Q}_n$	No change
1	0	1	0	1	Rset
1	1	0	1	0	Set
1	1	1	$\overline{Q}_n$	Q_n	Toggle

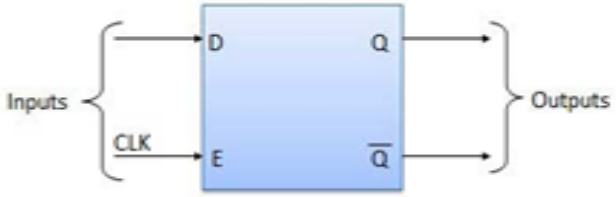
Operation

S.N.	Condition	Operation
1	J = K = 0 (No change)	When clock = 0, the slave becomes active and master is inactive. But since the S and R inputs have not changed, the slave outputs will also remain unchanged. Therefore outputs will not change if J = K = 0.
2	J = 0 and K = 1 (Reset)	Clock = 1 – Master active, slave inactive. Therefore outputs of the master become Q_n = 0 and Q_n bar = 1. That means S = 0 and R = 1. Clock = 0 – Slave active, master inactive. Therefore outputs of the slave become Q = 0 and Q bar = 1. Again clock = 1 – Master active, slave inactive. Therefore even with the changed outputs Q = 0 and Q bar = 1 fed back to master, its output will be Q1 = 0 and Q1 bar = 1. That means S = 0 and R = 1. Hence with clock = 0 and slave becoming active the outputs of slave will remain Q = 0 and Q bar = 1. Thus we get a stable output from the Master slave.
3	J = 1 and K = 0 (Set)	Clock = 1 – Master active, slave inactive. Therefore outputs of the master become Q_n = 1 and Q_n bar = 0. That means S = 1 and R = 0. Clock = 0 – Slave active, master inactive. Therefore outputs of the slave become Q = 1 and Q bar = 0. Again clock = 1 – then it can be shown that the outputs of the slave are stabilized to Q = 1 and Q bar = 0.
4	J = K = 1 (Toggle)	Clock = 1 – Master active, slave inactive. Outputs of master will toggle. So S and R also will be inverted. Clock = 0 – Slave active, master inactive. Outputs of slave will toggle. These changed output are returned back to the master inputs. But since clock = 0, the master is still inactive. So it does not respond to these changed outputs. This avoids the multiple togglingwhich leads to the race around condition. The master slave flip flop will avoid the race around condition.

Delay Flip Flop / D Flip Flop

Delay Flip Flop or D Flip Flop is the simple gated S-R latch with a NAND inverter connected between S and R inputs. It has only one input. The input data is appearing at the output after some time. Due to this data delay between i/p and o/p, it is called delay flip flop. S and R will be the complements of each other due to NAND inverter. Hence S = R = 0 or S = R = 1, these input condition will never appear. This problem is avoid by SR = 00 and SR = 1 conditions.

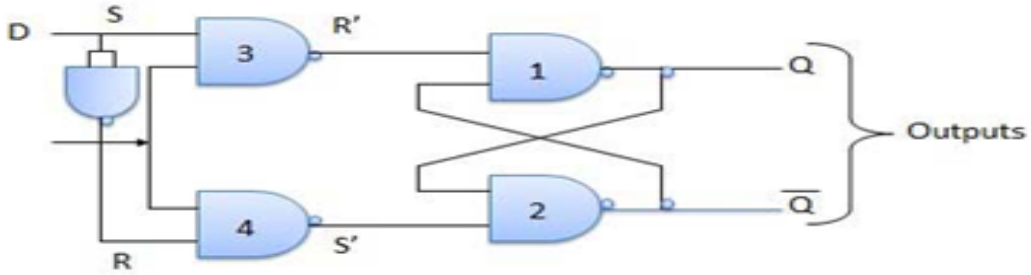
Block Diagram



Truth Table

Inputs		Outputs		Comments
E	D	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	0	1	Rset
1	1	1	0	Set

Circuit Diagram



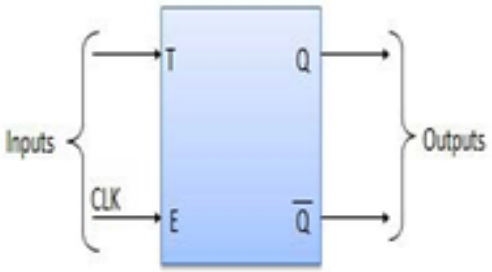
Operation

S.N.	Condition	Operation
1	**E = 0**	Latch is disabled. Hence no change in output.
2	**E = 1 and D = 0**	If E = 1 and D = 0 then S = 0 and R = 1. Hence irrespective of the present state, the next state is Q_{n+1} = 0 and Q_{n+1} bar = 1. This is the reset condition.
3	**E = 1 and D = 1**	If E = 1 and D = 1, then S = 1 and R = 0. This will set the latch and Q_{n+1} = 1 and Q_{n+1} bar = 0 irrespective of the present state.

Toggle Flip Flop / T Flip Flop

Toggle flip flop is basically a JK flip flop with J and K terminals permanently connected together. It has only input denoted by **T** as shown in the Symbol Diagram. The symbol for positive edge triggered T flip flop is shown in the Block Diagram.

Symbol Diagram



Truth Table

Inputs		Outputs		Comments
E	T	Q_{n+1}	$\overline{Q}_{n+1}$	
1	0	Q_n	$\overline{Q}_n$	No change
1	1	$\overline{Q}_n$	Q_n	Toggle

Block Diagram

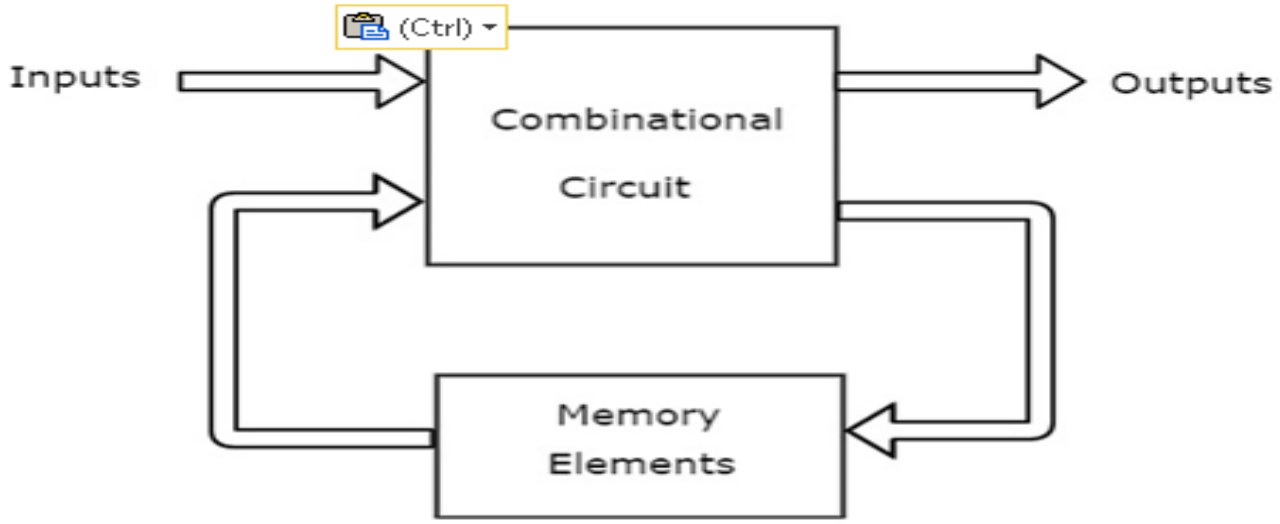
-

Operation

S.N.	Condition	Operation
1	**T = 0, J = K = 0**	The output Q and Q bar won't change
2	**T = 1, J = K = 1**	Output will toggle corresponding to every leading edge of clock signal.

SEQUENTIAL CIRCUIT

Block diagram



This sequential circuit contains a set of inputs and outputs. The outputs of sequential circuit depend not only on the combination of present inputs but also on the previous outputs. Previous output is nothing but the **present state**. Therefore, sequential circuits contain combinational circuits along with memory storage storage elements.

State Tables and State Diagrams

These also determine the next state of the circuit. The relationship that exists among the inputs, outputs, present states and next states can be specified by either the state table or the state diagram.

State Table

The state table representation of a sequential circuit consists of three sections labeled *present state*, *next state* and *output*. The present state designates the state of flip-flops before the occurrence of a clock pulse. The next state shows the states of flip-flops after the clock pulse, and the output section lists the value of the output variables during the present state.

State Diagram

In addition to graphical symbols, tables or equations, flip-flops can also be represented graphically by a state diagram. In this diagram, a state is represented by a circle, and the transition between states is indicated by directed lines (or arcs) connecting the circles.

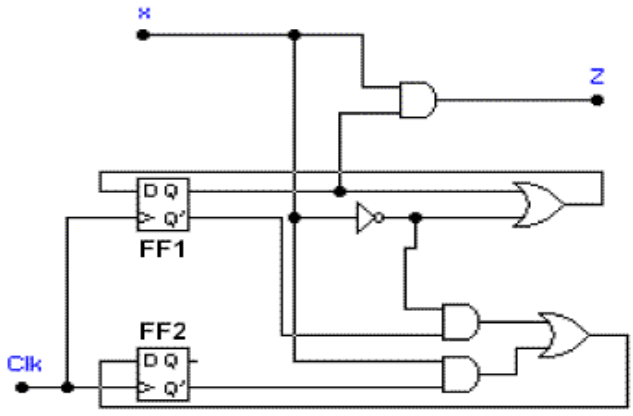
Example: Consider a sequential circuit shown in Figure. It has one input x, one output Z and two state variables Q_1Q_2
(Thus having four possible present states 00, 01, 10, 11).

The behavior of the circuit is determined By the following Boolean expressions:

$Z = x * Q_1$

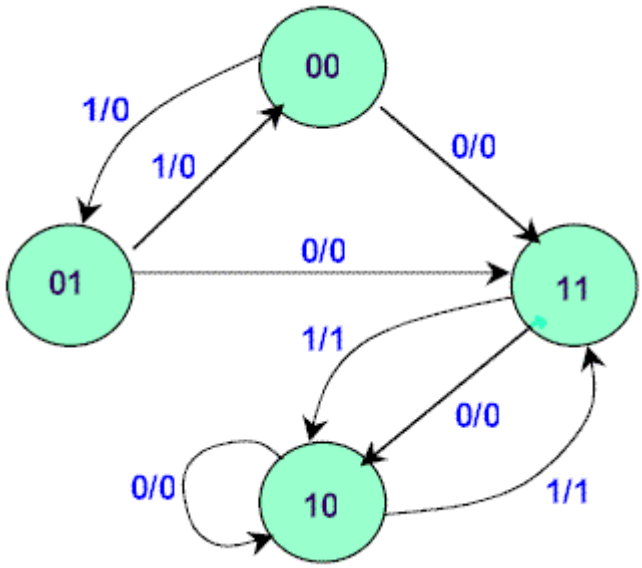
$D_1 = x' + Q_1$

$D_2 = x * Q_2' + x' * Q_1'$

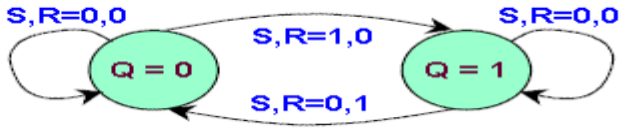
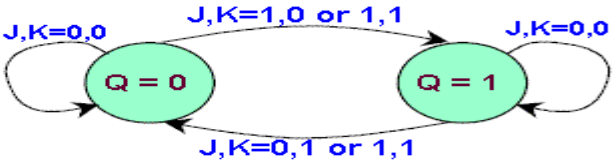
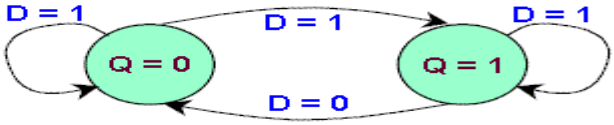
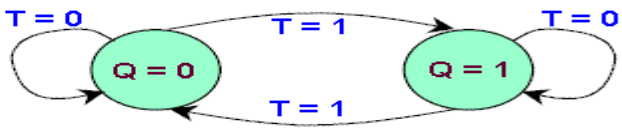


These equations can be used to form the state table. Suppose the present state (i.e. Q_1Q_2) = 00 and input x = 0. Under these conditions, we get Z = 0, D_1 = 1, and D_2 = 1. Thus the next state of the circuit D_1D_2 = 11, and this will be the present state after the clock pulse has been applied. The output of the circuit corresponding to the present state Q_1Q_2 = 00 and x = 1 is Z = 0. This data is entered into the state table as shown in Table 2.

Present State Q_1Q_2	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
0 0	1 1	0 1	0	0
0 1	1 1	0 0	0	0
1 0	1 0	1 1	0	1
1 1	1 0	1 0	0	1



State Diagrams of Various Flip-flops

NAME	STATE DIAGRAM
SR	
JK	
D	
T	

Define universal gate with example
State and prove DE Morgan's law
Draw the logic diagram of the Boolean function F=AB+A'B using NAND gates only
What is computer Architecture? Mention the types
Write the symbol, logical expression and truth table for NAND gate
Explain Full adder
State any two rules of Boolean algebra
What is state table and state diagram
What is SOP(POS) explain with example

Prove NAND and NOR gates as universal gates.
Explain half adder and full adder circuit with example
Define a half adder using only NAND gates.
Explain the working of J-K flip flop
Explain the working of R-S fil-flop
Explain the working of T and D flip flop
Explain the steps involved in design of the sequential circuits.

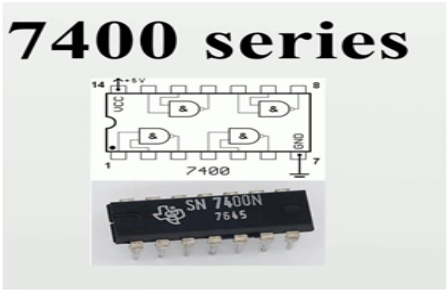
Simplify the Boolean function $F(A,B,C,D)=\sum(0,1,2,5,8,9,10)$ in both sum of products and product of sum.
Simplify $F(A,B,C,D)=\sum m(1,3,7,11,15)+\sum d(0,25)$ using k-map
Define k-map? Simplify the follower Boolean function using kmap $F(A,B,C,D)=\sum(0,2,4,6,10,11,12,13,14,15)$
What is a k-map? Explain different types of karnaugh maps

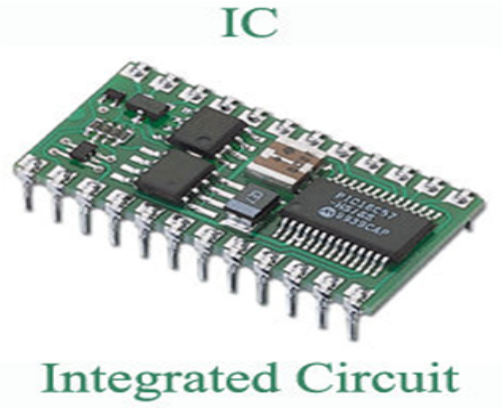
Chapter-2

DIGITAL INTEGRATED CIRCUITS

INTRODUCTION

DIGITAL LOGIC GATE is an electronic device that makes logical decisions based on the different combinations of digital signals present on its inputs





Integrated Circuits or IC they are commonly called No of transistors or gates Integrated circuits are categorized according to the number of logic gates or the complexity of the circuits within a single chip

Classification of ICs on basis of chip sizes

Integrated Circuits can be classified based on its integration scale. An integration scale denotes the number of components fitted into a standard Integrated Circuit.

Small Scale Integration

- The early developed ICs were the small-scale integrated circuits which contained only a few transistors.
- A number of transistors ranged from 2 to 10.
- Nowadays, a single small-scale integrated circuit chip contains about 3 to 30 gates.

Medium Scale Integration

- The next in line to the IC family were the medium scale integrated circuits which contained hundreds of transistors.
- A Medium Scale Integrated Circuit contains about 30 to 300 gates per chip.

Large Scale Integration

- Large Scale Integrated circuits were the next to MSI chips.
- Each LSI chip contained tens of thousands of transistors.
- A Large-Scale Integrated Circuit contains about 300 to 3000 gates per chip.

Very Large-Scale Integration

- Development of VLSI chip paved a way to the creation of the first microprocessor, by fabrication of a CPU on a single microchip.
- A VLSI chip contained about 1 to 4 million transistors.
- A single chip contains about more than 3000 gates.

Ultra-Large-Scale Integration

- When millions and billions of transistors are embedded on a single silicon chip, the integration technology is known as Ultra Large-Scale Integration.
- This technique was first used during the late 1980s, specifically for the development of Intel 8086 series.
- Another example of chips built on ULSI technology is Intel 486 and Pentium series of processors.

Classification of integrated circuits basis of applications

- ✓ **Analog: It works by processing continuous signals. They functions like amplification, active filtering, demodulation, mixing etc. It include sensors, power management circuits and operational amplifiers**
- ✓ **Digital: few thousand to millions of logic gates, flip-flops, multiplexers. Small size of these circuits allows high speed, low power dissipation, and cheaper manufacturing costs, uses zero and one process (microprocessor's DSPs and micro controllers**
- ✓ **Mixed signal: Combine both analog and digital circuits on a single chip to create functions such as A/D convertor and D/A converters, offer a smaller size and lower cost, signal interference**

Digital logic families

- ✓ **DL** Diode Logic
- ✓ **RTL** Resistor Transistor Logic
- ✓ **DTL** Diode transistor logic
- ✓ **HTL** High Threshold Logic
- ✓ **TTL** Transistor Transistor Logic
- ✓ **I2L** Integrated Injection Logic
- ✓ **ECL** Emitter Coupled Logic
- ✓ **MOS** Metal Oxide Semiconductor Logic
- ✓ **COMS** Complementary Metal Oxide Semiconductor Logic

Basic Concept

- ✓ **Fan in:** number of inputs a gate has, like two input And gate has fan in of two
- ✓ **Fan out:** number of gates that each gate can drive, while providing voltage levels in the guaranteed range, is called the standard load or fan out

  (normally as in the case of fan-in, the delay offered by a gate increases with the increase in fan-out)
- ✓ **Noise margin:** maximum noise voltage level that is tolerated by a gate is called noise margin
- ✓ **Power dissipation:** system defines battery life the greater the power dissipation ,shorter the battery life

Shift Registers

Flip flop can store can store one bit of information, to store multiple information we need multiple flip flops

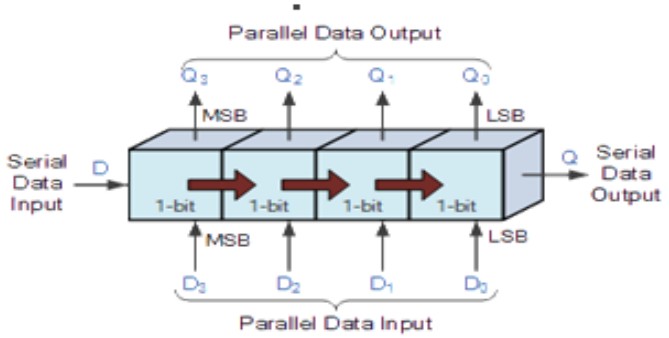
Register: group of flip flop which hold binary data

If the register is capable of shifting bits either towards right hand side or towards left hand side is known as **shift register.**

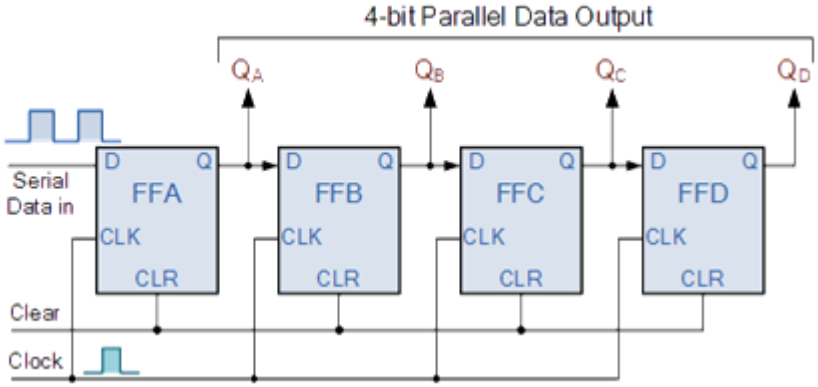
It another type of sequential logic circuit that is used for the storage or transfer of data in the form of binary numbers and then "shifts" the data out once every clock cycle based on the method used to load data onto and read data from shift registers, they are classified as

- Serial In – Serial Out shift register
- Serial In – Parallel Out shift register
- Parallel In – Serial Out shift register
- Parallel In – Parallel Out shift register

- **Serial-in to Parallel-out (SIPO)** - the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
- **Serial-in to Serial-out (SISO)** - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
- **Parallel-in to Serial-out (PISO)** - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- **Parallel-in to parallel-out (PIPO) - the** parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.
- The effect of data movement from left to right through a shift register can be presented graphically as:



4-bit Serial-in to Parallel-out Shift Register



Let's assume that all the flip-flops (FFA to FFD) have just been RESET (CLEAR input) and that all the outputs Q_A to Q_D are at logic level "0" ie, no parallel data output.

If a logic "1" is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting Q_A will be set HIGH to logic "1" with all the other outputs still remaining LOW at logic "0". Assume now that the DATA input pin of FFA has returned LOW again to logic "0" giving us one data pulse or 0-1-0.

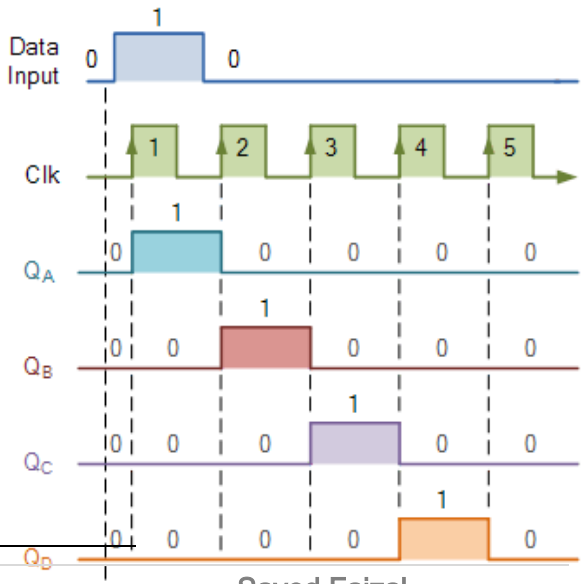
The second clock pulse will change the output of FFA to logic "0" and the output of FFB and Q_B HIGH to logic "1" as its input D has the logic "1" level on it from Q_A. The logic "1" has now moved or been "shifted" one place along the register to the right as it is now at Q_A.

When the third clock pulse arrives this logic "1" value moves to the output of FFC (Q_C) and so on until the arrival of the fifth clock pulse which sets all the outputs Q_A to Q_D back again to logic level "0" because the input to FFA has remained constant at logic level "0".

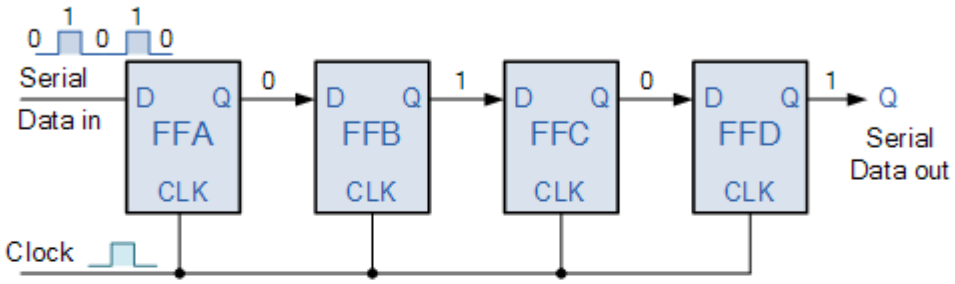
The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register. This data value can now be read directly from the outputs of Q_A to Q_D.

Then the data has been converted from a serial data input signal to a parallel data output.

Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0

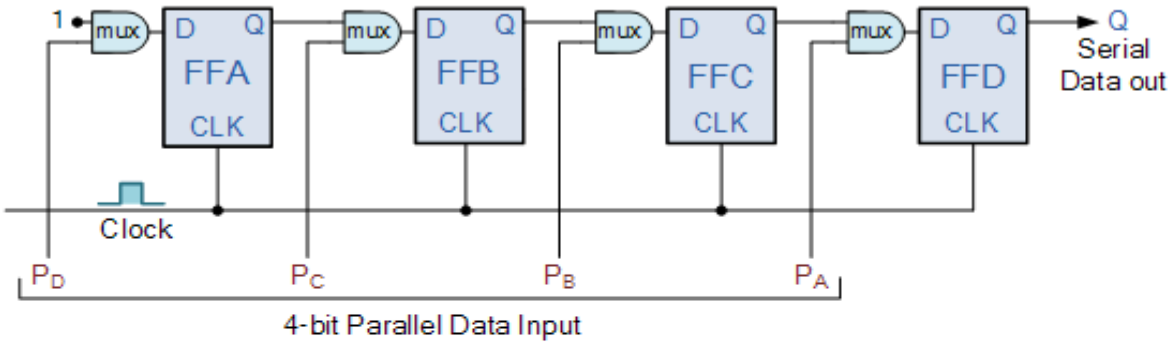


4-bit Serial-in to Serial-out Shift Register



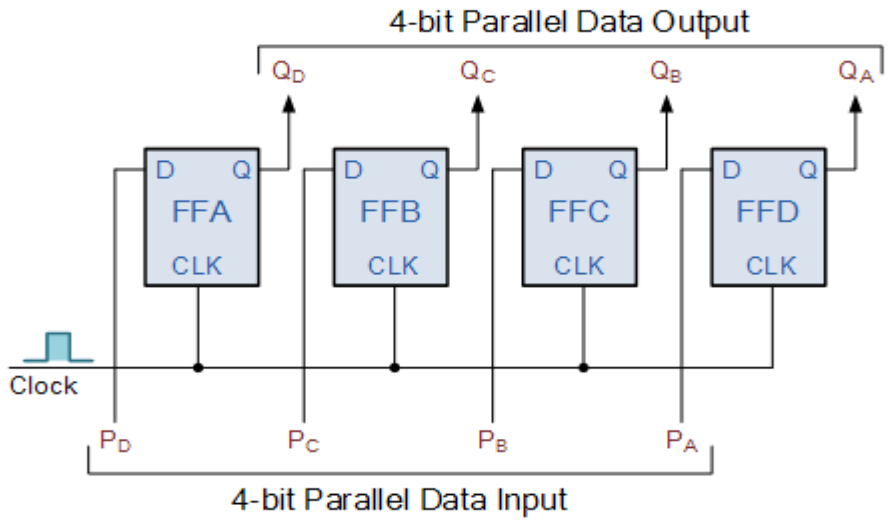
You may think what's the point of a SISO shift register if the output data is exactly the same as the input data. Well this type of **Shift Register** also acts as a temporary storage device or it can act as a time delay device for the data, with the amount of time delay being controlled by the number of stages in the register, 4, 8, 16 etc or by varying the application of the clock pulses. Commonly available IC's include the 74HC595 8-bit Serial-in to Serial-out Shift Register all with 3-state outputs.

4-bit Parallel-in to Serial-out Shift Register



As this type of shift register converts parallel data, such as an 8-bit data word into serial format, it can be used to multiplex many different input lines into a single serial DATA stream which can be sent directly to a computer or transmitted over a communications line. Commonly available IC's include the 74HC166 8-bit Parallel-in/Serial-out Shift Registers.

4-bit Parallel-in to Parallel-out Shift Register



The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk).

Similar to the Serial-in to Serial-out shift register, this type of register also acts as a temporary storage device or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

Rest of three perform working

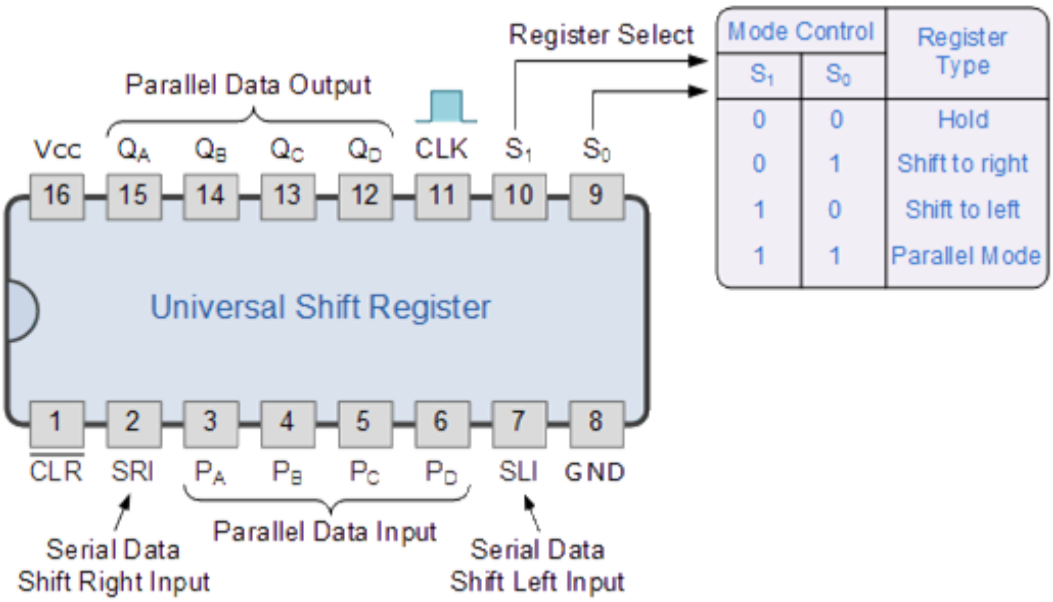
A Unidirectional shift register is capable of shifting in only one direction.

A bidirectional shift register is capable of shifting in both the directions.

Universal Shift Register

These universal shift registers can perform any combination of parallel and serial input to output operations but require additional inputs to specify desired function and to pre-load and reset the device. A commonly used universal shift register is the TTL 74LS194 as shown below.

4-bit Universal Shift Register 74LS194

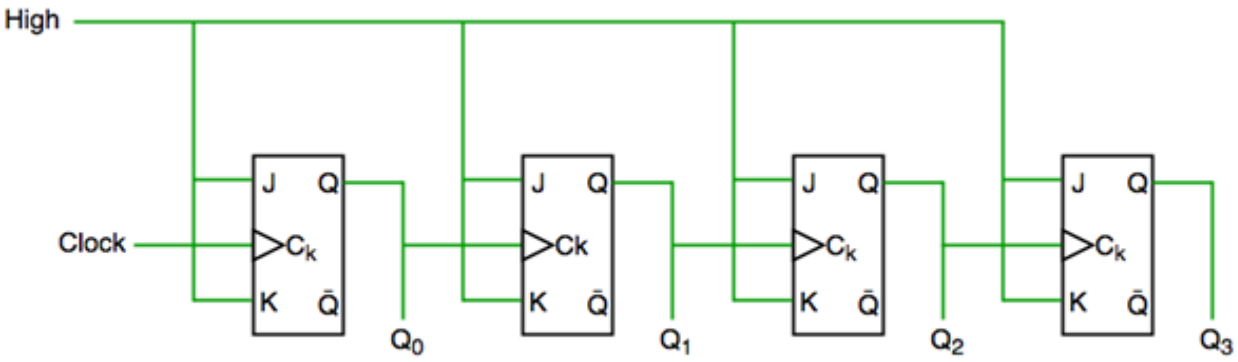


Universal shift registers are very useful digital devices. They can be configured to respond to operations that require some form of temporary memory storage or for the delay of information such as the SISO or PIPO configuration modes or transfer data from one point to another in either a serial or parallel format. Universal shift registers are frequently used in arithmetic operations to shift data to the left or right for multiplication or division.

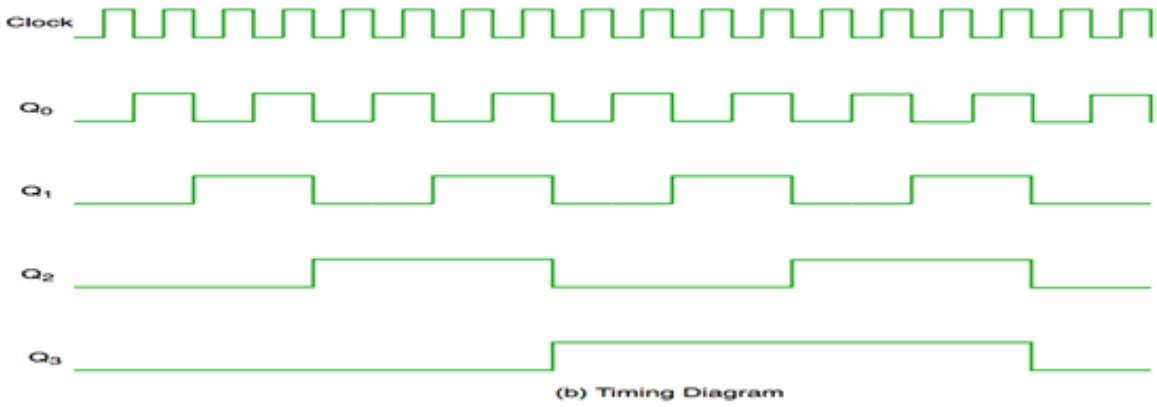
Counter is a sequential circuit. A digital circuit which is used for counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. Counters are of two types.

- Asynchronous or ripple counters
- Synchronous counters

- Asynchronous or ripple counters. In asynchronous counter we don't use universal clock, only first flip flop is driven by main clock and the clock input of rest of the following flip flop is driven by output of previous flip flops. We can understand it by following diagram



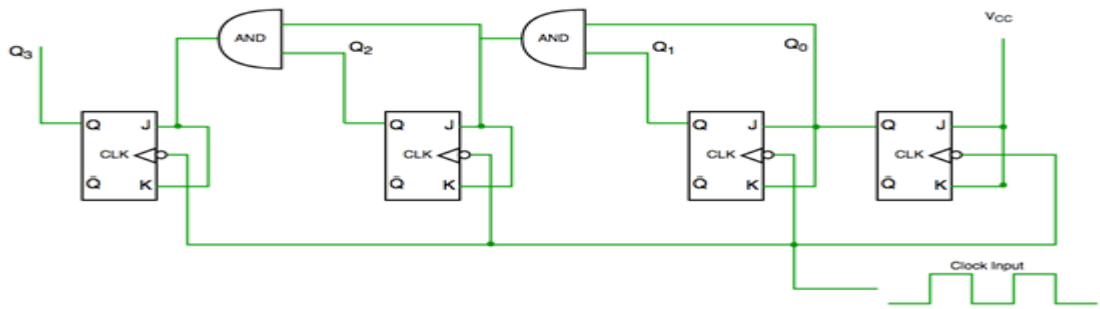
(a) Asynchronous counter

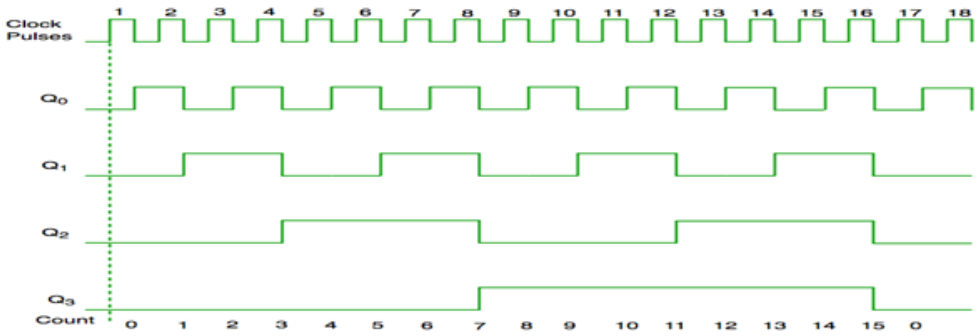


(b) Timing Diagram

It is evident from timing diagram that Q0 is changing as soon as the rising edge of clock pulse is encountered, Q1 is changing when rising edge of Q0 is encountered(because Q0 is like clock pulse for second flip flop) and so on. In this way ripples are generated through Q0,Q1, Q2,Q3 hence it is also called **RIPPLE counter.**

- **SYNCHRONOUS:** counters unlike the asynchronous counter, synchronous counter has one global clock which drives each flip flop so output changes in parallel. The one advantage of synchronous counter over asynchronous counter is, it can operate on higher frequency than asynchronous counter as it does not have cumulative delay because of same clock is given to each flip flop.





From circuit diagram we see that Q0 bit gives response to each falling edge of clock while Q1 is dependent on Q0, Q2 is dependent on Q1 and Q0 , Q3 is dependent on Q2,Q1 and Q0.

MULTIPLEXER

A data selector, more commonly called a Multiplexer, are combinational logic switching device that operate like a very fast acting multiple position rotary switch

They connect or control, multiple input lines called "channels" consisting of either 2,4,8 or 16 individual inputs, one at a time to an output.

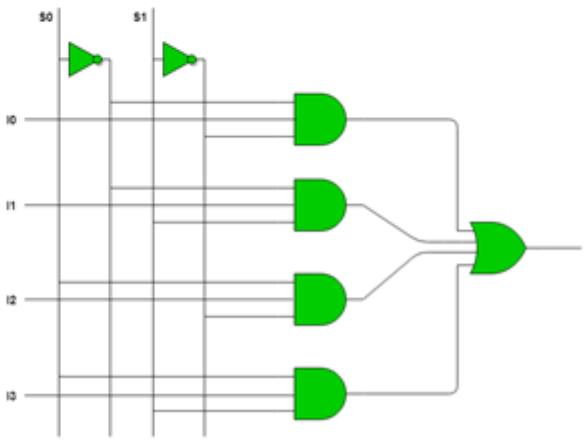
For example, a single 8-channel multiplexer would connect one of its eight inputs to the single data output. Multiplexers are used as one method of reducing the number of logic gates required in a circuit or when a single data line is required to carry two or more different digital signals.

Truth Table

S0	S1	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

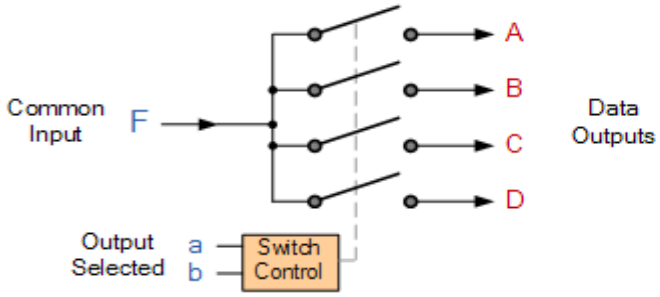
So, final equation,

Y = S0'.S1'.I0 + S0'.S1.I1 + S0.S1'.I2 + S0.S1.I3



DEMULTIPLEXER

The data distributor, known more commonly as a **Demultiplexer** or "Demux" for short, is the exact opposite of the Multiplexer we saw in the previous tutorial.
The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The **demultiplexer** converts a serial data signal at the input to a parallel data at its output lines as shown below.



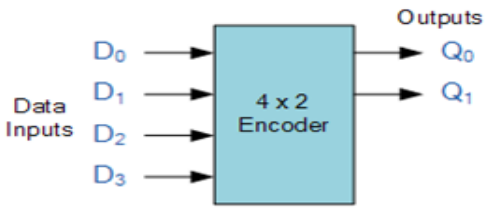
$$F = \overline{ab}A + \bar{a}bB + a\bar{b}C + abD$$

Output Select		Data Output Selected
a	b	
0	0	A
0	1	B
1	0	C
1	1	D

DIGITAL ENCODER

Digital Encoder more commonly called a **Binary Encoder** takes ALL its data inputs one at a time and then converts them into a single encoded output. So we can say that a binary encoder is a multi-input combinational logic circuit that converts the logic level "1" data at its inputs into an equivalent binary code at its output.

Generally, digital encoders produce outputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines. An "n-bit" binary encoder has 2^n input lines and n-bit output lines with common types that include 4-to-2, 8-to-3 and 16-to-4 line configurations.



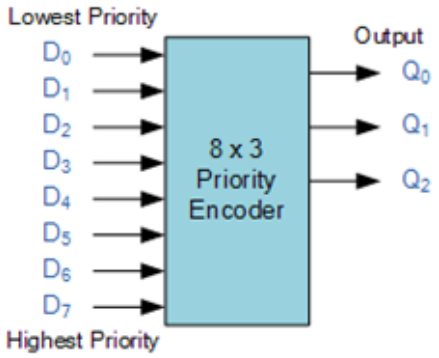
Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

One of the main disadvantages of standard digital encoders is that they can generate the wrong output code when there is more than one input present at logic level "1". For example, if we make inputs D_1 and D_2 HIGH at logic "1" both at the same time, the resulting output is neither at "01" or at "10" but will be at "11" which is an output binary number that is different to the actual input present. Also, an output code of all logic "0"s can be generated when all of its inputs are at "0" OR when input D_0 is equal to one.

PRIORITY ENCODER

- The **Priority Encoder** solves the problems mentioned above by allocating a priority level to each input.
- The *priority encoders* output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.
- The priority encoder comes in many different forms with an example of an 8-input priority encoder

8-to-3 Bit Priority Encoder



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X= dontcare

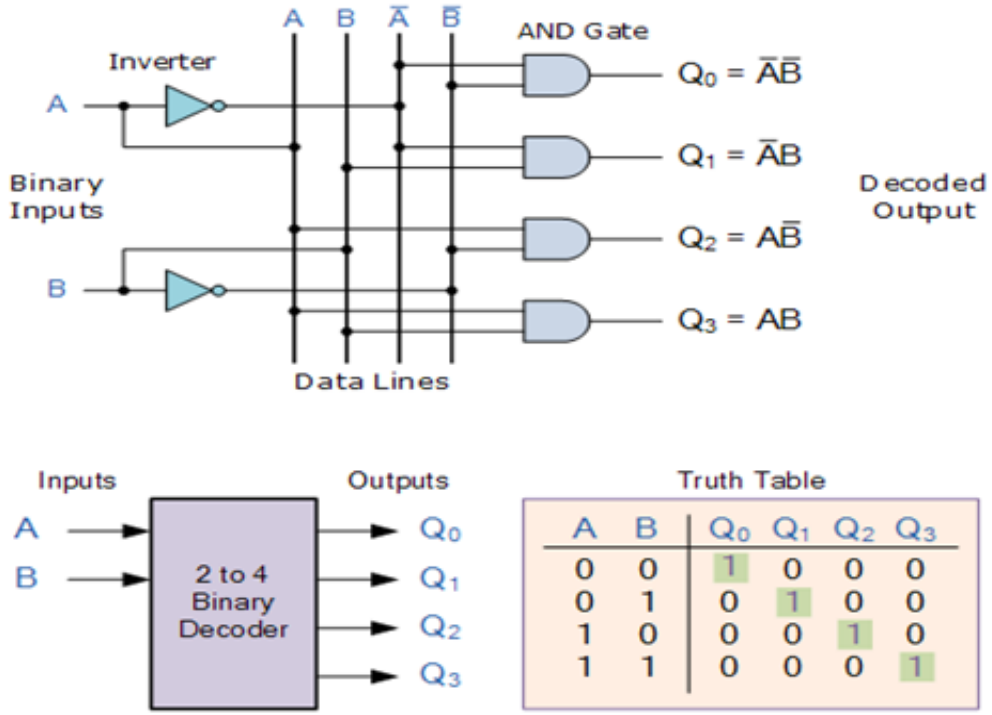
BINARY DECODERS

The name "Decoder" means to translate or decode coded information from one format into another, so a binary decoder transforms "n" binary input signals into an equivalent code using 2^n outputs.

Binary Decoders are another type of digital logic device that has inputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines, so a decoder that has a set of two or more bits will be defined as having an n-bit code, and therefore it will be possible to represent 2^n possible values.

A Binary Decoder converts coded inputs into coded outputs, where the input and output codes are different and decoders are available to "decode" either a Binary or BCD (8421 code) input pattern to typically a Decimal output code.

A 2-to-4 Binary Decoders



A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

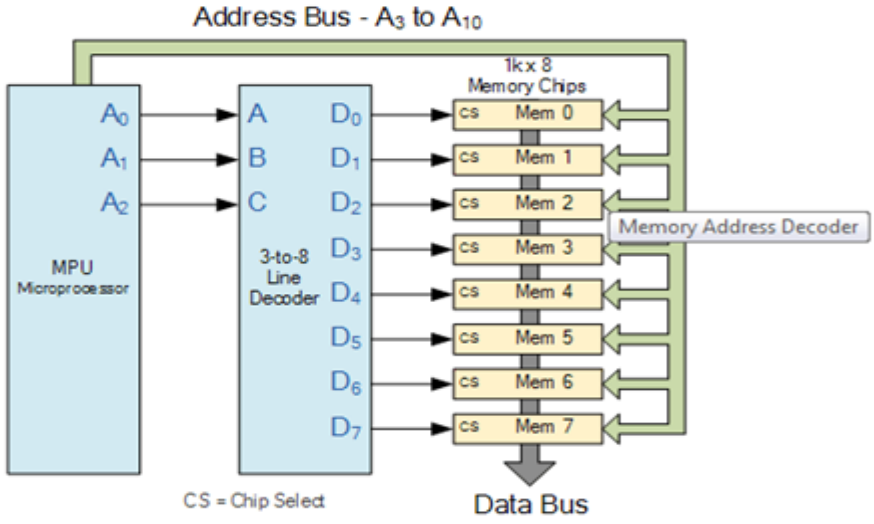
MEMORY ADDRESS DECODER

Binary Decoders are most often used in more complex digital systems to access a particular memory location based on an "address" produced by a computing device. In modern microprocessor systems the amount of memory required can be quite high and is generally more than one single memory chip alone.

One method of overcoming this problem is to connect lots of individual memory chips together and to read the data on a common "Data Bus". In order to prevent the data being "read" from each memory chip at the same time, each memory chip is selected individually one at time and this process is known as **Address Decoding**.

In this type of application, the address represents the coded data input, and the outputs are the particular memory element select signals. Each memory chip has an input called **Chip Select** or **CS** which is used by the MPU (micro-processor unit) to select the appropriate memory chip when required. Generally a logic "1" on the chip select (CS) input selects the memory device while a logic "0" on the input de-selects it.

Memory Address Decoding



Memory Unit

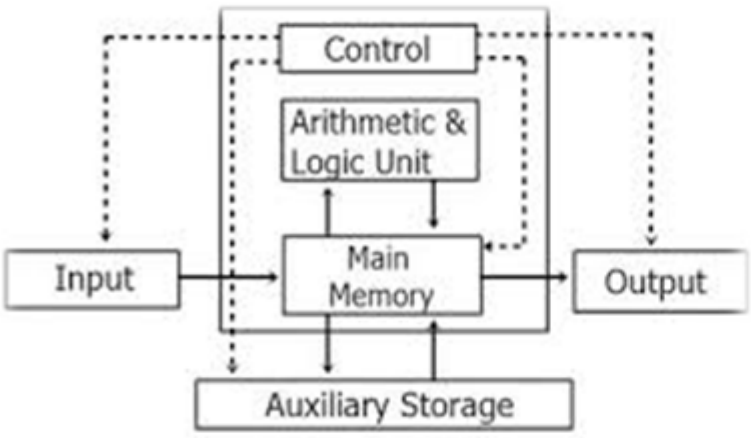
The computer system essentially comprises three important parts-

- Input device,
- central processing unit (CPU) and
- The output device.

The CPU itself is made of three components namely,

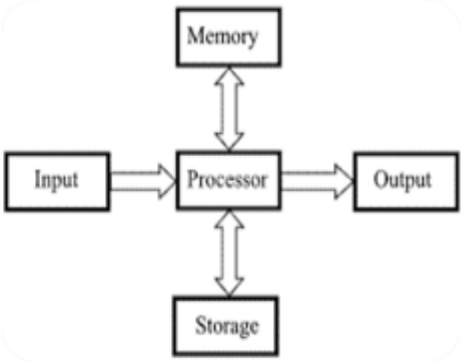
- The arithmetic logic unit (ALU),
- Memory unit, and
- The control unit.

In addition to these, auxiliary storage/secondary storage devices are used to store data and instructions on a long-term basis.



Block Diagram of Computer

The memory unit with other parts of the computer



All storage devices are characterized with the following features:

Speed

Volatility

Access method

Portability

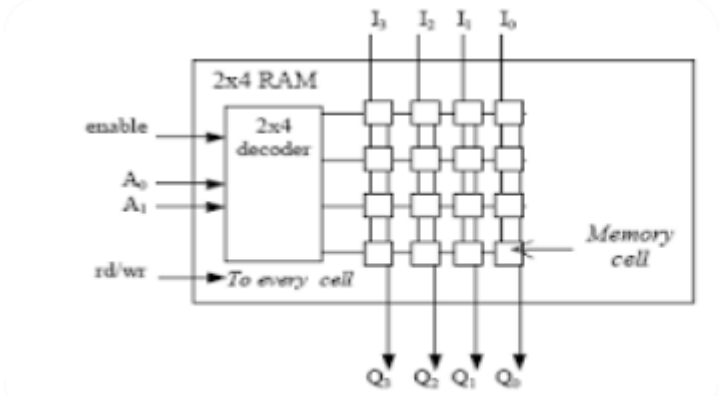
Cost and capacity

The Term Computer Memory is defined as one or more sets of chips that store Data/program instructions, either temporarily or permanently. It is critical processing component in any computer

Computer's memory can be classified into two types- RAM and ROM.

RAM

RAM'S content is not "programmed" before being inserted into an embedded system. Instead, the RAM contains no data when inserted in the embedded system; the system writes data to and then reads data from the RAM during its execution.



TYPES RAM

Static RAM

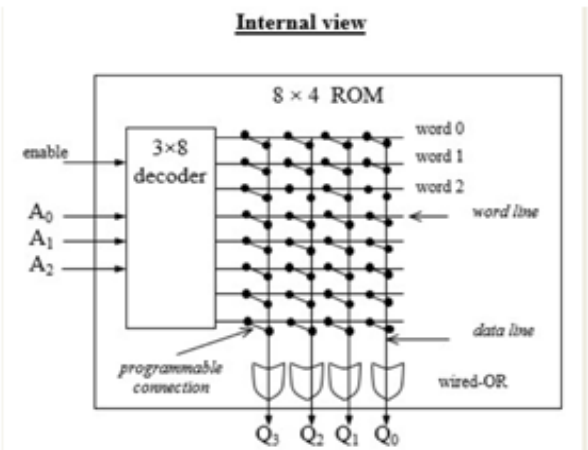
- Static RAM is faster but bigger than dynamic RAM.
- Static RAM, or SRAM, uses a memory cell consisting of a flip-flop to store a bit. Each bit thus requires about 6 transistors.
- This RAM type is called static because it will hold its data as long as power is supplied, in contrast to dynamic RAM.
- Static RAM is typically used for high-performance parts of a system (e.g., cache).

Dynamic RAM

- Dynamic RAM, or DRAM, uses a memory cell consisting of a MOS transistor and capacitor to store a bit
- Each bit thus requires only 1 transistor, resulting in more compact memory than SRAM. However, the charge stored in the capacitor leaks gradually, leading to discharge and eventually to loss of data.
- To prevent loss of data, each cell must regularly have its charge "refreshed typical DRAM cell minimum refresh rate is once every 15.625 microseconds

ROM

- ROM or Read Only Memory is a special type of memory which can only be read and contents of which are not lost even when the computer is switched off.
- It typically contains manufacturer's instructions.
- Among other things, ROM also stores an initial program called the 'bootstrap loader' whose function is to start the computer software operating, once the power is turned on.
- Read-only memories can be manufacturer - programmed or user-programmed.



TYPES ROM

MROM (Masked ROM): The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions. These kinds of ROMs are known as masked ROMs, which are inexpensive.

PROM (Programmable Read Only Memory): PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM program. Inside the PROM chip, there are small fuses which are burnt open during programming. It can be programmed only once and is not erasable.

EPROM (Erasable and Programmable Read Only Memory): EPROM can be erased by exposing it to ultra-violet light for duration of up to 40 minutes. Usually, an EPROM eraser achieves this function. During programming, an electrical charge is trapped in an insulated gate region. The charge is retained for more than 10 years because the charge has no leakage path. For erasing this charge, ultra-violet light is passed through a quartz crystal window (lid). This exposure to ultra-violet light dissipates the charge. During normal use, the quartz lid is sealed with a sticker.

EEPROM (Electrically Erasable and Programmable Read Only Memory): EEPROM is programmed and erased electrically. It can be erased and reprogrammed about ten thousand times. Both erasing and programming take about 4 to 10ms (millisecond). In EEPROM, any location can be selectively erased and programmed. EEPROMs can be erased one byte at a time, rather than erasing the entire chip. Hence, the process of reprogramming is flexible but slow.

Static RAM	Dynamic RAM
Made up of flip-flops.	Made up of capacitors.
Large in size.	Small in size.
Data store in the form of voltage.	Data store in the form of charge.
Much expensive as compare to dynamic RAM	Less expensive as compare to static RAM
Low storage capacity	High storage capacity.
Consume more power	Consume less power
Fast	Slow
Data sustain with time.	Data loses with time, so need refreshing circuit*.

1. What is a bidirectional shift register?(unidertional)
2. What is decoder expansion?
3. What is multiplexer? Explain the operation (Demultiplexer) **2 marks**
4. Define ROM and its types
5. Mention the different logic families of IC
6. Distinguish between Ram and Rom
7. Explain the advantage of registers
8. Define flip flop and why we use shift register

1. Explain ICs and its types with example **5 marks**
2. Design 4 to 1 multiplexer
3. Explain 4 bit register with logic diagram
4. Explain SISO shift register
5. Explain PIPO shift register with a neat diagram
6. Explain 4 bit shift register
7. Explain 8X3 priority encoder
8. Explain decoder expansion with neat diagram
9. Explain the basic computer register
10. What is binary counter? Explain a 4 bit synchronous counter with a neat block diagram
11. Explain with neat block diagram a 4-bit bidirectional shift register with parallel load

1. Simplify the Boolean function $F(A,B,C,D)=\sum(0,1,2,5,8,9,10)$ in both sum of products and product of sum.
2. Simplify $F(A,B,C,D)=\sum m(1,3,7,11,15)+\sum d(0,25)$ using k-map **5 marks**
3. Define k-map? Simplify the follower Boolean function using kmap $F(A,B,C,D)=\sum(0,2,4,6,10,11,12,13,14,15)$
4. What is a k-map? Explain different types of karnaugh maps
5. Design octal to binary encoder