ELEMENTARY PHYSICS AND DIGITAL ELECTRONICS

UNIT-4

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.

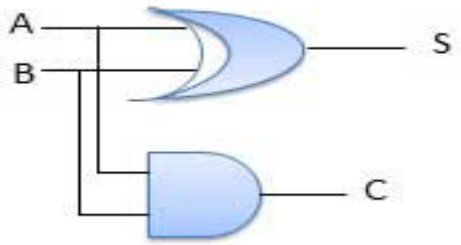
Block diagram

We're going to elaborate few important combinational circuits as follows.

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**.

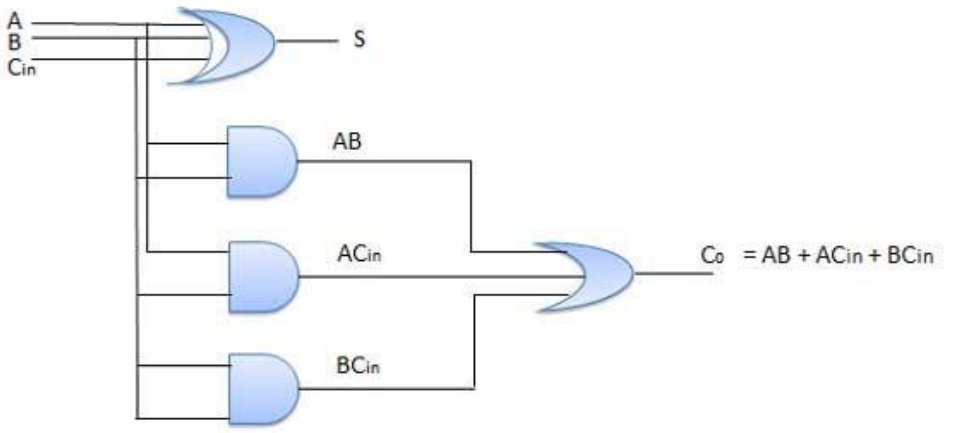
Block diagram



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



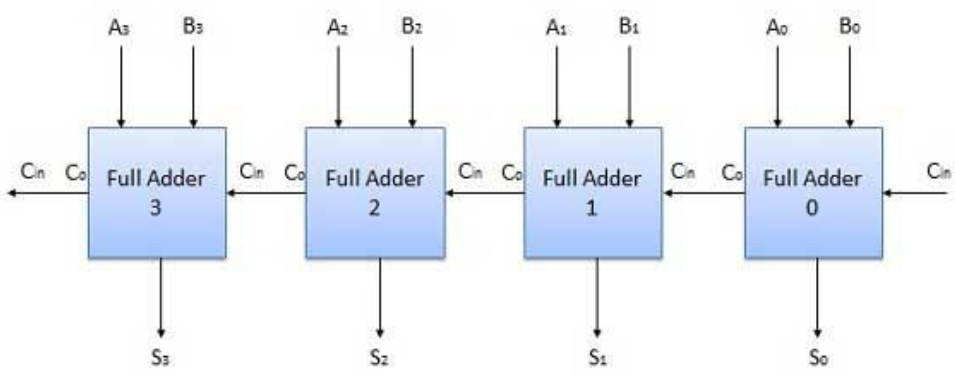
N-Bit Parallel Adder:

The Full Adder is capable of adding only two single digit binary number along with a carry input. But in practical we need to add binary numbers which are much longer than just one bit. To add two n-bit binary numbers we need to use the n-bit parallel adder. It uses a number of full adders in cascade. The carry output of the previous full adder is connected to carry input of the next full adder.

4 Bit Parallel Adder:

In the block diagram, A_0 and B_0 represent the LSB of the four bit words A and B. Hence Full Adder-0 is the lowest stage. Hence its C_{in} has been permanently made 0. The rest of the connections are exactly same as those of n-bit parallel adder is shown in fig. The four bit parallel adder is a very common logic circuit.

Block diagram:



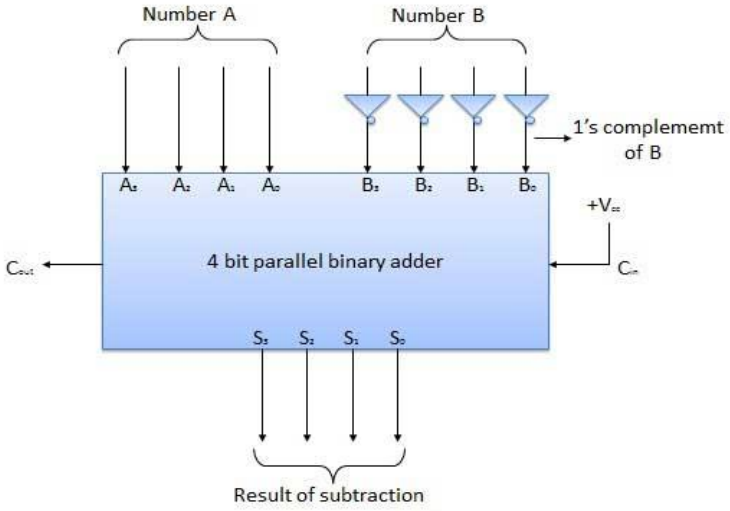
N-Bit Parallel Subtractor:

The subtraction can be carried out by taking the 1's or 2's complement of the number to be subtracted. For example we can perform the subtraction (A-B) by adding either 1's or 2's complement of B to A. That means we can use a binary adder to perform the binary subtraction.

4 Bit Parallel Subtractor:

The number to be subtracted (B) is first passed through inverters to obtain its 1's complement. The 4-bit adder then adds A and 2's complement of B to produce the subtraction. S_3 S_2 S_1 S_0 represents the result of binary subtraction (A-B) and carry output C_{out} represents the polarity of the result. If A > B then Cout = 0 and the result of binary form (A-B) then C_{out} = 1 and the result is in the 2's complement form.

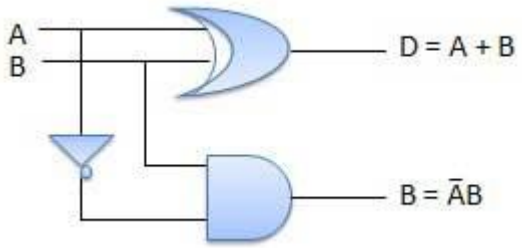
Block diagram:



Half Subtractors:

Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.

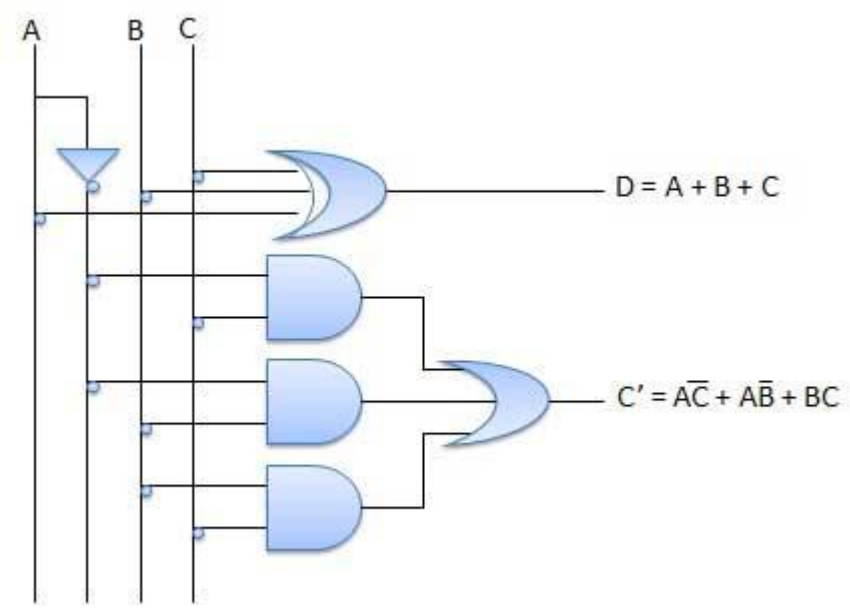
Circuit Diagram:



Full Subtractors:

The disadvantage of a half subtractor is overcome by full subtractor. The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output.

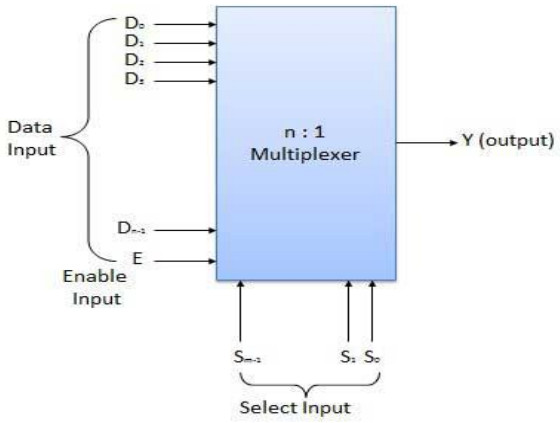
Circuit Diagram:



Multiplexers:

Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with 2m = n. It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs. Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal that means it will perform the required operation when it is low.

Block diagram



Multiplexers come in multiple variations

- 2 : 1 multiplexer

- 4 : 1 multiplexer
- 16 : 1 multiplexer
- 32 : 1 multiplexer

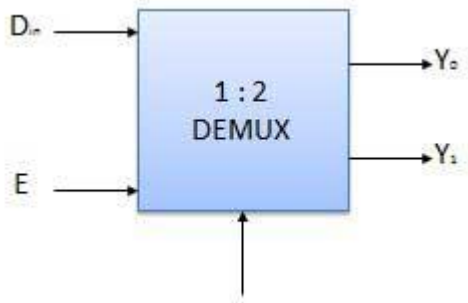
Demultiplexers:

A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. It has only one input, n outputs, m select input. At a time only one output line is selected by the select lines and the input is transmitted to the selected output line. A de-multiplexer is equivalent to a single pole multiple way switch as shown in fig.

Demultiplexers comes in multiple variations.

- 1 : 2 demultiplexer
- 1 : 4 demultiplexer
- 1 : 16 demultiplexer
- 1 : 32 demultiplexer

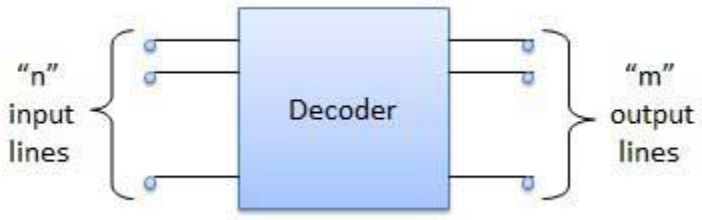
Block diagram



Decoder:

A decoder is a combinational circuit. It has n input and to a maximum m = 2n outputs. Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.

Block diagram



Examples of Decoders are following.

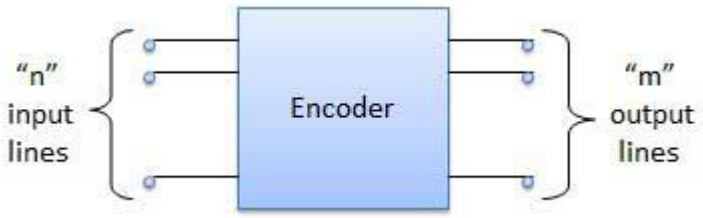
- Code converters

- BCD to seven segment decoders
- Nixie tube decoders
- Relay actuator

Encoder:

Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder. An encoder has n number of input lines and m number of output lines. An encoder produces an m bit binary code corresponding to the digital input number. The encoder accepts an n input digital word and converts it into an m bit another digital word.

Block diagram



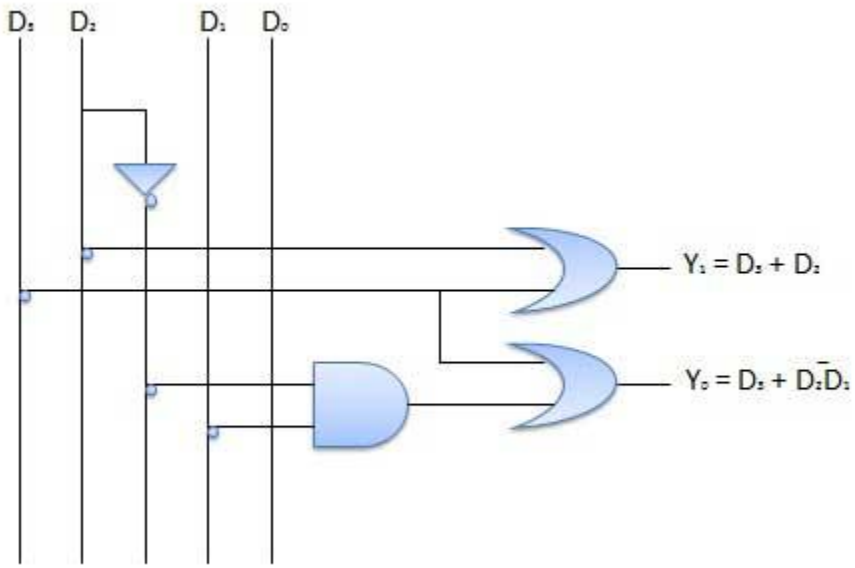
Examples of Encoders are following.

- Priority encoders
- Decimal to BCD encoder
- Octal to binary encoder
- Hexadecimal to binary encoder

Priority Encoder:

This is a special type of encoder. Priority is given to the input lines. If two or more input line are 1 at the same time, then the input line with highest priority will be considered. There are four input D_0, D_1, D_2, D_3 and two output Y_0, Y_1. Out of the four input D_3 has the highest priority and D_0 has the lowest priority. That means if $D_3 = 1$ then Y_1 $Y_1 = 11$ irrespective of the other inputs. Similarly if $D_3 = 0$ and $D_2 = 1$ then Y_1 $Y_0 = 10$ irrespective of the other inputs.

Logic Circuit



Parity Check:

Parity checking, which was created to eliminate data communication errors, has an easy to understand working mechanism. Parity bits are optional and there are no rules for where a parity bit has to be placed, but conventionally, parity bits are added at the end of the data transfer.

How Parity Checking Works:

Imagine a data transfer that looks like this: 1010001. This example has an odd number of 1s and and even number of 0s.

When an even parity checking is used, a parity bit with value 1 could be added to the data's right side to make the number of 1s even -- and the transmission would look like this: 10100011. If an odd parity check was used, the transmission would look like this: 10100010.

RAID

Redundant array of independent disks (RAID) also uses an enhanced form of parity check protection. A second set of parity data is written across all drives to avoid data loss in case of error.

When a RAID drive fails its parity check, data is rebuilt using parity information coupled with data on the other disks. The bits on the remaining drives are added up. If they add up to an odd number, the correct information on the failed drive has to be even, and vice-versa, for communication to continue.

Limitations

Parity checking is primarily used for communications, although more advanced protocols such as the Microcom Networking Protocols (MNP) and ITU-T V.42b has supplanted it as the standard in modem-to-modem communication.

Although parity checking provides a very basic method for detecting simple errors, it cannot, for example, detect errors caused by electrical noise changing the number of bits. It might happen, in fact, that both the receiving and sending bits are in error, offsetting each other.

Although the chance for this to happen in a PC is basically remote, in large computer systems where there's an essential need to ensure data integrity, a third bit could be allocated for parity checking.

Magnitude Comparator:

A magnitude digital Comparator is a combinational circuit that **compares two digital or binary numbers** in order to find out whether one binary number is equal, less than, or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and the other for B and have three output terminals, one for A > B condition, one for A = B condition, and one for A < B condition.

1-Bit Magnitude Comparator:

A comparator used to compare two bits is called a single-bit comparator. It consists of two inputs each for two single-bit numbers and three outputs to generate less than, equal to, and greater than between two binary numbers.

The truth table for a 1-bit comparator is given below:

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

From the above truth table logical expressions for each output can be expressed as follows:

A>B: AB'
A<B: A'B
A=B: A'B' + AB

2-Bit Magnitude Comparator:

A comparator used to compare two binary numbers each of two bits is called a 2-bit Magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to, and greater than between two binary numbers.

4-Bit Magnitude Comparator:

A comparator used to compare two binary numbers each of four bits is called a 4-bit magnitude comparator. It consists of eight inputs each for two four-bit numbers and three outputs to generate less than, equal to, and greater than between two binary numbers.

In a 4-bit comparator the condition of A>B can be possible in the following four cases:

1. If A3 = 1 and B3 = 0
2. If A3 = B3 and A2 = 1 and B2 = 0
3. If A3 = B3, A2 = B2 and A1 = 1 and B1 = 0
4. If A3 = B3, A2 = B2, A1 = B1 and A0 = 1 and B0 = 0

Similarly, the condition for A<B can be possible in the following four cases:

5. If A3 = 0 and B3 = 1
6. If A3 = B3 and A2 = 0 and B2 = 1
7. If A3 = B3, A2 = B2 and A1 = 0 and B1 = 1
8. If A3 = B3, A2 = B2, A1 = B1 and A0 = 0 and B0 = 1

The condition of A=B is possible only when all the individual bits of one number exactly coincide with corresponding bits of another number.

Cascading Comparator:

A comparator performing the comparison operation to more than four bits by cascading two or more 4-bit comparators is called a cascading comparator. When two comparators are to be cascaded, the outputs of the lower-order comparator are connected to corresponding inputs of the higher-order comparator.

Applications of Comparators:

1. Comparators are used in central processing units (CPUs) and microcontrollers (MCUs).
2. These are used in control applications in which the binary numbers representing physical variables such as temperature, position, etc. are compared with a reference value.
3. Comparators are also used as process controllers and for Servo motor control.
4. Used in password verification and biometric applications.

Sequential circuits:

The sequential circuit is a special type of circuit that has a series of inputs and outputs. The outputs of the sequential circuits depend on both the combination of present inputs and previous outputs. The previous output is treated as the present state. So, the sequential circuit contains the combinational circuit and its memory storage elements. A sequential circuit doesn't need to always contain a combinational circuit. So, the sequential circuit can contain only the memory element.

Difference between the combinational circuits and sequential circuits are given below:

S.No.	Combinational Circuits	Sequential Circuits
1)	The outputs of the combinational circuit depend only on the present inputs.	The outputs of the sequential circuits depend on both present inputs and present state(previous output).
2)	The feedback path is not present in the combinational circuit.	The feedback path is present in the sequential circuits.
3)	In combinational circuits, memory elements are not required.	In the sequential circuit, memory elements play an important role and require.
4)	The clock signal is not required for combinational circuits.	The clock signal is required for sequential circuits.
5)	The combinational circuit is simple to design.	It is not simple to design a sequential circuit.

Types of Sequential Circuits:

1. Asynchronous sequential circuits

The clock signals are not used by the **Asynchronous sequential circuits**. The asynchronous circuit is operated through the pulses. So, the changes in the input can change the state of the circuit. The asynchronous circuits do not use clock pulses. The internal state is changed when the input variable is changed. The un-clocked flip-flops or time-delayed are the memory elements of asynchronous sequential circuits. The asynchronous sequential circuit is similar to the combinational circuits with feedback.

2. Synchronous sequential circuits

In synchronous sequential circuits, synchronization of the memory element's state is done by the clock signal. The output is stored in either flip-flops or latches(memory devices). The synchronization of the outputs is done with either only negative edges of the clock signal or only positive edges.

Latches:

A **Latch** is a special type of logical circuit. The latches have **low** and **high** two stable states. Due to these states, latches also refer to as **bistable-multivibrators**. A latch is a storage device that holds

the data using the feedback lane. The latch stores 1 -bit until the device set to 1. The latch changes the stored data and constantly trials the inputs when the enable input set to 1.

Based on the enable signal, the circuit works in two states. When the enable input is high, then both the inputs are low, and when the enable input is low, both the inputs are high.

Types of Latches:

There are various types of latches used in digital circuits which are as follows:

1. SR Latch

The SR latch is a special type of asynchronous device which works separately for control signals. It depends on the S-states and R-inputs. The SR latch design by connecting two NOR gates with a cross loop connection. The SR latch can also be designed using the NAND gate.

2. Gated SR Latch

A Gated SR Latch is a special type of SR Latch having three inputs, i.e., Set, Reset, and Enable. The enable input must be active for the SET and RESET inputs to be effective. The ENABLE input of gated SR Latch enables the operation of the SET and RESET inputs. This ENABLE input connects with a switch. The Set-Reset inputs are enabled when this switch is on. Otherwise, all the changes are ignored in the set and reset inputs.

3. D Latch

The D latch is the same as D flip flop. The only difference between these two is the ENABLE input. The output of the latch is the same as the input passed to the Data input when the ENABLE input set to 1. At that time, the latch is open, and the path is transparent from input to output. If the ENABLE input is set to 0, the D latch's output is the last value of the latch, i.e., independent from the input D, and the latch is closed.

4. Gated D Latch

The Gated D Latch is another special type of gated latch having two inputs, i.e., DATA and ENABLE. When the enable input set to 1, the input is the same as the Data input. Otherwise, there is no change in output.

We can design the gated D latch by using gated SR latch. The set and reset inputs are connected together using an inverter. By doing this, the outputs will be opposite to each other.

5. JK Latch

The JK Latch is the same as the SR Latch. In JK latch, the unclear states are removed, and the output is toggled when the JK inputs are high. The only difference between SR latch JK latches is that there is no output feedback towards the inputs in the SR latch, but it is present in the JK latch.

6. T Latch

The T latch forms by shorting the JK latches inputs. The output of the T latch toggle when the input set to 1 or high.

RS flip-flop:

In this flip-flop circuit an additional control input is applied. This additional control input determines the when the state of the circuit is to be changed. This additional input is nothing but the clock pulse.

The RS flip-flop consists of basic flip-flop circuit along with two additional **NAND gates** and a clock pulse generator. The clock pulse acts as an enable signal for the two inputs. The output of the gates 3 and 4 remains at logic "1" until the clock pulse input is at 0.This is nothing but the quiescent condition of the flip-flop.

Information from S and R is allowed to reach the output only when clock pulse goes to 1.

Let's assume S=1, R=0 and CP=1. The set state is reached at this condition and since the clock pulse is 1, information from S and R is allowed to reach output.

From the truth table of NAND gate we can say that the output is 0 only when both the inputs are 1. In all the other case the output is 1.

So when S=1, R=0 &CP=1. Both the inputs to the gate 3 are 1 and hence its output is 0. This information (i.e.) 0 is passed to gate 1. Since one of the inputs of gate 1 is 0, we can say that the output Q=1. Since R=0, the output obtained at Q'=0.

Conclusion: When S=1, R=0 & CP=1 => Q=1 and Q'=0.

Now to change to reset state, the inputs must be S=0, R=1 & CP=1. The observed outputs are Q=0 & Q'=1.When the clock pulse returns to zero, the circuit remains in its previous state. This is applicable to both Set and Clear states.

Now when CP=1, inputs S=0 &R=0, that is when both the inputs are 0, the state of the circuit does not change.

When CP=1, S=1 & R=1, an indeterminate condition occurs. Because both the outputs Q and Q' remain at 1. This is not possible because both the outputs are complementary to each other. So it is better to avoid this condition during practise.

These results can be compared with those in characteristic table; Where S and R are inputs. Q is the output and Q (t+1) shows the next state. The characteristic table should be interpreted as: for the given present state Q, the inputs S and R, the application of a single clock pulse CP causes the flip-flop to go to the next state.

D Flip-Flop:

D flip-flops are used to eliminate the indeterminate state that occurs in RS Flip-flop. D flip-flop ensures that R and S are never equal to one at the same time. The D flip-flop has two inputs including the Clock pulse. D and CP are the two inputs of the D flip-flop.

The D input of the flip-flop is directly given to S. And the complement of this value is given as the R input. Similar to Rs flip-flop, the outputs of gate 3 and 4 remain at logic “1” until the clock pulse applied is 0. The value of D won’t affect the circuit until Cp is in 0. The value of D is sampled only when CP goes from 0 to 1. Now as soon as the value of Cp changes to 1, the value of D is sampled and the information is passed to the output.

Now assume that CP=1. Now if D=1, the output of gate 3 goes to 0 and this makes the output Q=1, Q’=0. This is the set state of the D flip-flop. When D=0, the output Q’=1 and Q=0, which is the reset state of the Flip-flop.

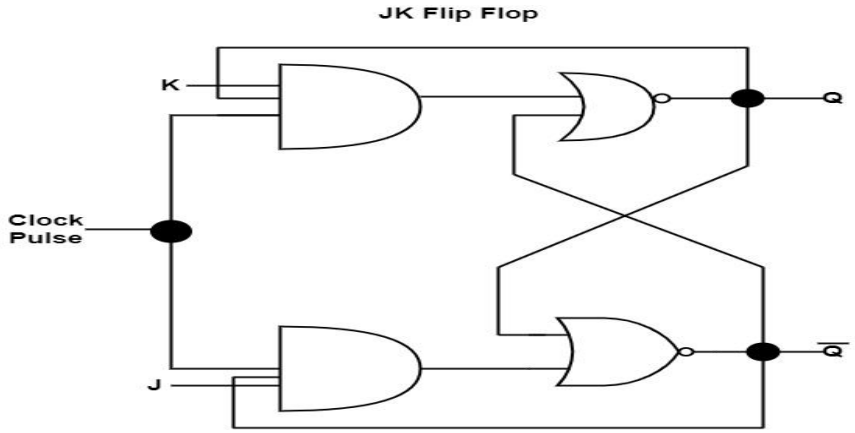
Since both S and R are given complementary values they can never be 1 at the same time. Thus we can avoid the indeterminate state that occurs in RS flip-flop.

When the clock pulse returns to zero, the previous state of the output is maintained (or) the output does not change its state unless it is enabled again by clock pulse. This Flip-flop is sometimes called Gated D-latch.

J-K flip-flop:

J-K flip-flop can be treated as an alteration of the S-R flip-flop. J represents SET, and 'K' represents CLEAR. In the JK flip-flop, the ‘S’ input is known as the ‘J’ input, and the ‘R’ input is known as the ‘K’ input. The output of the JK flip-flop does not modify if both ‘J’ and ‘K’ are ‘0’. If both the inputs are ‘1’, then the output dial to its free.

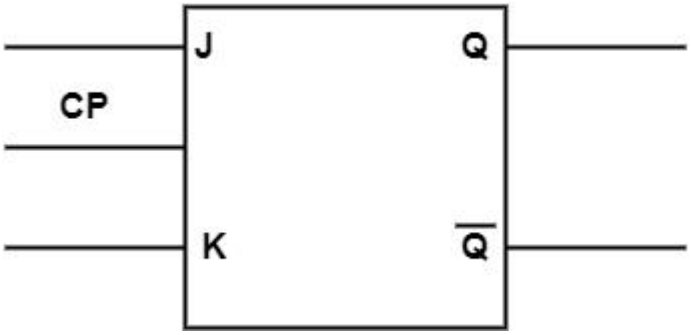
The figure shows the circuit diagram of a JK flip-flop.



The truth table of the JK flip-flop is displayed in the table.

S	R	Q_{N-1}
0	0	Q_N
0	1	0
1	0	1
1	1	Q_N--------QN^-

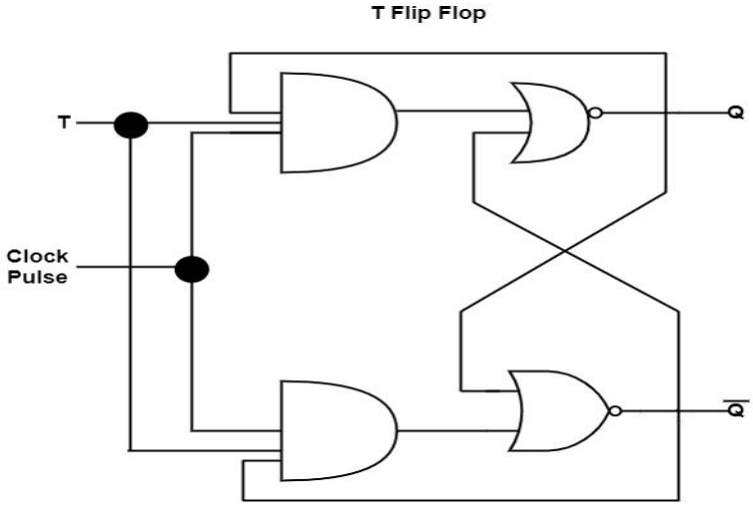
The logic symbol for the JK flip-flop is demonstrated in the diagram.



T flip-flop:

The T flip-flop is also called toggle flip-flop. It is a change of the JK flip-flop. The T flip flop is received by relating both inputs of a JK flip-flop. The T flip-flop is received by relating the inputs 'J' and 'K'. When T = 0, both AND gates are disabled. Therefore, there is no change in the output. When T= 1, the output toggles.

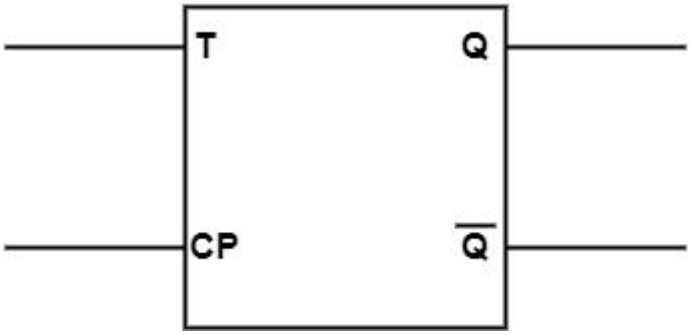
The diagram demonstrates the circuit diagram of a T flip-flop.



The truth table of T flip-flop is displayed in the table.

Q_N	T	Q_{N+1}
0	0	0
0	1	1
1	0	1
1	1	0

The logic symbol of the T flip-flop is shown in the figure.



Master Slave JK flip flop:

The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration. Out of these, one acts as the "master" and the other as a **"slave"**. The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop.

In addition to these two flip-flops, the circuit also includes an **inverter**. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop. In other words if CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and if CP=1 for master flip flop then it becomes 0 for slave flip flop.

Working of a master slave flip flop:

1. When the clock pulse goes to 1, the slave is isolated; J and K inputs may affect the state of the system. The slave flip-flop is isolated until the CP goes to 0. When the CP goes back to 0, information is passed from the master flip-flop to the slave and output is obtained.
2. Firstly the master flip flop is positive level triggered and the slave flip flop is negative level triggered, so the master responds before the slave.
3. If J=0 and K=1, the high Q' output of the master goes to the K input of the slave and the clock forces the slave to reset, thus the slave copies the master.
4. If J=1 and K=0, the high Q output of the master goes to the J input of the slave and the Negative transition of the clock sets the slave, copying the master.

5. If J=1 and K=1, it toggles on the positive transition of the clock and thus the slave toggles on the negative transition of the clock.
6. If J=0 and K=0, the flip flop is disabled and Q remains unchanged.

Registers:

A **Register** is a collection of flip flops. A flip flop is used to store single bit digital data. For storing a large number of bits, the storage capacity is increased by grouping more than one flip flops. If we want to store an n-bit word, we have to use an n-bit register containing n number of flip flops.

The register is used to perform different types of operations. For performing the operations, the CPU use these registers. The faded inputs to the system will store into the registers. The result returned by the system will store in the registers. There are the following operations which are performed by the registers:

Fetch:

It is used

- To take the instructions given by the users.
- To fetch the instruction stored into the main memory.

Decode:

The decode operation is used to interpret the instructions. In decode, the operation performed on the instructions is identified by the CPU. In simple words, the decode operation is used to decode the instructions.

Execute:

The execution operation is used to store the result produced by the CPU into the memory. After storing this result, it is displayed on the user screen.

Types of Registers:

1. MAR or Memory Address Register

The MAR is a special type of register that contains the memory address of the data and instruction. The main task of the MAR is to access instruction and data from memory in the execution phase. The MAR stores the address of the memory location where the data is to be read or to be stored by the CPU.

2. Program Counter

The program counter is also called an instruction address register or instruction pointer. The next memory address of the instruction, which is going to be executed after completing the execution of current instruction is contained in the program counter. In simple words, the program counter contains the memory address of the location of the next instruction.

3. Accumulator Register

The CPU mostly uses an accumulator register. The accumulator register is used to store the system result. All the results will be stored in the accumulator register when the CPU produces some results after processing.

4. MDR or Memory Data Register

Memory Data Register is a part of the computer's control unit. It contains the data that we want to store in the computer storage or the data fetched from the computer storage. The MDR works as a buffer that contains anything for which the processor is ready to use it. The MDR contains the copied data of the memory for the processor. Firstly the MDR holds the information, and then it goes to the decoder.

The data which is to be read out or written into the address location is contained in the **Memory Data Register**.

The data is written in one direction when it is fetched from memory and placed into the MDR. In write instruction, the data place into the MDR from another CPU register. This CPU register writes the data into the memory. Half of the minimal interface between the computer storage and the microprogram is the memory data address register, and the other half is the memory data register.

5. Index Register

The **Index Register** is the hardware element that holds the number. The number adds to the computer instruction's address to create an effective address. In CPU, the index register is a processor register used to modify the operand address during the running program.

6. Memory Buffer Register:

Memory Buffer Register is mostly called MBR. The MBR contains the Metadata of the data and instruction written in or read from memory. In simple words, it adds is used to store the upcoming data/instruction from the memory and going to memory.

7. Data Register

The data register is used to temporarily store the data. This data transmits to or from a peripheral device.

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.

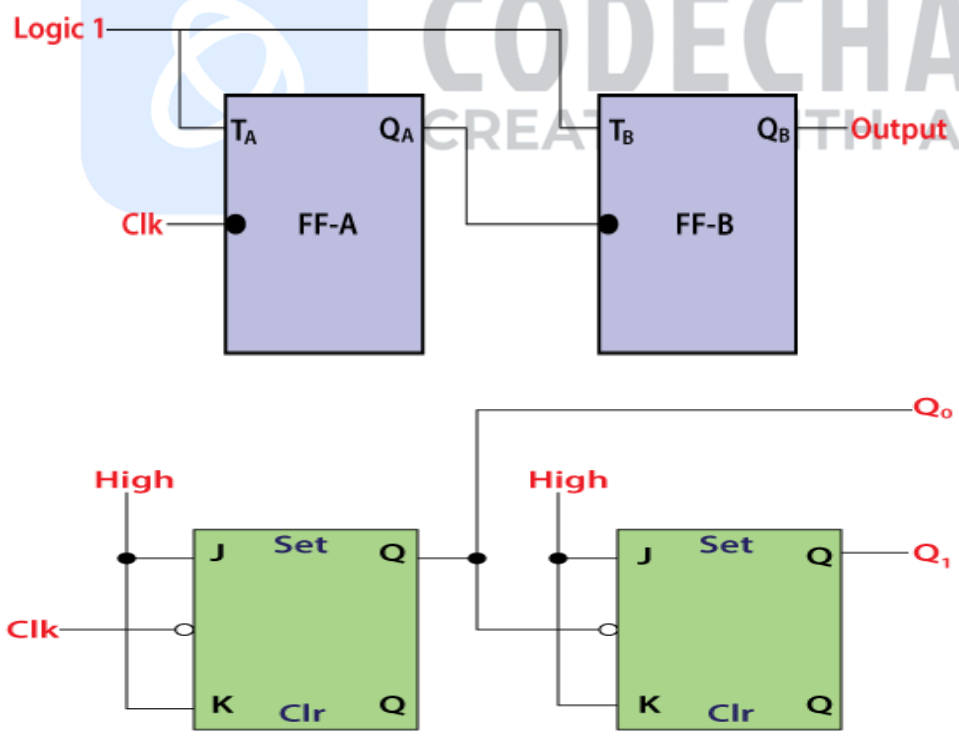
Truth Table

Clock	Counter output		State number	Decimal counter output
	Q_B	Q_A		
Initially	0	0	-	0
1st	0	1	1	1
2nd	1	0	2	2
3rd	1	1	3	3
4th	0	0	4	0

1. Asynchronous or ripple counters

The **Asynchronous counter** is also known as the **ripple counter**. Below is a diagram of the 2-bit **Asynchronous counter** in which we used two T flip-flops. Apart from the T flip flop, we can also use the JK flip flop by setting both of the inputs to 1 permanently. The external clock pass to the clock input of the first flip flop, i.e., FF-A and its output, i.e., is passed to clock input of the next flip flop, i.e., FF-B.

Block 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.

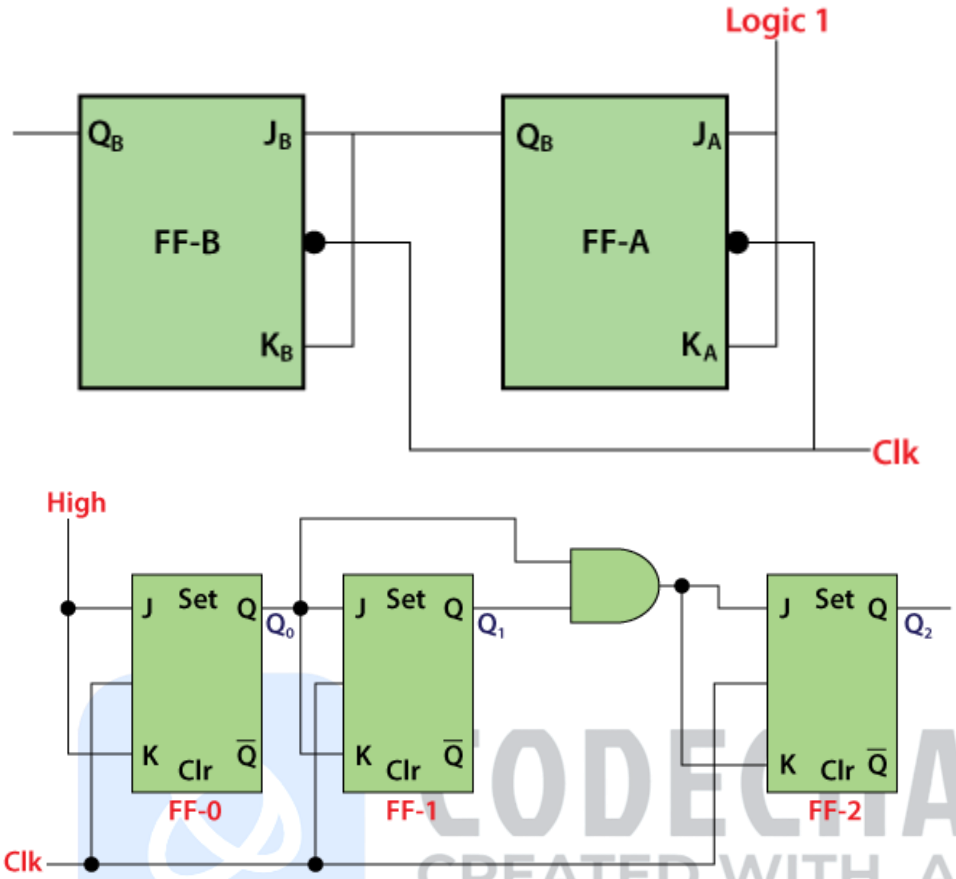
2. **Condition 2:** When the first negative clock edge passes.
   Operation: The first flip flop will toggle, and the output of this flip flop will change from 0 to 1. The output of this flip flop will be taken by the clock input of the next flip flop. This output will be taken as a positive edge clock by the second flip flop. This input will not change the second flip flop's output state because it is the negative edge triggered flip flop.
   So, $Q_A = 1$ and $Q_B = 0$
3. **Condition 3:** When the second negative clock edge is applied.
   Operation: The first flip flop will toggle again, and the output of this flip flop will change from 1 to 0. This output will be taken as a negative edge clock by the second flip flop. This input will change the second flip flop's output state because it is the negative edge triggered flip flop.
   So, $Q_A = 0$ and $Q_B = 1$.
4. **Condition 4:** When the third negative clock edge is applied.
   Operation: The first flip flop will toggle again, and the output of this flip flop will change from 0 to 1. This output will be taken as a positive edge clock by the second flip flop. This input will not change the second flip flop's output state because it is the negative edge triggered flip flop.
   So, $Q_A = 1$ and $Q_B = 1$
5. **Condition 5:** When the fourth negative clock edge is applied.
   Operation: The first flip flop will toggle again, and the output of this flip flop will change from 1 to 0. This output will be taken as a negative edge clock by the second flip flop. This input will change the output state of the second flip flop.
   So, $Q_A = 0$ and $Q_B = 0$

2. Synchronous counters

In the **Asynchronous counter**, the present counter's output passes to the input of the next counter. So, the counters are connected like a chain. The drawback of this system is that it creates the counting delay, and the propagation delay also occurs during the counting stage. The **synchronous counter** is designed to remove this drawback.

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



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 $\mathbf{Q_B = 0}$