13. **A multiplexer is**
(a) Control lines are used to select input line
(b) Selection lines are used to select the input line
(c) Both (a) and (b)
(d) None.

14. **In full subtractor the difference value is given by**
(a) Sum output of half-adder
(b) Carry output of a full-adder
(c) Difference output of a half-subtractor
(d) Sum output of a full-adder.

15. **Binary-to-octal decoder is same as**
(a) 2-line to 8-line decoder
(b) 3-line to 8-line decoder
(c) 4-line to 8-line decoder
(d) None of the above.

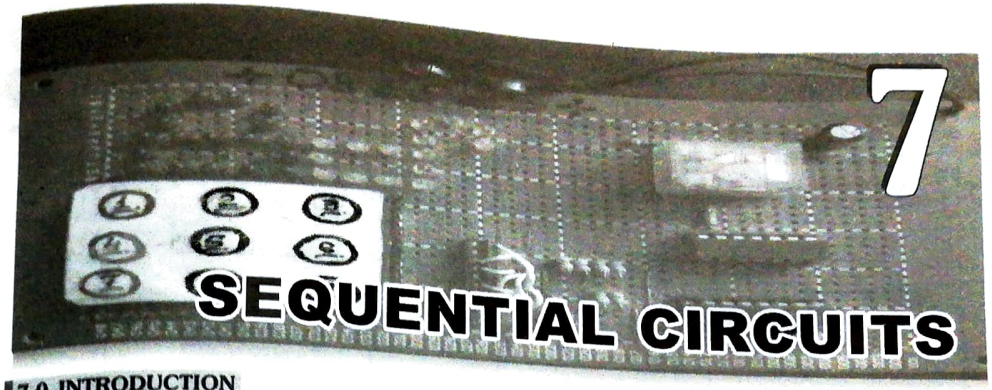
ANSWERS

1. (a)	2. (a)	3. (b)	4. (a)	5. (b)	6. (a)	7. (b)	8. (c)	9. (d)	10. (b)
11. (b)	12. (b)	13. (a)	14. (d)	15. (b)					

SUPPLEMENTARY PROBLEMS

1. Design full adder using decoder.
2. Implement Excess-3 to seven segment display.
3. Draw logic diagram of 4-bit ALU.
4. Implement $Y = \pi M (1, 5, 8, 11, 12)$ using MUX
5. Implement $Y = \Sigma m (0, 2, 5, 6) + \Sigma d (1)$ using DEMUX.
6. Implement using 4 to 16 decoder.
$Y = \Sigma m (1, 4, 5, 7, 9, 11)$
7. Implement $Y = \overline{A}BD + BC + \overline{B}CD$ using MUX.
8. Design 8-bit comparator using two 4-bit comparator.
9. Design 3-bit parity generator and checker.
10. Implement 8×1 MUX using 2×1 MUX.
11. Implement full Adder using Multiplexers.
12. Design 3-bit Gray to binary code converter using only NAND gates.

◆◆◆◆◆



7 SEQUENTIAL CIRCUITS

7.0 INTRODUCTION

Sequential circuits are those which follows specific order and each stage in these circuits depends on the result of previous stage. To retain the past results flip-flops are used.

Sequential circuit is a digital logic with feedback. There should be definite output state for any given input state.

The block diagram of a sequential circuit is as shown in fig. 7.1. Here memory elements are connected to the combinational circuit as a feedback path.

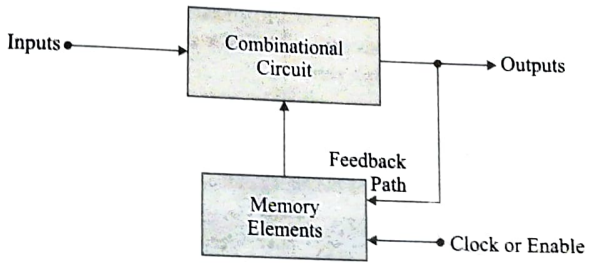


Fig. 7.1 Block diagram of Squential Circuit

Thus, a sequential circuit consists of a combinational circuit and memory elements through a feedback path which means it depends upon the present input states and past output states.

7.1 COMPARISON BETWEEN COMBINATIONAL CIRCUITS AND SEQUENTIAL CIRCUITS

Combinational Circuits	Sequential Circuits
1. In combinational circuits, output doesn't depends upon previous states.	1. In sequential circuits, output depends upon present inputs as well as previous inputs.
2. Memory elements are not present in combinational circuits.	2. Memory elements are present to store the past history of input variables in sequential circuits.
3. More hardware is required in combinational circuits.	3. Less hardware is required in sequential circuits as compared to combinational circuits.
4. Combinational circuits are easy to design as memory elements are absent.	4. Sequential circuits are difficult to design due to memory elements.
	5. Sequential circuits are slower than the combinational circuits.
5. Combinational circuits are faster in speed.	6. Sequential circuits are cheaper as compared to combinational circuits.
6. Combinational circuits are expensive in cost.	

Combinational Circuits	Sequential Circuits
7. Functional block diagram of a combinational circuit is as shown in fig. (*a*).	7. Functional block diagram of a sequential circuit is as shown in fig. (*b*).
8. Clock circuit is not used in combinational circuits.	8. Clock circuit is an essential requirement in sequential circuits.
9. Feedback path is not used in combinational circuits.	9. Feedback path is used for sequential circuits
10. Examples of combinational circuits are : Parallel adders, subtractors, code converters, comparators, multiplexers, demultiplexers, decoders, encoders etc.	10. Examples of sequential circuits are : Serial adders, flip-flops, counters, registers etc.

Fig. (*a*)

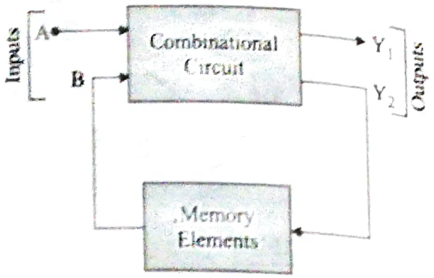
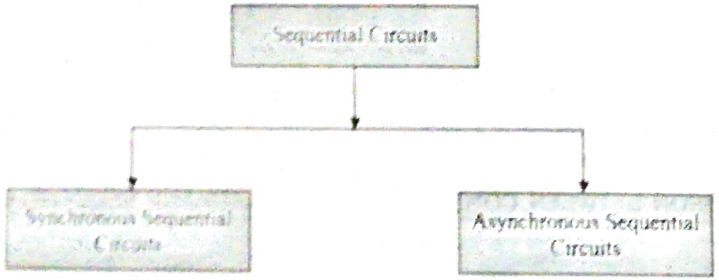


Fig. (*b*)

7.2 CLASSIFICATION OF SEQUENTIAL CIRCUITS

Sequential circuits are classified depending on the timing of their signals.



Comparison between synchronous and asynchronous sequential circuits

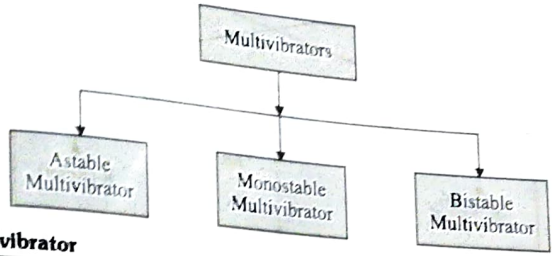
Synchronous Sequential Circuits	Asynchronous Sequential Circuits
1. Memory elements are clocked flip-flops in synchronous sequential circuits.	1. Memory elements are either unclocked flip-flops or time delay elements in asynchronous sequential circuits.
2. In synchronous sequential circuits the maximum operating speed of clock depends on time delays involved.	2. In asynchronous sequential circuits as clock is absent, it can operate faster as compared to synchronous circuits.
3. Memory element is affected due to change in input signal because of activation of clock signal	3. Memory element is affected due to change in input signal at any instant of time.
4. These circuits are easier to design.	4. These circuits are more difficult to design as compared to synchronous sequential circuits.

7.3 MULTIVIBRATORS

For synchronous sequential circuits, timing waveforms are required. The timing circuit generator produces rectangular waveforms.

Multivibrators are the digital circuits having feedback path. Bistable multivibrator hold state. It gives two states *i.e.* 0 or 1. Monostable multivibrator has only one stable state and Astable multivibrator has no stable state.

Multivibrators are classified as



7.3.1 Astable Multivibrator

It is free running multivibrator. It has two quasistable states which does not require any triggering.

Example of Astable Multivibrator

The inverter with feedback is an example of astable multivibrator as shown in figure 7.2.

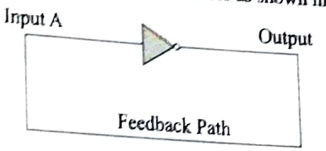


Fig. 7.2 Inverter with feedback path

When A = 0, output = 1 and this is feedback to input again, we get A = 1, output = 0 *i.e.* output is oscillating and there is no stable output.

General block diagram of multivibrator is as shown in fig. 7.3.

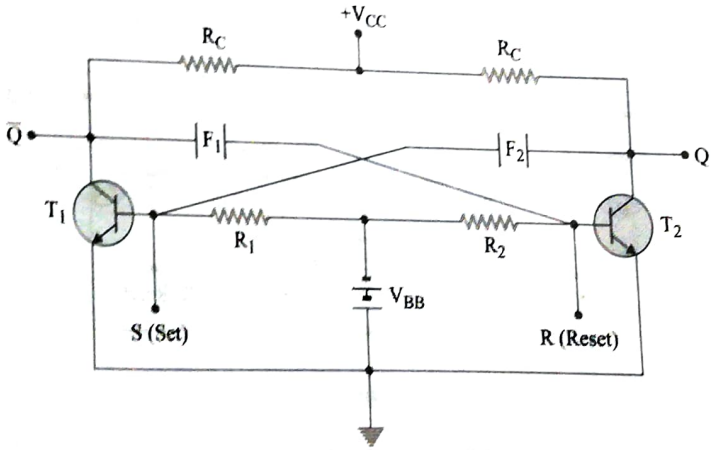


Fig. 7.3 General block diagram of multivibrator

Fig. 7.3 shows the general block diagram of multivibrator from which all multivibrators can be derived.

Astable multivibrator uses two capacitors in each feedback loop *i.e.* $F_1 = C_1$ and $F_2 = C_2$. It is as shown in fig. 7.4.

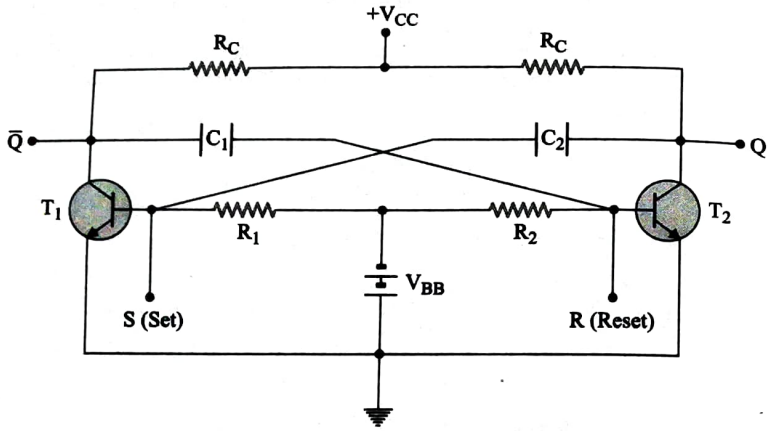


Fig. 7.4 Astable Multivibrator

Working : Let us assume T_1 is ON and T_2 is OFF. *i.e.* T_1 is in saturation state and cross coupled to T_2. If Q = 1, then $\overline{Q} = 0$, it remains same until triggered externally by pulse at S or R.

To change the states negative pulse is required at R and positive pulse is required at S. Then, Q becomes '0' and $\overline{Q} = 1$, as T_1 will be OFF and T_2 will be ON.

7.3.2 Monostable Multivibrator

It is one shot multivibrator and used for gating pulse whose width can be controlled. It has one stable state. From fig.7.3. which is a general block diagram, if we use a single storage element capacitor in one feedback loop *i.e.* $F_1 = C$ and one resistor at $F_2 = R$, then it becomes Monostable multivibrator, It is as shown in fig. 7.5.

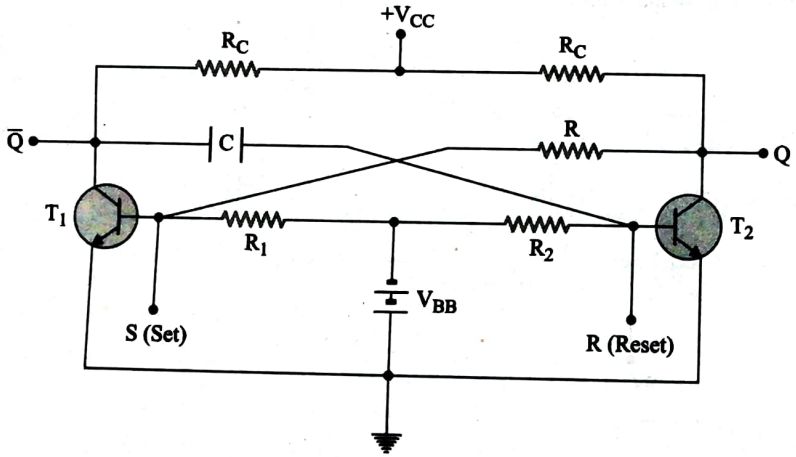


Fig. 7.5 Monostable Multivibrator

The truth table for an OR gate is as shown for its operation

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

For every combination of input, there is definite output state as shown in truth table of OR gate. Fig. 7.6. shows the output Y having feedback path for an OR gate.

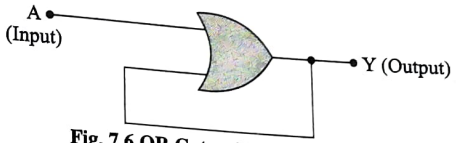


Fig. 7.6 OR Gate with Feedback path

Thus, the truth table becomes

A	Output
0	*
1	1

From truth table it is clear that if A = 1, output = 1 but if A=0, the output is not defined *i.e.* it depends upon the previous state. For example, if previous state is 1 and A = 0, output = 1 and if previous state is '0' and A = 0, output = 0. Thus, a latch circuit can be used in place of star *i.e.*, '*' to retain the output.

7.3.3 Bistable Multivibrator

It has two stable states and is called a flip flop. It has no storage elements like capacitor in the feedback loop of general block diagram (fig. 7.2). *i.e.* $F_1 = R_3$ and $F_2 = R_4$ but they all shunted by C_1 and C_2 for faster triggering speed as shown in fig. 7.7.

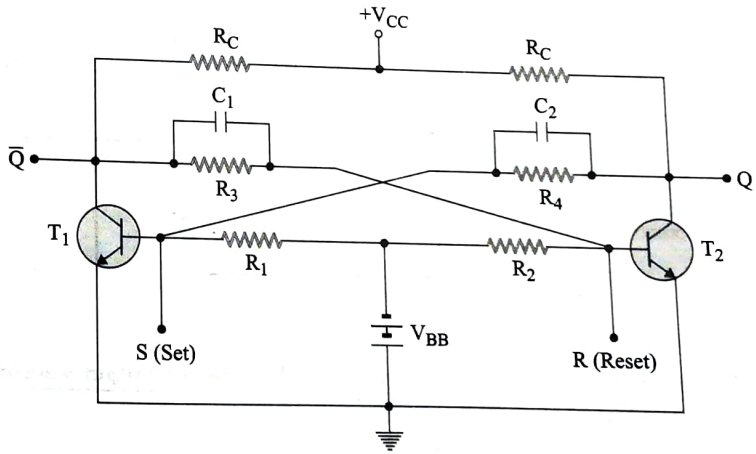


Fig. 7.7 Bistable Multivibrator

7.3.4 '555' Timer as Astable Multivibrator

555 timer is a device that can operate in various modes as it is a TTL compatible. It can be used as monostable and astable multivibrator. Figure 7.8 shows the 555 timer as Astable multivibrator

It gives a rectangular waveform at the output having two logic levels *i.e.*, High and Low and the time interval for each logic level is determined by the values of R and C connected in the circuit.

Initially, when the power is switch on, capacitor 'C' is charged with a time constant

$$\tau = RC = (R_A + R_B)C$$

During this point the output is High as flip-flop is set which is connected in 555 timer internal circuit. *i.e.*, $\bar{Q} = 0$, which has undamped the capacitor C.

When voltage across capacitor 'C' exceeds $\frac{2}{3}$ V_{CC}, the flip-flop is reset *i.e.*, $\bar{Q} = 1$. Thus, capacitor C is discharged through R_B with a time constant $\tau = R_BC$. Thus, the output goes Low at this point.

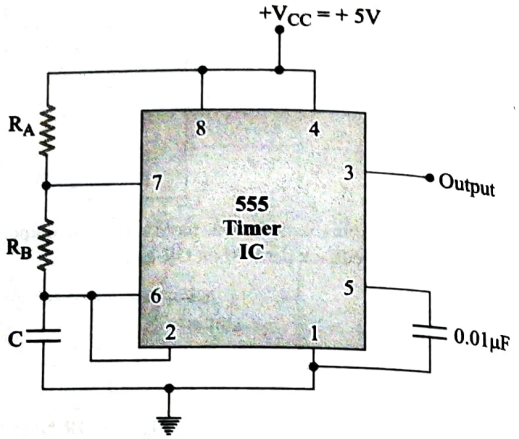


Fig. 7.8 555 Timer as Astable Multivibrator.

When it discharges to $\frac{1}{3}$ V_{CC}, the flip-flop is again set *i.e.*, $\bar{Q} = 0$ and thus, the output again goes High which has undamped the capacitor C.

In this way the capacitor C is periodically charged and discharged between $\frac{2}{3}$ V_{CC} and $\frac{1}{3}$ V_{CC} in order to repeat the cycle. Thus, when capacitor C is charged, output is High and when it is discharged, output is Low.

It will be illustrated from the waveforms during the operation as shown in fig. 7.9.

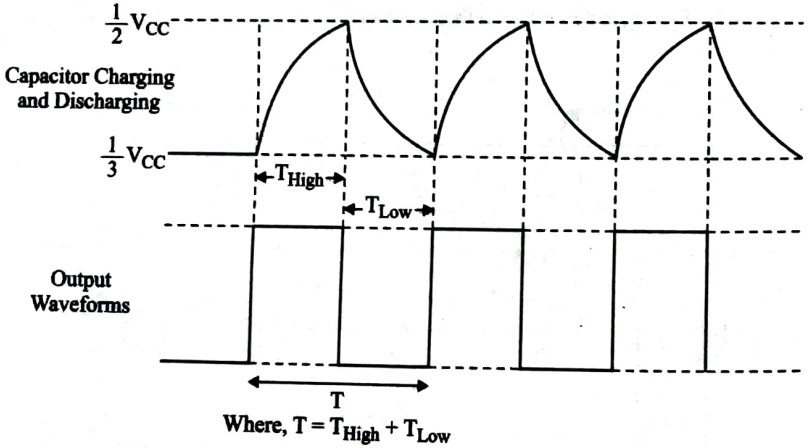


Fig. 7.9 555 Timer showing capacitor charging, discharging and output waveforms.

We know that, the voltage across the capacitor is given by

$$V_C = V_{CC}\,[1 - e^{-t/RC}]$$

The time T_{High} can be calculated as it is the time taken by the circuit to charge from $\frac{1}{3}$ V_{CC} to $\frac{2}{3}$ V_{CC} through resistor $(R_A + R_B)$ and is given by

$$\frac{1}{3}V_{CC} = \frac{2}{3}V_{CC}\,[1 - e^{-T_{High}/(R_A+R_B)C}]$$

$$1 = 2 - 2e^{-T_{High}/(R_A+R_B)C}$$

$$-1 = -2e^{-T_{High}/(R_A+R_B)C}$$

$$\frac{1}{2} = e^{-T_{High}/(R_A+R_B)C}$$

$$\therefore T_{High} = (R_A + R_B)\,C \log 2 = 0.693\,(R_A + R_B)\,C \quad \ldots(1)$$

The time T_{Low} can be calculated as it is the time taken by the circuit to discharge from $\frac{2}{3}$ V_{CC} to $\frac{1}{3}$ V_{CC} through resistance R_B and is given by

$$\frac{1}{3}V_{CC} = \frac{2}{3}V_{CC}\left[1 - e^{-T_{Low}/R_BC}\right]$$

It is similar to the equation (1) and can be written as

$$T_{Low} = R_BC \log 2$$

$$\therefore \quad T_{Low} = 0.693\,R_BC \quad \ldots(2)$$

Hence, the total time period of one cycle is given by

$$T = T_{High} + T_{Low} \quad \ldots(3)$$

Put equations (1) and (2) in equ. (3), we get

$$T = 0.693\,(R_A + R_B)\,C + 0.693\,R_BC$$

$$= 0.693\,(R_A + 2R_B)\,C \quad \ldots(4)$$

Also, the frequency is given by

$$f = \frac{1}{T} \quad \ldots(5)$$

Put equation (4) in (5), we get

$$f = \frac{1}{0.693\,(R_A + 2R_B)C}$$

$$= \frac{1.44}{(R_A + 2R_B)C} \quad \ldots(6)$$

Also, duty cycle is given by

$$D_{Cycle} = \frac{T_{High}}{T} \text{ or } \frac{T_{ON}}{T} \quad \ldots(7)$$

Put equation (1) and (4) in equation (7), we get

$$D_{Cycle} = \frac{T_{High}}{T}$$

$$= \frac{0.693\,(R_A + R_B)C}{0.693\,(R_A + 2R_B)\,C}$$

$$\therefore \quad D_{cycle} = \frac{R_A + R_B}{R_A + 2R_B}$$

7.4 LATCHES

As studied earlier, latch is a type of temporary storage device. Fig. 7.10. shows the circuit diagram for a latch formed by two NOR gates.

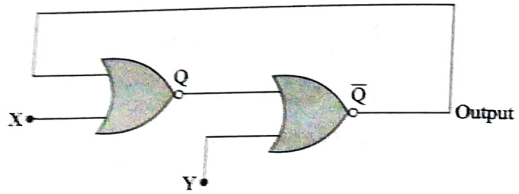


Fig. 7.10 Latch using NOR gates.

Where, X and Y are inputs and Q and $\overline{Q}$ are outputs. Q is the output of first NOR gate and $\overline{Q}$ is the complement output respectively.

Working

When X = 1 and Y = 0, Q = 0 and $\overline{Q}$ = 1 *i.e.* second NOR gate worked as inverter.

This circuit is simplest form of latch. For R-S latch put, X= R and Y = S. This circuit is also called as transparent latch because output respond immediately and input state will be latched (or temporarily stored).

7.4.1 S-R Latch

S-R latch is the simplest type of latch circuit and is known as Set-Reset latch. Universal gates are used to design the latch circuit. Latch circuit can be constructed either from two NAND gates or from two NOR gates. S-R latch is used in electronic timers.

7.4.1.1 S-R Latch using NAND Gates

It is as shown in figure 7.11 with two NAND gates cross-coupled to each other.

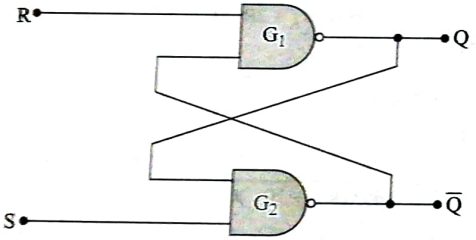


Fig. 7.11 Two input cross-coupled NAND gates S-R latch.

The two NAND gates are cross-coupled so that the output of G_1 *i.e.*, NAND 1, is connected to one of the inputs of G_2 *i.e.*, NAND 2, and vice-versa. Here, Q and $\overline{Q}$ are the latch outputs. Under normal conditions, these outputs will always be the inverse of each other. The latch inputs are set(S) and reset(R). The set input sets Q to logic '1' and the reset input resets Q to logic '0'.

There are four possible combinations for two inputs and are given in the following truth table

Inputs		Output	State
S	R	Q	
0	0	?	?
0	1	?	?
1	0	?	?
1	1	?	?

Here, only Q is taken for output because one flip-flop stores '1' bit at a time. Also, when Q = 1, the flip-flop is called set and when Q = 0, the flip-flop is called reset. So, Q is the output which is stored as a memory in the flip-flop or latch.

Let us consider the four possible combinations for its operation as shown in fig. 7.11 (*a*), (*b*), (*c*), (*d*), (*e*), (*f*) and (*g*).

Case I. When S = 0 and R = 0, we have

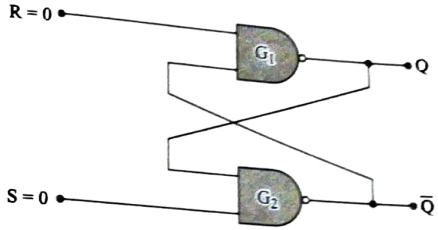


Fig. 7.11 (*a*)

As NAND gate has a property that if any of the input is '0', output is always '1'. Thus, we get Q = 1 and $\overline{Q}$ = 1. Which is not possible ($\because Q \neq \overline{Q}$). Thus, this state is avoided and is marked as invalid or indeterminate or ambiguous state.

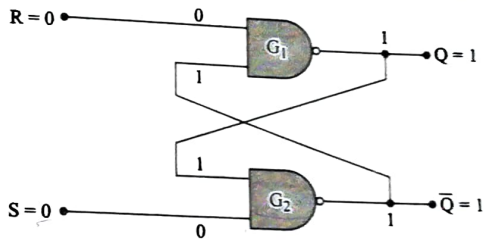


Fig. 7.11 (*b*)

When, R = 0, output of G_1 *i.e.*, Q = 1 and when S = 0, output of G_2 *i.e.*, $\overline{Q}$ = 1. Also, from fig. 7.11 (*b*) $Q = \overline{0.1} = \overline{0} = 1$ and $\overline{Q} = \overline{1.0} = \overline{0} = 1$. Thus, Q = 1 and $\overline{Q}$ = 1. Which is an Invalid state.

Case II. When S = 0 and R = 1, we have

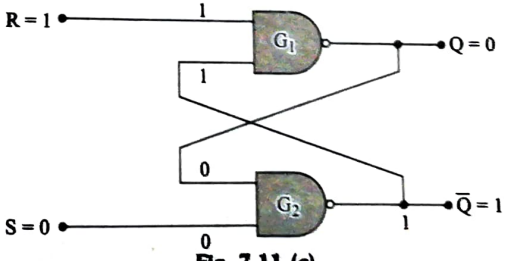


Fig. 7.11 (c)

From NAND gates property, we have any of the input is '0' output is always '1'. Also, output of G_2 *i.e.*, $\overline{Q}$ = 1. Now output of G_2 is input to G_1 and another input to G_1 is R = 1. Thus, from fig. 7.11 (*c*), we get $Q = \overline{\overline{Q}.R} = \overline{1.1} = \overline{1} = 0$. So, it is clear that when S = 0 and R = 1, the output Q = 0 *i.e.*, Reset state.

Case III. When S = 1 and R = 0, we have

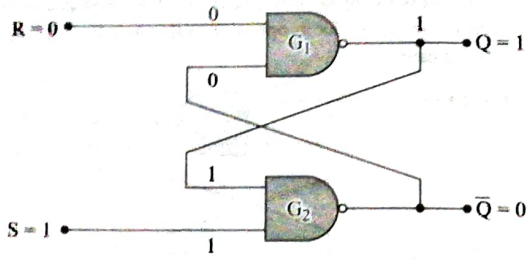


Fig. 7.11 (*d*)

From NAND gates property if any of the input is '0', output is always '1' we get output of G_1 *i.e.*, Q = 1. Similarly, from fig. 7.11 (*d*), we get, $\bar{Q} = \overline{Q.S} = \overline{1.1} = \bar{1} = 0$. So, it is clear that when S = 1 and R = 0, the output Q = 1 *i.e.* Set state

Case IV. When S = 1 and R = 1, we have

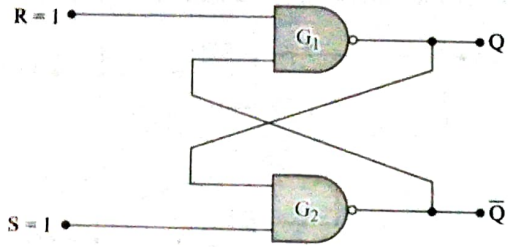


Fig. 7.11 (*e*)

This is a special case in which previously stored data is required for its operation. The previously stored data may be '0' or '1'. Let us take, if prior data is '0', we have

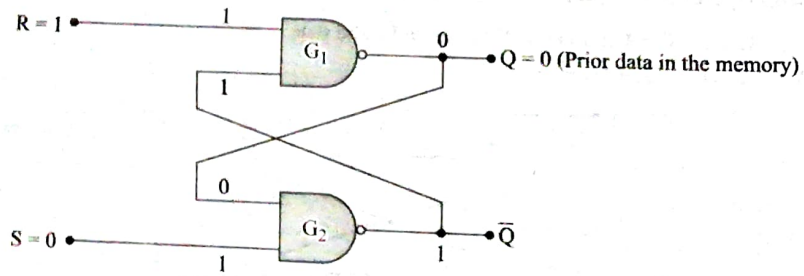


Fig. 7.11 (*f*)

When previously stored data is assumed as '0' as shown in fig. 7.11 (*f*), the output of NAND gate G_2 is $\bar{Q} = \overline{0.1} = \bar{0} = 1$ *i.e.*, $\bar{Q} = 1$. Similarly, output of G_1 is $Q = \overline{1.1} = \bar{1} = 0$ *i.e.*, Q = 1. Thus, we get the same data at the output which was assumed earlier. Hence, there is no change state when S = 1, R = 1 for Q = 0 (prior data in memory).

Let us consider if prior data is '1', we have

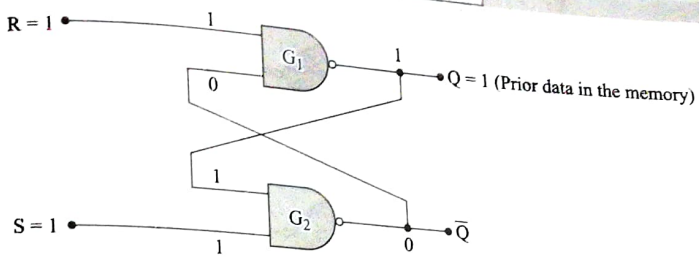


Fig. 7.11 (*g*)

When previously stored data is assumed as '1', from fig. 7.11 (*g*) the output of NAND gate G_2 is $\bar{Q} = \overline{1.1} = \bar{1} = 0$ *i.e.*, $\bar{Q} = 0$. Similarly, output of G_1 is $Q = \overline{1.0} = \bar{0} = 1$ *i.e.*, Q = 1. Thus, we get the same data at the output which was assumed earlier. Hence, there is no change state when S = 1, R = 1 for Q = 1 (prior data in memory).

So, it is clear that when S = 1 and R = 1, the output Q gives No change state.

The truth table from above cases can be written as

Inputs		Output	State
S	R	Q	
0	0	X	Invalid
0	1	0	Reset *i.e.*, '0'
1	0	1	Set *i.e.*, '1'
1	1	Q	No Change

7.4.1.2 S-R Latch using NOR Gates

S-R latch using NOR gates is similar to the S-R NAND latch except that the Q and $\bar{Q}$ outputs have reversed positions.

It is as shown in figure 7.12

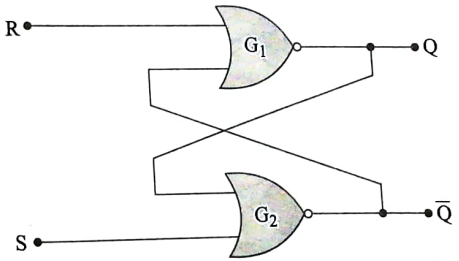


Fig. 7.12 Two input cross-coupled NOR gates S-R latch.

It make use of two cross-coupled NOR gates so that the output of G_1 *i.e.*, NOR 1 is connected to one of the inputs of G_2 *i.e.*, NOR 2, and vice-versa. Its operation is similar to that of S-R NAND latch.

It has four possible combinations as shown in fig. 7.12 (*a*), (*b*), (*c*), (*d*), (*e*) and (*f*), let us discuss one by one.

Case I. When $S = 0$ and $R = 0$, we have

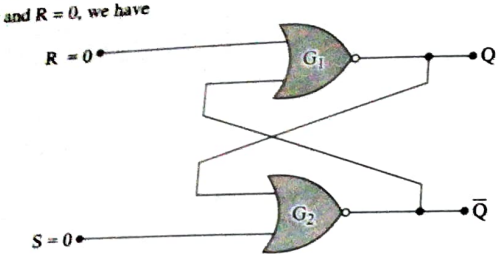


Fig. 7.12 (*a*)

As NOR gate has a property that if any of the input is '1', output is always '0'. Here, both the inputs are '0' thus, previously stored data is required for its operation. The previously stored data may be '0' or '1'. Let us assumed that if the prior data is '0', we have

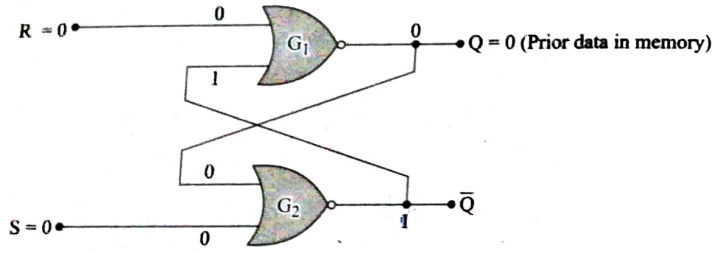


Fig. 7.12 (*b*)

When previously stored data is assumed as '0' as shown in fig. 7.12 (*b*), the output of NOR gate G_2 is $\bar{Q} = \overline{0+0} = \bar{0} = 1$ *i.e.*, $\bar{Q} = 1$. Similarly, output of G_1 is $Q = \overline{0+1} = \bar{1} = 0$ *i.e.*, $Q = 0$. Thus, we get the same data at the output which was assumed earlier. Hence, there is no change state when $S = 0$, $R = 0$ for $Q = 0$ (prior data in the memory). Let us consider if prior data is '1', we have

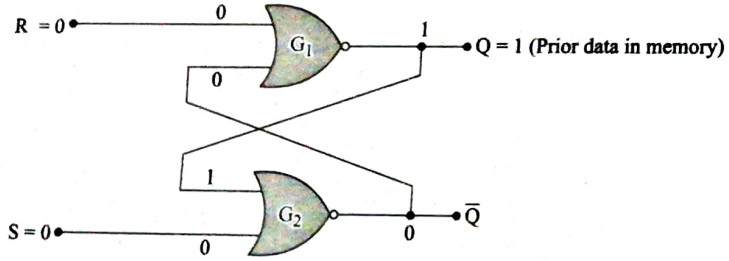


Fig. 7.12 (*c*)

When previously stored data is assumed as '1' as shown in fig. 7.12 (*c*), the output of NOR gate G_2 is $\bar{Q} = \overline{1+0} = \bar{1} = 0$ *i.e.*, $\bar{Q} = 0$. Similarly, output of G_1 is $Q = \overline{0+0} = \bar{0} = 1$ *i.e.*, $Q = 1$. Thus, we get the same data at the output which was assumed earlier. Hence, there is no change state when $S = 0$, $R = 0$ for $Q = 1$ (prior data in memory).

So, it is clear that when $S = 0$ and $R = 0$, the output Q give No change state.

Case II. When $S = 0$ and $R = 1$, we have

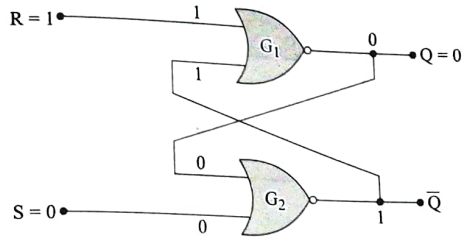


Fig. 7.12 (*d*)

From NOR gates property, if any of the input is '1' output is always '0'. From fig. 7.12 (*d*) output of G_1 *i.e.*, $Q = 0$. Now output of G_1 is input to G_2 and another input to G_1 is $S = 0$. Thus, we get $\bar{Q} = \overline{Q+S} = \overline{0+0} = \bar{0} = 1$.

So, it is clear that when $S = 0$ and $R = 1$, the output $Q = 0$ *i.e.*, Reset state.

Case III. When $S = 1$ and $R = 0$, we have

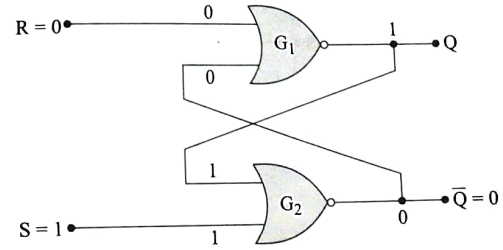


Fig. 7.12 (*e*)

From NOR gates property if any of the input is '1', output is always '0'. From fig. 7.12 (*e*), we get output of G_2 *i.e.*, $\bar{Q} = 0$. Similarly, $Q = \overline{\bar{Q}+R} = \overline{0+0} = \bar{0} = 1$. So, it is clear that when $S = 1$ and $R = 0$, the output $Q = 1$ *i.e.*, Set state.

Case IV. When $S = 1$ and $R = 1$, we have

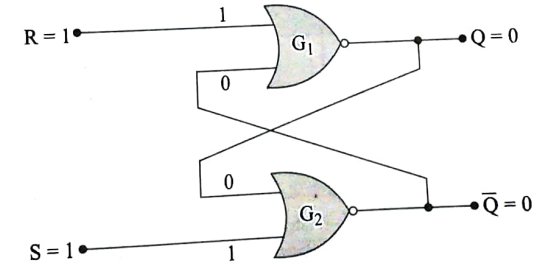


Fig. 7.12 (*f*)

From NOR gates property if any of the input is '1', output is always '0'. From fig. 7.12 (*f*), we get output of G_1 and G_2 both are '0' *i.e.*, $Q = 0$ and $\bar{Q} = 0$ which is not possible ($\because Q \neq \bar{Q}$). Hence, it is an invalid state.

The truth table from above cases can be written as

Inputs		Output	State
S	R	Q	
0	0	Q	No Change
0	1	0	Reset *i.e.*, '0'
1	0	1	Set *i.e.*, '1'
1	1	X	Invalid

7.4.2 Gated Latches (Clocked Flip-Flops)

The latches in which there is no clock or enable signal *i.e.*, the output can change state at any time when the input conditions are changed are called as asynchronous latches. In case of gated latches enable (EN) input signal is required and the output will change state whenever enable input signal is High. Thus, gated latches are also called as clocked or synchronous latches.

7.4.2.1 Gated S-R Latch

Gated S-R latch needed an enable (EN) input signal. Enable signal may be a clock and due to this reason it is also known as clocked S-R latch. When enable (EN) input is high, the change in output will take place. When enable (EN) input is low, there is no change in the outputs. Its logic symbol and internal structure is as shown in fig. 7.13 (*a*) and (*b*)

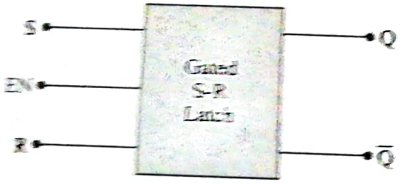


Fig. 7.13 (a) Logic Symbol.

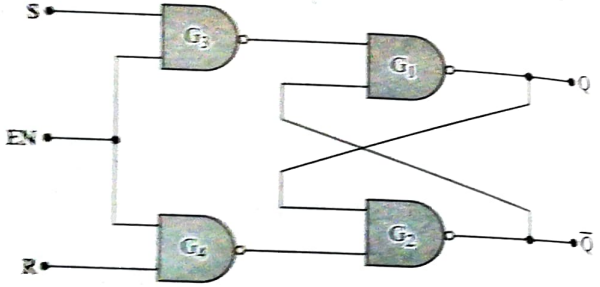


Fig. 7.13 (b) Internal circuit using NAND gates.

The truth table of S-R latch is as shown

Inputs			Output	State
EN	S	R	Q	
1	0	0	Q	No Change
1	0	1	0	Reset *i.e.*, '0'
1	1	0	1	Set *i.e.*, '1'
1	1	1	X	Invalid
0	X	X	Q	No Change

It is clear from the truth table that when EN = 1, S–R latch works for four possible combinations *i.e.*, as a normal latch. But when EN = 0, output of gates G_3 and G_4 is always high irrespective of inputs S and R. Thus, the output dosen't change for EN = 0, therefore, no change state occurs.

From fig. 7.13 (*b*) *i.e.*, the internal circuit using NAND gates another equivalent circuit can be designed by using AND and NOR gates. It is as shown in fig. 7.14.

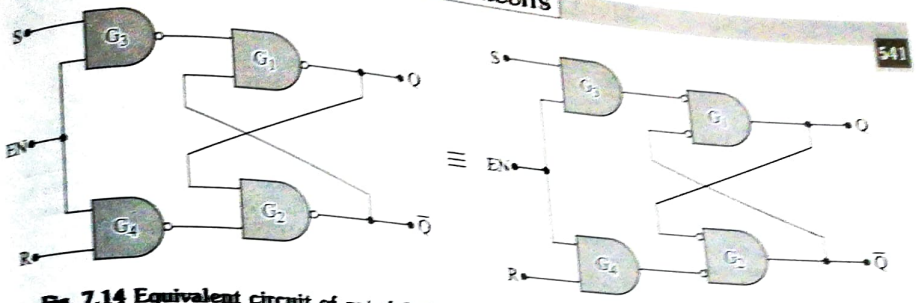


Fig. 7.14 Equivalent circuit of gated S-R latch using AND gates and bubbled AND gates.

In fig. 7.14, if the bubbled AND gates are replaced by NOR gates then the circuit of S–R latch can be designed by using AND and NOR gates only as shown in fig. 7.15.

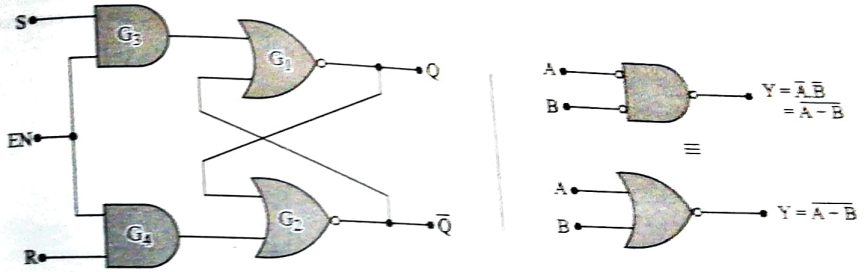


Fig. 7.15 Equivalent circuit of gated S–R latch using AND and NOR gates as bubbled AND gate is equivalent to NOR gate.

7.4.2.2 Gated D-Latch

It is designed from S–R latch by putting an inverter or NOT gate between S and R inputs and connected together to make a single input D as shown in fig. 7.16 (*a*) and (*b*).

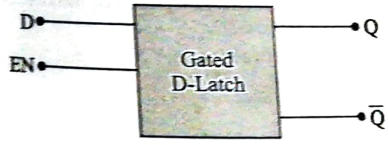


Fig. 7.16(a) Logic Symbol.

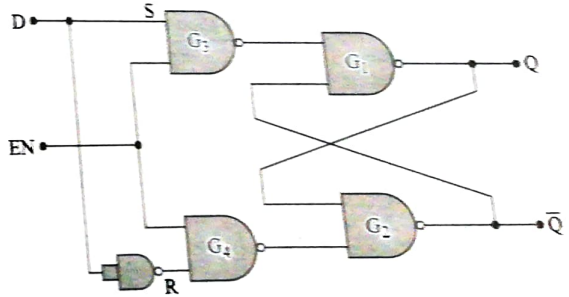


Fig. 7.13 (b) Internal circuit using NAND gates.

D latch is used whenever S = R = 0 and S = R = 1 combinations are not required. The input D is called the data input and thus it is called as D-latch. Its truth table is as shown

Inputs		Output	State
EN	D	Q	
1	0	0	Reset
1	1	1	Set
0	X	Q	No Change

Thus, when D = 0, we have S = 0 and R = 1, which will provide reset state *i.e.*, Q = 0 for enable, EN = 1. Similarly, when D = 1, we have S = 1 and R = 0, which will provide set state *i.e.*, Q = 1 for enable, EN = 1. But when enable input EN = 0 is provided, output of gates G_3 and G_4 is always high irrespective of input D. Thus, the output doesn't change by EN = 0, therefore, no change state occurs.

7.5 FLIP-FLOPS

Flip-flop is a sequential circuit having two stable output states and it can stay in any one of the two stable states unless external input is applied. So, it is the basic memory element used to store a bit of information *i.e.* either '0' or '1'. In other words, flip-flop is a sequential circuit which is used to store single bit of information or data at a time *i.e.*, either '0' or '1'. Single flip-flop stores single bit, thus for storage of 4-bit data, four flip-flops are required.

Flip -flop is different from latch by the method used for changing states. It is used in arithmetic circuits because of its high speed. It is useful in registers, accumulators with address, counter applications, error detection and data conversion.

7.5.1 Concept of Flip-Flop

The basic idea of the feedback in sequential circuits is to store or hold the value as shown in figure 7.17.

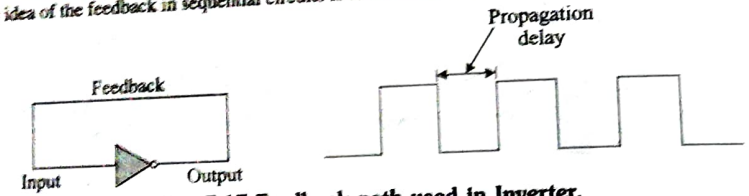


Fig. 7.17 Feedback path used in Inverter.

But output keeps toggling *i.e.*, output changes from 0 to 1 then 1 to 0 then 0 to 1 and so on. To overcome this problem cascading of two inverters are used as shown in fig. 7.18.

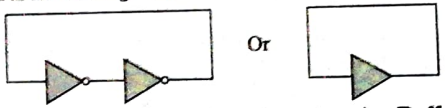


Fig. 7.18 Cascading of two Inverters (or Buffer)

But again the problem occurs that the buffer output cannot be known. As we studied earlier this problem is overcome by cross coupled NAND or cross coupled NOR gates. To control the input we can use enable signal. For the case of LATCH, level sensitive control is used and in case of flip flops, edge sensitive control is used.

Fig. 7.19 shows the circuit of two level sensitive memory elements with phase shift between them.

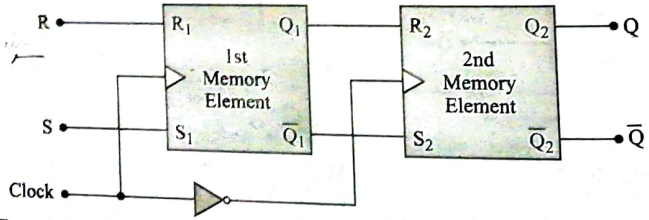


Fig. 7.19 Two level sensitive memory elements with phase shift.

The input R–S is given to second R–S latch when, Clock (CLK) changes their state. This state change of CLK from LOW to HIGH is rising edge and HIGH to LOW is falling edge.

Fig. 7.20, shows how clock pulse represents with rise and fall times.

The width of pulse in TTL showed in fig. 7.20 is not less than 30ns but the rise and fall showed are less than 30ns. The order of rise and fall time should be same to overcome oscillations in logic circuits.

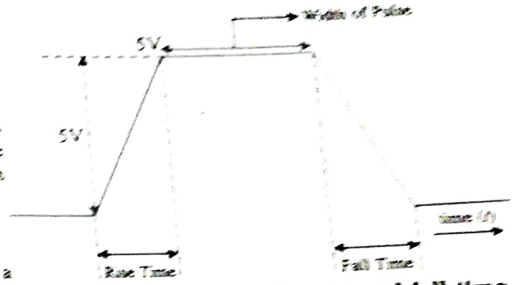


Fig. 7.20 Clock Pulse with rise and fall time.

7.5.2 Types of Clocks

Clock is a digital signal in the form of a square wave or a rectan-gular pulse train. There are basic two types of clock signals *i.e.*, square wave clock signal (or square clock) and rectangular pulse train clock signal (or pulse clock). These are as shown in fig. 7.21 (*a*) and (*b*).

(*a*) Square wave clock signal

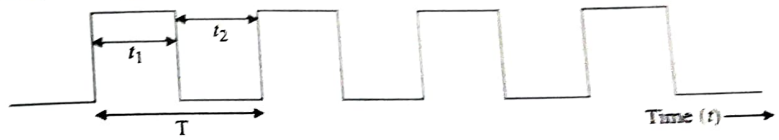


Fig. 7.21 (*a*) Square wave clock signal.

Here in fig. 7.21 (*a*) one complete cycle is given by $T = t_1 + t_2$.

where, T is the time period of one complete cycle (*i.e.*, positive half cycle and negative half cycle), t_1 is the width of first half cycle and t_2 is the width of second half cycle. When the width $t_1 = t_2$, then the clock is known as a square clock.

(*b*) Rectangular pulse train clock signal

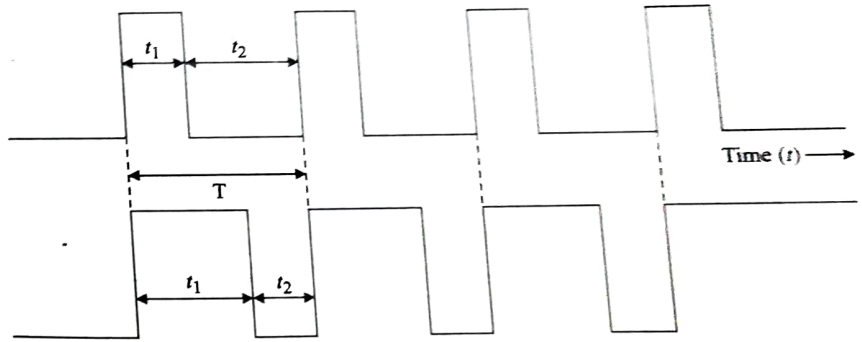


Fig. 7.21 (*b*) Rectangular pulse train clock signals.

Here in fig. 7.21 (*b*) one complete cycle is given by

$$T = t_1 + t_2$$

When the width $t_1 \neq t_2$, then the clock is known as pulse clock.

The clock is used in the sequential circuits to provide timing and synchronization. It means the digital system in which clock is used will change state only when the clock makes a transition. There are two types of transitions.

(*a*) Positive going transition (PGT). (*b*) Negative going transition (NGT).

When clock changes from 0 to 1 states, then it is called as positive going transition (PGT) and when clock changes from 1 to 0 states, then it is called as negative going transition (NGT). It will be more clear from the waveforms as shown in fig. 7.22.

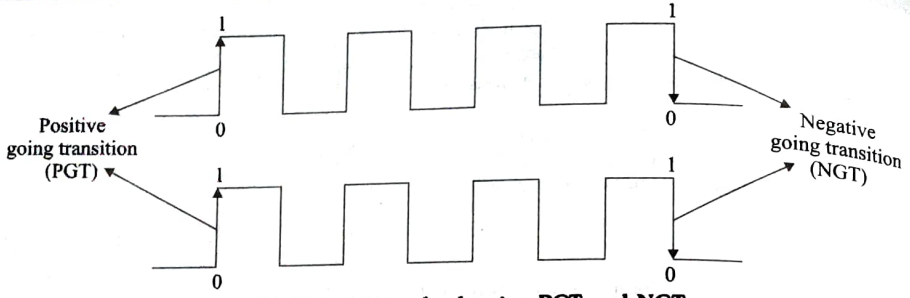


Fig. 7.22 Clock signals showing PGT and NGT.

Positive going transition is also called as rising edge and negative going transition is also called as falling edge. In clocked flip-flops the clock is abbreviated by clk or CLK or CK or CP. The clock can be activated by PGT or NGT as shown in fig. 7.23.

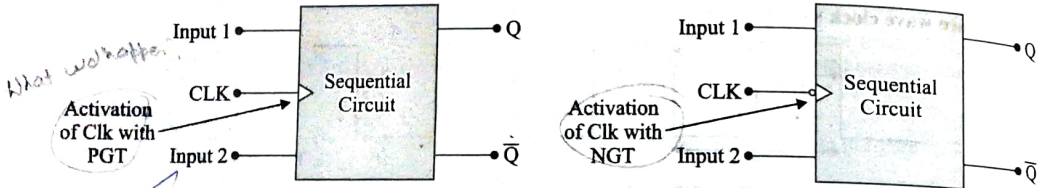
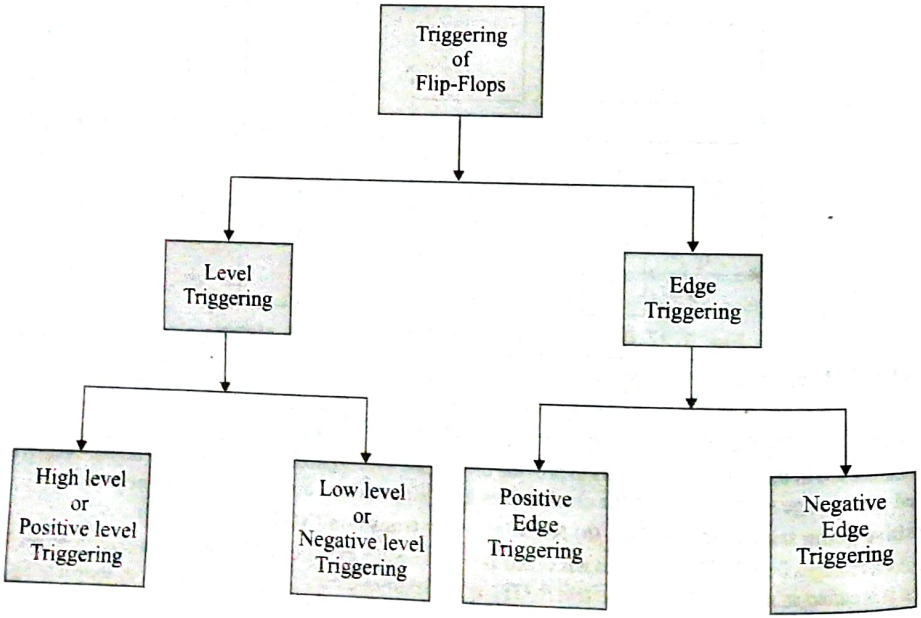


Fig. 7.23 Activation of clock signal in sequential circuit with PGT and NGT.

7.5.3 Triggering of Flip-Flops

Trigger is change of state from one to another in which the state of a flip-flop is switched by a momentary change in the input signal. Due to change of state, transition occurs and then it is said to trigger the flip-flop. Thus, triggering is the process of change of state of flip-flop by applying input signal.

Triggering of flip-flops is of basic two types :



In level triggering the input signal will affect the flip-flop for logic 1 or high logic of the clock. But in edge triggering the input signal will affect the flip-flop for either positive going edge or negative going edge of the clock pulse. Fig. 7.24 (*a*) and (*b*) shows the waveforms for level and edge triggering.

(*a*) Level Triggering

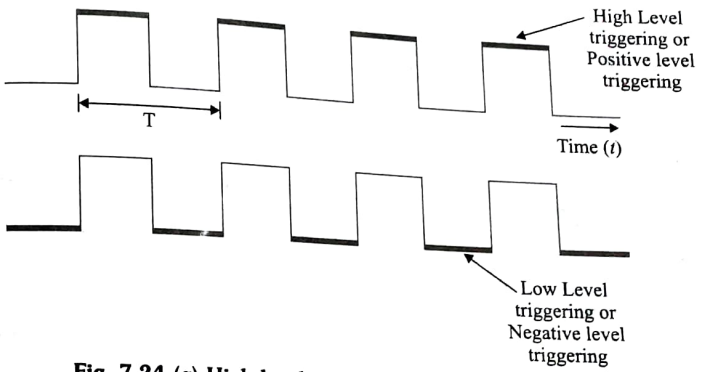


Fig. 7.24 (*a*) High level and low level triggering waveforms.

(*b*) Edge Triggering

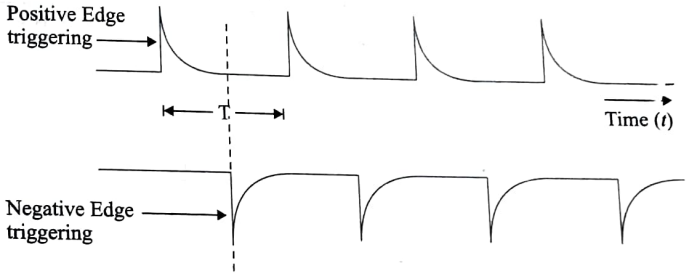


Fig. 7.24 (*b*) Positive edge and negative edge triggering waveforms.

Edge triggering is obtained by using a differentiator circuit, which make use of R and C passive components. As we know RC circuit gives time constant, thus can be used for edge triggering.

Differentiator circuit using R and C is as shown in fig. 7.25 with input and output waveforms.

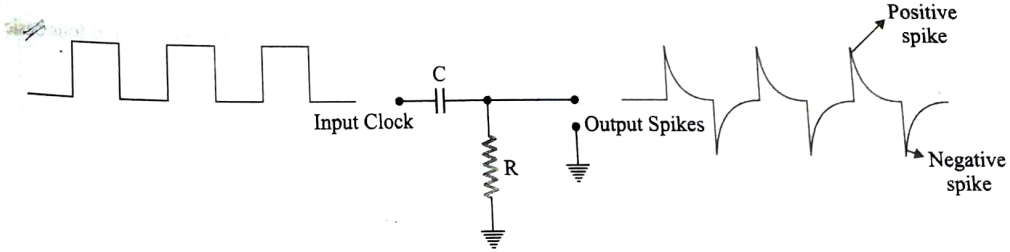


Fig. 7.25 RC differentiator circuit with input and output waveforms.

The signal used at the input is '*ac*' instead of '*dc*'. Because of the holding action of *dc*, it cannot be used. In order to get the positive and negative edge triggered clocks a diode is used in series with the RC circuit at its output stage as shown in fig. 7.26 (*a*) and (*b*).

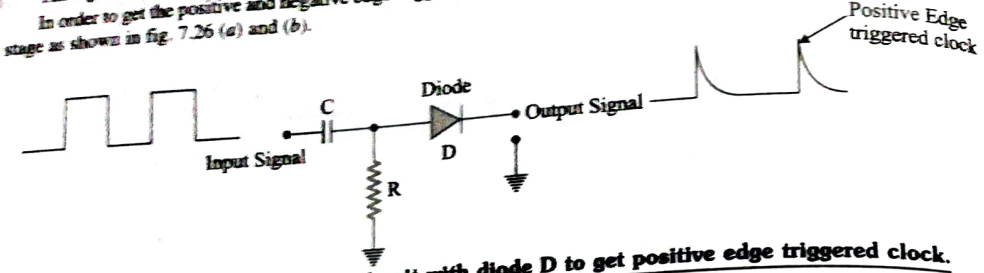


Fig. 7.26 (*a*) RC differentiator circuit with diode D to get positive edge triggered clock.

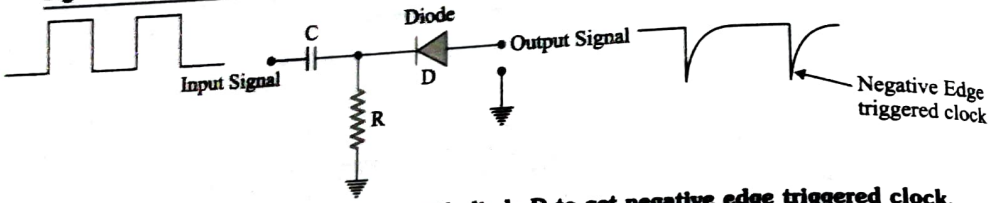


Fig. 7.26 (*b*) RC differentiator circuit with diode D to get negative edge triggered clock.

Thus, the input clock signal with output spikes or edges from an RC differentiator circuit is as shown in fig. 7.27.

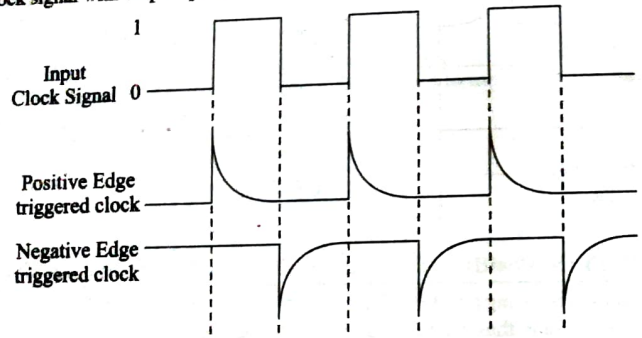
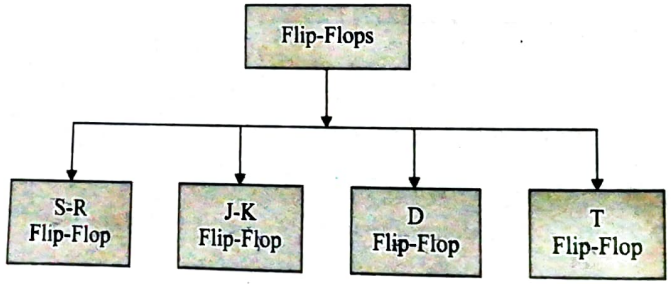


Fig. 7.27 Input clock signal and output edge triggered waveforms for RC differentiator circuit.

7.5.4 Types of Flip-Flops

Flip-flops are the basic memory element for storage of data in binary form. Flip-flops are classified into four basic types



7.5.4.1 S-R Flip-Flop

An S-R flip-flop is the set-reset flip-flop. It consists of gated S-R latch with clock circuit instead of enable signal. The clock circuit is a differentiator circuit connected at the clock input of flip-flop. It may be positive edge triggered or negative edge triggered. The functional block diagram of S-R flip-flop *i.e.*, logic symbol and its internal structure is as shown in fig. 7.28.

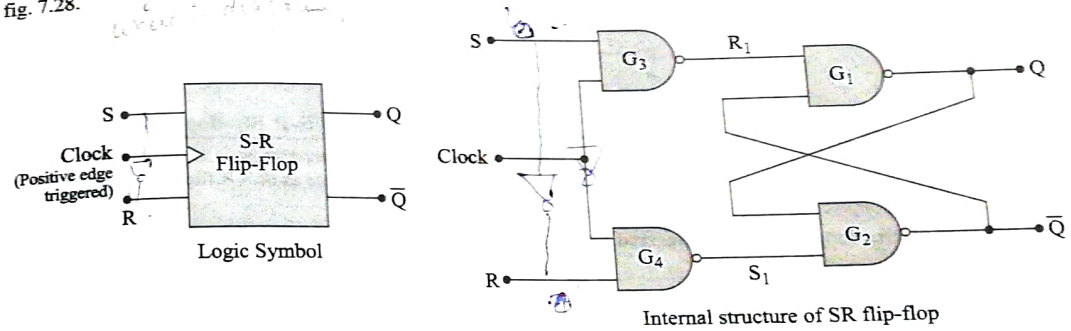


Fig. 7.28 Logic symbol and internal structure of S–R flip-flop.

The S-R flip-flop can change states only when the signal applied to its clock input makes transition from 0 to 1. The operation of S-R flip-flop depends on different combinations of S and R inputs.

Case I. When, S = 0 and R = 0, we have

Output of gates G_3 and G_4 are always '1' irrespective of clock signal. Thus, '1' is input to gates G_1 and G_2, which provides the output for No Change. As it is a NAND latch. Therefore, when S = R = 0, output Q is a No Change state.

Case II. When, S = 0 and R = 1, we get

Output of gate G_3 is always '1' and G_4 is '0' as second input is clock. Thus, the inputs for gates G_1 and G_2 are '1' and '0' respectively. From NAND latch, we get output Q = 0, which is a Reset state.

Case III. When, S = 1 and R = 0, we get

Output of gate G_4 is always '1' and G_3 is '0'. Thus, the inputs for gates G_1 and G_2 are '0' and '1' respectively. From NAND latch, we get output Q = 1, which is a Set state.

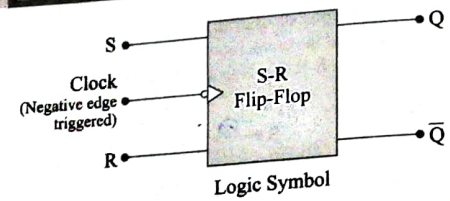
Case IV. When, S = 1 and R = 1, we get

Output of both gates *i.e.*, G_3 and G_4 are '0'. Thus, the inputs for gates G_1 and G_2 are '0'. From NAND latch, we get outputs $Q = \overline{Q} = 1$, which is not possible. Hence, Invalid state.

From above combinations the truth table for S-R flip-flop can be obtained as shown

Inputs			Output	State	
Clk	S	R	Q		
↑	0	0	Q	No Change	Positive Edge Triggered Clock
↑	0	1	0	Reset	
↑	1	0	1	Set	
↑	1	1	X	Invalid	
0	X	X	Q	No Change	Flip-flop is disabled
1	X	X	Q	No Change	
↓	X	X	Q	No Change	

Similarly, the truth table for S-R flip-flop using negative edge triggered clock can be obtained, the only differ that it responds during the negative edge of the clock input. The functional block diagram of negative edge trigge flip-flop is as shown in fig. 7.29.



Logic Symbol

Fig. 7.29 Logic symbol or functional block diagram of S–R flip-flop.

The only difference in clock is that, during positive bubble is absent and when negative edge triggered clock is used bubble is added as shown in the logic symbol. Its internal structure using gates is same as of S-R flip-flop using positive edge triggered clock.

Timing waveforms for positive edge triggered S-R flip-flop are shown in fig. 7.30, assuming any data for S and R inputs, we have

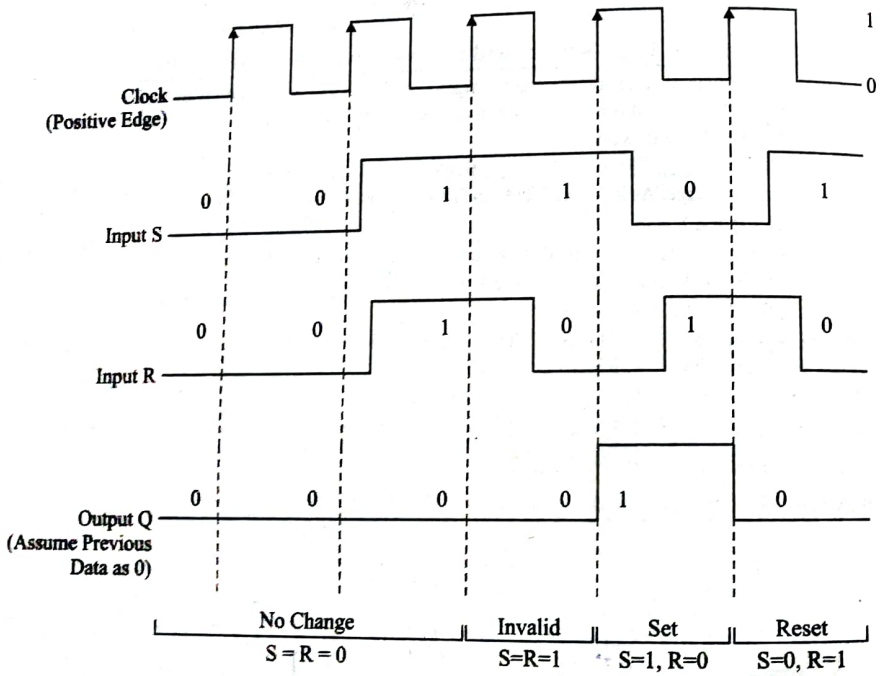


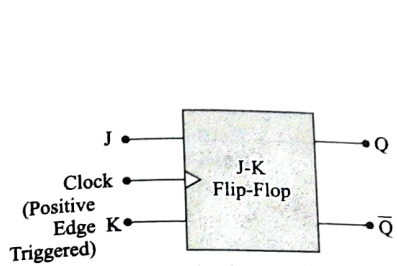
Fig. 7.30 Timing waveforms for positive edge trigger S–R flip-flop.

The change will occur only during the positive edge of the clock pulse. The output remains previous one until next positive edge of clock takes place.

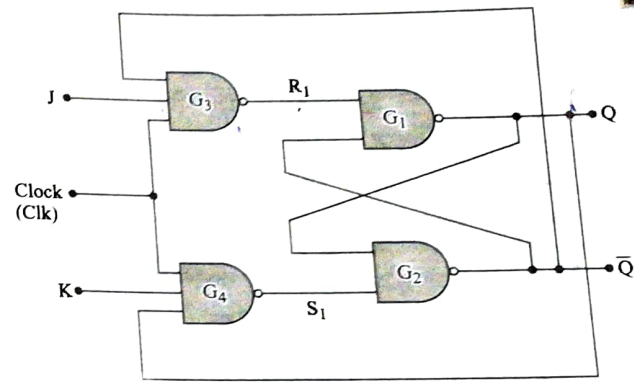
7.5.4.2 J-K Flip-Flop

The disadvantage of S-R flip-flop is its invalid state *i.e.*, $Q = \overline{Q}$, which is not possible. To overcome this disadvantage K flip-flop is designed. The J and K designations for the synchronous control inputs doesn't have any known significance.

The operation of J-K flip-flop is similar to that of S-R flip-flop except the invalid state of S-R flip-flop *i.e.*, it overcomes invalid state. J-K flip-flop will be designed by doing slight changes in S-R flip-flop as shown in fig. 7.31.



Logic Symbol



Internal structure of SR flip-flop

Fig. 7.31 Logic symbol and internal structure of J-K flip-flop.

If the output Q of S-R flip is connected at the input of gate G_4 and output $\overline{Q}$ is connected at the input of gate G_3, then the S-R flip-flop changes to J-K flip-flop. It overcomes the invalid state of S-R flip-flop. The expressions for R_1 and S_1 are given by

$$R_1 = \overline{Clk.J.\overline{Q}} \quad \text{and} \quad S_1 = \overline{Clk.K.Q}$$

The operation of J-K flip-flop depends on the different combinations of J and K inputs.

Case I. When J = 0 and K = 0, we have

Outputs of gates G_3 and G_4 are always '1' irrespective of clock signal and output feedbacks. Thus, '1' is input for gates G_1 and G_2, which gives output for No Change. As it is a NAND latch. Therefore, when S = R = 0, output Q is a No Change state.

Case II. When J = 0 and K = 1, we have

Output of gate G_3 is always '1' and G_4 is '0'. It is clear from the expressions for R_1 and S_1 as shown

$$R_1 = \overline{Clk.J.\overline{Q}} \quad \text{and} \quad S_1 = \overline{Clk.K.Q}$$

If the previous state of Q and $\overline{Q}$ is given by

Q = 1 and $\overline{Q}$ = 0, then the expressions for R_1 and S_1 becomes

$$R_1 = \overline{Clk.J.\overline{Q}} = \overline{1.0.0} = \overline{0} = 1$$

$$S_1 = \overline{Clk.K.Q} = \overline{1.1.1} = \overline{1} = 0$$

Thus, from S-R latch if $R_1 = 0$ and $S_1 = 0$, we have

$$Q = 0 \text{ and } \overline{Q} = 1$$

Therefore, when J = 0 and K = 1, output Q = 0 *i.e.*, Reset state.

Case III. When J = 1 and K = 0, we have

Output of gate G_4 is always '1' and G_3 is '0'.

If the previous state of Q and $\overline{Q}$ is given by

Q = 0 and $\overline{Q}$ = 1, then the expressions for R_1 and S_1 becomes

$R_1 = \overline{Clk.J.\bar{Q}} = \overline{1.1.1} = \bar{1} = 0$

$S_1 = \overline{Clk.K.Q} = \overline{1.0.0} = \bar{0} = 1$

Thus, from S-R latch if $R_1 = 0$ and $S_1 = 1$, we have

$Q = 1$ and $\bar{Q} = 0$.

Therefore, when J = 1 and K = 0, output Q = 1 *i.e.*, Set state.

Case IV. When J = 1 and K = 1, we have

This is special case in which previously stored data is required for its operation. The data may be '0' or '1'.

Let us take if prior data is '0' *i.e.*, Q = 0 and $\bar{Q} = 1$, we have

$R_1 = \overline{Clk.J.\bar{Q}} = \overline{1.1.1} = \bar{1} = 0$

$S_1 = \overline{Clk.K.Q} = \overline{1.1.0} = \bar{0} = 1$

Thus, $R_1 = 0$ and $S_1 = 1$

From S-R latch, we get output Q = 1 *i.e.*, Set state.

Let us take if prior data is '1' *i.e.*, Q = 1 and $\bar{Q} = 0$, we have

$R_1 = \overline{Clk.J.\bar{Q}} = \overline{1.1.0} = \bar{0} = 1$

$S_1 = \overline{Clk.K.Q} = \overline{1.1.1} = \bar{1} = 0$

Thus, $R_1 = 1$ and $S_1 = 0$

From S-R latch, we get output Q = 0 *i.e.*, Reset state.

It is clear from above discussion that when J = K = 1, toggling takes place *i.e.*, if previous data was '0' with the passage of clock at same inputs J = K = 1, the output toggles from 0 to 1. Also, if previous data was '1' with the passage of clock at same inputs J = K = 1, the output again toggles from 1 to 0. Thus, outputs are inverted every time when clock pulse is provided.

Therefore, when J = K = 1, the output Q is a Toggle state.

The truth table for J-K flip-flop (positive-edge triggered) is as shown

Inputs			Outputs		State	
Clk	J	K	Q	$\bar{Q}$		
↑	0	0	Q	$\bar{Q}$	No Change	Positive Edge Triggered Clock
↑	0	1	0	1	Reset	
↑	1	0	1	0	Set	
↑	1	1	$\bar{Q}$	Q	Toggle	
0	X	X	Q	$\bar{Q}$	No Change	Flip-flop is disabled
1	X	X	Q	$\bar{Q}$	No Change	
↓	X	X	Q	$\bar{Q}$	No Change	

Fig. 7.32 shows the timing waveforms for positive edge triggered J-K flip-flop, assuming any data for J and K inputs, we have

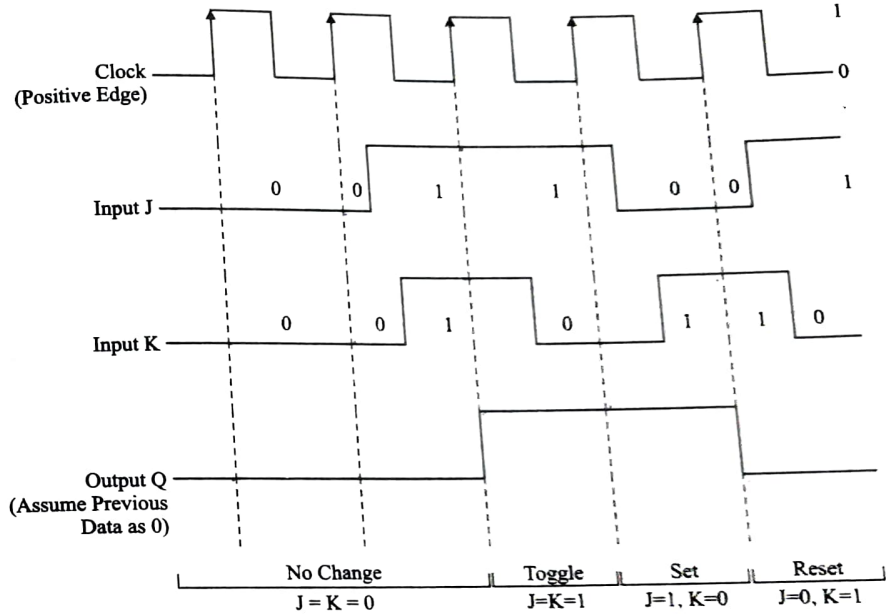


Fig. 7.32 Timing waveforms for positive edge triggered J-K flip-flop.

7.5.4.3 D Flip-Flop (Delay Flip-Flop)

It can be designed from S-R flip-flop and J-K flip-flop by putting an inverter or NOT gate between S and R (or) J and K inputs and connected together to make a single input D as shown in fig. 7.33 (*a*) and (*b*).

Using S-R flip-flop

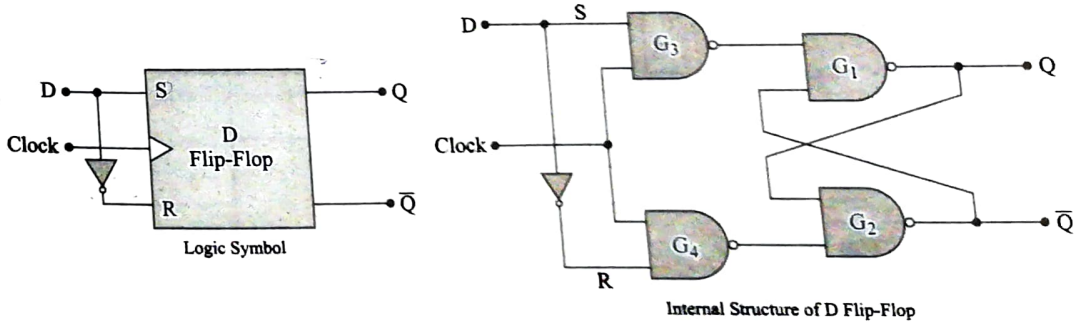


Fig. 7.33 (*a*) Logic symbol and internal structure of D flip-flop using S-R flip-flop.

Using J-K flip-flop

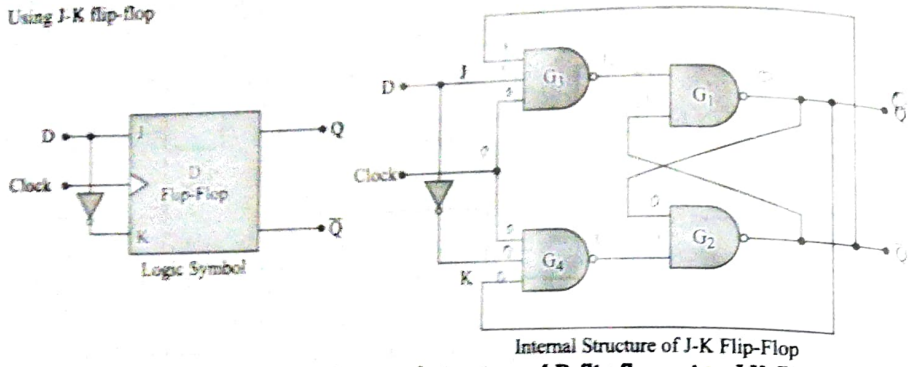


Fig. 7.33 (*b*) Logic symbol and internal structure of D flip-flop using J-K flip-flop.

D flip-flop is used whenever S = R = 0 and S = R = 1 (or) J = K = 0 and J = K = 1 combinations are not required for particular flip-flop used. The input D is called the data input and thus it is named as D-flip-flop. For D = 0, either S = 0 and R = 1 or J = 0 and K = 1. For this input output Q is always Reset *i.e.*, 0. Similarly, for D = 1, either S = 1 and R = 0 or J = 1 and K = 0. For this input Q is always set *i.e.*, 1.

Its truth table is as shown

Inputs		Output	State
Clk	D	Q	
↑	0	0	Reset
↑	1	1	Set
0	X	Q	No Change

It is similar to D latch. It is important because if we want to store 0 or 1 in the flip-flop, then, same inputs are applied to D. Thus, it is very useful in registers for storage of data.

Fig. 7.34 shows the timing waveforms for positive edge triggered D flip-flop, assuming any data for D input, we have

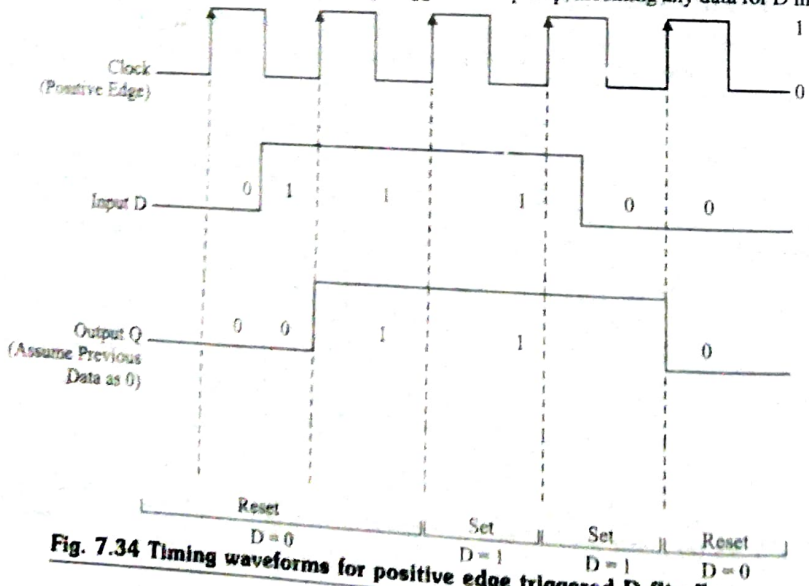


Fig. 7.34 Timing waveforms for positive edge triggered D flip-flop.

7.5.4.4 T Flip-Flop (Toggle Flip-Flop)

It can be designed using J-K flip-flop by connecting J and K inputs together to make a single input T as shown in fig. 7.35.

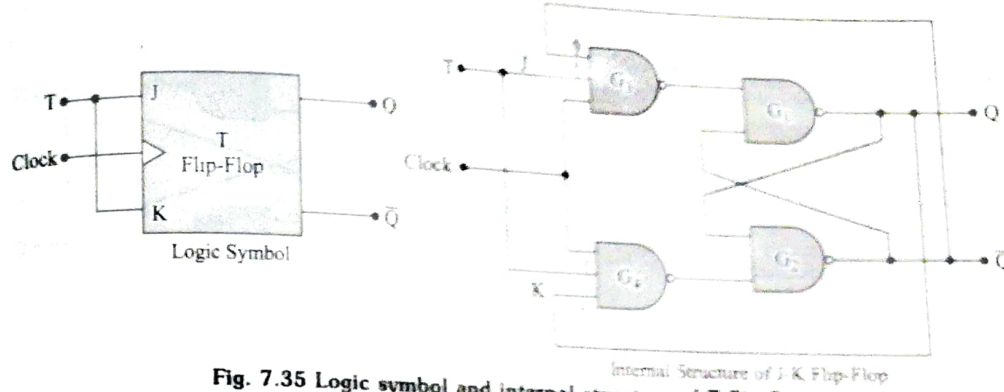


Fig. 7.35 Logic symbol and internal structure of T flip-flop

Its operation is similar to the inputs of J-K flip-flop with J = K = 0 (or) J = K = 1 as inputs.

Due to toggling property it is known as toggle flip-flop and are extensively used in counters to count the pulses. Truth table and timing waveforms are as shown in fig. 7.36.

Inputs		Output	State
Clk	T	Q	
↑	0	Q	No change
↑	1	$\bar{Q}$	Toggle
0	X	Q	No change

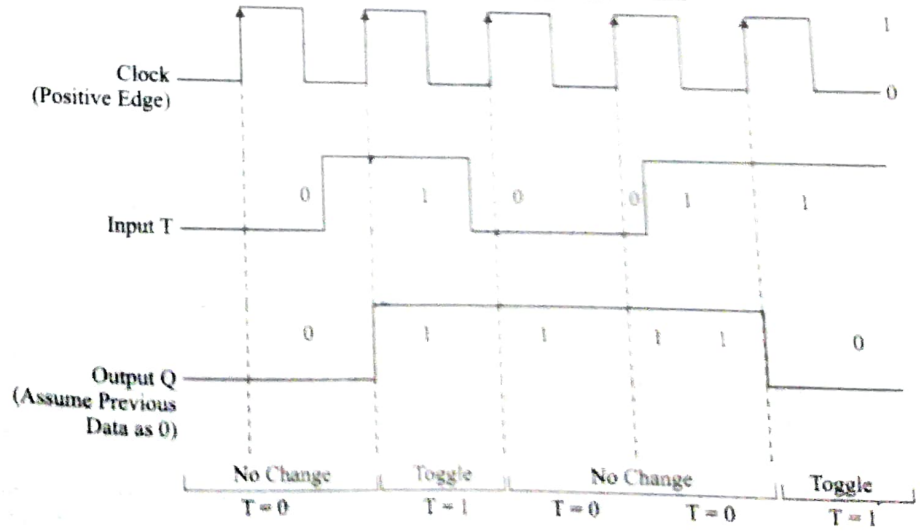


Fig. 7.36 Timing waveforms of T flip-flop at positive edge trigged clock.

7.5.4.5 Race Around Condition

Race around condition occurs in J-K latch, when J = K = 1 *i.e.*, when the J-K latch is in toggle state. Latches are controlled by enable (EN) signal and are level triggered. Figure 7.37 shows the J-K latch circuit with logic symbol.

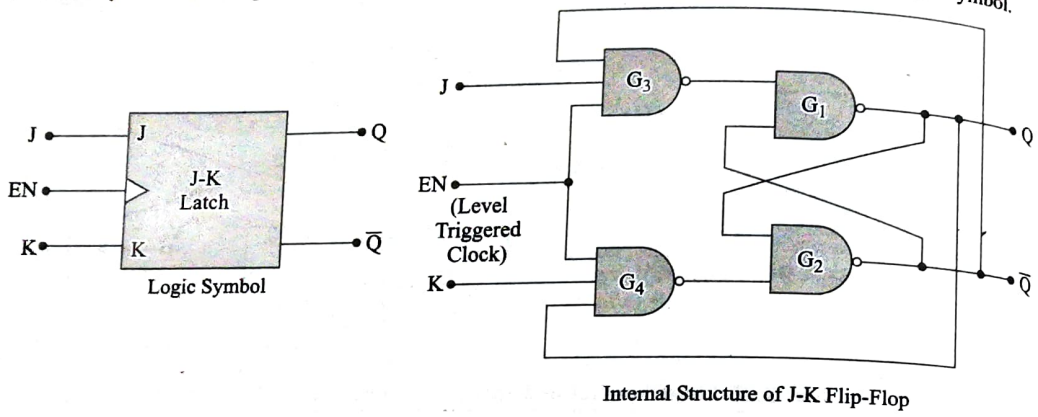


Fig. 7.37 Logic symbol and internal structure of J-K flip flop at level triggered clock.

This J-K latch circuit is not practical. The output is feedback to input and thus change in output results in input change. During positive half cycle of clock pulse, when J = K = 1, the output keep on changing from 0 to 1, then 1 to 0, then 0 to 1 and so on. This is called the toggle state. The output toggles continuously and at the end of pulse its state will be uncertain *i.e.*, whether '0' comes first or '1' comes first at the output we don't know. There is a race between '0' and '1'. This condition is called as race around condition. It is clear from the waveforms as shown in fig. 7.38.

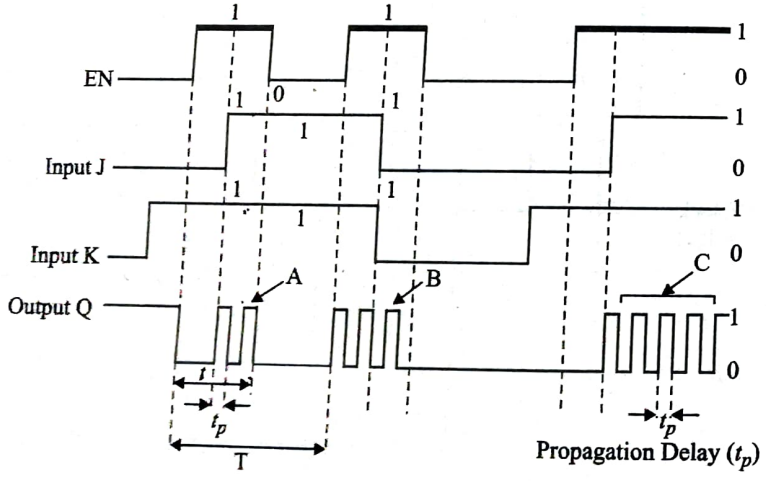


Fig. 7.38 Timing waveforms of J-K flip flop at level triggered clock.

Here, A, B and C are the points showing the continuous toggling of output when, inputs are J = 1 and K = 1 during the enable signal (EN) = 1. When EN = 0, there is no change condition thus, we get the previous output for Low enable signal. Also, the output transition occurs after some propagation delay time.

The race-around condition can be avoided if enable signal is reduced than propagation delay of flip-flop *i.e.*, $t < t_p < T$, as shown in the waveforms of fig. 7.38) but it is not practically feasible. Thus, master-slave flip-flop is used to overcome this problem.

7.5.4.6 Elimination of Race Around Condition

The elimination of race around condition in J-K latch can be done in two ways

1. Using the edge triggered or pulse triggered clocks in flip-flops.
2. Using the Master-Slave flip-flops.

7.5.4.6.1 Edge Triggered or Pulse Triggered Clocks in Flip-Flops

To avoid race around condition in the J-K latch edge triggered or pulse triggered clocks are used as shown fig. 7.39 (*a*), (*b*) and 7.40 (*a*), (*b*).

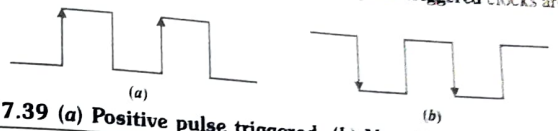


Fig. 7.39 (*a*) Positive pulse triggered. (*b*) Negative pulse triggered.

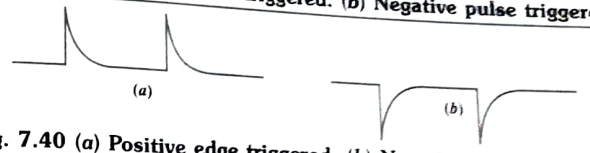


Fig. 7.40 (*a*) Positive edge triggered (*b*) Negative edge triggered.

Fig. 7.41 shows the waveforms for positive edge triggered J-K flip-flop.

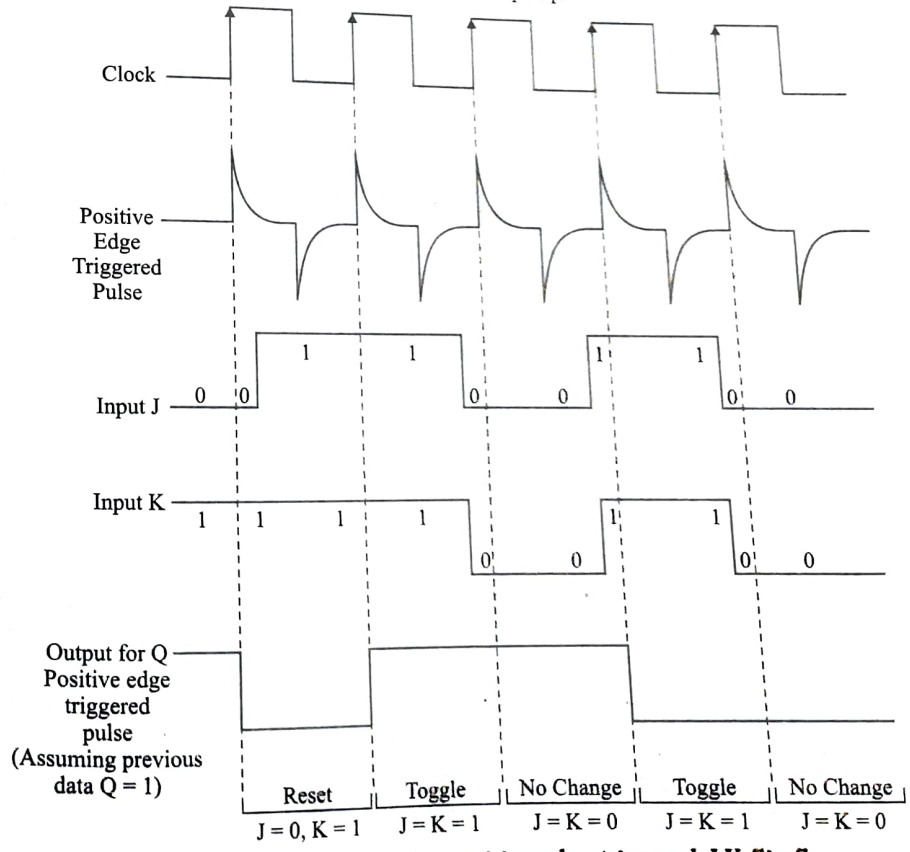


Fig. 7.41 Timing waveforms for positive edge triggered J-K flip-flop.

From waveforms of fig. 7.41 it is clear that by using positive edge triggered clock, the change in the output occurs only at the positive edge of clock signal and change will remain until next positive edge takes place. But in case of level triggered clocks, there is lot of time for the inputs to change during the whole level. Thus, multiple toggling occurs in level triggered clocks. In case of edge triggered clocks the time for inputs to change is at the edge only not at the whole level. Thus, edge triggered clocks will eliminate the concept of multiple toggling and hence, eliminate the race around condition.

Fig. 7.42 shows the clocked JK flip-flop using RC differentiator circuit.

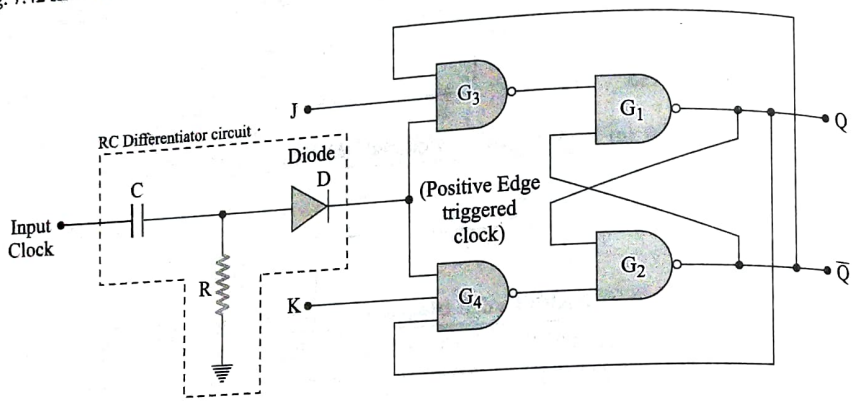


Fig. 7.42 The clock used may be edge triggered or pulse triggered.

7.5.4.6.2 Master Slave Flip-Flops

A Master Slave flip-flop is constructed from two flip-flops. One flip-flop acts as a master and the other acts as slave. Slave follows the output of master. The combination of master and slave is known as master slave flip-flop. It is used to convert the level triggered flip-flop to edge triggered flip-flop. Thus, used to eliminate the race around condition.

There are three basic types of master-slave flip-flops *i.e.*, S–R, D and J-K master-slave flip-flops. Mostly, J-K master slave flip-flop is used because of its easy availability in integrated circuit forms.

7.5.4.6.2.1 Master Slave S-R flip-flop

The functional block diagram for an S-R master slave flip-flop is as shown in fig. 7.43.

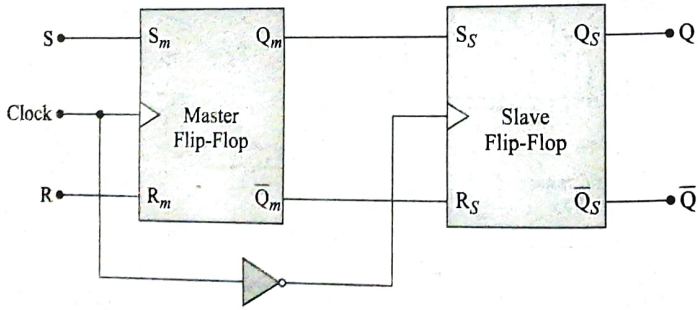


Fig. 7.43 Functional block diagram of S-R master slave flip-flop.

The internal structure or circuit diagram of an S-R Master-Slave flip-flop is as shown in fig. 7.44.

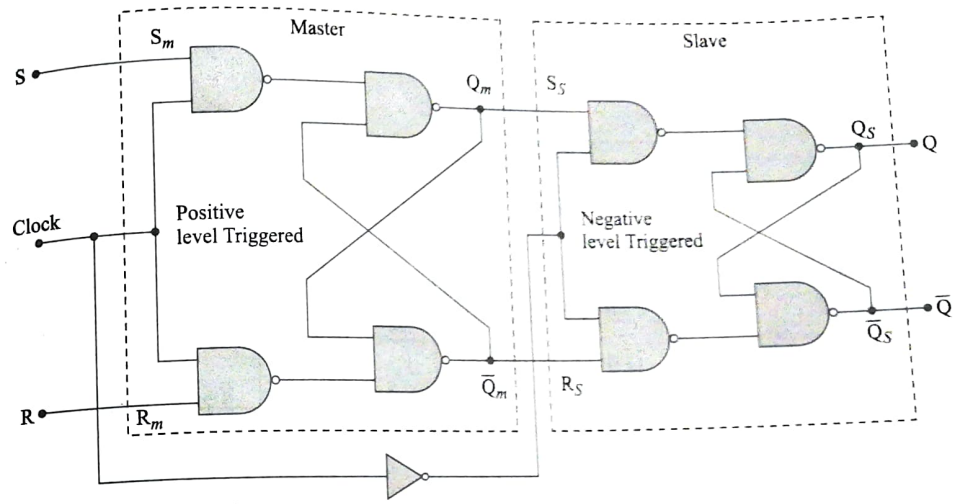


Fig. 7.44 Internal structure of S-R Master-Slave flip-flop.

It consist of one master flip-flop, one slave flip-flop and one inverter. In fig. 7.44 both the S-R flip flops are positive level triggered but the inverter used at the input acts as an input clock for slave flip-flop which forces it to trigger at the negative level.

The outputs of master Q_m and $\overline{Q}_m$ are dependent upon the inputs S and R to the master at positive clock pulse. Slave follows the output of master thus, master outputs Q_m and $\overline{Q}_m$ acts as inputs to slave S_s and R_s *i.e.*, $Q_m = S_s$ and $\overline{Q}_m = R_s$. But the slave uses negative clock pulse because of inverter to get the final outputs *i.e.*, Q and $\overline{Q}$. From fig. 7.44 it is clear that during half time period of clock pulse master will work and at the same time during negative time period *i.e.*, second half of clock pulse slave will work because both are interconnected. Fig. 7.45 timing waveforms will illustrate the operation of master slave S-R flip-flop.

From timing waveforms of fig. 7.45 during first time period of clock pulse when clock is positive, S = 1 and R = 0, th output of master sets to '1', at the same time during negative half cycle of clock master is isolated and slave is active whic follows the output of master, thus slave also sets to '1'.

Similarly, during second time period of clock pulse when clock is positive, S = 0 and R = 1, the output of master rese to '0', at the same time during negative half cycle of second clock master is isolated and slave is active which follows t output of master, thus slave also resets to '0'.

In the similar way other time periods of clock pulse gives master no change, slave no change, master sets, slave se master resets and slave resets as shown in fig. 7.45.

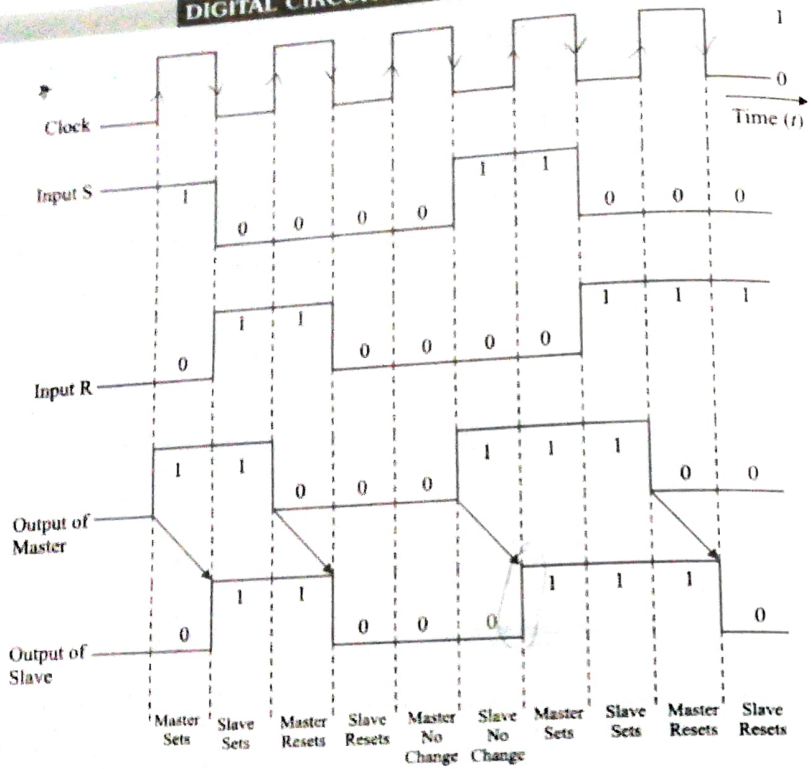


Fig. 7.45 Input and output timing waveforms for Master-Slave S-R flip-flop.

7.5.4.6.2.2 Master Slave J-K flip-flop

The functional block diagram for an J-K Master-Slave flip-flop is as shown in fig. 7.46.

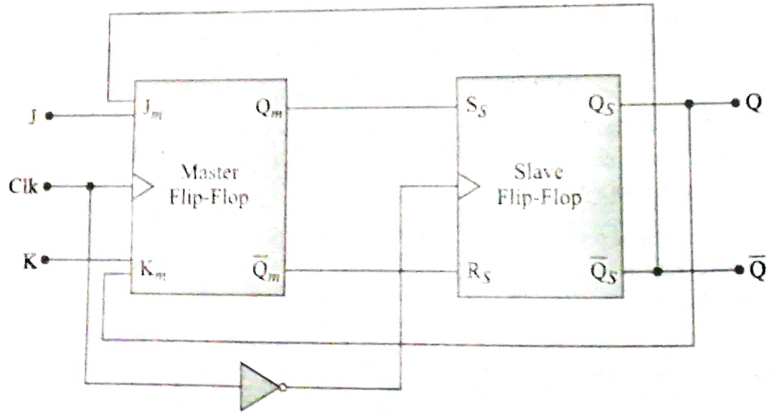


Fig. 7.46 Functional block diagram of J-K Master-Slave flip-flop.

The internal structure or circuit diagram of master slave J-K flip-flop is as shown in fig. 7.47.

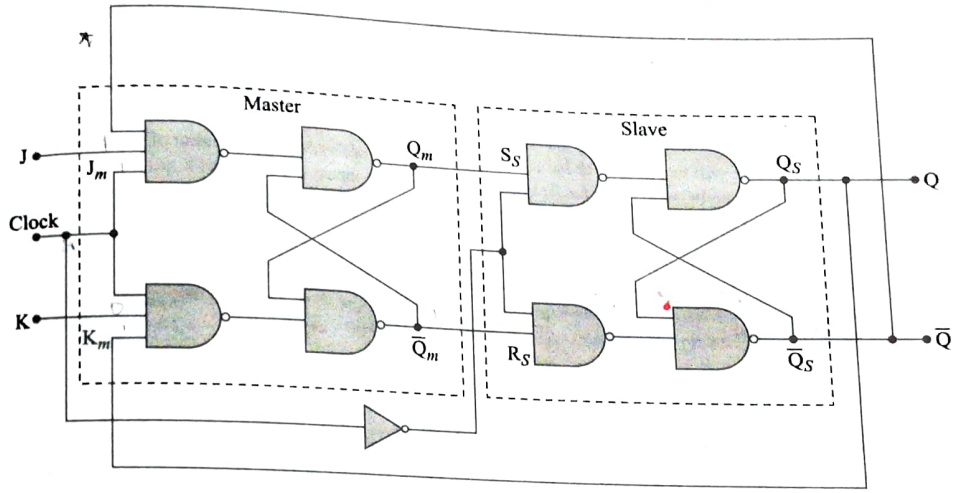


Fig. 7.47 Internal structure of J-K Master-Slave flip-flop.

It consists of one master J-K flip flop, one slave S-R flip-flop and one inverter. In fig. 7.47 clock signal is connected directly to the master J-K flip-flop, but it is connected through inverter to the slave S-R flip-flop. Thus, the master J-K flip-flop is triggered on the positive clock pulse but the inverter used at the input of slave S-R flip-flop forces it to trigger at the negative clock pulse. Also the output of slave S-R flip-flop is connected as a third input for master J-K flip-flop. Thus, overall circuit is called as master slave J-K flip-flop.

There are two conditions for clock either '1' or '0'. When clock is '1' *i.e.*, positive level, the master is active and slave is inactive or isolated. But when clock is '0' *i.e.*, negative level, the slave is active and master is inactive or isolated. It will eliminate the race around condition as for half time period master works and for next half time period slave works. There are four possible combinations for J-K master-slave flip-flop.

Case I. When J = 0, K = 0.

For clock = 1, the master J-K flip-flop works in active mode and the slave S-R flip-flop will be inactive or isolated. During J = K = 0, the output of master will not change, as there is no change condition when J = K = 0 in a J-K flip-flop. As output of J-K is followed by S-R thus, it also remains in No Change condition.

For clock = 0, the slave S-R flip-flop works in active mode and the master J-K flip-flop will be inactive or isolated. Thus, the condition remains No Change for J = K = 0.

Case II. When J = 0 and K = 1.

For clock = 1, the master J-K flip-flop works in active mode and the slave S-R flip-flop will be inactive or isolated. This will give output $Q_m = 0$ and $\overline{Q}_m = 1$, which are the input for slave *i.e.*, $S_s = 0$ and $R_s = 1$. This state is a Reset state. Thus, $S_s = 0$ and $R_s = 1$ when again fed to master, we have $J_m = 0$, $K_m = 1$ and $Q_m = 0$, $\overline{Q}_m = 1$. That means $S_s = 0$ and $R_s = 1$. Thus, $Q = 0$ and $\overline{Q} = 1$ will be the states for condition J = 0 and K = 1.

Case III. When J = 1 and K = 0.

For clock = 1, the master J-K flip-flop will work in active mode and the slave S-R flip flop will be inactive or isolated. This will give output $Q_m = 1$ and $\overline{Q}_m = 0$, which are the inputs for slave *i.e.*, $S_s = 1$ and $R_s = 0$. This state is Set state. Thus,

$S_s = 1$ and $R_s = 0$ when again fed to master, we have $J_m = 1$, $K_m = 0$ and $Q_m = 1$, $\overline{Q}_m = 0$. That means $S_s = 1$ and $R_s = 0$.

Thus, $Q = 1$ and $\overline{Q} = 0$ will be the states for condition J = 1 and K = 0.

Case IV. When J = 1 and K = 1.

For clock = 1, the master J-K flip-flop will work in active mode and the slave S-R flip flop will be inactive or isolated. This will give output of master in toggle form *i.e.*, previous data will invert. The state is Toggle state, thus S and R data is also inverted.

For clock = 0, the master will be inactive or isolated but slave will be active, again the output of slave toggles. Thus, the output inverts from previously stored data *i.e.*, it toggles whenever the inputs are J = 1 and K = 1.

Fig. 7.48 shows the timing waveforms for a master-slave J-K flip-flop (assuming any data for inputs J and K).

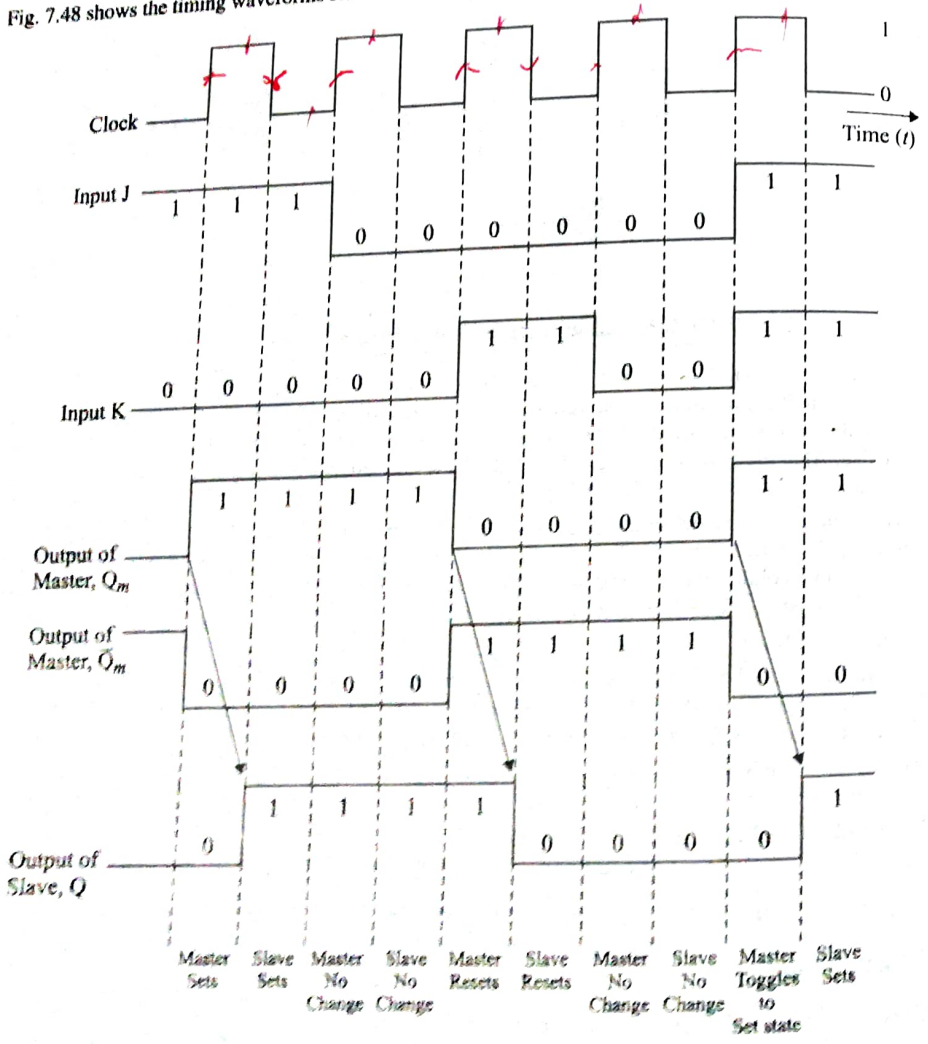


Fig. 7.48 Timing waveforms for Master-Slave J-K flip-flop.

Truth table of J-K master slave flip flop in as shown

Inputs			Output	State
Clk	J	K	Q	
⎍	0	0	Q	No change
⎍	0	1	0	Reset
⎍	1	0	1	Set
⎍	1	1	$\overline{Q}$	Toggle

7.5.4.7 Asynchronous or Direct Inputs or Preset and Clear Inputs

Synchronous inputs are the inputs which we have discussed in the previous articles of S-R, J-K, D and T flip-flops, because the data on these inputs are transferred to the output of flip flop's only when the clock is applied or only on the triggering of clock pulse *i.e.*, the data or information are transferred synchronously with the clock pulse.

Asynchronous inputs are the inputs which are used to Set the flip flop to '1' or preset the flip-flop to '1' and Reset the flip flop to '0' or clear the flip-flop to '0' at any time independent of the other inputs provided to the flip-flop. Thus, these inputs are also called as direct inputs. These inputs are connected directly to the latch circuit of flip-flop which is to be preset or clear at a time. It will override the effect of the synchronous inputs and clock pulse provided to the flip-flop.

Fig. 7.49 and 7.50 shows the logic symbols and internal structures of S-R and J-K flip-flop with preset and clear signals or inputs.

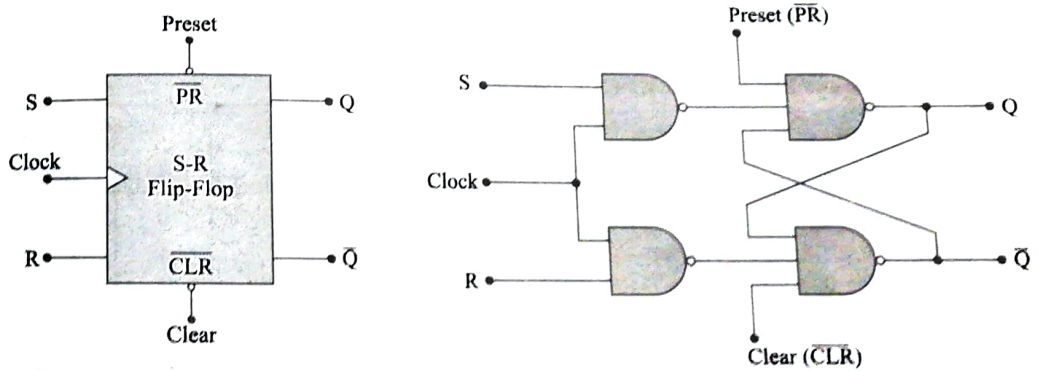


Fig. 7.49 Logic symbol and internal structure of S-R flip-flop with preset and clear inputs.

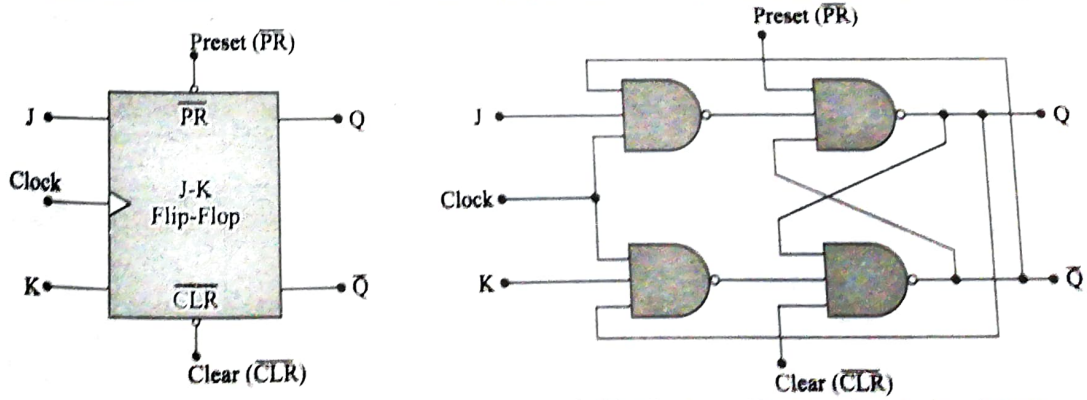


Fig. 7.50 Logic symbol and internal structure of J-K flip-flop with preset and clear inputs.

From fig. 7.49 and 7.50 it is clear that the preset ($\overline{PR}$) and ($\overline{CLR}$) inputs are connected to the latch circuit and are active low signals. Following table will clear the operation for preset and clear inputs used in flip-flops.

Inputs		Flip-Flop Response
Preset ($\overline{PR}$)	Clear ($\overline{CLR}$)	
0	0	Not used
0	1	Flip-Flop is set *i.e.*, Q = 1
1	0	Flip-Flop is reset *i.e.*, Q = 0
1	1	Flip-Flop works in normal clocked operation

Case I. When preset = 0 and clear = 0, the state at the output of latch circuit is invalid, thus it is not used.

Case II. When preset = 0 and clear = 1, the output Q will immediately set to '1' regardless of other inputs to flip-flop Thus, the clock input has no affect on the flip-flop when preset ($\overline{PR}$) = 0.

Case III. When preset = 1 and clear = 0, the output Q will immediately cleared to '0' or reset to '0' regardless of other inputs to flip-flop. Thus, the clock input has no affect on the flip-flop when clear ($\overline{CLR}$) = 0.

Case IV. When preset = 1 and clear = 1, the asynchronous inputs becomes inactive (*i.e.*, because of active-low signals) The flip-flops when used at these states will work as a normal clocked flip-flops or synchronous input clocked flip-flops.

7.5.5 Truth Table and Excitation Tables of all Flip-Flops

The truth tables and excitation tables of flip-flops SR, JK, D and T are as shown below. These truth tables are useful to specify the next states when present states are known.

7.5.5.1 Truth Table and Excitation Table for S-R Flip-Flop

Inputs		Output
S	R	Q_{n+1}
0	0	No change
0	1	0 (Reset)
1	0	1 (Set)
1	1	Indeterminate

(a) **Truth table of S-R flip-flop**

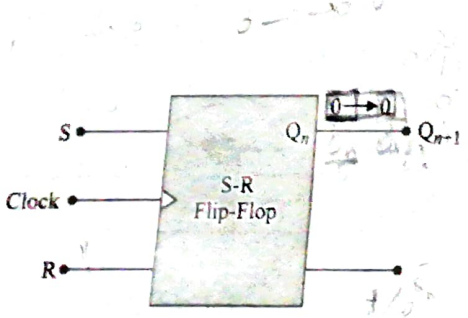
Present State	Next State	Inputs	
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

(b) **Excitation table of S-R flip-flop**

Explanation of excitation table of S-R flip-flop

Here, Q_n is the present state and Q_{n+1} is the next state. The excitation table is derived from truth table by considering the present and next states.

Case I. When present state $Q_n = 0$ and we want next state $Q_{n+1} = 0$. Then the values of S and R are given by

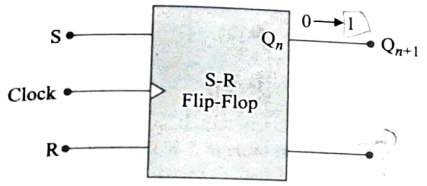


From truth table of S-R flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	S	R
0	0 (No change)	0	0
0	0 (Reset)	0	1

Thus, R = 0 (or) 1, but S is always 0 for present state $Q_n = 0$ and next state $Q_{n+1} = 0$. Hence, R = don't care *i.e.*, denoted by 'X' and S = 0.

Case II. When present state $Q_n = 0$ and we want next state $Q_{n+1} = 1$. Then the values of S and R are given by

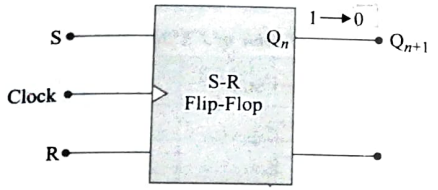


From truth table of S-R flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	S	R
0	1 (Set)	1	0

Thus, S = 1 and R = 0 is the condition for set state with present state $Q_n = 0$ and next-state $Q_{n+1} = 1$.

Case III. When present state $Q_n = 1$ and we want next state $Q_{n+1} = 0$. Then the values of S and R are given by

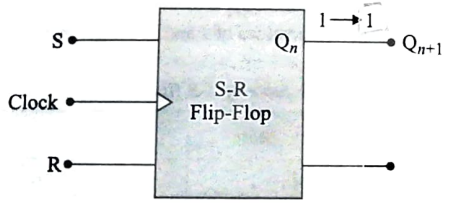


From truth table of S-R flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	S	R
1	0 (Reset)	0	1

Thus, S = 0 and R = 1 is the condition for reset state with present state $Q_n = 1$ and next state $Q_{n+1} = 0$.

Case IV. When present state $Q_n = 1$ and we want next state $Q_{n+1} = 1$. Then the values of S and R are given by



From truth table of S-R flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	S	R
1	1 (No change)	0	0
1	1 (Set)	1	0

Thus, R = 0, but S = 0 (or) 1 for present state $Q_n = 1$ and next state $Q_{n+1} = 1$. Hence, R = 0 and S = don't care *i.e.*, denoted by 'X'.

Thus, the excitation table for S-R flip-flop is given by

Case	Present State	Next State	Inputs	
	Q_n	Q_{n+1}	S	R
Case I	0	0	0	X
Case II	0	1	1	0
Case III	1	0	0	1
Case IV	1	1	X	0

7.5.5.2 Truth Table and Excitation Table for J-K Flip-Flop

Inputs		Output
J	K	Q_n
0	0	No change
0	1	0 (Reset)
1	0	1 (Set)
1	1	Toggle

(a) **Truth table of J-K flip-flop**

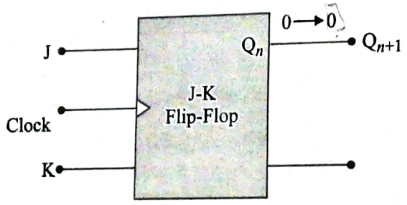
Present State	Next State	Inputs	
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(b) **Excitation table of J-K flip-flop**

Explanation of excitation table of J-K flip-flop.

Here, Q_n is the present state and Q_{n+1} is the next state. The excitation table is derived from truth table by considering the present and next states.

Case I. When present state $Q_n = 0$ and we want next state $Q_{n+1} = 0$. Then the values of J and K are given by

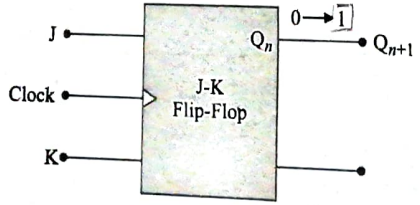


From truth table of J-K flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	J	K
0	0 (No change)	0	0
0	0 (Reset)	0	1

Thus, J = 0 and K = 0 (or) 1 *i.e.*, don't care 'X' for present state $Q_n = 0$ and next state $Q_{n+1} = 0$.

Case II. When present state $Q_n = 0$ and we want next state $Q_{n+1} = 1$. Then the values of J and K are given by

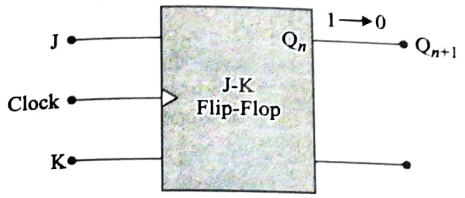


From truth table of J-K flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	J	K
0	1 (Set)	1	0
0	1 (Toggle)	1	1

Thus, J = 1 and K = 0 (or) 1 *i.e.*, don't care 'X' for present state $Q_n = 0$ and next state $Q_{n+1} = 1$.

Case III. When present state $Q_n = 1$ and we want next state $Q_{n+1} = 0$. Then the values of J and K are given by

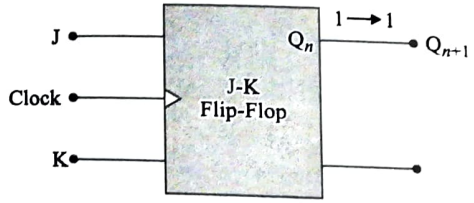


From truth table of J-K flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	J	K
1	0 (Reset)	0	1
1	0 (Toggle)	1	1

Thus, J = 0 (or) 1 *i.e.*, don't care 'X' and K = 1 for present state $Q_n = 1$ and next state $Q_{n+1} = 0$.

Case IV. When present state $Q_n = 1$ and we want next state $Q_{n+1} = 1$. Then the values of J and K are given by



From truth table of J-K flip-flop

Present State	Next State	Inputs	
Q_n	Q_{n+1}	J	K
1	1 (No change)	0	0
1	1 (Set)	1	0

Thus, J = don't care *i.e.*, 'X' and K = 0 for present state $Q_n = 1$ and next state $Q_{n+1} = 1$.

Thus, the excitation table for J-K flip-flop is given by

Case	Present State	Next State	Inputs	
	Q_n	Q_{n+1}	J	K
Case I	0	0	0	X
Case II	0	1	1	X
Case III	1	0	X	1
Case IV	1	1	X	0

7.5.5.3 Truth Table and Excitation Table for D Flip-Flop

Input	Output
D	Q_n
0	0
1	1

(a) **Truth table of D flip-flop**

Present state	Next state	Input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

(b) **Excitation table of D flip-flop**

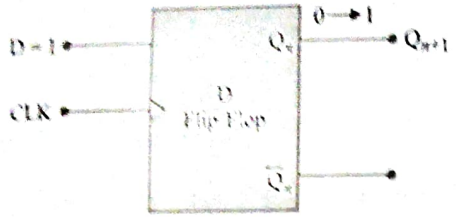
Explanation of excitation table of D flip-flop

Case I. When $Q_n = 0$, $Q_{n+1} = 0$, then D = 0 as this is a Reset state (Reset means '0'). Hence

$$D = 0 \left[\begin{array}{l} i.e.\ S = 0,\ R = 1 \\ or\ J = 0,\ K = 1 \end{array}\right]$$

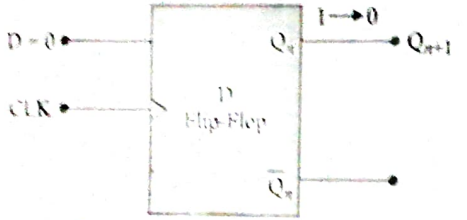
Case II. When $Q_n = 0$, $Q_{n+1} = 1$, then D = 1 as this is a Set state (Set means '1'). Hence

$$D = 1 \left[\begin{array}{l} i.e.\ S = 1,\ R = 0 \\ or\ J = 1,\ K = 0 \end{array}\right]$$



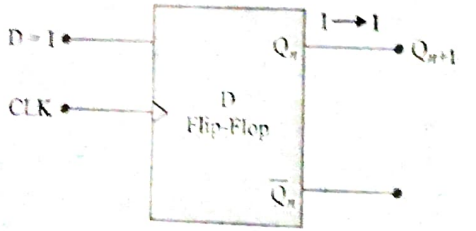
Case III. When $Q_n = 1$, $Q_{n+1} = 0$, then D = 0 as this is a Reset state (Reset means '0'). Hence

$$D = 0 \left[\begin{array}{l} i.e.\ S = 0,\ R = 1 \\ or\ J = 0,\ K = 1 \end{array}\right]$$



Case IV. When $Q_n = 1$, $Q_{n+1} = 1$, then D = 1 as this is a Set state (Set means '1'). Hence

$$D = 1 \left[\begin{array}{l} i.e.\ S = 1,\ R = 0 \\ or\ J = 1,\ K = 0 \end{array}\right]$$



The function table derived from state table is given by

D	Q_{n+1}
0	Q_n
1	Q_n

Where, Q_n is the present state and Q_{n+1} is the next state. Thus, it is concluded that in D flip-flop if '0' is input, then '0' will be stored and if '1' is input , then '1' will be stored.

Thus, the excitation table for D flip-flop is given by

Case	Present State	Next State	Input
	Q_n	Q_{n+1}	D
Case I	0	0	0
Case II	0	1	1
Case III	1	0	0
Case IV	1	1	1

7.5.5.4 Truth Table and Excitation Table for T Flip-Flop

Input	Output
T	Q_n
0	No change
1	Toggle

(a) Truth table of T flip-flop

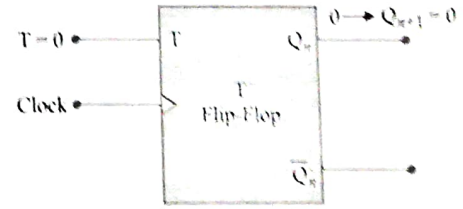
Present state	Next state	Input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) Excitation table of T flip-flop

Explanation of excitation table of T flip-flop

The excitation table is derived from truth table by considering the present and next states.

Case I. When $Q_n = Q_{n+1} = 0$, then T = 0 as this is Hold-on state. Thus, output remains '0' *i.e.* No change.

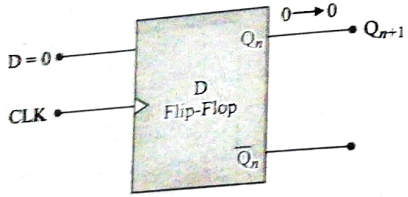


Case II. When $Q_n = 0$, $Q_{n+1} = 1$, then T = 1 as this is a Toggle state. Thus, output toggles from '0' to '1'.

Explanation of excitation table of D flip-flop

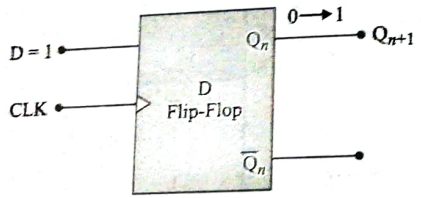
Case I. When $Q_n = 0$, $Q_{n+1} = 0$, then D = 0 as this is a Reset state (Reset means '0'). Hence

$$D = 0 \begin{bmatrix} i.e. S = 0, R = 1 \\ \text{or } J = 0,\ K = 1 \end{bmatrix}$$



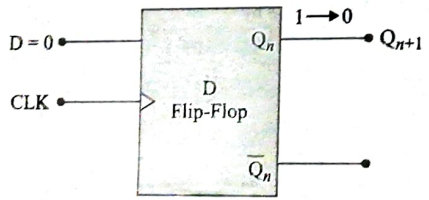
Case II. When $Q_n = 0$, $Q_{n+1} = 1$, then D = 1 as this is a Set state (Set means '1'). Hence

$$D = 1 \begin{bmatrix} i.e. S = 1, R = 0 \\ \text{or } J = 1,\ K = 0 \end{bmatrix}$$



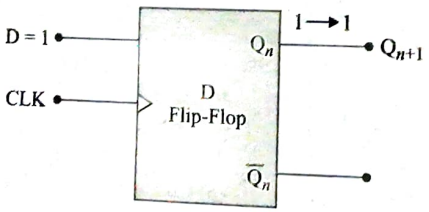
Case III. When $Q_n = 1$, $Q_{n+1} = 0$, then D = 0 as this is a Reset state (Reset means '0'). Hence

$$D = 0 \begin{bmatrix} i.e. S = 0, R = 1 \\ \text{or } J = 0,\ K = 1 \end{bmatrix}$$



Case IV. When $Q_n = 1$, $Q_{n+1} = 1$, then D = 1 as this is a Set state (Set means '1'). Hence

$$D = 1 \begin{bmatrix} i.e. S = 1, R = 0 \\ \text{or } J = 1,\ K = 0 \end{bmatrix}$$



The function table derived from state table is given by

D	Q_{n+1}
0	Q_n
1	Q_n

Where, Q_n is the present state and Q_{n+1} is the next state. Thus, it is concluded that in D flip-flop if '0' is input, then '0' will be stored and if '1' is input , then '1' will be stored.

Thus, the excitation table for D flip-flop is given by

Case	Present State	Next State	Input
	Q_n	Q_{n+1}	D
Case I	0	0	0
Case II	0	1	1
Case III	1	0	0
Case IV	1	1	1

7.5.5.4 Truth Table and Excitation Table for T Flip-Flop

Input	Output
T	Q_n
0	No change
1	Toggle

(a) **Truth table of T flip-flop**

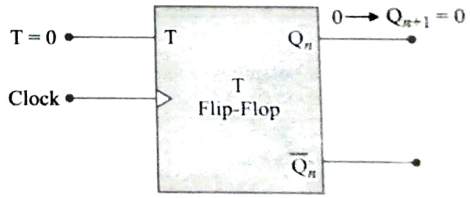
Present state	Next state	Input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) **Excitation table of T flip-flop**

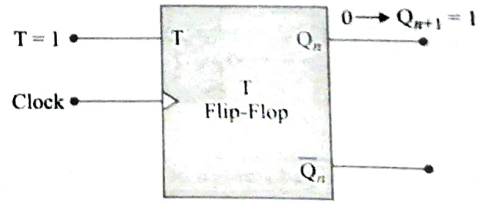
Explanation of excitation table of T flip-flop

The excitation table is derived from truth table by considering the present and next states.

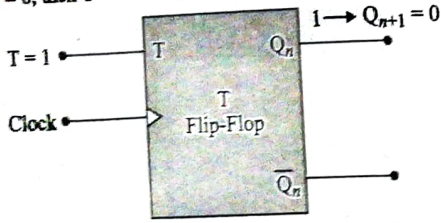
Case I. When $Q_n = Q_{n+1} = 0$, then T = 0 as this is Hold-on state. Thus, output remains '0' *i.e.* No change.



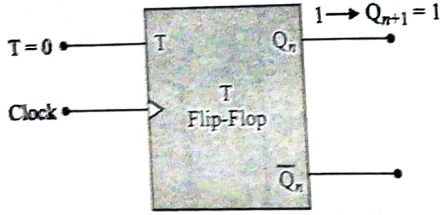
Case II. When $Q_n = 0$, $Q_{n+1} = 1$, then T = 1 as this is a Toggle state. Thus, output toggles from '0' to '1'.



Case III. When $Q_n = 1$, $Q_{n+1} = 0$, then T = 1 as this is again a Toggle state. Thus, output toggles from '1' to '0'.



Case IV. When $Q_n = 1$, $Q_{n+1} = 1$, then T = 0 as this is also a Hold state. Thus, output remains '1' *i.e.*, No change.



Hence, the function table from state table is given by

T_n	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$

Thus, the excitation table for T flip-flop is given by

Case	Present State	Next State	Input
	Q_n	Q_{n+1}	T
Case I	0	0	0
Case II	0	1	1
Case III	1	0	1
Case IV	1	1	0

7.5.6 Flip-Flop Conversions

Conversion of one flip-flop into another flip-flop is possible by adding some extra gates or connections. Block diagram for conversion of one flip-flop into another is as shown in fig. 7.51.

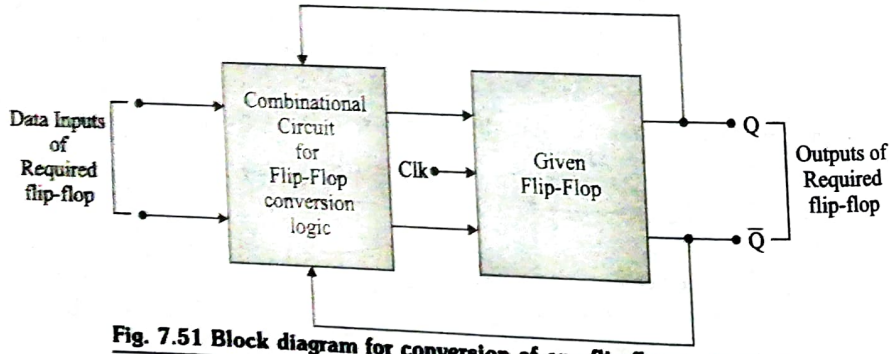


Fig. 7.51 Block diagram for conversion of one flip-flop to another.

Conversion of one flip-flop to another flip flop is done by following the given steps :

1. Write the excitation table as follows :

Inputs of Required flip flop	Present State (Q_n)	Next State (Q_{n+1})	Inputs of Given Flip-Flop

Present state (Q_n) is one time 0 and one time 1 for a particular input of required flip-flop. Then from the truth table of required flip-flop mark the next states (Q_{n+1}) for given present states (Q_n). At last from the standard excitation table of given flip-flop mark its inputs by taking present (Q_n) and next states (Q_{n+1}).

2. Put the values of inputs of given flip-flop from the excitation table in the K-map for minimization of the circuit. The variables for K-map are inputs of required flip-flop and the present state (Q_n).
3. Implement the circuit using the block diagram of fig. 7.51 and the minimized expressions obtained from the K-maps.

The various flip-flop conversions are as follows :

7.5.6.1 SR Flip-Flop to JK Flip-Flop

Here given flip flop is SR and the flip-flop to convert is JK *i.e.*, conversion of SR flip flop to JK flip-flop.

Step 1 : Write the excitation table

Inputs of Required Flip-Flop		Present state	Next state	Inputs of Given Flip-Flop	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	?	?	?
0	0	1	?	?	?
0	1	0	?	?	?
0	1	1	?	?	?
1	0	0	?	?	?
1	0	1	?	?	?
1	1	0	?	?	?
1	1	1	?	?	?

Truth table of J-K flip flop is as shown

Inputs		Output
J	K	Q
0	0	No Change
0	1	0, Reset
1	0	1, Set
1	1	Toggle

When J = K = 0 we get No change state, that means whatever in the present state (Q_n) will be in the next state (Q_{n+1}) without any change. Thus, the excitation table will become

Inputs of Required Flip Flop		Present State	Next State	Inputs of given Flip-Flop	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	?	?
0	0	1	1		

No Change

Similarly, when J = 0 and K = 1, we have Reset state *i.e.*, output is always '0'. Thus, we have

Inputs of Required Flip Flop		Present State	Next State	Inputs of given Flip-Flop	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	?	?
0	0	1	1		
0	1	0	0	?	?
0	1	1	0		

No Change

Reset State

Similarly, when J = 1, K = 0 and J = K = 1, we have Set state and Toggle state. For Set state the output is always '1' and for Toggle state the output *i.e.*, next state is Toggle of present state *i.e.*, if $Q_n = 0$, $Q_{n+1} = 1$ and if $Q_n = 1$, $Q_{n+1} = 0$. It is as shown in the excitation table.

Inputs of Required Flip Flop		Present State	Next State	Inputs of given Flip-Flop	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	?	?
0	0	1	1		
0	1	0	0	?	?
0	1	1	0		
1	0	0	1	?	?
1	0	1	1		
1	1	0	1	?	?
1	1	1	0		

Set State

Toggle

Toggle State

The standard excitation table is given by

Present State	Next State	Inputs	
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Now present state (Q_n) and next state (Q_{n+1}) are given, mark the inputs of given flip-flop from its standard excitation table.

The overall excitation table for SR to JK flip-flop coversion in designed with the help of excitation table of S-R flip flop as shown.

Present State	Next State	Inputs	
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Overall exitation table for SR to JK flip-flop conversion is as shown

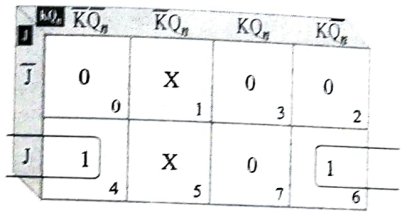
Inputs of Required Flip Flop		Present State	Next State	Inputs of given Flip-Flop	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

The complete excitation table is given by

Inputs		Present State	Next State	Flip-Flop	Inputs
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	0	1
1	1	1	0	0	1

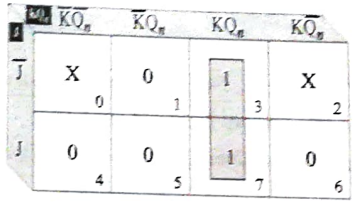
Step 2 : K-map Minimization using input variables as J, K and Q_n is as shown

For S →



$\therefore S = J\bar{Q}_n$

For R →



$\therefore R = KQ_n$

Step 3 : SR to JK implementation from K-map minimization is as shown in fig. 7.52.

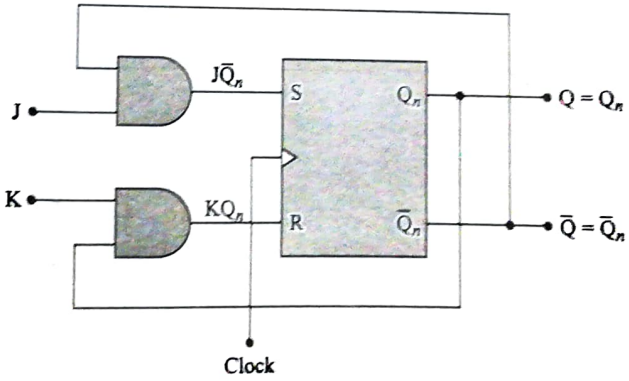


Fig. 7.52 SR to JK flip-flop

The internal structure of SR to JK flip-flop conversion is as shown in fig. 7.53.

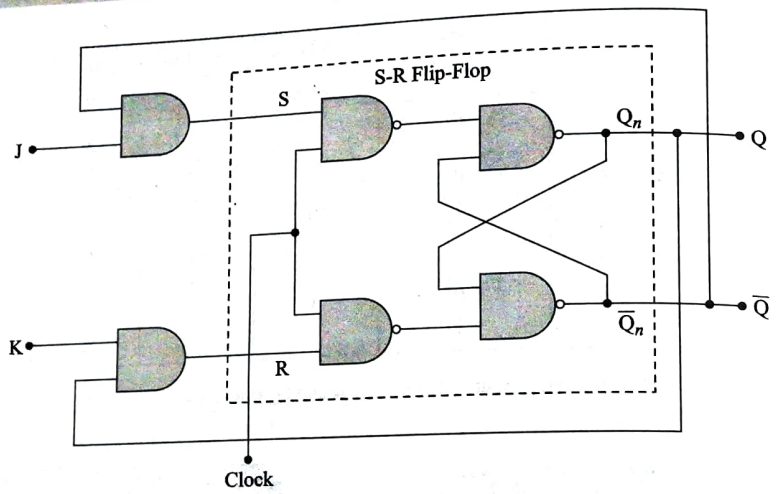


Fig. 7.53 Internal structure of SR to JK flip-flop conversion.

7.5.6.2 SR Flip-Flop to D Flip-Flop

Step 1. The excitation table of conversion for SR to D flip-flop is as shown

Input	Present State	Next State	Flip-Flop Inputs	
D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Step 2. K-map Minimization is as shown

For S →

D \ Q_n	$\overline{Q}_n$	Q_n
$\overline{D}$	0 (0)	0 (1)
D	1 (2)	X (3)

$\therefore S = D$

For R →

D \ Q_n	$\overline{Q}_n$	Q_n
$\overline{D}$	X (0)	1 (1)
D	0 (2)	0 (3)

$\therefore R = \overline{D}$

Note : Put 'X' = '1' as don't care condition.

Step 3. SR to D implementation is as shown in fig. 7.54.

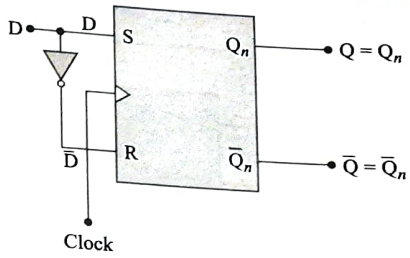


Fig. 7.54 SR to D flip-flop.

The internal structure of SR to D flip flop conversion is as shown in fig. 7.55.

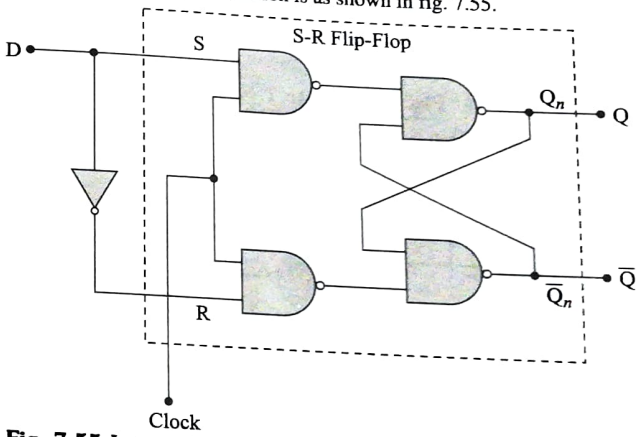


Fig. 7.55 Internal structure of SR to D flip-flop conversion.

7.5.6.3 SR Flip-Flop to T Flip-Flop

Step 1. The excitation table of conversion for SR to T flip-flop is as shown

Input	Present State	Next State	Flip-Flop Inputs	
T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

Step 2. K-map Minimization

For S →

T \ Q_n	$\overline{Q}_n$	Q_n
$\overline{T}$	0 (0)	X (1)
T	1 (2)	0 (3)

$\therefore S = T\overline{Q}_n$

For R →

T \ Q_n	$\overline{Q}_n$	Q_n
$\overline{T}$	X (0)	0 (1)
T	0 (2)	1 (3)

$\therefore R = TQ_n$

Step 3. SR to T implementation is as shown in fig. 7.56.

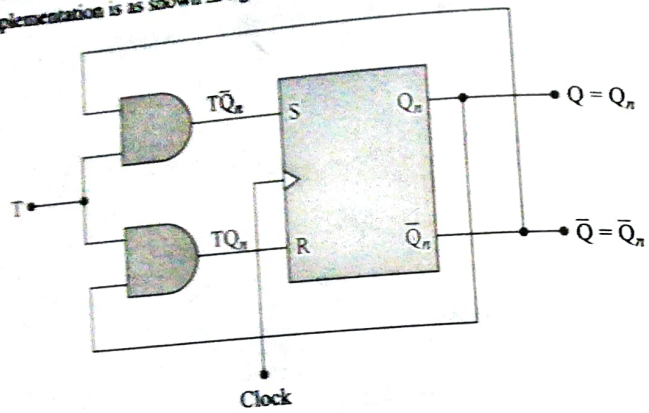


Fig. 7.56 SR to T Flip-flop.

The internal structure of SR to T flip flop is as shown in fig. 7.57.

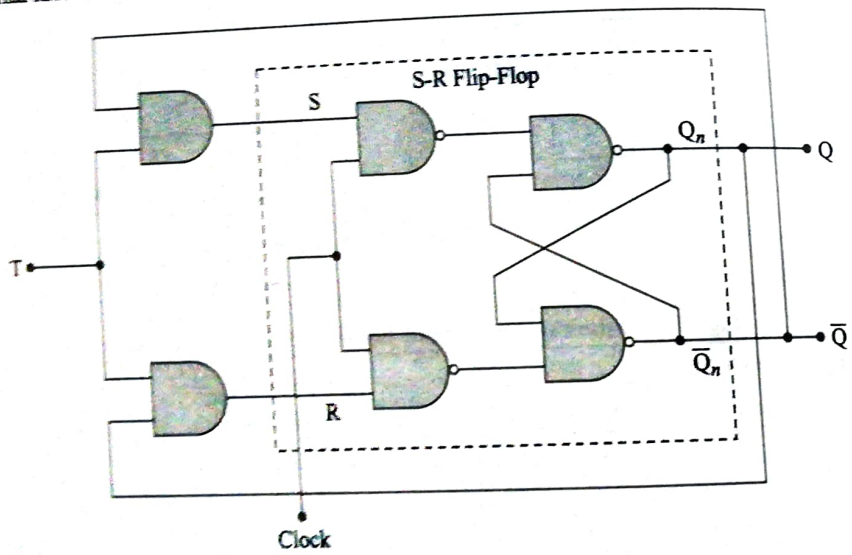


Fig. 7.57 Internal structure of SR to T flip-flop conversion.

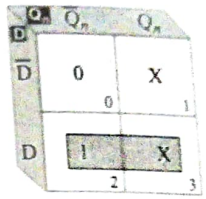
7.5.6.4 JK Flip-Flop to D Flip-Flop

Step 1. Write the excitation table for JK to D flip-flop, we have

Input	Present State	Next State	Flip-Flop Inputs	
D	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

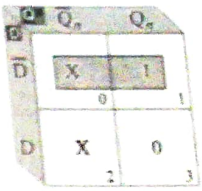
Step 2. K-map Minimization is as shown

For J →



$\therefore J = D$

For K →



$\therefore K = \bar{D}$

Step 3. JK to D implementation is as shown in fig. 7.58.

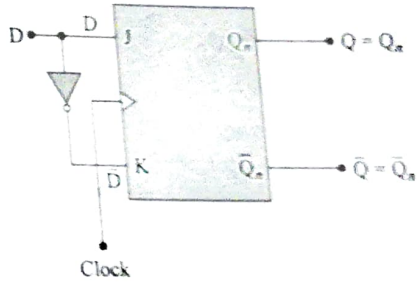


Fig. 7.58 JK to D flip-flop.

The internal structure of JK to D flip-flop conversion is as shown in fig. 7.59.

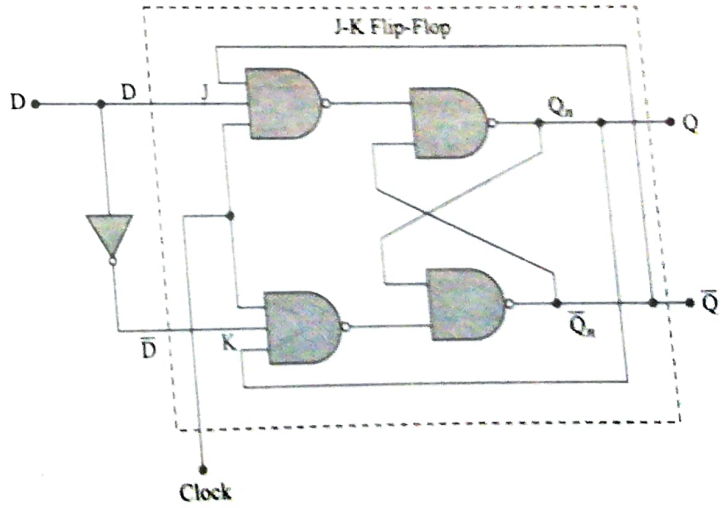


Fig. 7.59 Internal structure of JK to D flip-flop conversion.

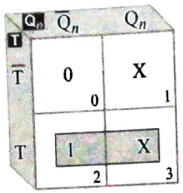
7.5.6.5 JK Flip-Flop to T Flip-Flop

Step 1. Write the excitation table for JK to T flip flop, we have

Input	Present State	Next State	Flip-Flop Inputs	
T	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

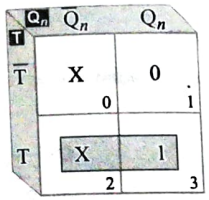
Step 2. K-map Minimization is as shown

For J →



∴ J = T

For K →



∴ K = T

Step 3. JK to T implementation is as shown in fig. 7.60.

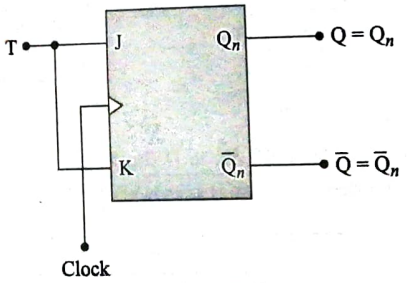


Fig. 7.60 JK to T flip-flop.

The internal structure of JK to T flip flop conversion is as shown in fig. 7.61.

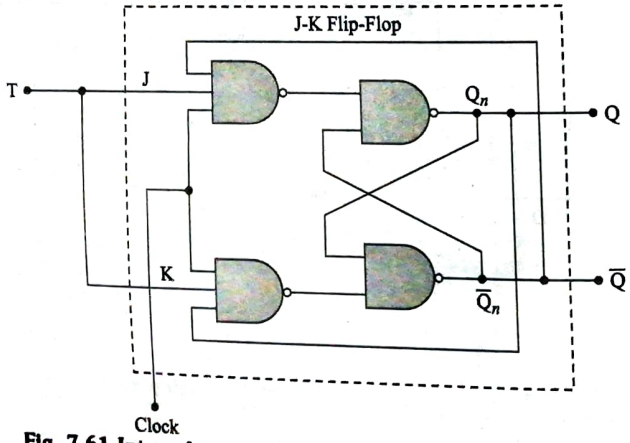


Fig. 7.61 Internal structure of JK to T flip-flop conversion.

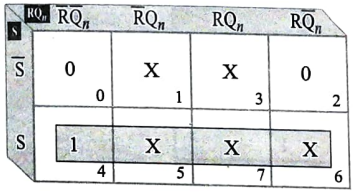
7.5.6.6 JK Flip-Flop to SR Flip-Flop

Step 1. Write the excitation table

Inputs		Present State	Next State	Flip-Flop Inputs	
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	0	Invalid	X	X
1	1	1	Invalid	X	X

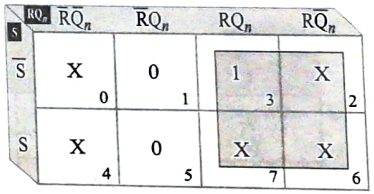
Step 2. K-map Minimization is as shown

For J →



∴ J = S

For K →



∴ K = R

Step 3. JK to SR implementation is as shown in fig. 7.62.

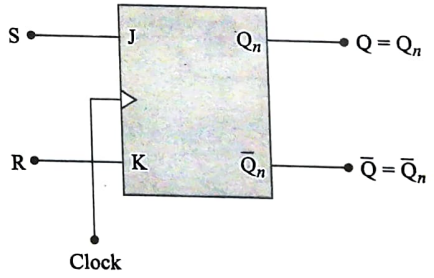


Fig. 7.62 JK to SR flip-flop.

The internal structure of JK to SR flip-flop conversion is as shown in fig. 7.63.

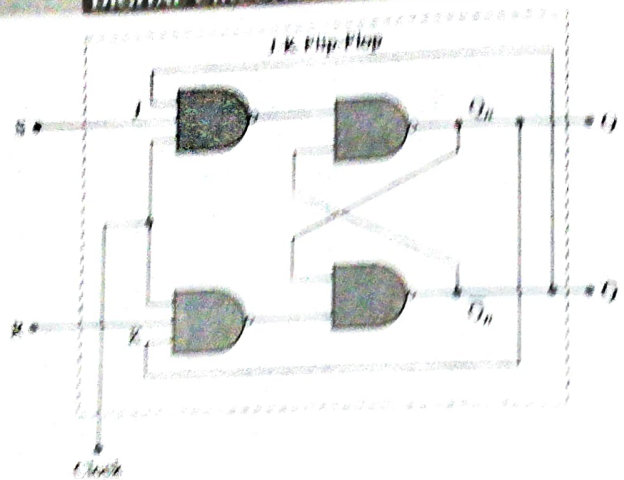


Fig. 7.4(f) Internal structure of JK to SR flip-flop conversion

7.5.4.7 D Flip Flop to T Flip Flop

Step 1: Write the excitation table for D to T flip flop, as shown

Input	Present State	Next State	Flip Flop Input
T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Step 2: K-map simplification is as shown

$$\therefore D = T\bar{Q}_n + \bar{T}Q_n$$

$$= T \oplus Q_n$$

Step 3: D to T implementation is as shown in Fig. 7.4(g)

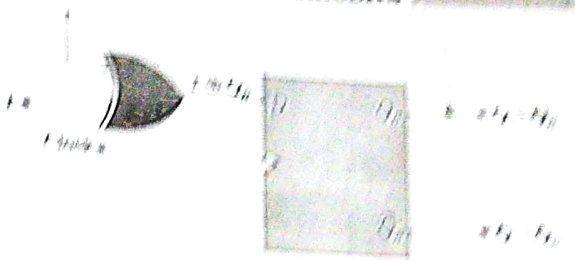


Fig. 7.4(g) D to T flip flop

The internal structure of D to T flip flop connection is as shown in Fig. 7.4(h)

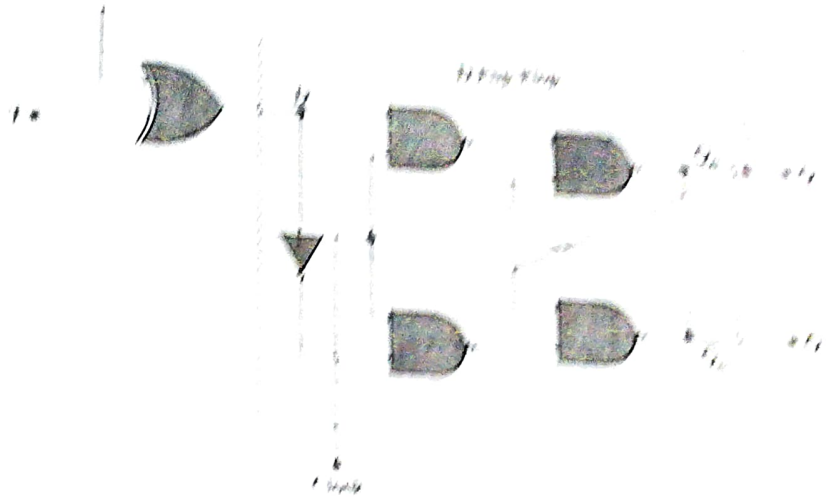


Fig. 7.4(h) Internal structure of D to T flip flop conversion

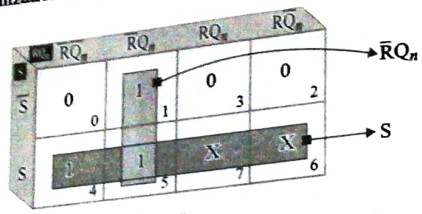
7.5.4.8 JK Flip Flop to SR Flip Flop

Step 1: Write the excitation table

Inputs		Present State	Next State	Flip Flop Input
S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	Invalid	X
1	1	1	Invalid	X

Step 2. K-map Minimization is as shown

For D →



$\therefore D = \bar{R}Q_n + S$

Step 3. D to SR implementation is as shown in fig. 7.66.

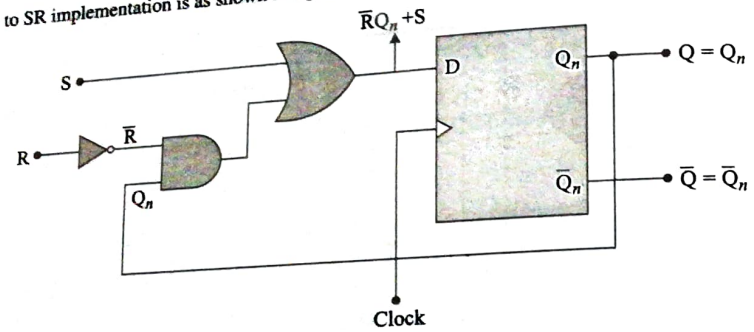


Fig. 7.66 D to SR flip-flop.

The internal structure of D to SR flip-flop conversion is as shown in fig. 7.67.

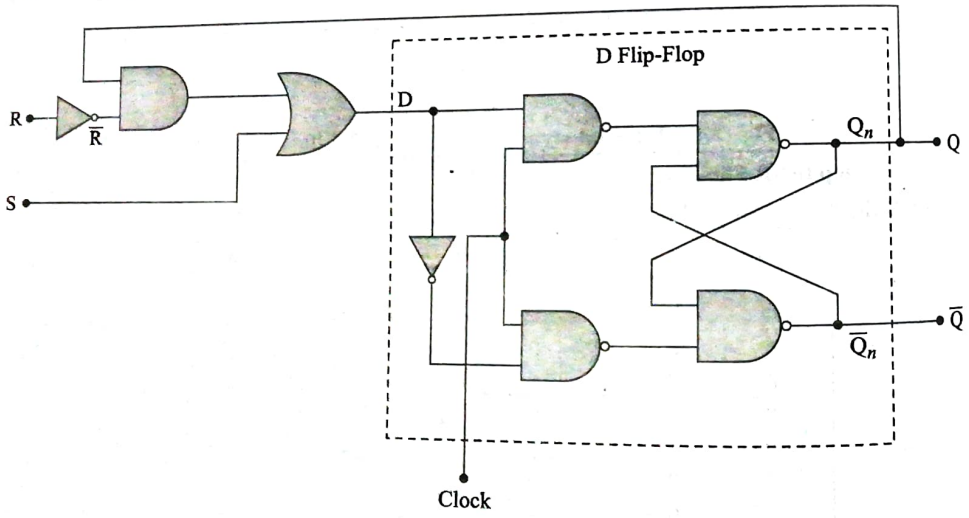


Fig. 7.67 Internal structure of D to SR flip-flop conversion.

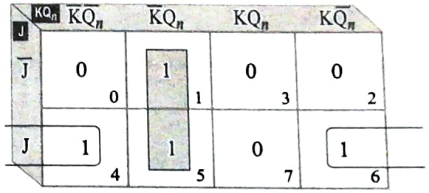
7.5.6.9 D Flip-Flop to JK Flip-Flop

Step 1. Write the excitation table for D to JK flip flop, we have

Inputs		Present State	Next State	Flip-Flop Input
J	K	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

Step 2. K-map Minimization is as shown

For D →



$\therefore D = J\bar{Q}_n + \bar{K}Q_n$

Step 3. D to JK implementation is as shown in fig. 7.68.

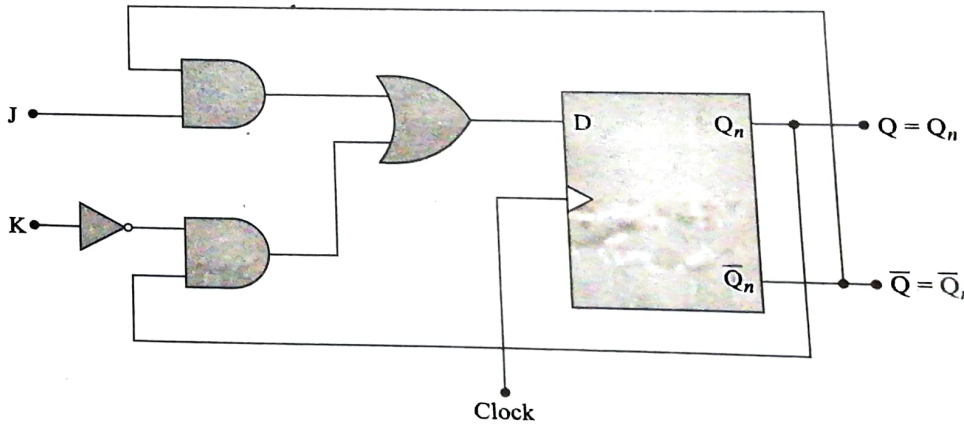


Fig. 7.68 D to JK flip-flop.

The internal structure of D to JK flip-flop conversion is as shown in fig. 7.69.

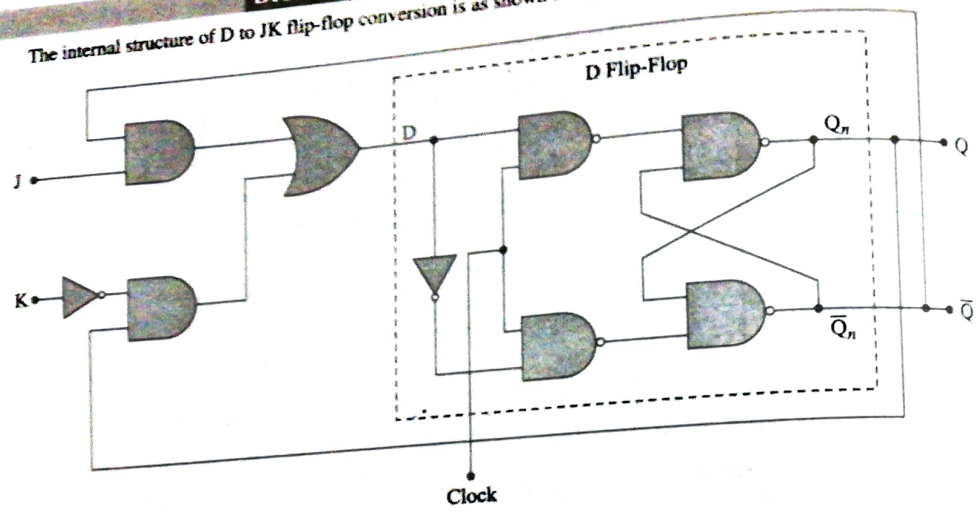


Fig. 7.69 Internal structure of D to JK flip-flop conversion.

7.5.6.10 T Flip-Flop to D Flip-Flop

Step 1. Write the excitation table of T to D flip-flop, we have

Input	Present State	Next State	Flip-Flop Input
D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Step 2. K-map Minimization is as shown

For T →

$$\therefore T = \overline{D}Q_n + D\overline{Q}_n$$

$$= D \oplus Q_n$$

Step 3. T to D implementation is as shown in fig. 7.70

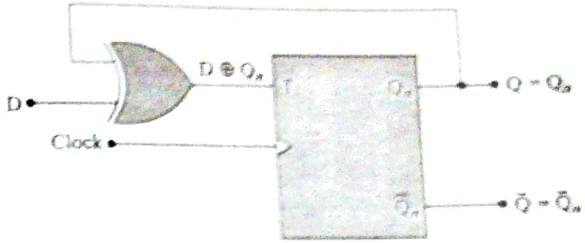


Fig. 7.70 T to D flip-flop

The internal structure of T to D flip-flop conversion is as shown in fig. 7.71

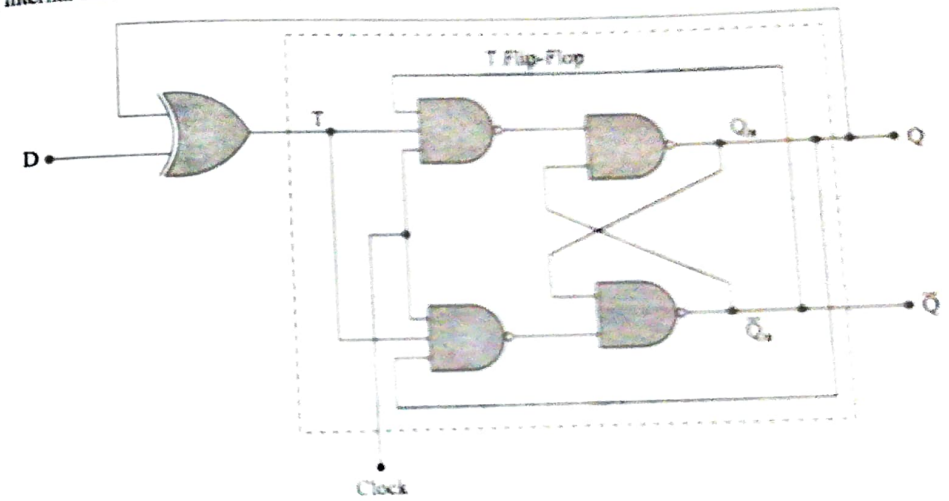


Fig. 7.71 Internal structure of T to D flip-flop conversion.

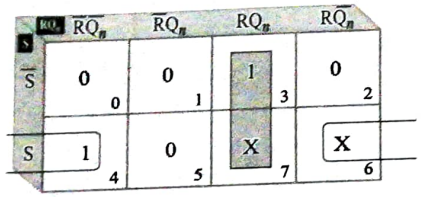
7.5.6.11 T Flip-Flop to SR Flip-Flop

Step 1. Write the excitation table for T to SR flip-flop conversion, we get

Inputs		Present state	Next state	Flip-Flop Input
S	R	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	Invalid	X
1	1	1	Invalid	X

Step 2. K-map Minimization is as shown

For T →



$\therefore\ T = S\overline{Q}_n + RQ_n$

Step 3. T to SR implementation is as shown in fig. 7.72.

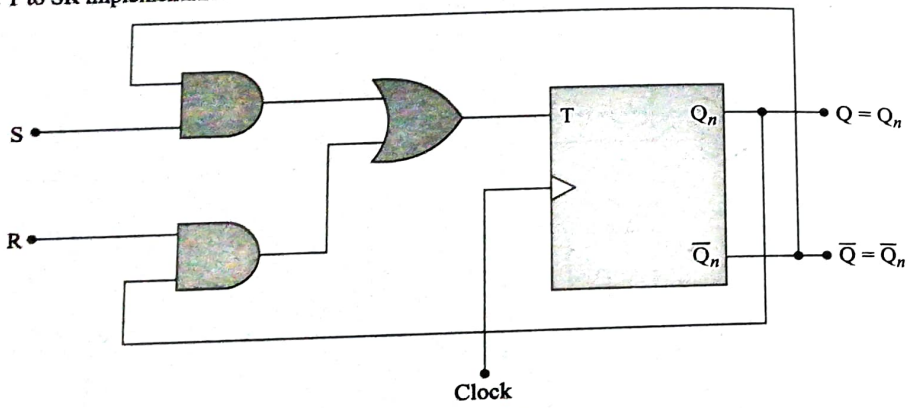


Fig. 7.72 T to SR flip-flop.

The internal structure of T to SR flip-flop conversion is as shown in fig. 7.73.

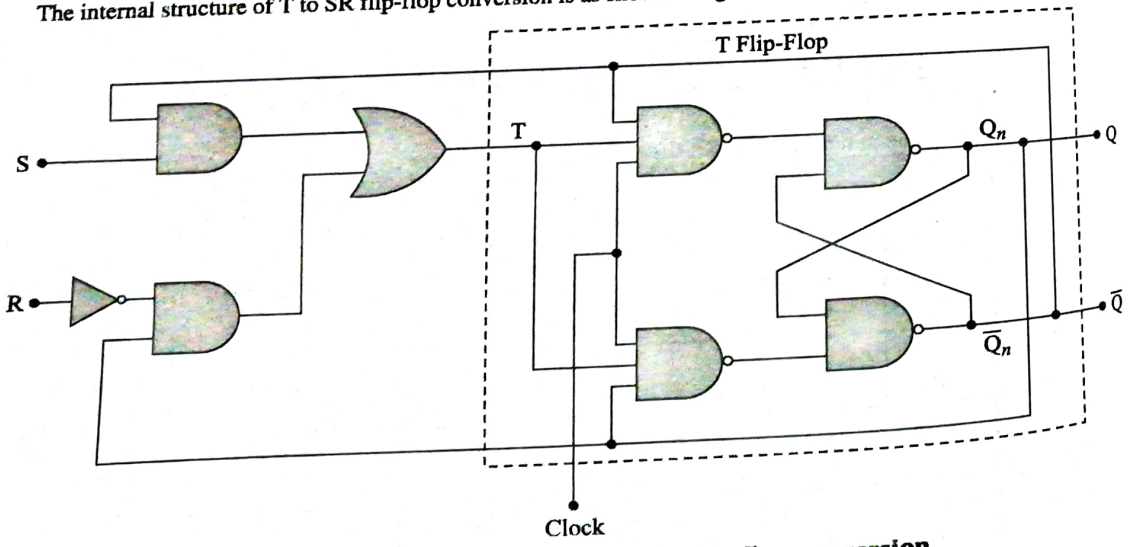


Fig. 7.73 Internal structure of T to SR flip-flop conversion.

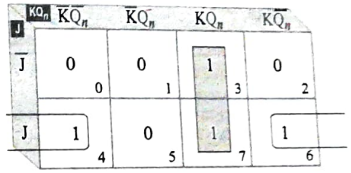
7.5.6.12 T Flip-Flop to JK Flip-Flop

Step 1. Write the excitation table for T to JK flip-flop, we have

Inputs		Present state	Next state	Flip-Flop Input
J	K	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

Step 2. K-map Minimization

For T →



$\therefore\ T = J\overline{Q}_n + KQ_n$

Step 3. T to JK implementation is as shown in fig. 7.74.

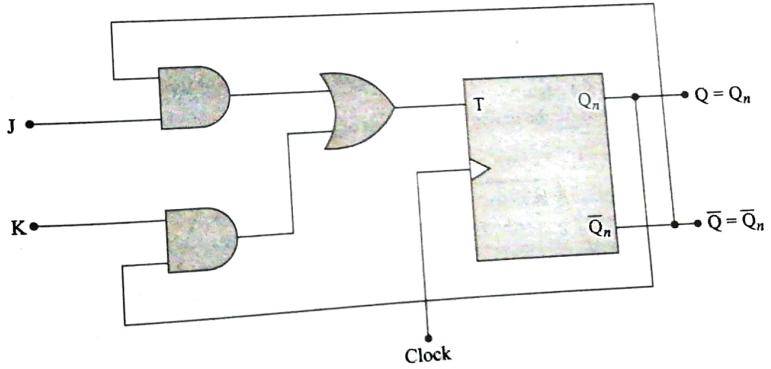


Fig. 7.74 T to JK flip-flop.

The internal structure for T to JK flip flop conversion is as shown in fig. 7.75.

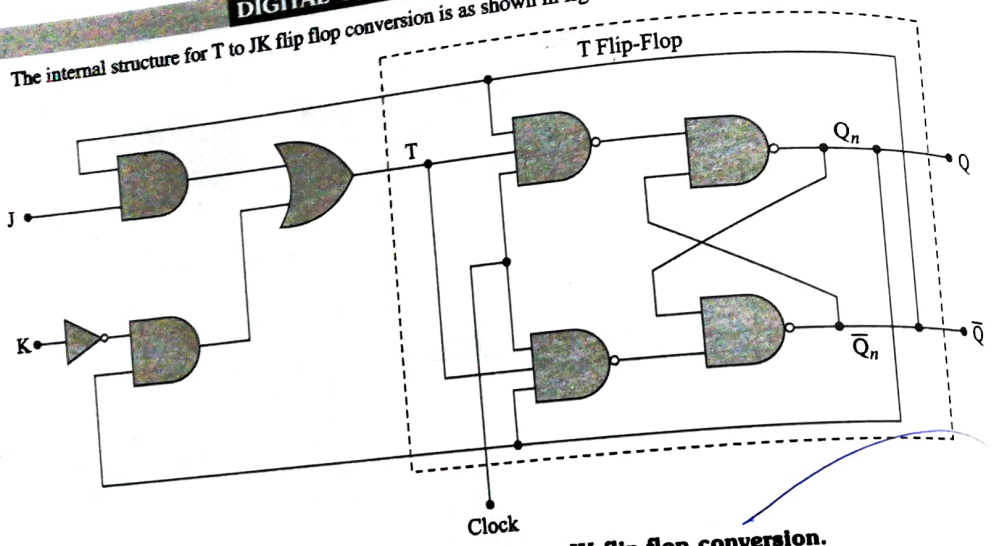


Fig. 7.75 Internal structure for T to JK flip-flop conversion.

7.5.7 Applications of Flip-Flops

There are large number of applications of flip-flops. Few important applications or uses of flip flops are :

1. These are used for the storage of data or information *i.e.*, they are used as a memory element. Parallel and serial data storage is done by using flip-flops.
2. These are used in registers where transfer of data takes place from one flip-flop to another.
3. These are used in counters and timers.
4. These are used in frequency division and delay element.
5. These are used in the elimination of keyboard debounce.

7.5.8 Flip-Flop Timings

The transfer of data from the flip-flop inputs to the output of that flip-flop takes a small interval of time in case of edge triggered flip-flop. It is during the transition from 1 to 0 (↓) state of a clock signal or from 0 to 1 (↑) state and in this the data lock out occurs at the end of the edge.

Fig. 7.76 shows the waveforms for the timing specifications of an edge triggered flip-flop

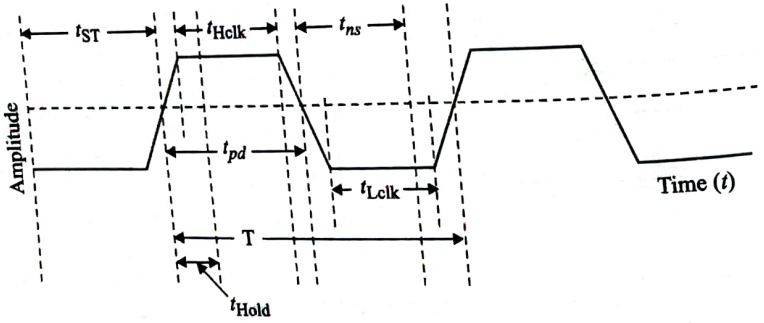


Fig. 7.76 Timing specifications of flip-flops in a clock signal.

t_{ST} = Setup time
t_{Hold} = Hold time
t_{pd} = Propagation delay of flip-flop in nsec.
t_{ns} = Propagation delay of next state decoder in nsec.
t_{Hclk} = High clock pulse width
t_{Lclk} = Low clock pulse width.

Important definitions for flip-flop timings or timing specifications of flip-flops are

(1) Setup Time : It is the minimum time interval in which the data bit must be present immediately before the active transition of the clock signal and during this time the data input must be maintained at the proper level.

An undesired output is obtained if $T < t_{ST}$ *i.e.*, if time interval is less than the setup time..

(2) Hold Time : It is the time interval immediately after the active transition of the clock signal and during this time the data input must be maintained at the proper level.

An undesired output is obtained if $T < t_{Hold}$.

(3) Propagation Delay : When clock signal is applied, the desired output is obtained after some time. This time is called as propagation delay of flip-flop *i.e.*, t_{pd}.

(4) Maximum Frequency : This is the maximum frequency of the clock signal for proper and stable operation of the flip-flop.

Mathematically f_{max} is given by $$f_{max} = \frac{1}{T} \leq \frac{1}{t_{ns} + t_{pd} + t_{ST}}$$

(5) High Clock Pulse Width (t_{Hclk}) : It is the minimum time period for which the clock must remain HIGH.

(6) Low Clock Pulse Width (t_{Lclk}) : It is the minimum time period for which the clock must remain LOW.

GLOSSARY

Active All-low **:** Active low means the signal must be low to activate any chip.

Astable Multivibrator **:** It is a digital circuit that oscillates between two unstable output states.

Bistable Multivibrator **:** It is a name used to describe a flip-flop.

Clear **:** It is an input to flip-flop at its clear pin or reset pin to clear the flip-flop at a time to 0 *i.e.*, Q = 0.

Clock **:** These are the digital signals in the form of rectangular pulse train or square wave.

Clocked flip-flops **:** Flip-flops that are allowed to respond to its excitation inputs only during the application clock pulse (or) flip-flops which have a clock input.

D flip-flop **:** It is a type of flip-flop in which output follows the input when triggered.

Edge-triggered **:** It is the triggering of flip-flop state which changes on the rising or falling edge of the clock pu *i.e.*, positive or negative edge triggering.

Enable **:** It is an input signal which allow circuit to perform its normal operation.

Flip-Flop **:** It is a bistable sequential device used to store single bit of information or data at a time *i.e.*, eithe or '0' at a time.

Latch **:** It is a sequential device that checks all of its inputs continuously and changes its outputs accordingly time independent of clock signal. Enable signal is used in latches.

Master-Slave flip-flop **:** A cascade of two flip-flops. One acts as master that trigger at High clock and anoth as slave follows the master, when clock goes Low.

Propagation delay **:** When input is applied we get the desired output after some time. It is due to the rea propagation delay of flip-flops. Each gate has its own propagation delay.

Reset **:** It is an asynchronous input used to give Q = 0 immediately.

Race around condition **:** It occurs in J-K flip-flop, when the clock is level triggered or enable at J = Multiple toggling takes place during the half time period of clock and at last output is uncertain.

Timing diagram **:** Depiction of logic levels as related to time.

Trigger **:** Input signal to a flip-flop which cause change of state is called trigger.