UNIT – III

Digital Electronics: Logic Gates, Simple combinational circuits–Half and Full Adders, BCD Adder. Latches and Flip-Flops (S-R, JK and D), Shift Registers and Counters. Introduction to Microcontrollers and their applications (Block diagram approach only).

****

Logic Gates

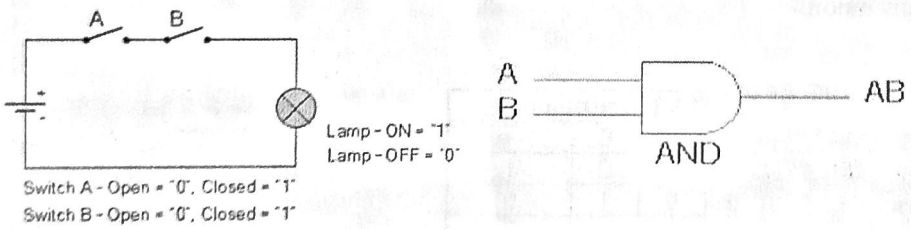
Boolean functions may be practically implemented by using electronic gates. The following points are important to understand.

- Electronic gates require a power supply.
- Gate **INPUTS** are driven by voltages having two nominal values, e.g. 0V and 5V representing logic 0 and logic 1 respectively.
- The **OUTPUT** of a gate provides two nominal values of voltage only, e.g. 0V and 5V representing logic 0 and logic 1 respectively. In general, there is only one output to a logic gate except in some special cases.
- There is always a time delay between an input being applied and the output responding.

Logic gates

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables.

AND gate

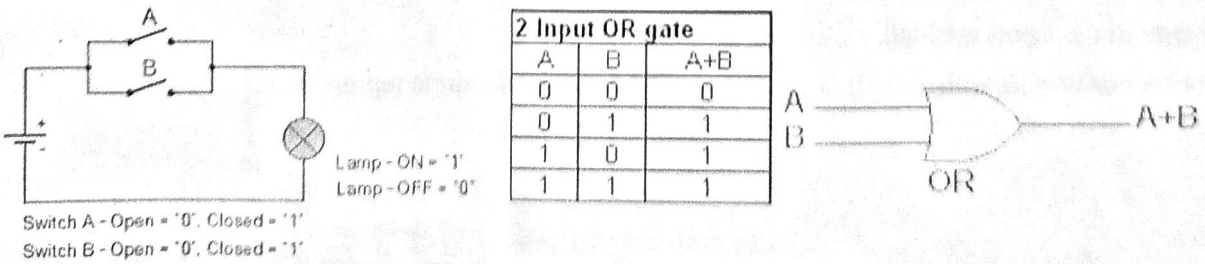


2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3.1: AND Gate operation

The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB

OR gate

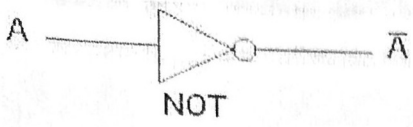


2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

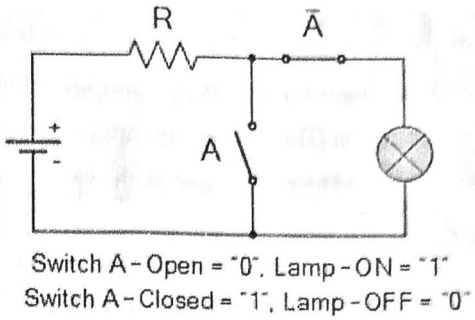
Figure 3.2: OR Gate operation

The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation

NOT gate



NOT gate	
A	$\bar{A}$
0	1
1	0

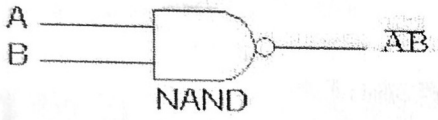


Switch A - Open = "0", Lamp - ON = "1"
Switch A - Closed = "1", Lamp - OFF = "0"

Figure 3.3: NOT Gate operation

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs.

NAND gate

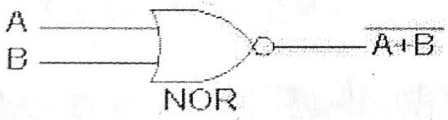


2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Figure 3.4:NAND Gate operation

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

NOR gate



2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

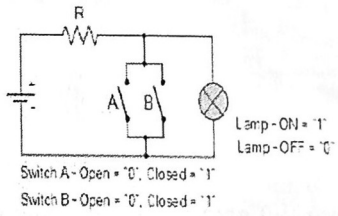
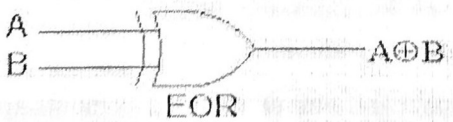


Figure 3.5:NOR Gate operation

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high.

The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

EXOR gate

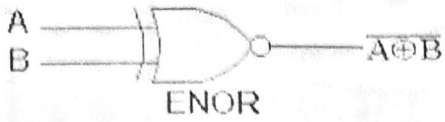


2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Figure 3.6: EXOR Gate operation

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (⊕) is used to show the EOR operation.

EXNOR gate

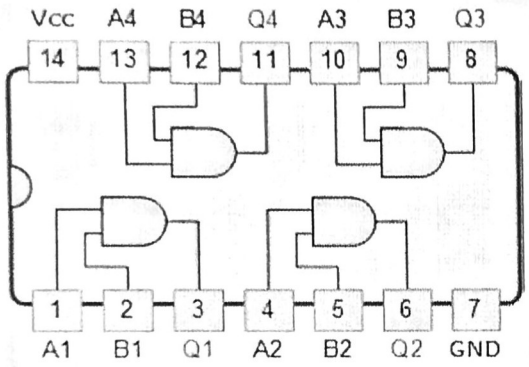


2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

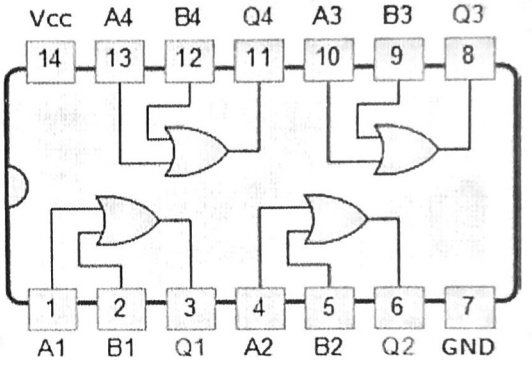
Figure 3.7: ENOR Gate operation

The '**Exclusive-NOR**' gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

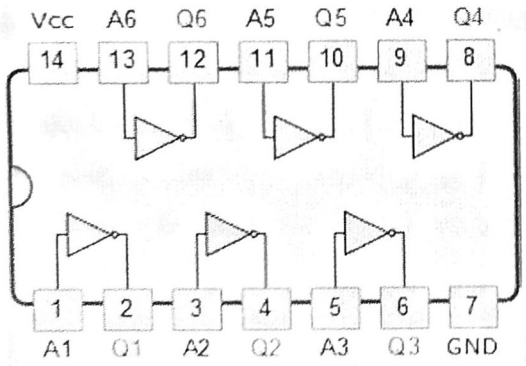
The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.



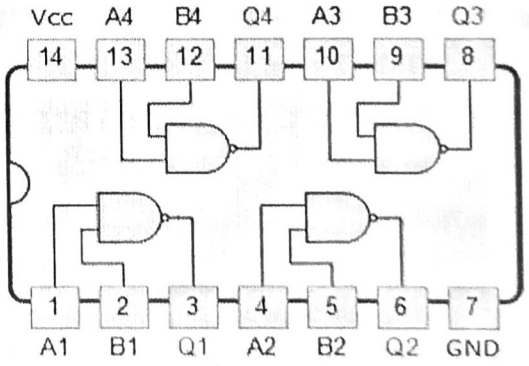
7408 Quad 2-input AND Gate



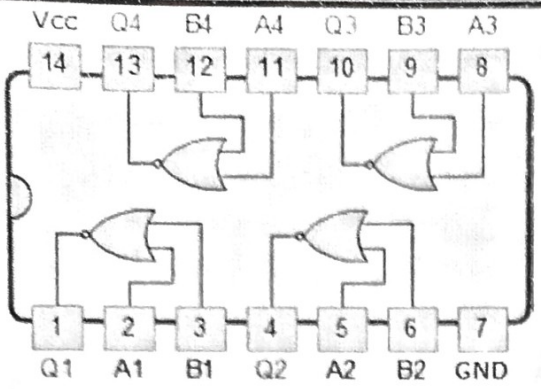
7432 Quad 2-input Logic OR Gate



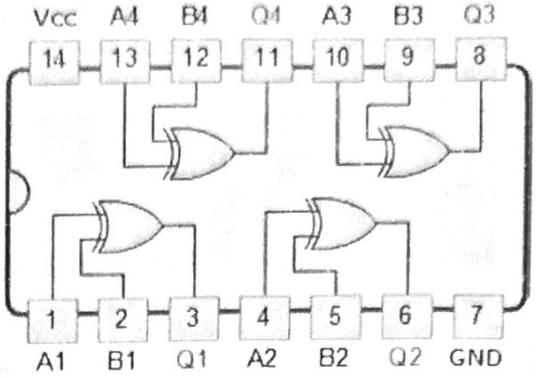
7404 NOT Gate or Inverter



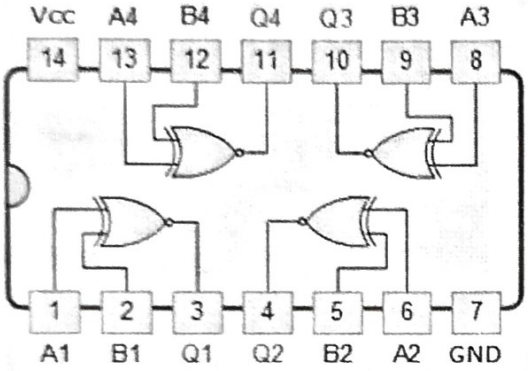
7400 Quad 2-input Logic NAND Gate



7402 Quad 2-input NOR Gate



7486 Quad 2-input Exclusive-OR Gate



74266 Quad 2-input Ex-NOR Gate

Figure 3.8:Pin diagram of Logic Gates

HALF ADDER

The simplest adder, called a *half adder*, adds two 1-bit operands A and B, producing a 2-bit sum. The sum can range from 0 to 2, which requires two bits to express. The low-order bit of the sum may be named HS (half sum), and the high-order bit may be named C_O (carry out). We can write the following equations for SUM and CARRY:

SUM<=A XOR B;

CARRY<=A AND B;

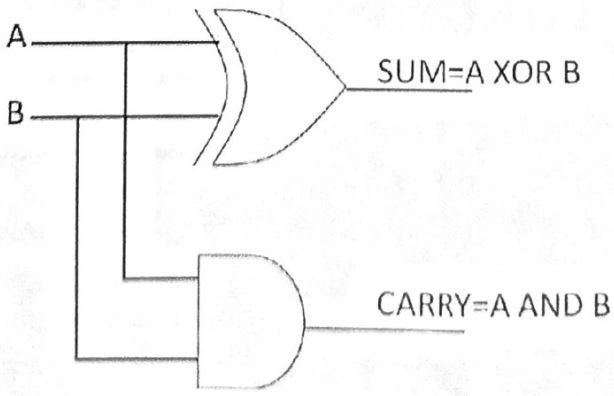
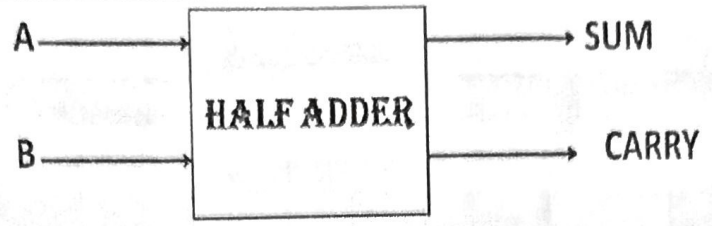
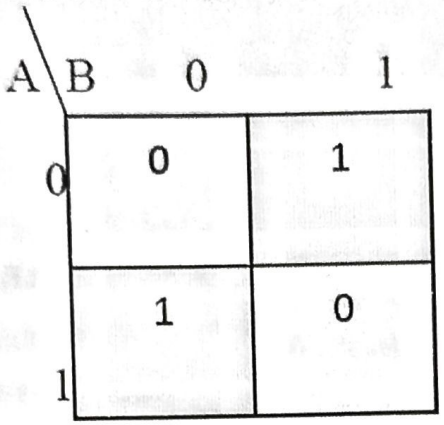
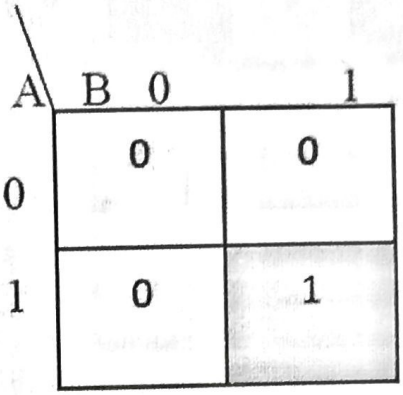


Figure 3.9:Half Adder Logic diagram







TRUTH TABLE

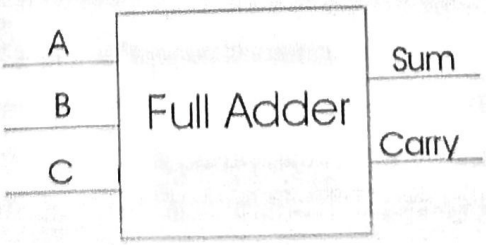
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FULL ADDER

Full adder is a digital circuit used to calculate the sum of three binary bits which is the main difference between this and half adder. Full adders are complex and difficult to implement when compared to half adders. Two of the three bits are same as before which are A, the augend bit and B, the addend bit. The additional third bit is carry bit from the previous stage and is called Carry – in generally represented by CIN. It calculates the sum of three bits along with the carry. The output carry is called Carry – out and is represented by COUT.

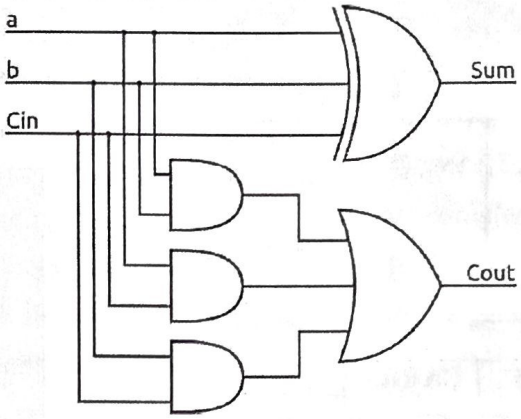
SUM<=A XOR B XOR C;

CARRY<=AB+BC+AC;



A \ BC_{IN}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

A \ BC_{IN}	00	01	11	10
0	0	0	1	0
1	0	1	1	1



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

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

$$= C_{in}\,(A \odot B) + \bar{C}_{in}\,(A \oplus B)$$

$$= C_{in}\,(\overline{A \oplus B}) + \bar{C}_{in}\,(A \oplus B)$$

$$= C_{in} \oplus (A \oplus B)$$

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 3.10: Full Adder Logic diagram

BCD Adder

BCD stand for binary coded decimal. Suppose, two 4-bit numbers A and B. The value of A and B can varies from 0(0000 in binary) to 9(1001 in binary) because we are considering decimal numbers.

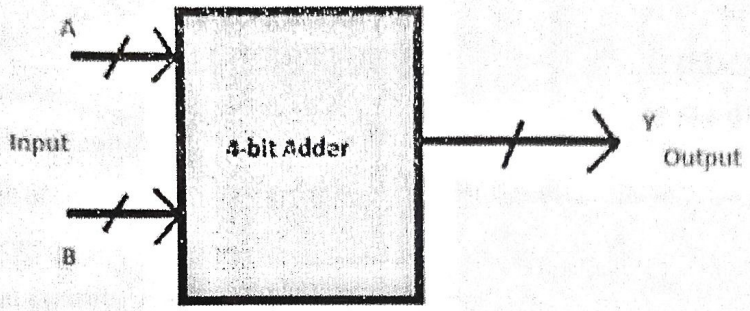


Figure 3.11: BCD Addition

The output will varies from 0 to 18, if not considering the carry from the previous sum. But if considering the carry, then the maximum value of output will be 19 (i.e. 9+9+1 = 19).

When we are simply adding A and B, then we get the binary sum. Here, to get the output in BCD form, we will use BCD Adder.

Decimal	Binary Sum					BCD Sum				
	C'	S3'	S2'	S1'	S0'	C	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

When C'= 1, it indicates that the sum is greater than 9 or if there is a carry. When C'= 1, a binary 0110 is added to the binary sum through the second adder. A decimal parallel adder that adds n' decimal digit needs n' BCD adder stages. A BCD adder that adds two BCD digits and produces a sum digit in BCD is shown in Figure. The two decimal digits, together with the input carry, are first added in the top four-bit adder to produce the binary sum. When the output carry is equal to 0, nothing is added to the binary sum. When it is equal to 1, binary 0110 is added to the binary sum through the bottom four-bit adder. The output carry

generated from the bottom adder can be ignored, since it supplies information already available at the output carry terminal.

The conditions are:

1. If C' = 1 (Satisfies 16-19)
2. If $S_3'.S_2' = 1$ (Satisfies 12-15)
3. If $S_3'.S_1' = 1$ (Satisfies 10 and 11)

So, our logic is

$$\mathbf{C' + S_3'.S_2' + S_3'.S_1' = 1}$$

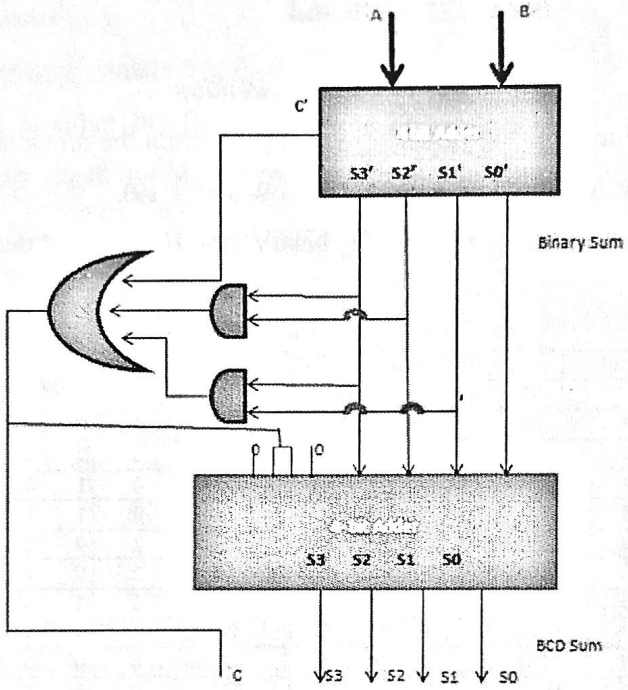


Figure 3.12: BCD Adder

Introduction to Sequential Circuits

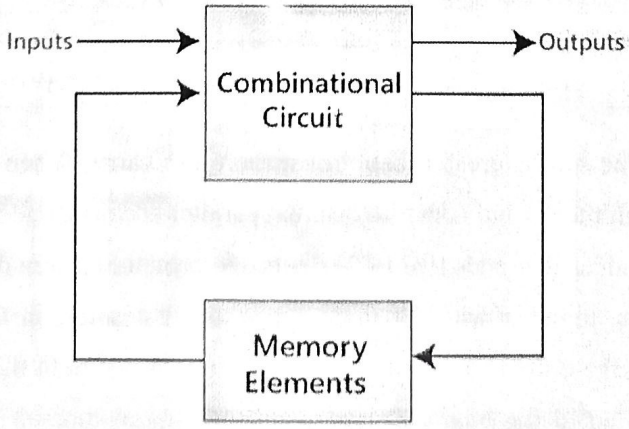


Figure 3.13. Sequential Circuit

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.

Types of Sequential Circuits

Asynchronous sequential circuits

The clock signals are not used by the **Asynchronous sequential circuits**. The internal state is changed when the input variable is changed.

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.

Clock signal

A clock signal is a periodic signal in which ON time and OFF time need not be the same. When ON time and OFF time of the clock signal are the same, a square wave is used to represent the clock signal. Below is a diagram which represents the clock signal:

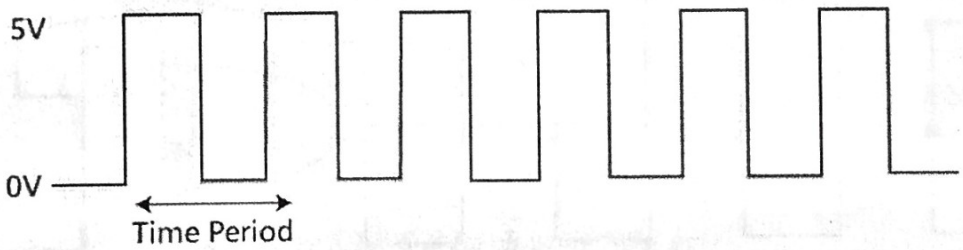


Figure 3.14: Clock Signal

Types of Triggering

These are two types of triggering in sequential circuits:

Level triggering

The logic High and logic Low are the two levels in the clock signal. In level triggering, when the clock pulse is at a particular level, only then the circuit is activated. There are the following types of level triggering:

Positive level triggering

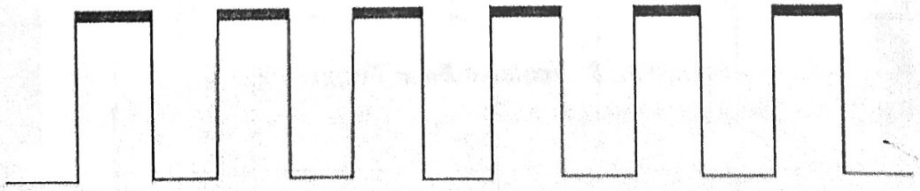


Figure 3.15: Positive Level Triggering

In a positive level triggering, the signal with Logic High occurs. So, in this triggering, the circuit is operated with such type of clock signal.

Negative level triggering

In negative level triggering, the signal with Logic Low occurs. So, in this triggering, the circuit is operated with such type of clock signal.

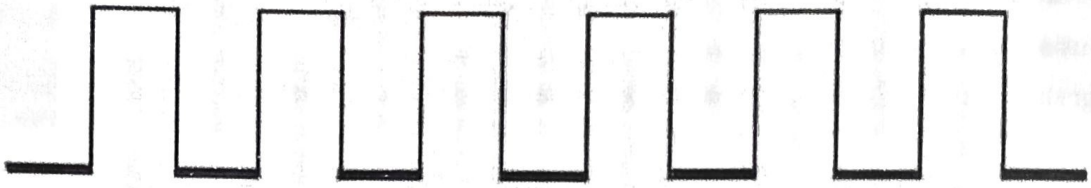


Figure 3.16: Negative Level Triggering

Edge triggering

In clock signal of edge triggering, two types of transitions occur, i.e., transition either from Logic Low to Logic High or Logic High to Logic Low. Based on the transitions of the clock signal, there are the following types of edge triggering.

Positive edge triggering

The transition from Logic Low to Logic High occurs in the clock signal of positive edge triggering. So, in positive edge triggering, the circuit is operated with such type of clock signal.

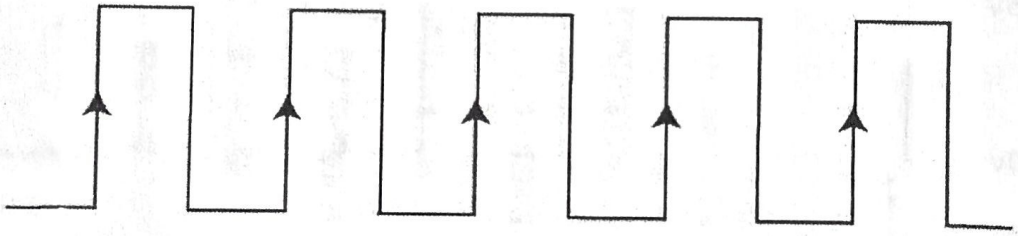


Figure 3.17: Positive Edge Triggering

Negative edge triggering

The transition from Logic High to Logic low occurs in the clock signal of negative edge triggering. So, in negative edge triggering, the circuit is operated with such type of clock signal.

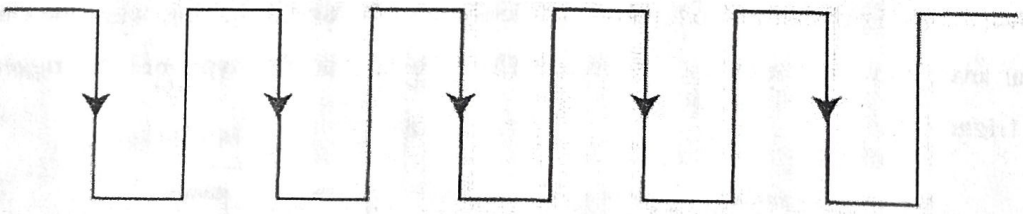


Figure 3.18: Negative Edge Triggering

LATCHES

There are two types of memory elements based on the type of triggering that is suitable to operate it.

- Latches
- Flip-flops

Latches operate with enable signal, which is level sensitive. Whereas, flip-flops are edge sensitive. **A Latch** is a special type of logical circuit. The latches have **low** and **high** two stable statesWhen 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

- **SR Latch**
- **D latch**
- **JK Latch**
- **T Latch.**

S-R Latch

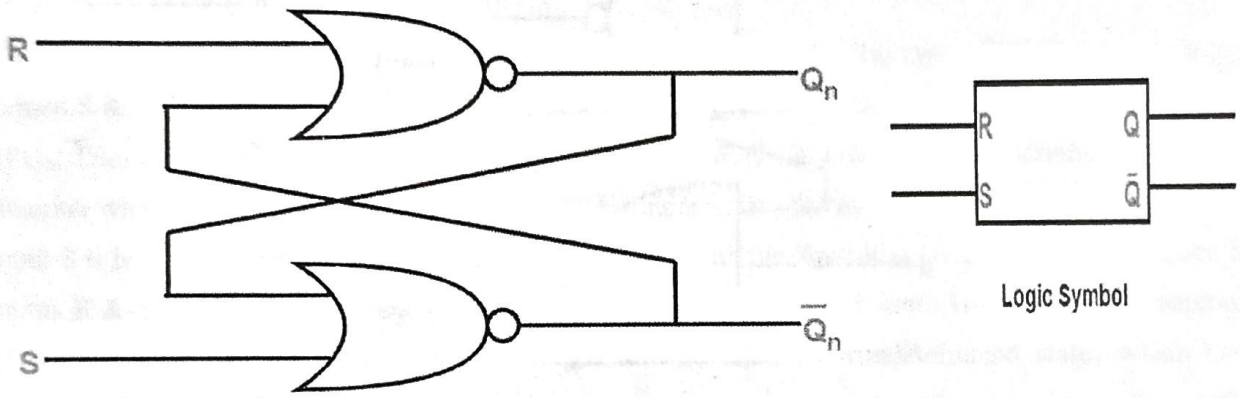


Figure 3.19: S-R Latch using NOR gate

While neither constructing a latch using NOR gates, it is compulsory to consider-

- Reset input R in normal output Q_n.
- Set input S in complemented output Q'_n.

INPUTS			OUTPUTS	REMARKS
R	S	Q_n (Present State)	Q_{n+1} (Next State)	States and Conditions
0	0	X	Q_n	Hold state condition R = S = 0
0	1	X	1	Set state condition R = 0 , S = 1
1	0	X	0	Reset state condition R = 1 , S = 0
1	1	X	Indeterminate	Indeterminate state condition R = S = 1

The circuit has two inputs, S and R, and two outputs, labeled Q_n and Q'_n, where Q'_n is normally the complement of Q_n.

S=1, R=0—Q_n=1, Q'_n=0 This state is also called the SET state.

S=0, R=1— Q_n =0, Q'_n=1 This state is known as the RESET state.

In both the states, the outputs are compliments of each other and that the value of Q follows the value of S.

S=0, R=0— Q_n & Q'_n = Remember

If both the values of S and R are switched to 0, then the circuit remembers the value of S and R in their previous state.

S=1, R=1— Q_n =0, Q'_n =0 [Invalid]

This is an invalid state because the values of both Q and Q' are 0. They are supposed to be compliments of each other. Normally, this state must be avoided.

S-R Latch using NAND Gate

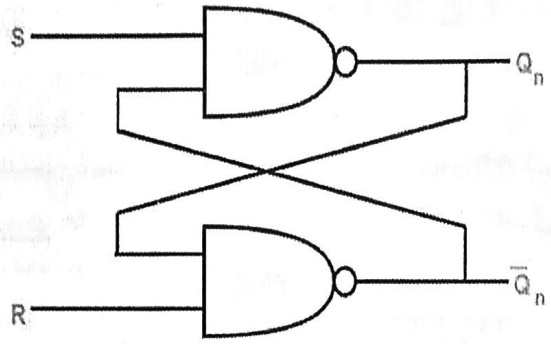
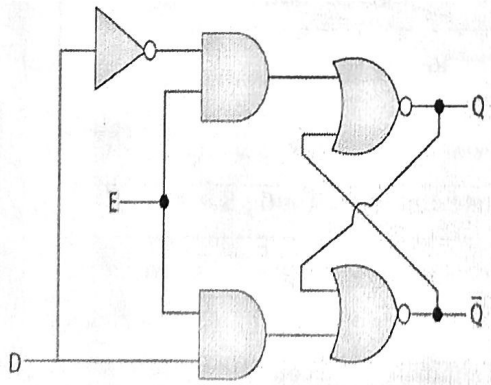


Figure 3.20: S-R Latch using NAND gate

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. Below are the circuit diagram and the truth table of the D latch.



E	D	Q	Q'
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

Figure 3.21: D Latch Logic circuit & Truth Table

Basics of Flip Flop

- A Flip Flop is a memory element that is capable of storing one bit of information.
- It is also called as **Bistable Multivibrator** since it has two stable states either 0 or 1.

Flip flops are of different types depending on how their inputs and clock pulses cause transition between two states.

There are 4 basic types of flip flops-

1. SR Flip Flop
2. JK Flip Flop
3. D Flip Flop
4. T Flip Flop

S-R Flip Flop

- SR flip flop is the simplest type of flip flops.
- It stands for **Set Reset flip flop.**
- It is a clocked flip flop.

The name SR represents the SET and RESET function of the flip flop. This type of flip flop has two inputs named S & R for SET & RESET respectfully & and two outputs name Q_n & Q'_n, whereas Q'_n is the invert of Q_n. The SET function represents when output Q_n is high & Q'_n is low. RESET function represents clear function when output Q_n low & Q'_n High. At each trigger pulse, the output of SR flip flop sets when the input S is high and input R is low. And clears the output when the input R is HIGH & S is low. When both inputs R & S are LOW, the output status Q_n & Q'_n Remains unchanged. Both HIGH input combination is considered forbidden (invalid) as they will produce race condition (undetermined state) which causes ambiguity in the system.

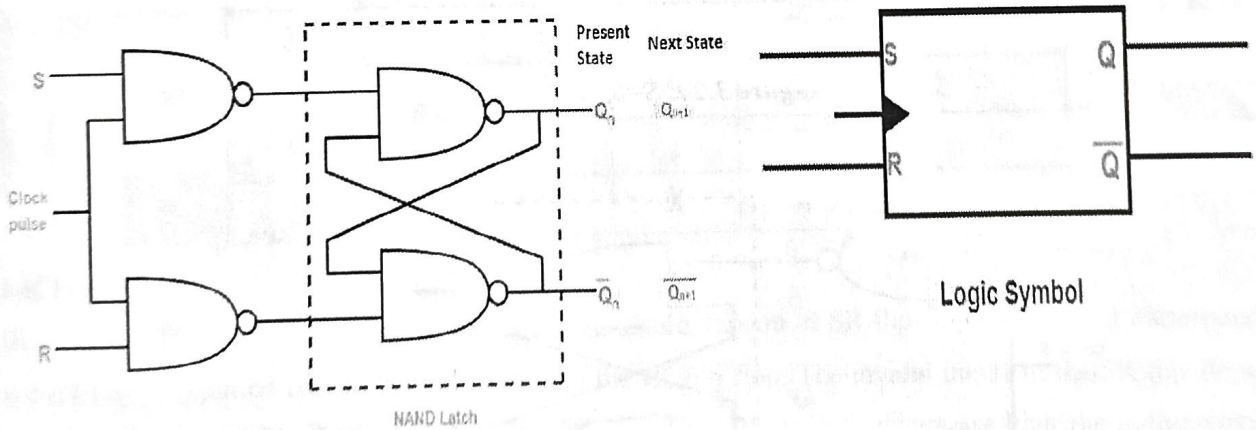
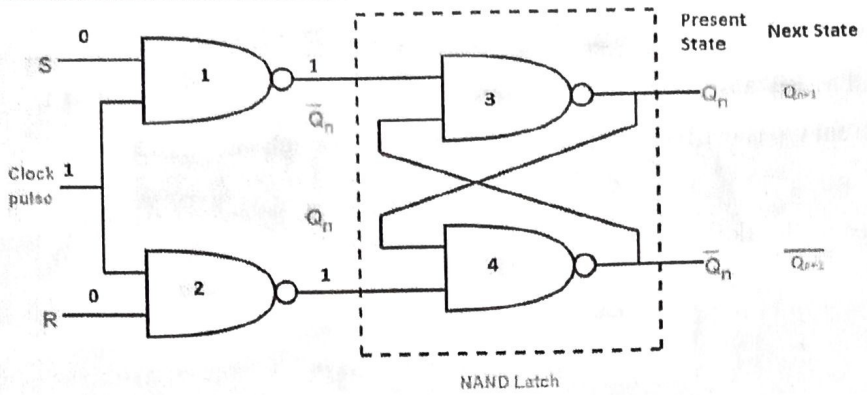


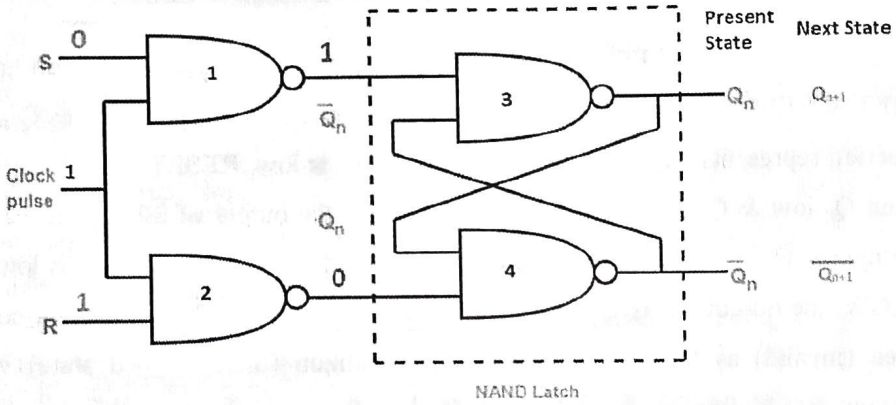
Figure 3.22: S-R Flip Flop Logic diagram



$Q_{n+1} = 1.\overline{Q_n} = Q_n$

$\overline{Q_{n+1}} = 1.Q_n = \overline{Q_n}$

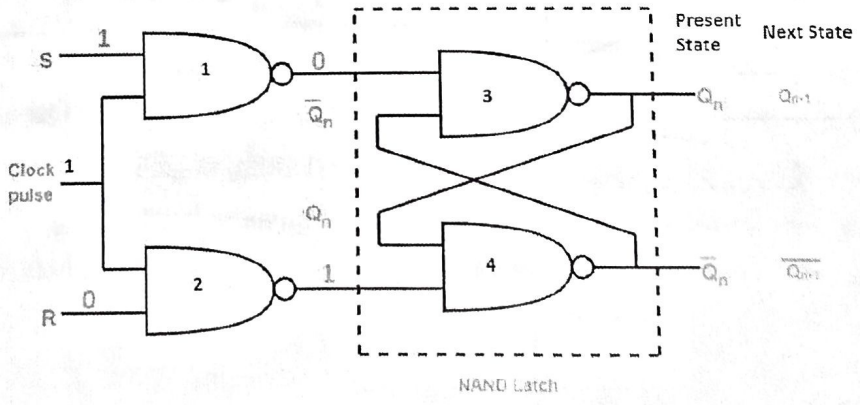
Figure 3.23: S=0, R=0



$Q_{n+1} = 1.\overline{Q_n} = Q_n = 0$

$\overline{Q_{n+1}} = 0.Q_n = \overline{0} = 1$

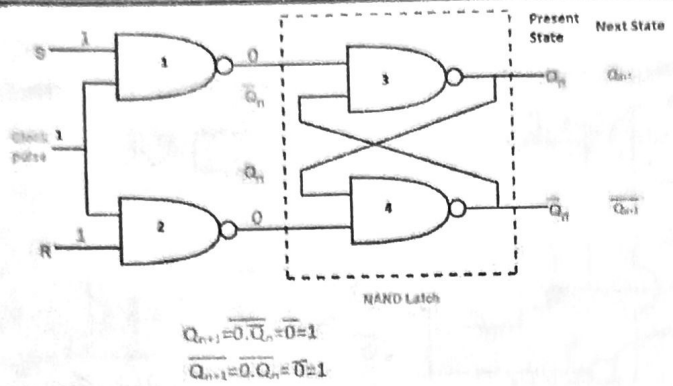
Figure 3.24: S=0, R=1



$Q_{n+1} = 0.\overline{Q_n} = \overline{0} = 1$

$\overline{Q_{n+1}} = 1.Q_n = \overline{Q_n} = 0$

Figure 3.25: S=1, R=0



$Q_{n+1}=\overline{0.\overline{Q_n}}=\overline{0}=1$

$\overline{Q_{n+1}}=\overline{0.Q_n}=\overline{0}=1$

Figure 3.26: S=1, R=1

The circuit has two inputs, S and R, and two outputs, labeled Q_n. and Q'_n, where Q'_n is normally the complement of Q_n.

S=1, R=0—Q_n.=1, Q'_n=0 This state is also called the SET state.

S=0, R=1— Q_n =0, Q'_n=1 This state is known as the RESET state.

In both the states, the outputs are compliments of each other and that the value of Q follows the value of S.

S=0, R=0— Q_n & Q'_n = Remember

If both the values of S and R are switched to 0, then the circuit remembers the value of S and R in their previous state.

S=1, R=1— Q_n =1, Q'_n =1 [Invalid]

This is an invalid state because the values of both Q and Q' are 0. They are supposed to be compliments of each other. Normally, this state must be avoided.

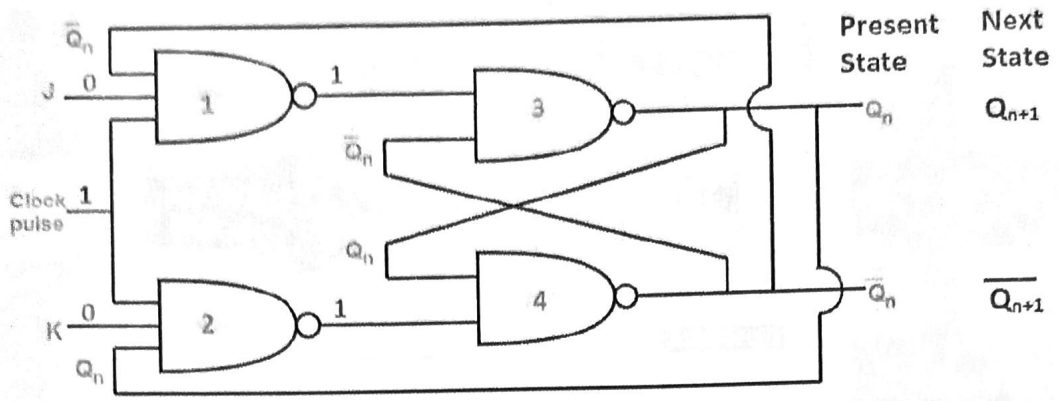
INPUTS			OUTPUTS		ACTIONS
CLK	S	R	Qn+1	$\overline{Qn+1}$	
1	0	0	Qn	$\overline{Qn}$	No Change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	X	X	Indeterminate

J-K Flip Flop

JK flip flop can also be considered a better and modified version of SR flip-flop with J input corresponding to SET and K input corresponding to RESET of the JK flip flop. The invalid inputs of the SR flip-flops are used in this type of flip flop for a meaning full function. When both inputs are high the output status is toggled during each clock cycle. When the inputs are same, the JK flip-flop retains its original status even after clock pulses.

In JK flip flop,

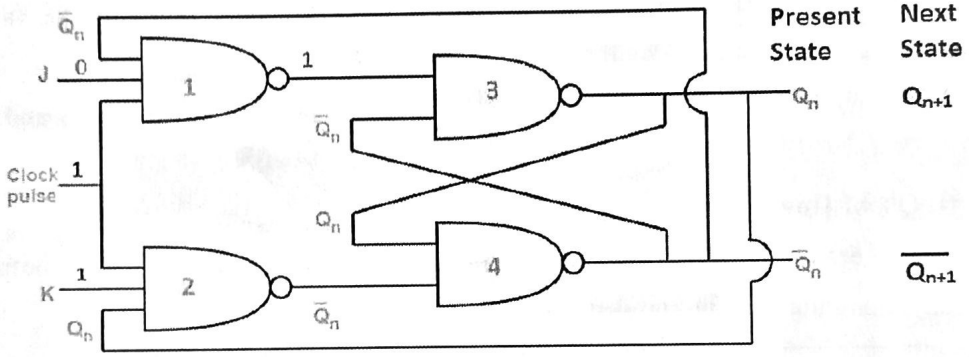
- Input J behaves like input S of SR flip flop which was meant to set the flip flop.
- Input K behaves like input R of SR flip flop which was meant to reset the flip flop.



$Q_{n+1}=\overline{1.\overline{Q_n}}=\overline{\overline{Q_n}}=Q_n$

$\overline{Q_{n+1}}=\overline{1.Q_n}=\overline{Q_n}$

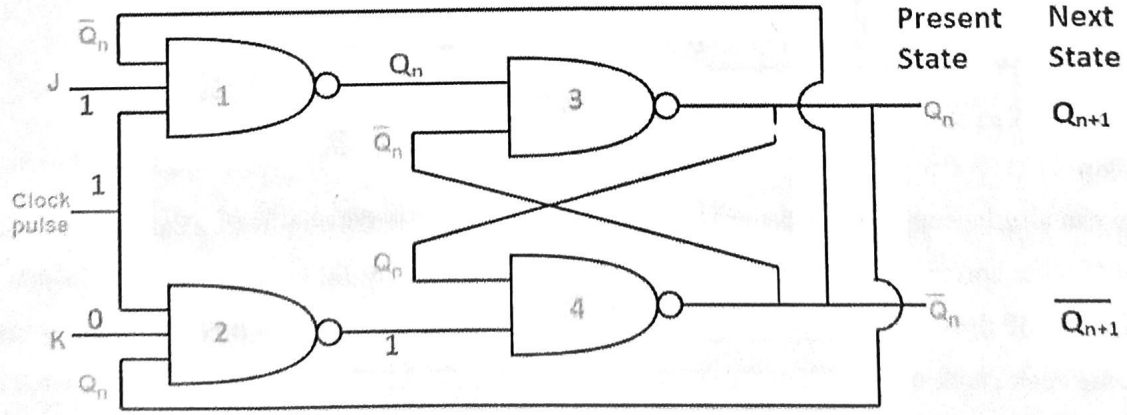
Figure 3.27: J=0, K=0



$Q_{n+1}=\overline{1.\overline{Q_n}}=\overline{\overline{Q_n}}=Q_n = 0$

$\overline{Q_{n+1}}= \overline{\overline{Q_n}\,Q_n} = \overline{0}=1$

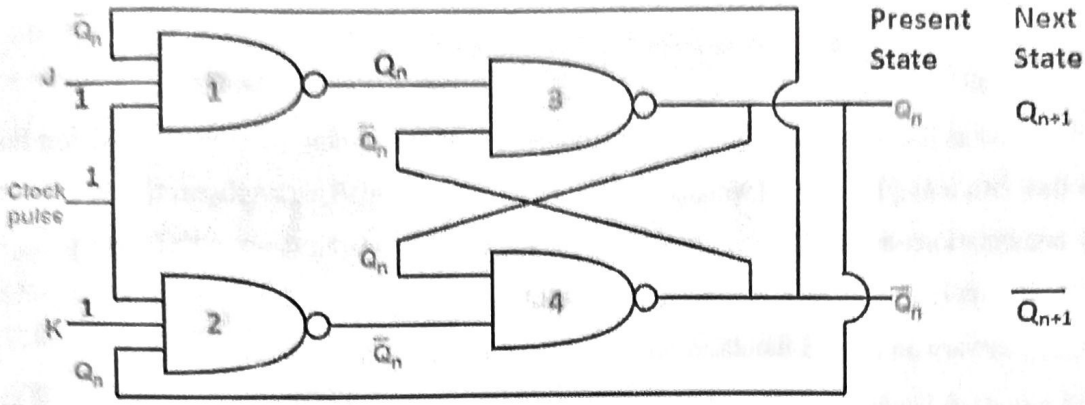
Figure 3.28: J=0, K=1



$Q_{n+1}= \overline{\overline{Q_n}\,Q_n} = \overline{0}=1$

$\overline{Q_{n+1}}=\overline{1.Q_n}=\overline{Q_n}=0$

Figure 3.29: J=1, K=0



$\overline{Q_{n+1}} = \overline{\overline{Q_n}\, Q_n} = \overline{0} = 1$

$Q_{n+1} = \overline{1.\overline{Q_n}} = \overline{\overline{Q_n}} = Q_n = 0$

Figure 3.30: J=1, K=1

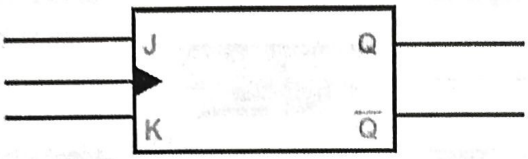


Figure 3.31: J K Flip Flop Logic diagram

INPUTS			OUTPUTS	REMARKS
J	K	Q_n (Present State)	Q_{n+1} (Next State)	States and Conditions
0	0	X	Q_n	Hold State condition J = K = 0
0	1	X	0	Reset state condition J = 0 , K = 1
1	0	X	1	Set state condition J = 1 , K = 0
1	1	X	Q'_n	Toggle state condition J = K = 1

The D-type Flip Flop

A D type (Data or delay flip flop) has a single data input in addition to the clock input as shown in figure.

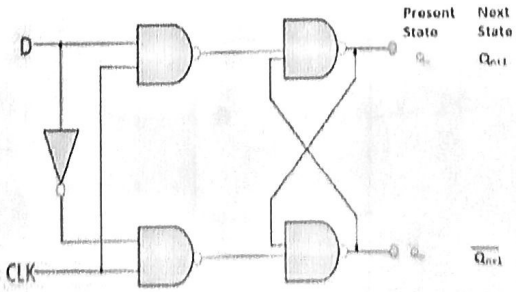


Figure 3.32: D Flip Flop

The D input goes directly to S input and its complement through NOT gate is applied to the R input. This kind of flip flop prevents the value of D from reaching the output until a clock pulse occurs.

D flip flop is the same as SR flip flop with only a minor change. In D flip-flop, the inputs of SR flip flop are combined together into a single input D with one of the input R inverted. This configuration eliminates the invalid inputs combinations as there cannot be the same inputs. During the clock pulse, D flip flop SET output when its input is High & Resets when the input is LOW. It is easier to configure as compared to SR Flip flop because there are no Invalid inputs.

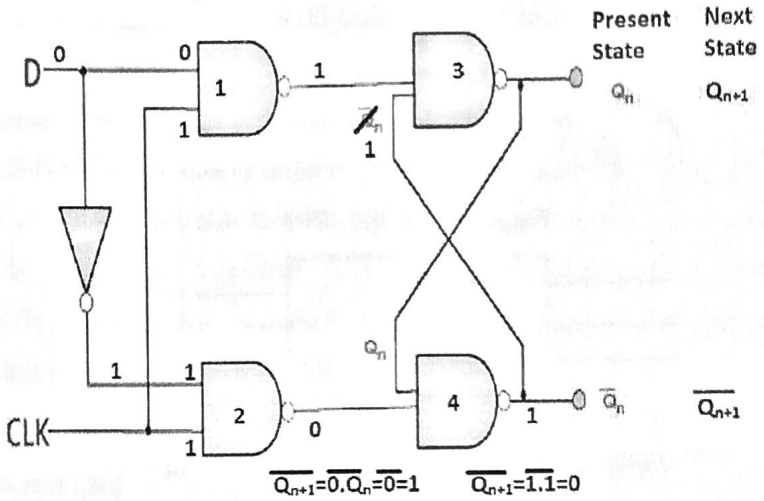


Figure 3.33: D=0 Flip Flopoperation

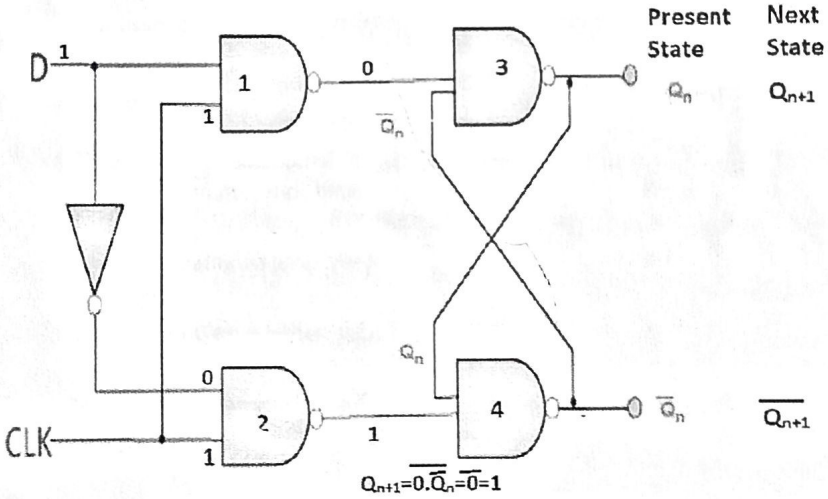


Figure 3.34: D=1 Flip Flopoperation

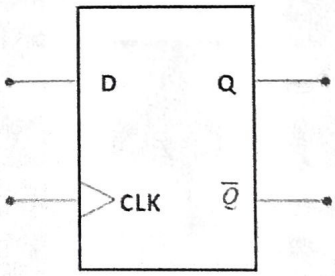


Figure 3.35: DFlip Floplogic symbol

Clk	D	Q	Q	Description
↓ » 0	X	Q	Q	Memory no change
↑ » 1	0	0	1	Reset Q » 0
↑ » 1	1	1	0	Set Q » 1

SHIFT REGISTERS

A group of flip flops which is used to store multiple bits of data and the data is moved from one flip flop to another is known as **Shift Register**. The bits stored in registers shifted when the clock pulse is applied within and inside or outside the registers. To form an n-bit shift register, we have to connect n number of flip flops. So, the number of bits of the binary number is directly proportional to the number of flip flops. The flip flops are connected in such a way that the first flip flop's output becomes the input of the other flip flop.

A **Shift Register** can shift the bits either to the left or to the right. A **Shift Register**, which shifts the bit to the left, is known as **"Shift left register"**, and it shifts the bit to the right, known as **"Right left register"**. The shift register is classified into the following types:

- Serial In Serial Out
- Serial In Parallel Out
- Parallel In Serial Out
- Parallel In Parallel Out

Serial IN Serial OUT Shift Register

In "Serial Input Serial Output", the data is shifted "IN" or "OUT" serially. In SISO, a single bit is shifted at a time in either right or left direction under clock control.

Initially, all the flip-flops are set in "reset" condition i.e. $Q_3 = Q_2 = Q_1 = Q_0 = 0$. If pass the binary number 1111, the LSB bit of the number is applied first to the Dn bit. The D_3 input of the third flip flop, i.e., FF-3, is directly connected to the serial data input D_3. The output Q_3 is passed to the data input D_2 of the next flip flop. This process remains the same for the remaining flip flops.

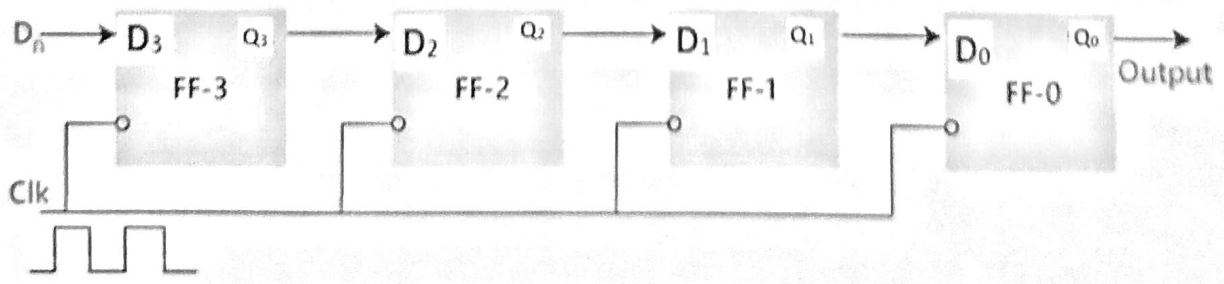


Figure 3.36:Serial In Serial Out shift register

Operation

When the clock signal application is disabled, the outputs $Q_3\ Q_2\ Q_1\ Q_0 = 0000$. The LSB bit of the number is passed to the data input D_3. We will apply the clock, and this time the value of D_3 is 1. The first flip flop, i.e., FF-3, is set, and the word is stored in the register at the first falling edge of the clock. Now, the stored word is 1000.

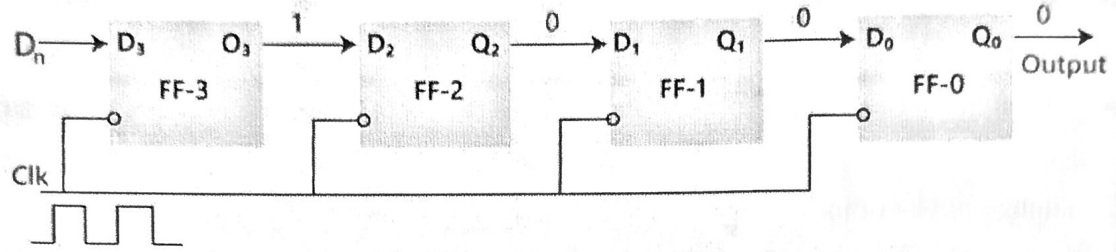


Figure 3.37:Serial In Serial Out shift register during first clock cycle

The next bit of the binary number, i.e., 1, is passed to the data input D_2. The second flip flop, i.e., FF-2, is set, and the word is stored when the next negative edge of the clock hits. The stored word is changed to 1100.

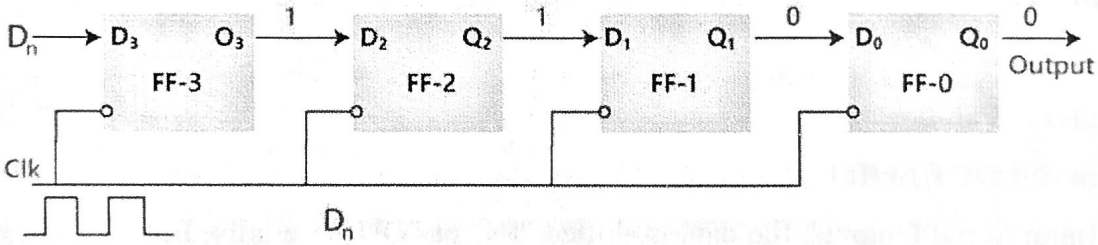


Figure 3.38:Serial In Serial Out shift register during second clock cycle

The next bit of the binary number, i.e., 1, is passed to the data input D_1, and the clock is applied. The third flip flop, i.e., FF-1, is set, and the word is stored when the negative edge of the clock hits again. The stored word is changed to 1110.

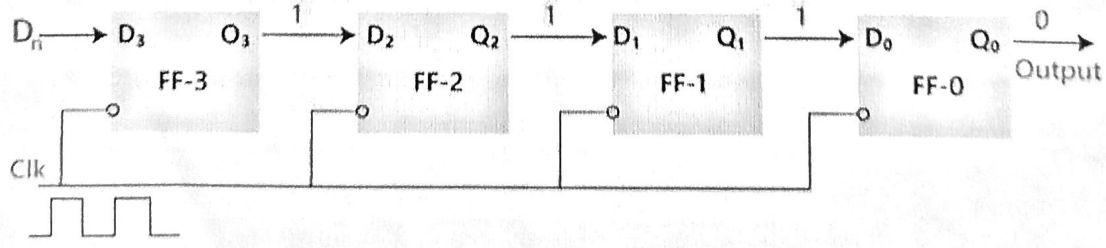


Figure 3.39:Serial In Serial Cut shift register during third clock cycle

Similarly, the last bit of the binary number, i.e., 1, is passed to the data input D_0, and the clock is applied. The last flip flop, i.e., FF-0, is set, and the word is stored when the clock's negative edge arrives. The stored word is changed to 1111.

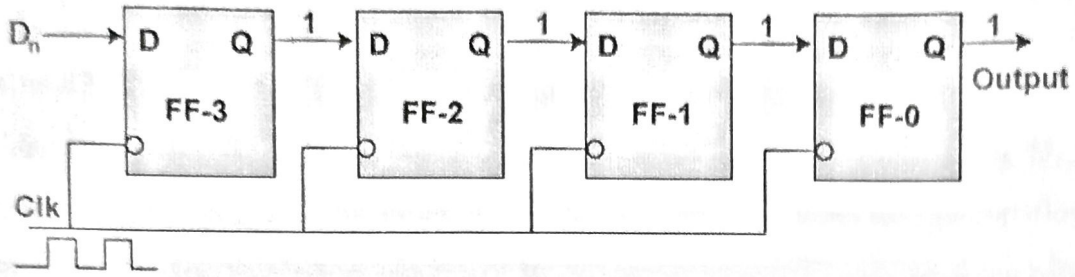


Figure 3.40:Serial In Serial Out shift register during fourth clock cycle

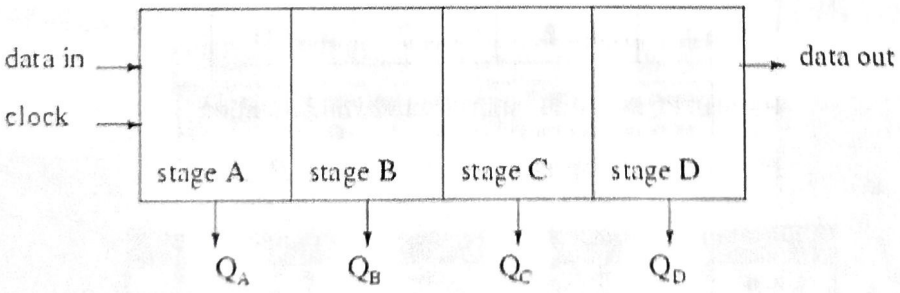
Truth Table

	Clk	D_n-Q_3	Q_3-D_2	Q_2-D_1	Q_1-D_0	Q_0
Initially			0	0	0	0
(1)	↓	1	1	0	0	0
(2)	↓	1	1	1	0	0
(3)	↓	1	1	1	1	0
(4)	↓	1	1	1	1	1

→ Direction of data travel

Serial IN Parallel OUT Shift Register

For this kind of register, data bits are entered serially in the same manner. The difference is the way in which the data bits are taken out of the register. Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously.



Serial-in, parallel-out shift register with 4-stages

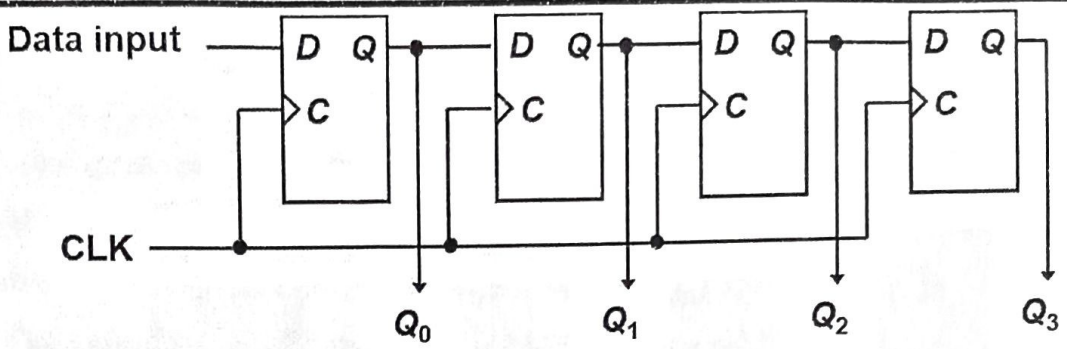
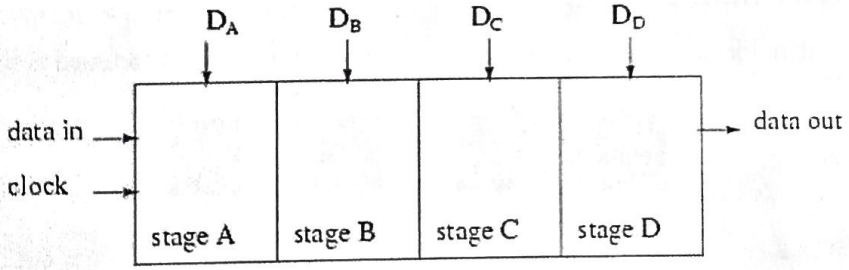


Figure 3.41:Serial In Parallel Out shift register

Clear	FF0	FF1	FF2	FF3
1001	0	0	0	0
	1	0	0	0
	0	1	0	0
	0	0	1	0
	1	0	0	1

Parallel IN Serial OUT Shift Register

In the **"Parallel IN Serial OUT"** register, the data is entered in a parallel way, and the outcome comes serially. The input of the flip flop is the output of the previous Flip Flop. The input and outputs are connected through the combinational circuit. Through this combinational circuit, the binary input D_0, D_1, D_2, D_3 are passed. The **shift mode** and the **load mode** are the two modes in which the **"PISO"** circuit works.



Parallel-in, serial-out shift register with 4-stages

Figure 3.42:Parallel In Serial Out Shift Register

	Q_0	Q_1	Q_2	Q_3	
Clear	0	0	0	0	
Write	1	0	0	1	
Shift	1	0	0	1	
	1	1	0	0	1
	1	1	1	0	01
	1	1	1	1	001
	1	1	1	1	1001

D_0, D_1, D_2 and D_3 are the parallel inputs, where D_0 is the most significant bit and D_3 is the least significant bit. To write data in, the mode control line is taken to LOW and the data is clocked in. The data can be shifted when the mode control line is HIGH as SHIFT is active high. The register performs right shift operation on the application of a clock pulse.

Parallel IN Parallel OUT Shift Register

In **"Parallel IN Parallel OUT"**, the inputs and the outputs come in a parallel way in the register. The inputs B_0, B_1, B_2, and B_3, are directly passed to the data inputs D_0, D_1, D_2, and D_3 of the respective flip flop. The bits of the binary input are loaded to the flip flops when the negative clock edge is applied. The clock pulse is required for loading all the bits. At the output side, the loaded bits appear.

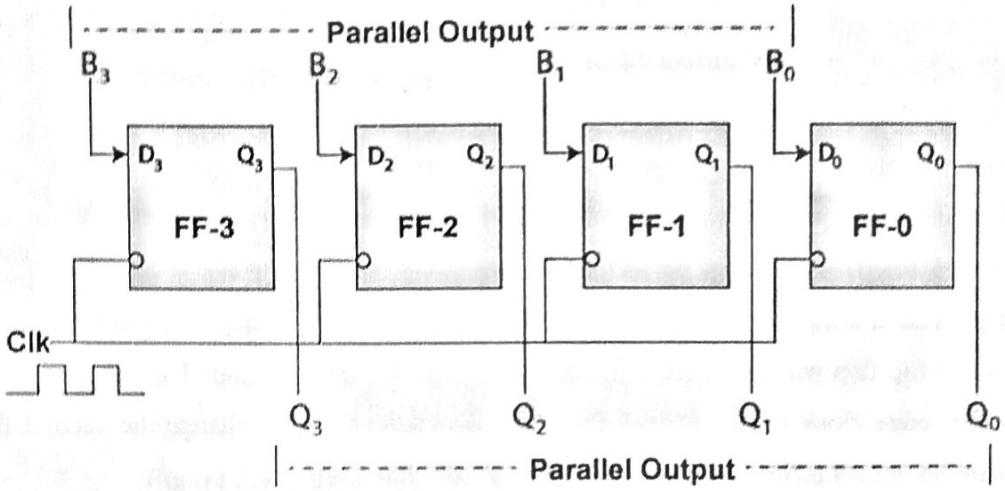


Figure 3.43:Parallel In Parallel Out Shift Register

Counters

A special type of sequential circuit used to count the clock 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. 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

There are the following types of counters:

- Asynchronous Counters
- Synchronous Counters

Asynchronous or ripple counters

The **Asynchronous counter** is also known as the **ripple counter**. we can 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.

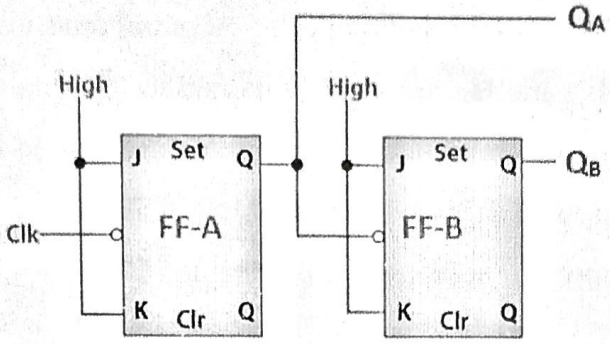


Figure 3.44: Asynchronous Counter

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 $Q_B = 0$

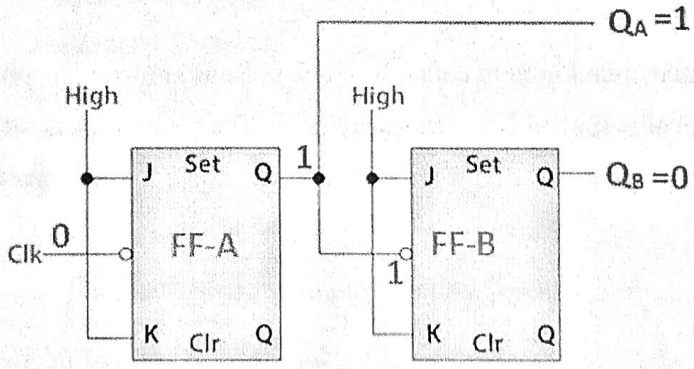


Figure 3.45: Asynchronous Counter with first clock cycle

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 $Q_B = 1$.

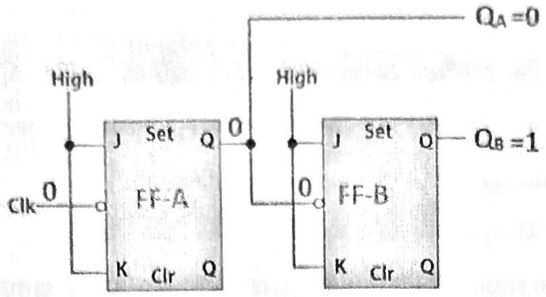


Figure 3.46: Asynchronous Counter with second clock cycle

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 $Q_B = 1$

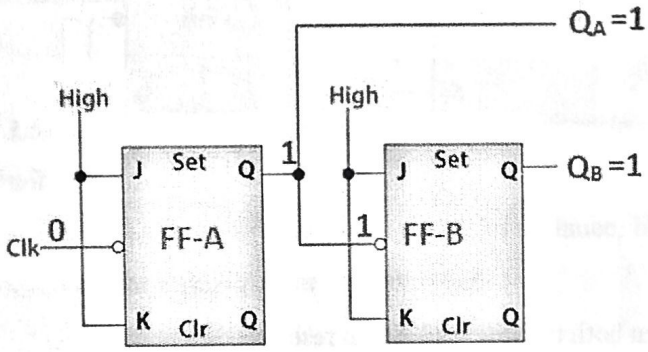


Figure 3.47: Asynchronous Counter with third clock cycle

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 $Q_B = 0$

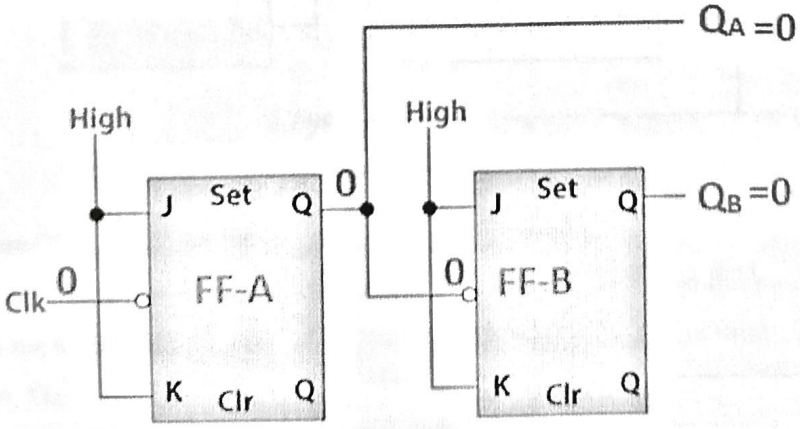


Figure 3.48: Asynchronous Counter with fourth clock cycle

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.

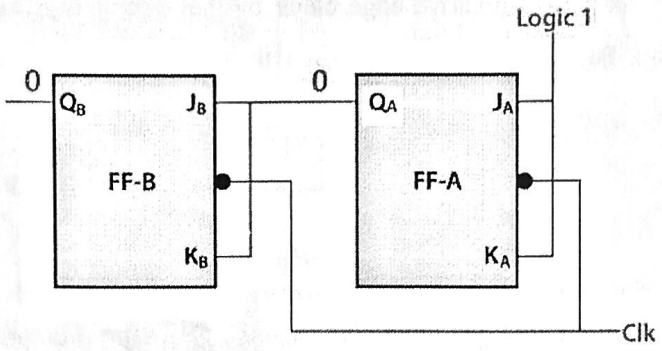


Figure 3.49: Synchronous Counter

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 $\mathbf{Q_B = 0}$

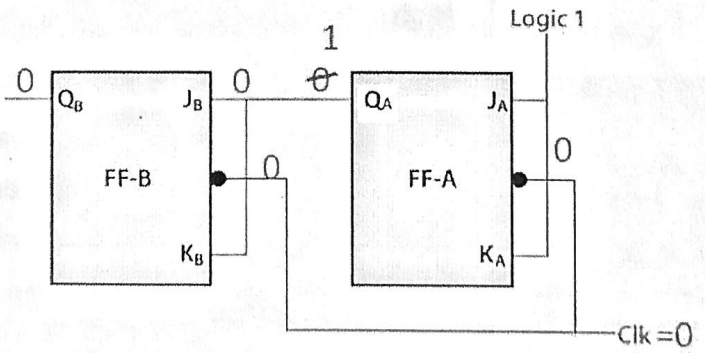


Figure 3.50: Synchronous Counter with first clock cycle

3. **Condition 3:** 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, $Q_A = 0$ and $Q_B = 1$

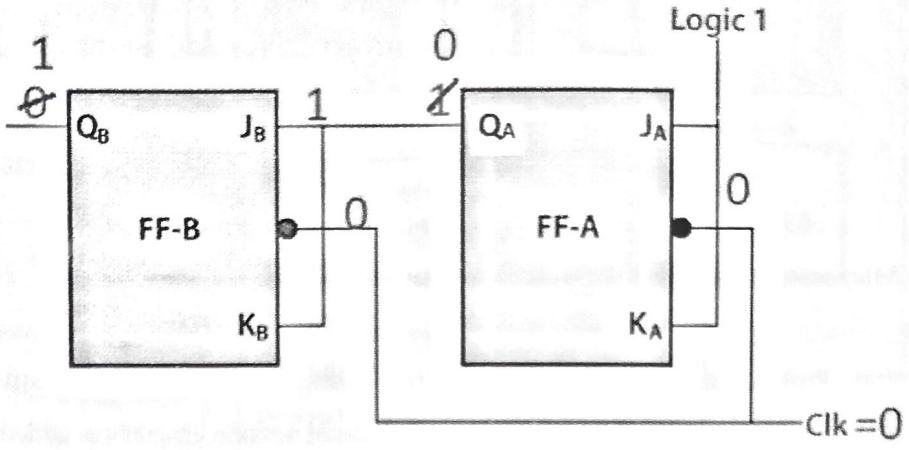


Figure 3.51: Synchronous Counter with second clock cycle

4. **Condition 4:** 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, $Q_A = 1$ and $Q_B = 1$

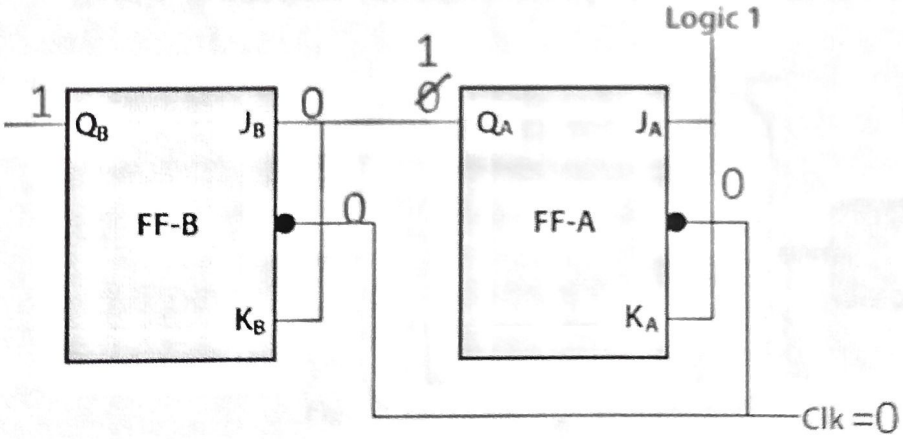


Figure 3.52: Synchronous Counter with third clock cycle

5. **Condition 5:** 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, $Q_A = 0$ and $Q_B = 0$

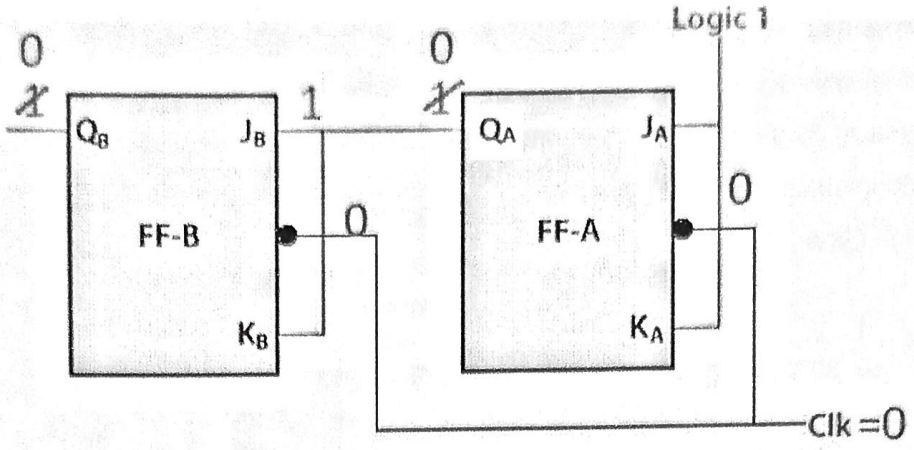


Figure 3.53: Synchronous Counter with fourth clock cycle

Introduction to Microcontrollers and their applications

Microprocessor

Computer's Central Processing Unit (CPU) built on a **single Integrated Circuit (IC)** is called a **microprocessor**. It is a general purpose device and an additional outside circuitry is added to make it work as a microcomputer.

A microprocessor consists of an ALU, control unit and register array. Where **ALU** performs arithmetic and logical operations on the data received from an input device or memory. Control unit controls the instructions and flow of data within the computer and, **register array** consists of registers which are used to store data for mathematical and logical operations. The ROM/RAM in which the program is to be stored, the I/O pins, timers etc all have to be added to the microprocessor separately using discrete devices.

E.g – 8085, 8086

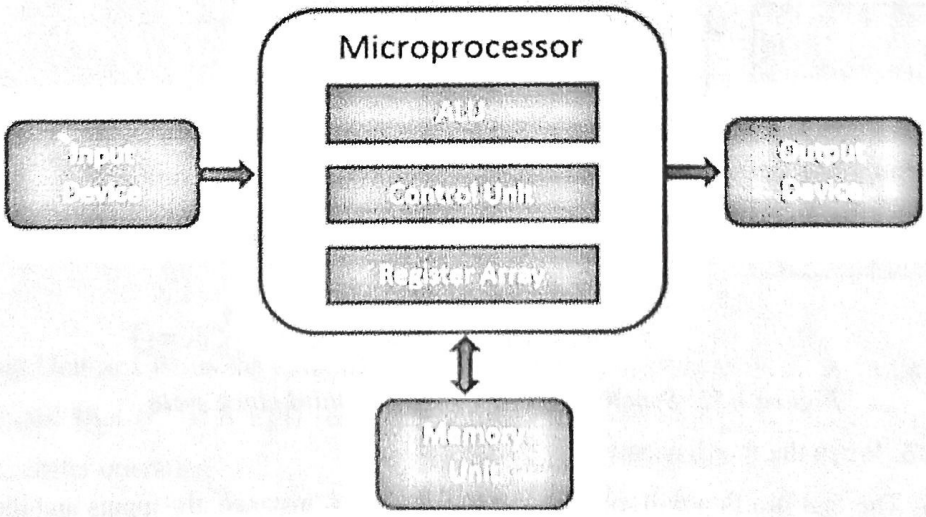


Figure 3.54: Block diagram of Microprocessor

Basic Components of a Microcontroller

A **microcontroller** includes a processor, the ALU, Instruction Decoder, registers, code storage space(flash memory/ROM), RAM, general purpose I/O pins, timers, interrupt controller, serial communication module etc. That is, the microcontroller is a microprocessor with some basic modules.

E.g – 8051, Atmega8, PIC 18F4550

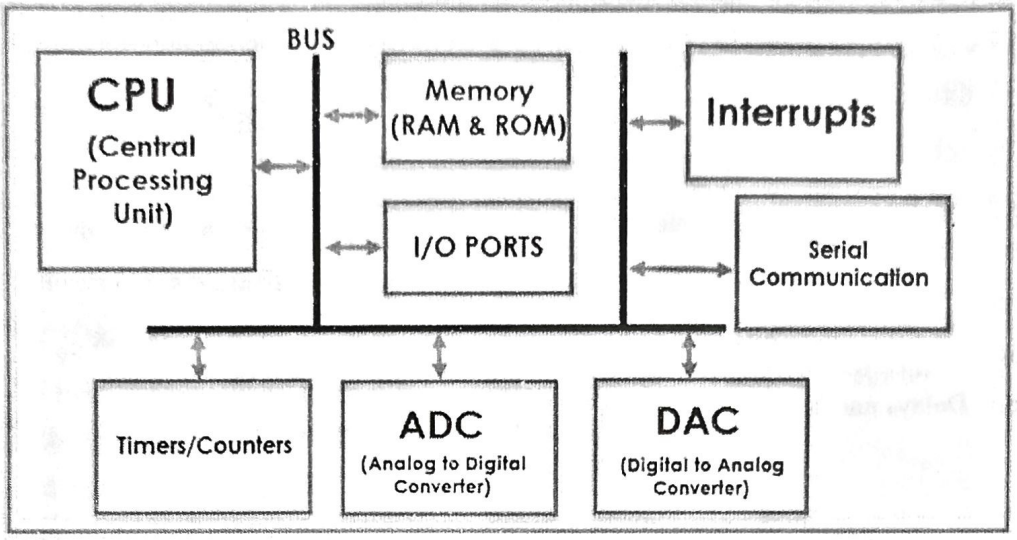


Figure 3.55: Block diagram of Microcontroller

The major components of a Microcontroller are:

- The CPU (Central Processing Unit)
- The Memory and
- The I/O Ports

CPU

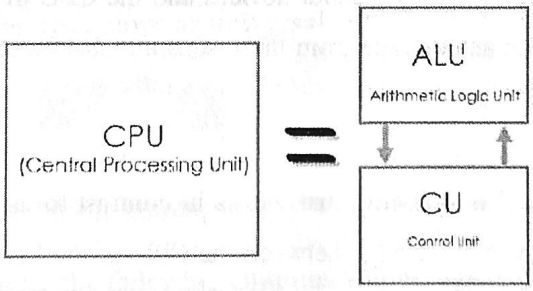


Figure 3.55: Central Processing Unit

Central Processing Unit or CPU is the brain of the Microcontroller. It consists of an Arithmetic Logic Unit (ALU) and a Control Unit (CU). A CPU reads, decodes and executes instructions to perform Arithmetic, Logic and Data Transfer operations.

Memory

Any Computational System requires two types of Memory: Program Memory and Data Memory. Program Memory contains the program i.e. the instructions to be executed by the CPU. Data Memory is required to store temporary data while executing the instructions. Usually, Program Memory is a Read Only Memory or

ROM and the Data Memory is a Random Access Memory or RAM. Data Memory is sometimes called as Read Write Memory (R/W M).

I/O Ports

The interface for the Microcontroller to the external world is provided by the I/O Ports or Input/Output Ports. Inputs device like Switches, Keypads, etc. provide information from the user to the CPU in the form of Binary Data. The CPU, upon receiving the data from the input devices, executes appropriate instructions and gives response through Output Devices like LEDs, Displays, Printers, etc.

Bus

A System bus is a group of connecting wire that connect the CPU with other peripherals like Memory, I/O Ports and other supporting components.

Timers/Counters

One of the important components of a Microcontroller are the Timers and Counters. They provide the operations of Time Delays and counting external events.

Serial Port

One of the important requirement of a Microcontroller is to communicate with other device and peripherals (external). Serial Port proves such interface through serial communication.

Interrupts

A very important feature of a Microcontroller is Interrupts and its Interrupt Handling Mechanism. Interrupts can be external, internal, hardware related or software related.

ADC (Analog to Digital Converter)

Analog to Digital Converter is a circuit that converts Analog signals to Digital Signals. The ADC Circuit forms the interface between the external Analog Input devices and the CPU of the Microcontroller. Almost all sensors are analog devices and the analog data from these sensors must be converted in to digital data for the CPU to understand.

DAC (Digital to Analog Converter)

DIgital to Analog Converter or DAC is a circuit, that works in contrast to an ADC i.e. it converts Digital Signals to Analog Signals. DAC forms the bridge between the CPU of the Microcontroller and the external analog devices.

Applications of Microcontrollers

There are huge number of applications of Microcontrollers. In fact, the entire embedded systems industry is dependent on Microcontrollers. The following are few applications of Microcontrollers.

- **Household appliances:** Microwave oven, washing machine,
- **Office and commercial appliances:** Fax machine, photocopier
- **Telecommunication:** Telephones, phone answering machines, mobile phones
- **Entertainment and gaming:** Televisions, VCRs, music players
- Automotive industry: Fuel injection, ABS

- **Industrial automation and manufacturing:** Motor control systems, data acquisition and supervisory systems, industrial robots,
- **Electronic measurement instruments:** Digital multimeters, logic analyzers
- **Biomedical systems:** ECG recorder, blood cell analyzers
- **Computer systems:** Keyboard controller, CD drive or hard disk Military weapons, guidance and positioning

MODEL QUESTIONS

1. What is the function of full adder? Draw and explain various implementations.
2. What is the function of half adder? Draw and explain various implementations.
3. Explain the operation of BCD adder with the help of truth table.
4. Explain the working of RS flip flop with the help of truth table.
5. Draw the logic diagram of a JK flip flop and using excitation table, explain its operation.
6. What are the different types of shift registers? Explain any one type of shift register.
7. Explain the working of asynchronous using J-K flip flop.
8. Explain the working of synchronous using J-K flip flop.
9. Define microcontroller? Explain the feature of microcontroller with block diagram.

MULTIPLE CHOICE QUESTIONS

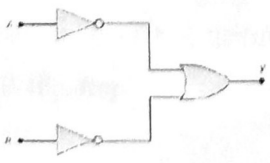
The output of a two-input AND gate is high.....

a. Only if both the inputs are high
b. Only if both the inputs are low
c. Only if one input is high and the other is low
d. If at least one input is low

Which of these sets of logic gates are known as universal gates?

a. XOR, NAND, OR
b. OR, NOT, XOR
c. NOR, NAND, XNOR
d. NOR, NAND

Which logic gate is represented by the following combination of logic gates?



a. OR
b. NAND
c. AND
d. NOR