Unit-1

Digital Computers

A Digital computer can be considered as a digital system that performs various computational tasks.

The first electronic digital computer was developed in the late 1940s and was used primarily for numerical computations.

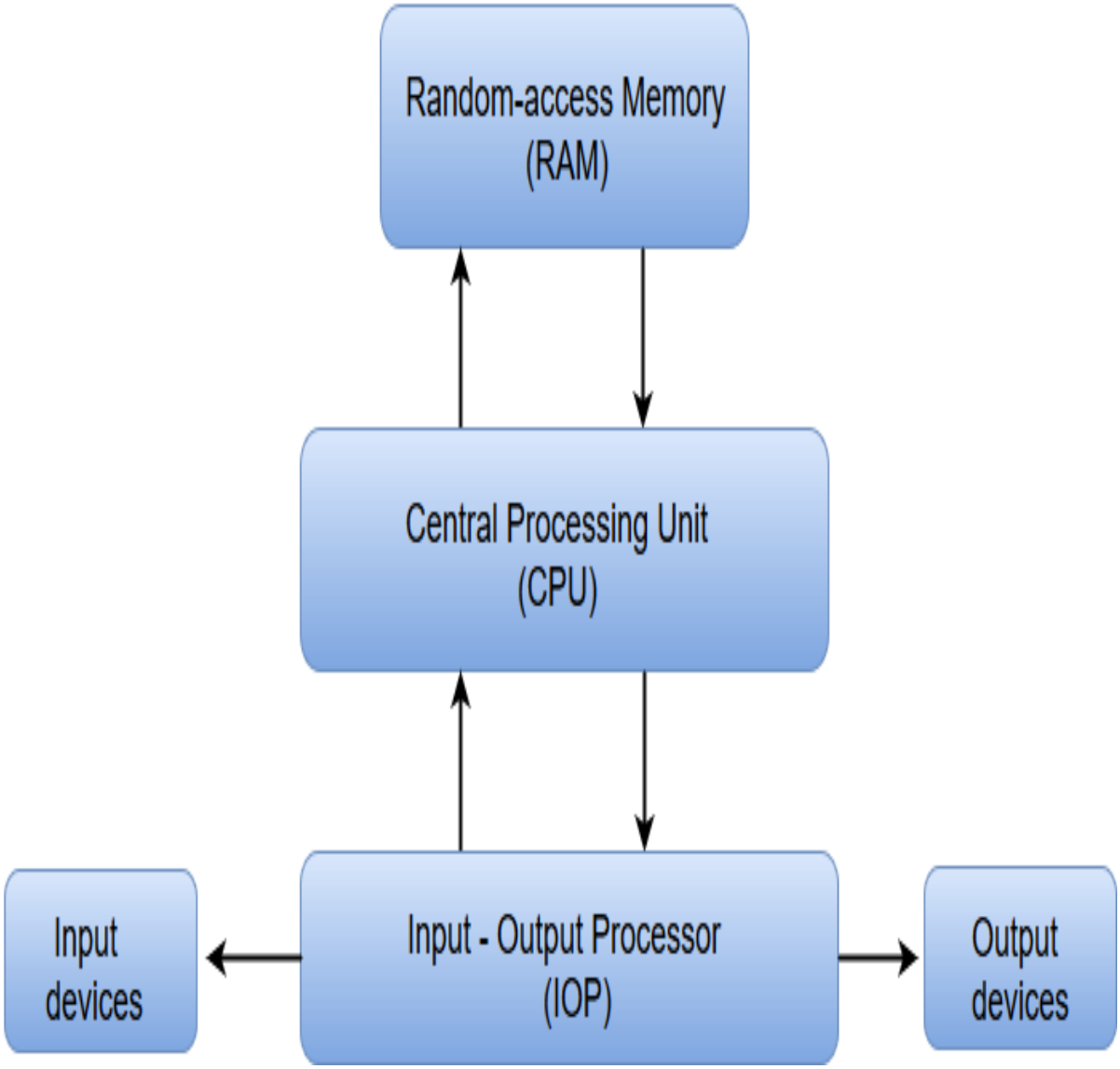
By convention, the digital computers use the binary number system, which has two digits: 0 and 1. A binary digit is called a bit.

A computer system is subdivided into two functional entities: Hardware and Software.

The hardware consists of all the electronic components and electromechanical devices that comprise the physical entity of the device.

The software of the computer consists of the instructions and data that the computer manipulates to perform various data-processing tasks.

Block diagram of a digital computer:



- The Central Processing Unit (CPU) contains an arithmetic and logic unit for manipulating data, a number of registers for storing data, and a control circuit for fetching and executing instructions.

- The memory unit of a digital computer contains storage for instructions and data.

- The Random Access Memory (RAM) for real-time processing of the data.

- The Input-Output devices for generating inputs from the user and displaying the final results to the user.

- The Input-Output devices connected to the computer include the keyboard, mouse, terminals, magnetic disk drives, and other communication devices.

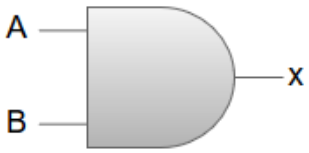
Logic Gates

- The logic gates are the main structural part of a digital system.
- Logic Gates are a block of hardware that produces signals of binary 1 or 0 when input logic requirements are satisfied.
- Each gate has a distinct graphic symbol, and its operation can be described by means of algebraic expressions.
- The seven basic logic gates includes: AND, OR, XOR, NOT, NAND, NOR, and XNOR.
- The relationship between the input-output binary variables for each gate can be represented in tabular form by a truth table.
- Each gate has one or two binary input variables designated by A and B and one binary output variable designated by x.

1) AND GATE:

The AND gate is an electronic circuit which gives a high output only if all its inputs are high. The AND operation is represented by a dot (.) sign.

AND Gate:



Algebraic Function: x = AB

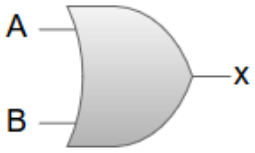
Truth Table:

A	B	x
0	0	0
0	1	0
1	0	0
1	1	1

2)OR GATE:

The OR gate is an electronic circuit which gives a high output if one or more of its inputs are high. The operation performed by an OR gate is represented by a plus (+) sign.

OR Gate:



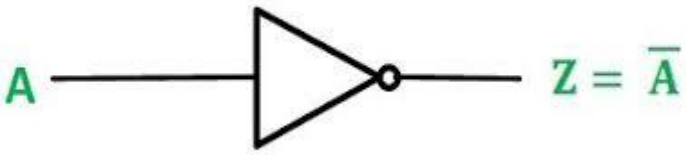
Algebraic Function: x = A + B

Truth Table:

A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

3)NOT GATE:

The NOT gate is an electronic circuit which produces an inverted version of the input at its output. It is also known as an **Inverter**.



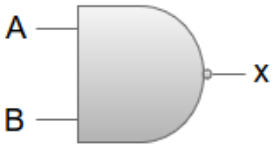
Truth Table

INPUT	OUTPUT
A	NOT A
0	1
1	0

4) NAND GATE:

The NOT-AND (NAND) gate which is equal to an AND gate followed by a NOT gate. The NAND gate gives a high output if any of the inputs are low. The NAND gate is represented by a AND gate with a small circle on the output. The small circle represents inversion.

NAND Gate:



Algebraic Function: x = (AB)'

Truth Table:

A	B	x
0	0	1
0	1	1
1	0	1
1	1	0

5) NOR GATE:

The NOT-OR (NOR) gate which is equal to an OR gate followed by a NOT gate. The NOR gate gives a low output if any of the inputs are high. The NOR gate is represented by an OR gate with a small circle on the output. The small circle represents inversion.

NOR Gate:

Algebraic Function: x = (A+B)'

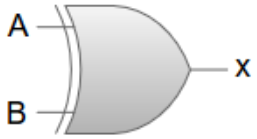
Truth Table:

A	B	x
0	0	1
0	1	0
1	0	0
1	1	0

6) Exclusive-OR/ XOR GATE:

The 'Exclusive-OR' gate is a circuit which will give a high output if one of its inputs is high but not both of them. The XOR operation is represented by an encircled plus sign.

XOR Gate:



Algebraic Function: $x = A \oplus B$

or

$x = A'B + AB'$

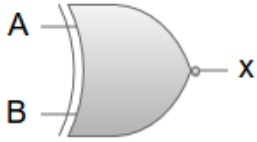
Truth Table:

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

7) EXCLUSIVE-NOR:

The 'Exclusive-NOR' gate is a circuit that does the inverse operation to the XOR gate. It will give a low output if one of its inputs is high but not both of them. The small circle represents inversion.

Exclusive-NOR Gate:



Algebraic Function: $x = (A \oplus B)'$

or

$x = A'B' + AB$

Truth Table:

A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Algebra

Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1.

It is also called as **Binary Algebra** or **logical Algebra**. Boolean algebra was invented by **George Boole** in 1854.

Rule in Boolean Algebra

Following are the important rules used in Boolean algebra.

- Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
- Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as $\overline{B}$. Thus if B = 0 then $\overline{B}$ = 1 and B = 1 then $\overline{B}$ = 0.
- ORing of the variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as A + B + C.
- Logical ANDing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

Boolean Laws

There are 6 types of Boolean Laws.

1) Commutative law

Any binary operation which satisfies the following expression is referred to as commutative operation.

(i) $A.B = B.A$ (ii) $A + B = B + A$

Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

2) Associative law

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

(i) $(A.B).C = A.(B.C)$ (ii) $(A + B) + C = A + (B + C)$

3) Distributive law

Distributive law states the following condition.

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

4) AND law

These laws use the AND operation. Therefore they are called as **AND** laws.

(i) $A.0 = 0$ (ii) $A.1 = A$

(iii) $A.A = A$ (iv) $A.\overline{A} = 0$

5) OR law

These laws use the OR operation. Therefore they are called as **OR** laws.

(i) $A + 0 = A$ (ii) $A + 1 = 1$

(iii) $A + A = A$ (iv) $A + \overline{A} = 1$

6) INVERSION law

This law uses the NOT operation. The inversion law states that double inversion of variable results in the original variable itself.

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

Rules for Boolean Algebra

1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, B, or *C* can represent a single variable or a combination of variables.

The below table lists the most basic identities of Boolean algebra. All the identities in the table can be proven by means of truth tables.

• x + 0 = x	• x + y = y + x
• x * 0 = 0	• x * y = y * x
• x + 1 = 1	• x + (y + z) = (x + y) + z
• x * 1 = x	• x * (y * z) = (x * y) * z
• x + x = x	• x * (y + z) = (x * y) + (x * z)
• x * x = x	• x + (y * x) = (x + y) * (x + z)
• x + x' = 1	• (x + y)' = x' * y'
• x * x' = 0	• (x * y)' = x' + y'

Examples:

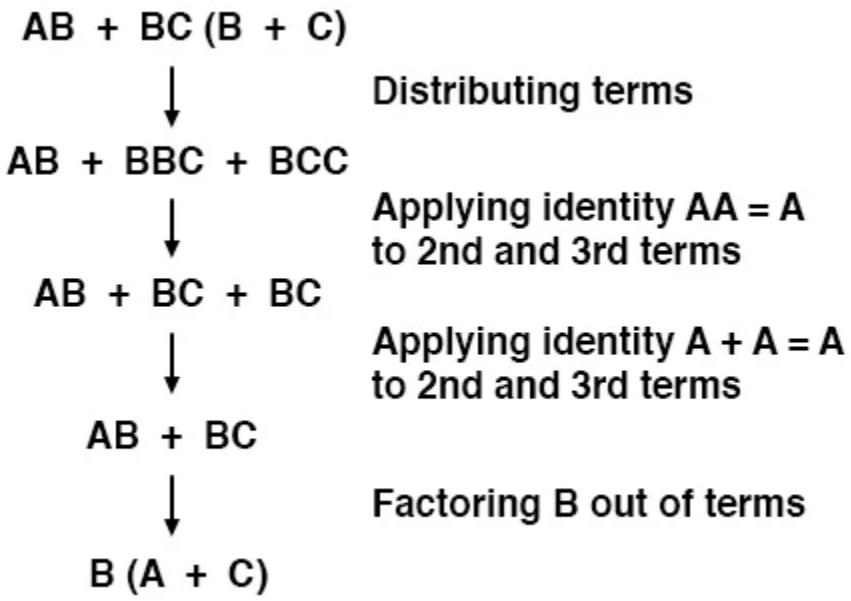
1) **A+AB**

=A (1+B) (take A is common)

=A (1) (1+B=1)

=A

2) **AB+BC(B+C)**



Combinational Circuits

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 does 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.

Half Adder

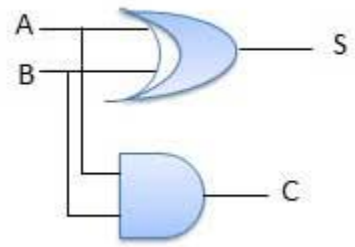
Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary numbers A and B. It is the basic building block for addition of two **single** bit numbers. This circuit has two outputs **carry** and **sum**.

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

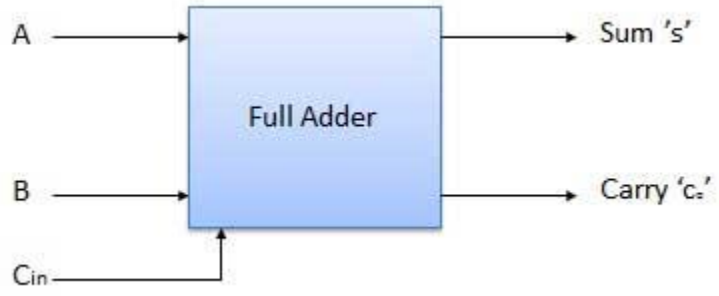
Circuit Diagram



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.

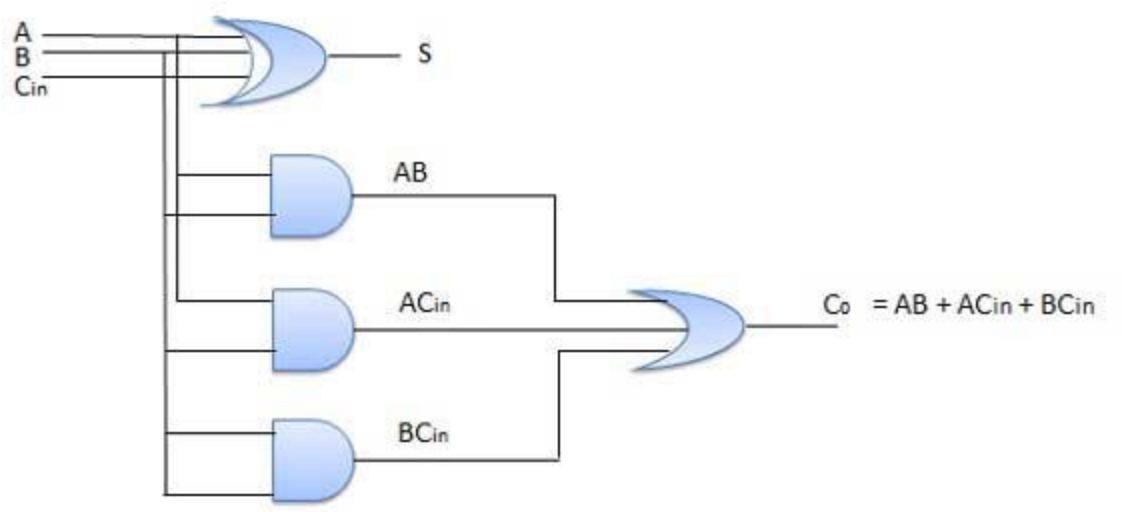
Block diagram



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

A flip flop is an electronic circuit with two stable states that can be used to store binary data.

The stored data can be changed by applying varying inputs.

Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.

Flip-flops and latches are used as data storage elements. It is the basic storage element in sequential logic.

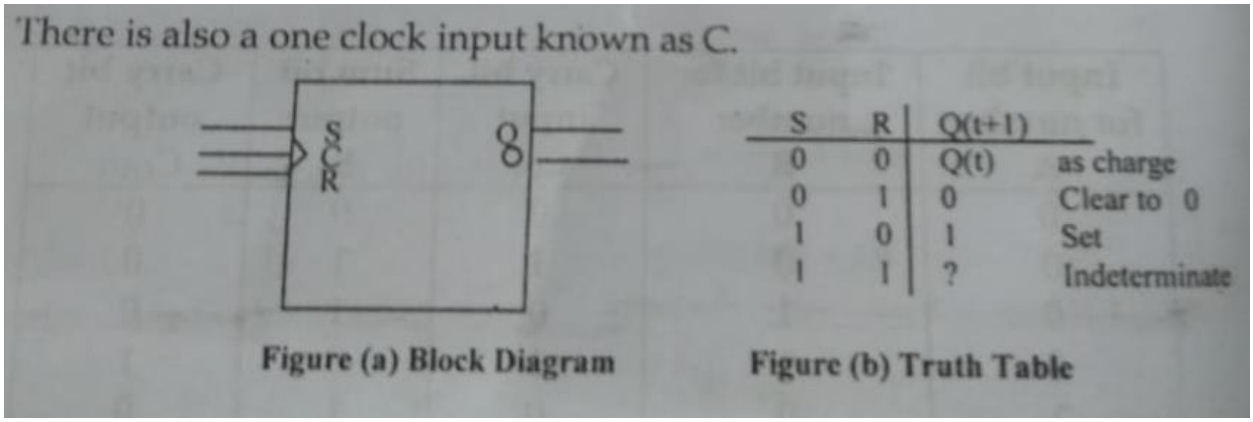
There are 4 types of flip flops:

- SR Flip-Flop
- D Flip-Flop
- JK Flip-Flop
- T Flip-Flop

1) SR Flip-Flop

SR flip-flop operates with only positive clock transitions or negative clock transitions. Whereas, SR latch operates with enable signal.

The **circuit diagram** of SR flip-flop is shown in the following figure.



This circuit has two inputs S & R and two outputs Qtt & Qtt'.
The operation of SR flipflop is similar to SR Latch.
But, this flip-flop affects the outputs only when positive transition of the clock signal is applied instead of active enable.

When S=0 and R=1 then, the output of flip flop goes to 0.

When S=1 and R=0 then, the output of flip flop goes to 1.

When S=1 and R=1 then, the output of flip flop goes to undetermined.

When S=0 and R=0 then, the output of flip flop goes as change.

2) D Flip-Flop

D flip-flop operates with only positive clock transitions or negative clock transitions. Whereas, D latch operates with enable signal. That means, the output of D flip-flop is insensitive to the changes in the input, D except for active transition of the clock signal. The **circuit diagram** of D flip-flop is shown in the following figure.

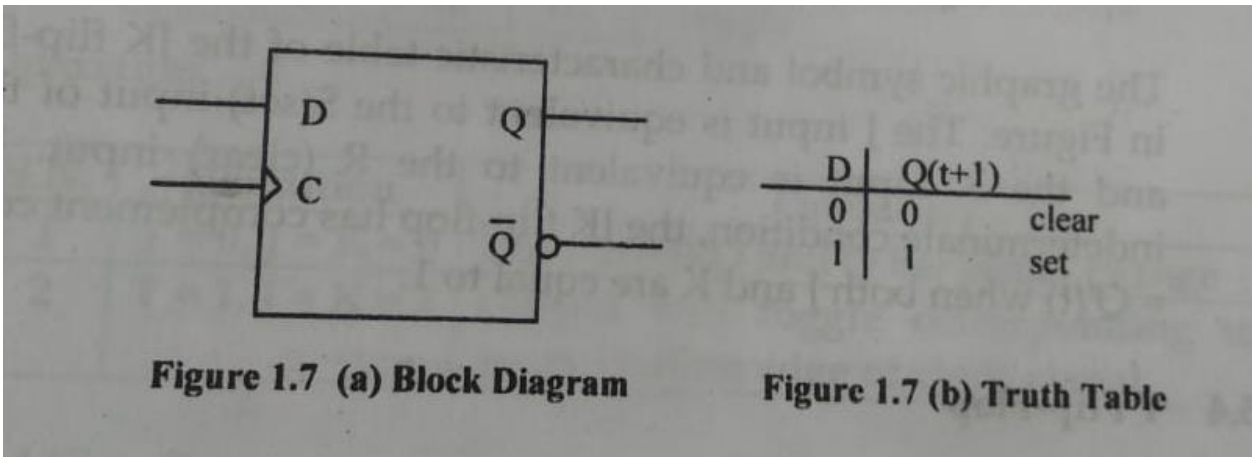


Figure 1.7 (a) Block Diagram Figure 1.7 (b) Truth Table

Therefore, D flip-flop always Hold the information, which is available on data input, D of earlier positive transition of clock signal. From the above state table, we can directly write the next state equation as

When D=1 then, the output of flip flop goes to 1.

When D=0 then, the output of flip flop goes to 0.

3) JK Flip-Flop

JK flip-flop is the modified version of SR flip-flop. It operates with only positive clock transitions or negative clock transitions.

The **circuit diagram** of JK flip-flop is shown in the following figure.

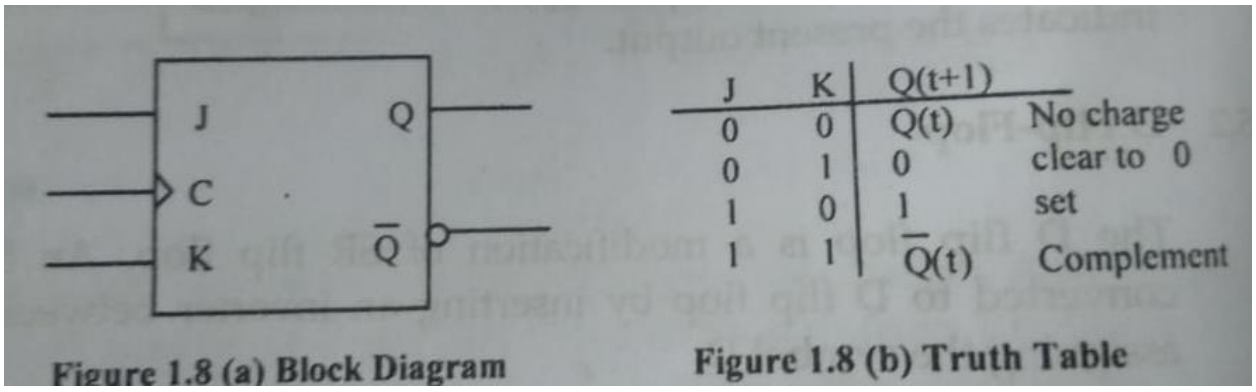


Figure 1.8 (a) Block Diagram Figure 1.8 (b) Truth Table

This circuit has two inputs J & K and two outputs Qtt & Qtt'. The operation of JK flip-flop is similar to SR flip-flop. Here, we considered the inputs of SR flip-flop as **S = J Q**tt**'** and **R = KQ**tt in order to utilize the modified SR flip-flop for 4 combinations of inputs.

When J=0 and K=0 then, the output of flip flop goes no change.

When J=0 and K=1 then, the output of flip flop goes to 0.

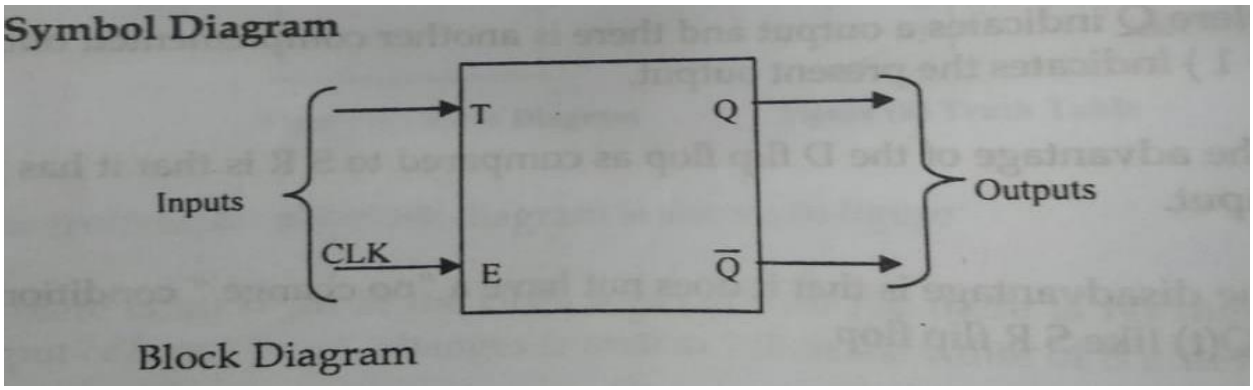
When J=1 and K=0 then, the output of flip flop goes to 1.

When =1 and K=1 then, the output of flip flop goes to complement.

4) T 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 transitions or negative clock transitions.

The **circuit diagram** of T flip-flop is shown in the following figure.



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

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.

This circuit has single input T and two outputs Qtt & Qtt'. The operation of T flip-flop is same as that of JK flip-flop. Here, we considered the inputs of JK flip-flop as **J = T** and **K = T** in order to utilize the modified JK flip-flop for 2 combinations of inputs.

When T=0 then output goes to no change.
When T=1 then output goes to no Toggle.

Integrated Circuits

An integrated circuit (IC) is manufactured using silicon material and mounted in a ceramic or plastic container (known as Chip). The basic components of an IC consist of electronic circuits for the digital gates. The various gates are interconnected inside an IC to form the required circuit.

The following categories can broadly classify an Integrated Circuit (IC):

SSI (Small Scale Integration Devices)

These type of devices contain several independent gates in a single package. The inputs and outputs of these gates are connected directly to the pins in the package. The number of logic gates are usually less than 10 and are limited by the number of pins available in the IC.

MSI (Medium Scale Integration Devices)

These type of devices has a complexity of approximately 10 to 200 gates in a single package. The basic components include decoders, adders, and registers.

LSI (Large Scale Integration Devices)

LSI devices contain about 200 to a few thousand gates in a single package. The basic components of an LSI device include digital systems, such as processors, memory chips, and programmable modules.

VLSI (Very Large Scale Integration Device)

This type of devices contains thousands of gates within a single package. The most common example of a VLSI device is a complex microcomputer chip.

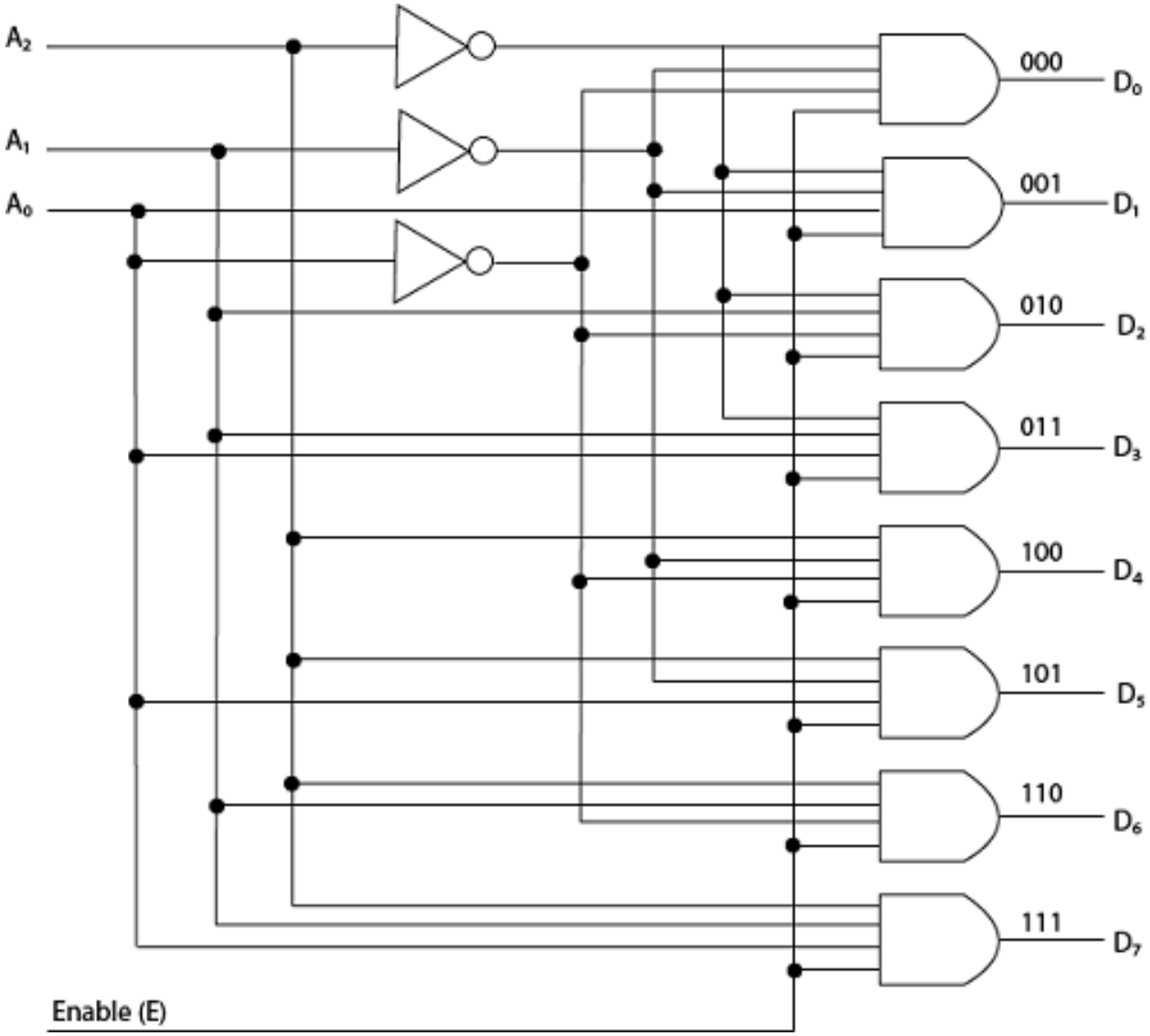
Decoders

A Decoder can be described as a combinational circuit that converts binary information from the 'n' coded inputs to a maximum of 2^n different outputs.

The most preferred or commonly used decoders are n-to-m decoders, where m<= 2^n.

An n-to-m decoder has n inputs and m outputs and is also referred to as an n * m decoder.

The following image shows a 3-to-8 line decoder with three input variables which are decoded into eight output, each output representing one of the combinations of the three binary input variables.



The three inverter gates provide the complement of the inputs corresponding to which the eight AND gates at the output generates one binary combination for each input.

The most common application of this decoder is binary-to-octal conversion.

The truth table for a 3-to-8 line decoder can be represented as:

x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

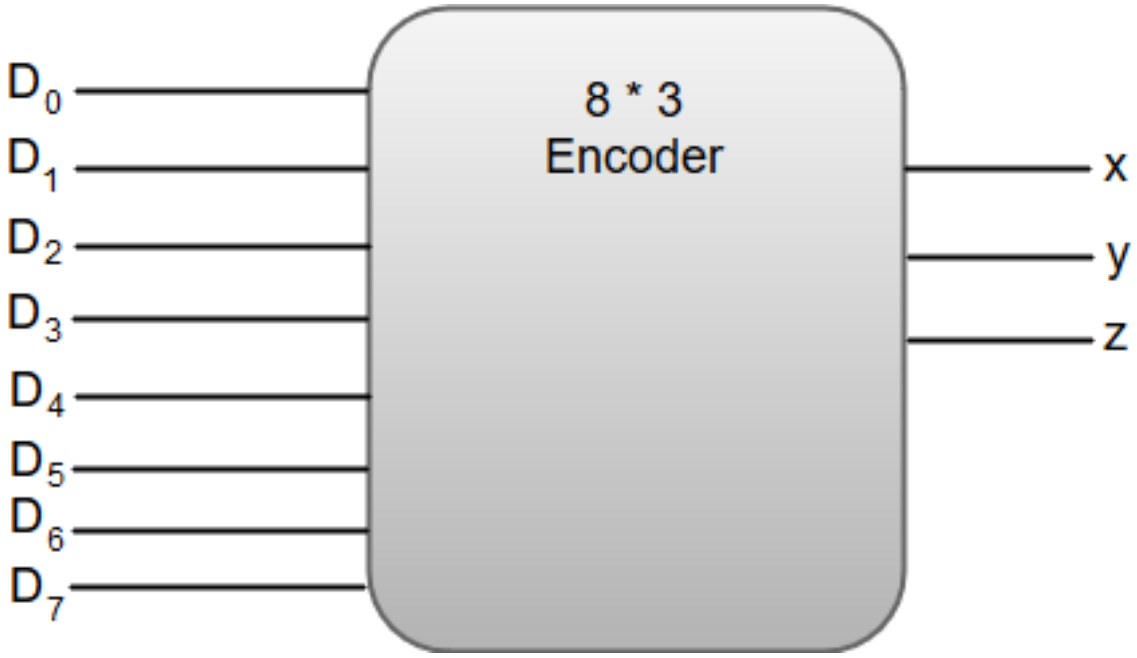
Encoders

An encoder can also be described as a combinational circuit that performs the inverse operation of a decoder. An encoder has a maximum of 2^n (or less) input lines and n output lines.

In an Encoder, the output lines generate the binary code corresponding to the input value.

The most common application of an encoder is the **Octal-to-Binary** encoder. Octal to binary encoder takes eight input lines and generates three output lines.

The following image shows the block diagram of an 8 * 3 line encoder.



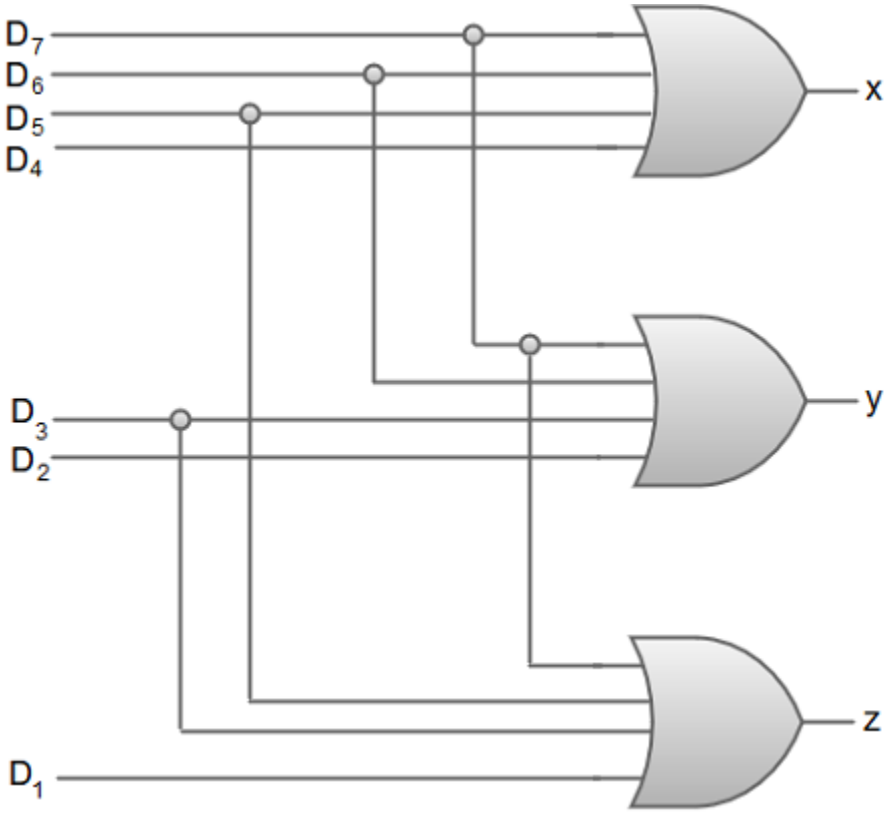
The truth table for an 8 * 3 line encoder can be represented as:

D7	D6	D5	D4	D3	D2	D1	D0	x	y	z
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

From the truth table, we can write the Boolean function for each output as:

```
x = D4 + D5 + D6 + D7
y = D2 + D3 + D6 + D7
z = D1 + D3 + D5 + D7
```

The circuit diagram for an 8 * 3 line encoder can be represented by using two input OR gates.



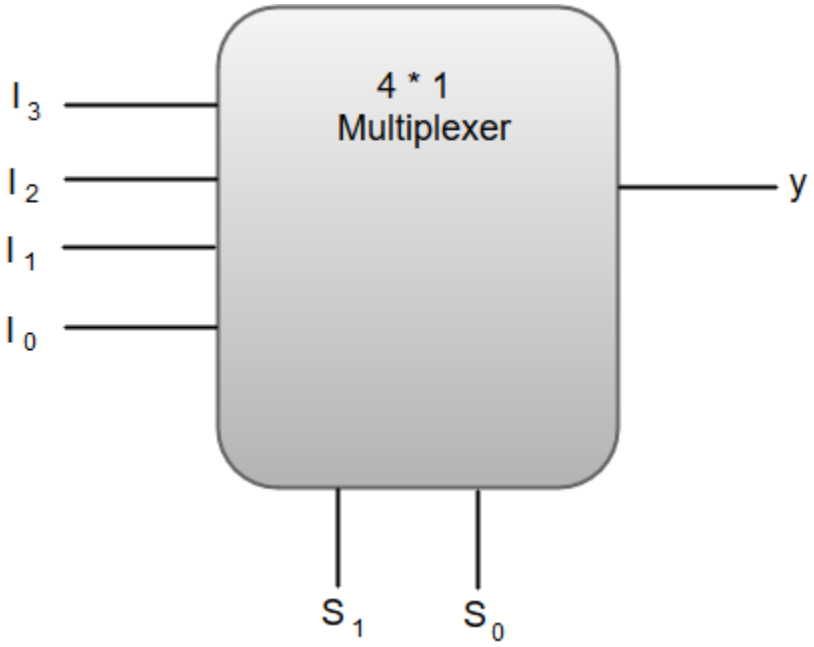
Multiplexers

A Multiplexer (MUX) can be described as a combinational circuit that receives binary information from one of the 2^n input data lines and directs it to a single output line.

The selection of a particular input data line for the output is decided on the basis of selection lines.

The multiplexer is often called as data selector since it selects only one of many data inputs.

The following image shows the block diagram of a 4 * 1 Multiplexer.



Out of these four input data lines, a particular input data line will be connected to the output based on the combination of inputs present at these two selection lines.

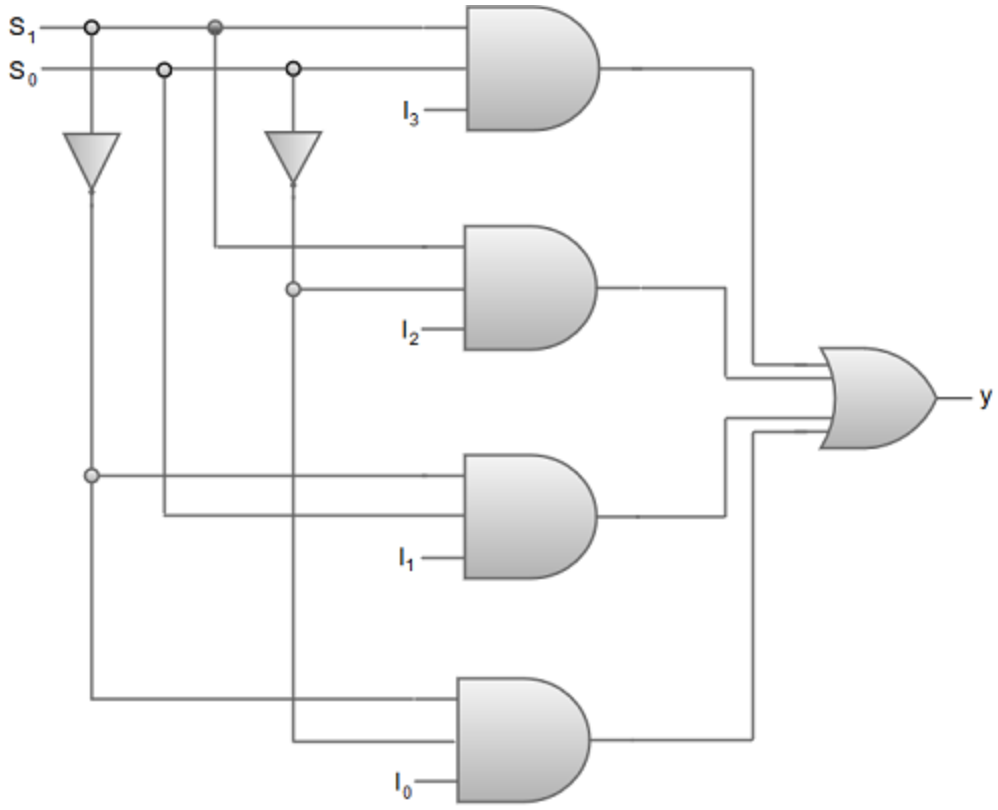
The function table for a 4 * 1 Multiplexer can be represented as:

S1	S0	y
0	0	I0
0	1	I1
1	0	I2
1	1	13

From the function table, we can write the Boolean function for the output (y) as:

```
y = S1'S0'I0 + S1' S0'I1 + S1S0'I2 + S1S0I3
```

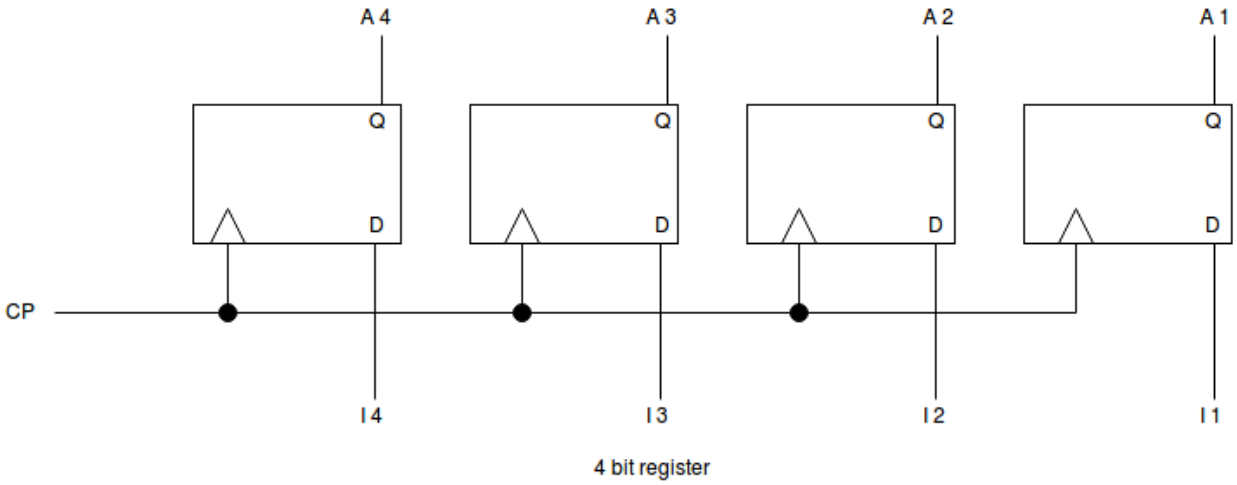
The above equation for output 'y' can be implemented using inverters, three-input AND gates and an OR gate.



$$y = S_1' S_0' I_0 + S_1' S_0 I_1 + S_1 S_0' I_2 + S_1 S_0 I_3$$

Registers

- A Register is a fast memory used to accept, store, and transfer data and instructions that are being used immediately by the CPU.
- A Register can also be considered as a group of flip-flops with each flip-flop capable of storing one bit of information.
- A register with *n flip-flops* is capable of storing binary information of n-bits.
- The flip-flops contain the binary information whereas the gates control the flow of information, i.e. when and how the information?s are transferred into a register.
- Different types of registers are available commercially. A simple register consists of only flip-flops with no external gates.
- The transfer of new data into a register is referred to as loading the register.



4 bit register

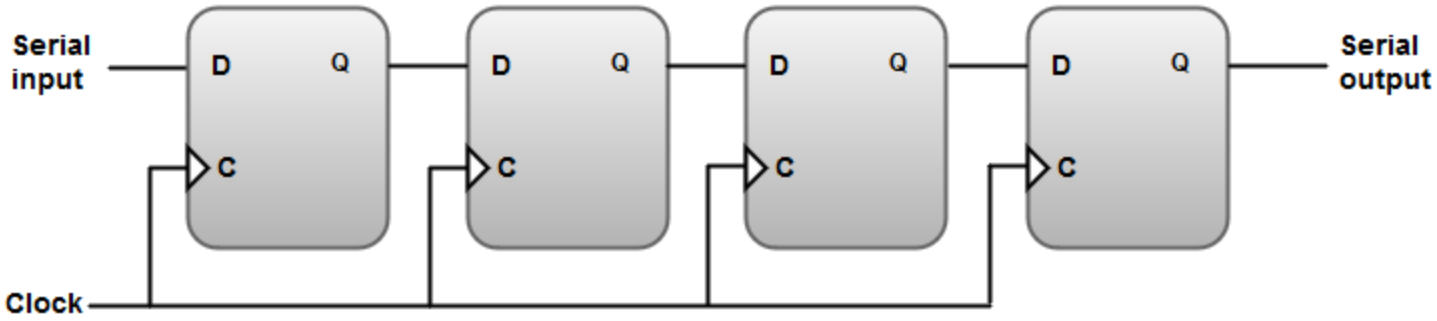
- The above figure shows a register constructed with four D-type flip-flops and a common clock pulse-input.
- The clock pulse-input, CP, enables all flip-flops so that the information presently available at the four inputs can be transferred into the four-bit register.

Shift – Registers

- Shift - Registers are capable of shifting their binary information in one or both directions. The logical configuration of a Shift - Register consists of a series of flip-flops, with the output of one flip-flop connected to the input of the next flip-flop.

The following image shows the block diagram of a Shift - Register and its configuration.

4 - bit Shift-Register:



The basic configuration of a Shift - Register contains the following points:

- The most general Shift - Registers are often referred to as **Bidirectional Shift register with parallel load**.
- A common clock is connected to each register in series to synchronize all operations.
- A serial input line is associated with the left-most register, and a serial output line is associated with the right-most register.
- A control state is connected which leaves the information in the register unchanged even though clock pulses are applied continuously.

Binary Counters

A special type of sequential circuit used to count the pulse is known as a counter, or a collection of flip flops where the clock signal is applied is known as counters.

The counter is one of the widest applications of the flip flop. Based on the clock pulse, the output of the counter contains a predefined state.

The number of the pulse can be counted using the output of the counter.

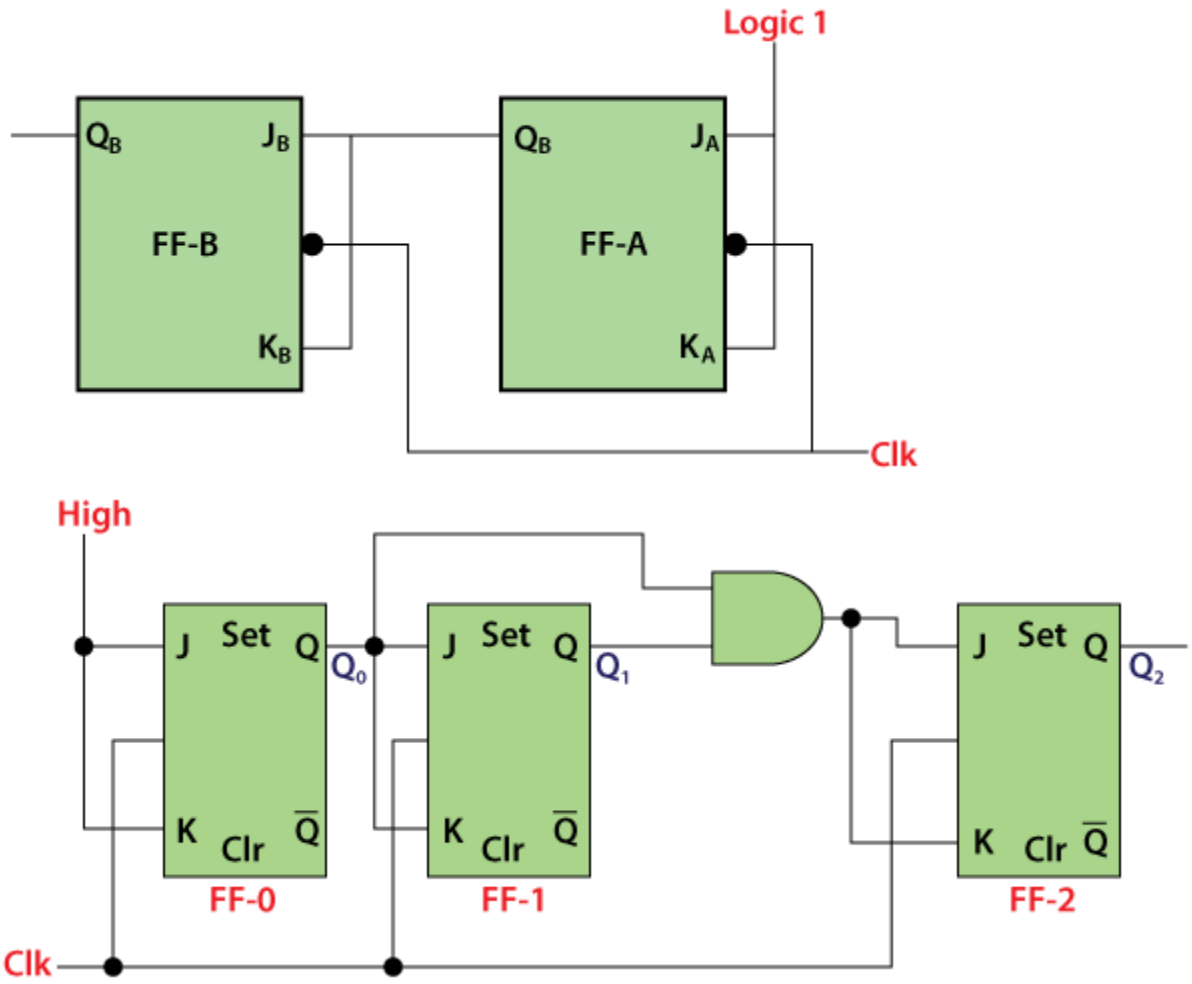
There are the following types of counters:

- Asynchronous Counters
- Synchronous Counters

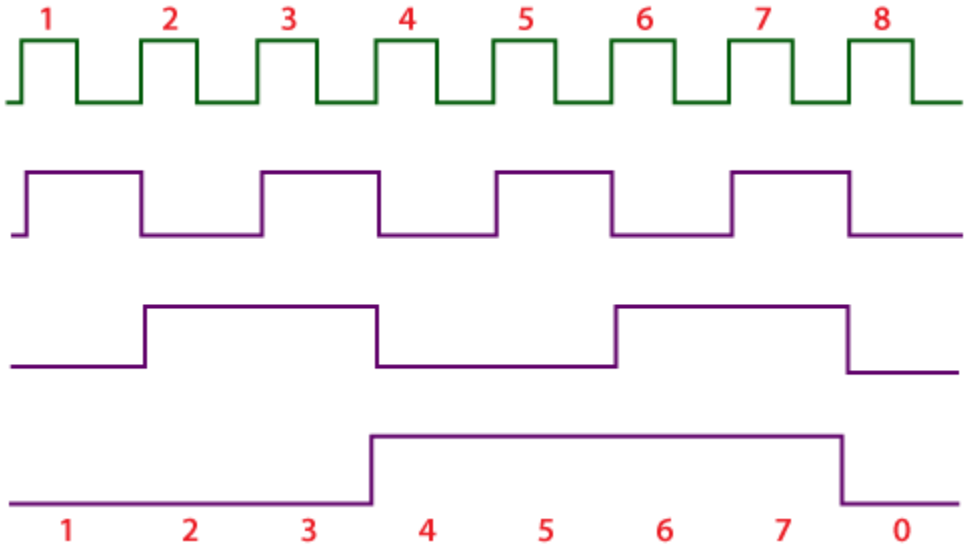
4-bit Synchronous Counters

In the **synchronous counter**, the same clock pulse is passed to the clock input of all the flip flops. The clock signals produced by all the flip flops are the same as each other. Below is the diagram of a 2-bit synchronous counter in which the inputs of the first flip flop, i.e., FF-A, are set to 1. So, the first flip flop will work as a toggle flip-flop. The output of the first flip flop is passed to both the inputs of the next JK flip flop.

Logical Diagram



Signal Diagram



Operation

1. **Condition 1:** When both the flip flops are in reset condition. **Operation:** The outputs of both flip flops, i.e., Q_A Q_B, will be 0. So, $\mathbf{Q_A = 0}$ and $\mathbf{Q_B = 0}$
2. **Condition 2:** When the first negative clock edge passes. **Operation:** The first flip flop will be toggled, and the output of this flip flop will be changed from 0 to 1. When the first negative clock edge is passed, the output of the first flip flop will be 0. The clock input of the first flip flop and both of its inputs will set to 0. In this way, the state of the second flip flop will remain the same. So, $\mathbf{Q_A = 1}$ and $\mathbf{Q_B = 0}$
3. **Condition 2:** When the second negative clock edge is passed. **Operation:** The first flip flop will be toggled again, and the output of this flip flop will be changed from 1 to 0. When the second negative clock edge is passed, the output of the first flip flop will be 1. The clock input of the first flip flop and both of its inputs will set to 1. In this way, the state of the second flip flop will change from 0 to 1. So, $\mathbf{Q_A = 0}$ and $\mathbf{Q_B = 1}$
4. **Condition 2:** When the third negative clock edge passes. **Operation:** The first flip flop will toggle from 0 to 1, but at this instance, both the inputs and the clock input set to 0. Hence, the outputs will remain the same as before. So, $\mathbf{Q_A = 1}$ and $\mathbf{Q_B = 1}$

5. **Condition 2:** When the fourth negative clock edge passes. **Operation:** The first flip flop will toggle from 1 to 0. At this instance, the inputs and the clock input of the second flip flop set to 1. Hence, the outputs will change from 1 to 0. So, $\mathbf{Q_A = 0}$ and $\mathbf{Q_B = 0}$