5. Sequential Circuits

Integrated circuits (ICs)

In the electronic circuits we have studied so for, separate resistors and transistors have been soldered and mechanically connected to form the circuits. Such circutes are generally known as discrete circuits. During the years following 1960, new techniques were developed for the construction of integrated circuits (ICs).

An integrated circiut (IC) is one inwhich the whole electronic system (such as amplifier) is produced on one small piece of silicon. (Monolithic - one stone). An IC is thus a device that includes its own resistors, transistors and diodes, fabricated in one chip.

The final form of the electronic system in the single chip of silicon can carry out the complete function (of amplification) of the system. Since the components are microscopically small, a manufacturer can place hundreds of these compontents in the space of a single discrete tranisistor. The advantage if ICs over discrete circuits are the following :

i) Large saving of space. The resistors and transistors necessary for each digital circuit are formed on each chip by a series of processing steps. In this way, identical digital circuits are manufactured simultaneousely on the same silicon wafer. A wafer is divided into thousands of chips.

ii) Low cost of production, since large number of identical devices can be fabricated simultaneously.

iii) Increaingly reliable since all the inter connections are made in the initial manufacturing process.

iv) Frequency response is extremly good, since there are no lengthly wire connections.

An integrated circuit (IC) fabricated entirely on a single silicon chip is called a monolithic IC. There is another type of IC known as hybrid IC which contains one or more monolithic circuits, interconnected with external resistors and capacitors using thin-film or thick-film techniques.

The monolithic ICs have power limitations. They typically have a maximum power of less than 1 W. When higher power is needed one can use hybird ICs.

IC 7400 devices

These devices make use TTL logic circuits. The 74 XX operate over a temperature range of 0°C to 75°C. The XX portion of the IC number refers to a specific device.

XX = 04 stands for inverter. 7404 is a TTL inverter. Some standared TTL digital circuits are given below:

7400 → quad (four) 2-input NAND gate

7402 → quad 2-input NOR gate

7404 → hex (six) inverter

7408 → quad 2-input AND gate

7432 → quad 2-input OR gate

Logic Gate Flip-Flops

A Flip-flop is a bistable circuit that has two stable states. The output of the flip-flop is either a 'low' or 'high' voltage i.e., a 0 or 1 state. This output stays low or high. To change the state of output, the circuit must be driven by an input pulse, called 'trigger pulse'. Untill the trigger arrives, the output voltage remains low or high indefinitely. Since the state does not change even after the removal of the input, the flip-flop is essentially a 1-bit memory (or storage) unit. Such memory units are highly useful in digital computers, to store a binary number or to keep the informations of previous counts and additions until needed.

There are different forms of flip-flops (FF) such as R-S flip flop, J-K flip-flop, Master slave flip-flop and D flip-flop. Most flip-flops are of the clocked type, in which the change of state takes place at some define rate. Flip-flops are used in registers and digital counters.

A general block diagram representing a FF is shown in the figure 127. The value of the output, marked as Q is called the state of the flip-flop i.e., when Q = 1. the state of the FF is 1 and when Q = 0, the state is 0.

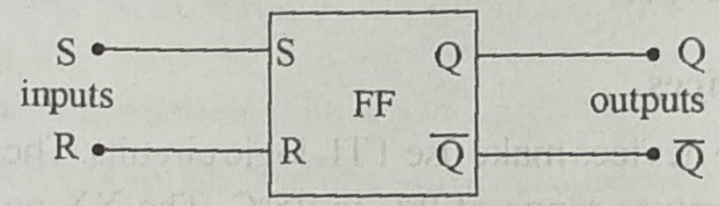


Fig. 127 *Flip flop*

Flip flops usually have two outputs, one equal to the state (Q) of the FF and the other equal to its complement ($\overline{Q}$). Once the state of a flip flop is set, it remains this way until changed by the input. Thus, a flip-flops 'remembers' its inputs. Memory circuits can be built using many flip-flops. Flip-flops are the fundamental building block for sequential logic circuit. 'Sequential' means that the output of each FF feds the input to the next FF to decide its action.

The R-S Flip-Flop

The fundamental (or basic) flip-flop is called the R-S flip-flop or R-S latch. Its block diagram is given below. The input leads are labelled R and S which stands for reset (or clear) and set respectively. The value of Q is the state of the flip-flop. Both input and output signals consist of voltage levels which correspond to 0s and 1s.

The R-S flip-flop operates in the following way:

When R = 0 and S = 1 the flip-flop is set i.e., Q = 1. This makes $\overline{Q}$ = 0. With Q = 1, if the input S = 1 and R = 0 is applied, then there is no change in the state of the FF. This is because the FF is already set.

When R = 1 and S = 0 the flip-flop is reset i.e., Q = 0 and $\overline{Q}$ = 1. On further application of R = 1 and S = 0 the state of the FF remains unchanged in the reset state.

The input R = 0 and S = 0 causes the state of the FF to be unchanged. i.e., does not change from its previous value. This is because there is no new setting or resetting.

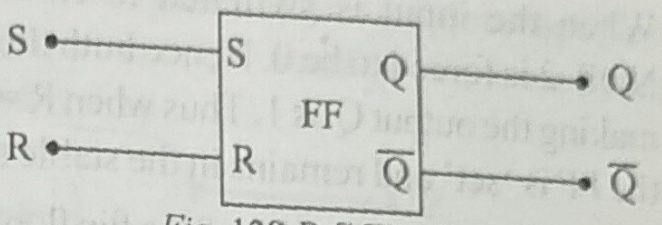


Fig. 128 R-S Flip flop

The input R = 1 and S = 1 is not allowed since such an input attempts to both set and reset the FF at the same time, which violates the basic definition of a flip-flop that requires $\overline{Q}$ to be the complement of Q.

We can summarise the above operations in the truth table.

R	S	Q	Comment
0	0	NC	no change
0	1	1	set
1	0	0	reset
1	1	*	not allowed

Implementation of R-S Flip Flip using two NOR Gates

The R-S flip-flop can be implemented using two NOR gates as shown in figure 129.

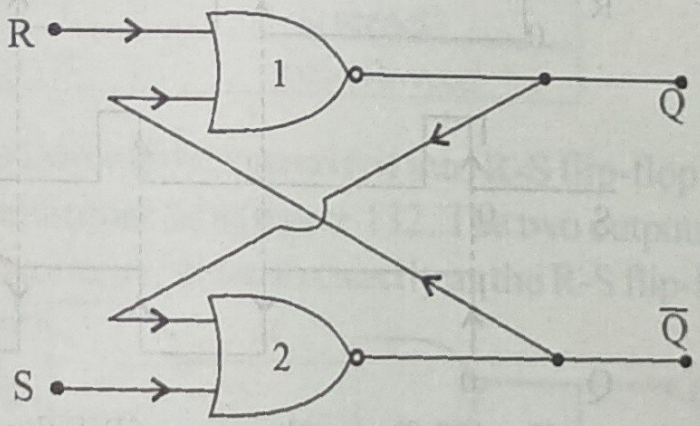


Fig. 129 R-S flip flop using NOR gates

The NOR gates (under positive logic) are inter-connected so that the output of NOR-1 is fed as one of the inputs to NOR-2 and vice versa.

When the input is switched as R = 1 and S = 0, the output of NOR-1 is 0 (at Q). Hence both the inputs to the NOR-2 are zero. This makes the output $\overline{Q}$ as 1. Thus when R = 1 and S = 0, we have Q = 0. The state of the FF is 'Reset' and remains in the stable state with Q = 0.

When the input is switched to R = 0 and S = 1, the output of NOR-2 is forced to be 0. Hence both the inputs to the NOR -1 are zero making the output Q as 1. Thus when R = 0 and S = 1, we have Q =1 and the FF is 'set' and remains in the stable state with Q =1.

When R = 0 and S = 0 the flip flop remains unchanged, since a 0 at the input of a NOR gate has no effect on its output, the FF remains in its existing state. i.e.,Q remains unchanged. Thus when R = 0 and S = 0,

Q = unchanged.

Finally, the input condition R =1 and S = 1 is not allowed, as it forces the output of both NOR gates to the low state. i.e., Q = 0 and $\overline{Q}$ = 0 at the same time, which violates the requirement of a FF. This condition is never used as it leads to unpredictable operation of the FF. This is referred to as "racing problem".

Thus the truth table of the R-S flip-flop is completely explained. The timing diagram of the R-S flip-flop is shown in figure 130. The figure 130 shows how the input signals interact to produce the output signal.

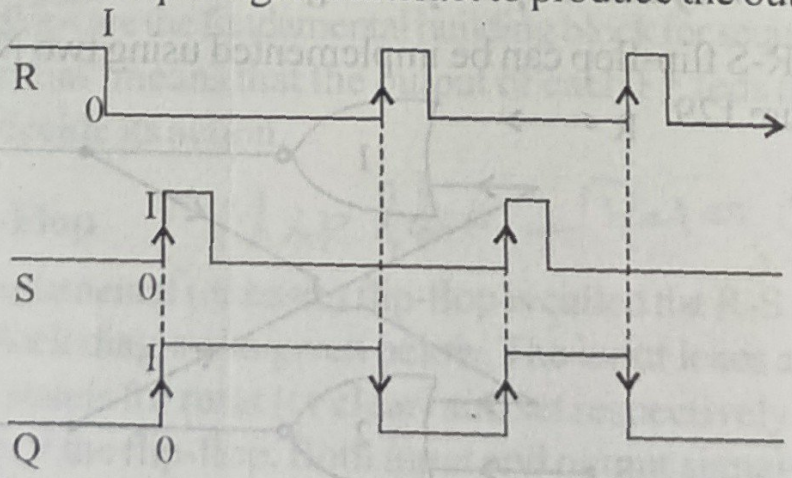


Fig. 130 Timing diagram of R-S flip flop.

The Q output goes high, when S (set) goes high. Q remain high, even after S goes low. Q returns to low when R goes high and stays low even after R returns to low.

Implementation of R-S Flip Flop using NAND gates

An R-S flip-flop using NAND gates is shown in the figure 131. The inputs are $\overline{S}$ and $\overline{R}$ as marked in the figure and we call this FF as $\overline{R}\overline{S}$ flip-flop.

A low to any input to a NAND gate will give a high output. Thus a low on the $\overline{S}$ input will set the flip-flop.(Q = 1 and $\overline{Q}$ = 0). A low on the $\overline{R}$ input will reset it. (Q = 0 and $\overline{Q}$ = 1).

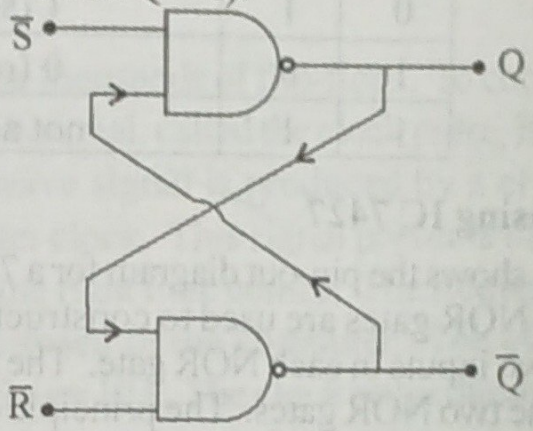


Fig. 131 $\overline{R}$-$\overline{S}$ FF using NAND gates

Both $\overline{R}$ and $\overline{S}$ simulatanously low is not allowed, since this will force both Q and $\overline{Q}$ high.

$\overline{R}$	$\overline{S}$	Q
1	1	last stage (no change)
1	0	1 (set)
0	1	0 (reset)
0	0	not allowed

The $\overline{R}$-$\overline{S}$ flip flop can be converted into R-S flip-flop by using NAND invertor at each input as in figure 132. The two outputs are now $\overline{R}$ and $\overline{S}$. The resulting circuit behaves exactly as the R-S flip-flop.

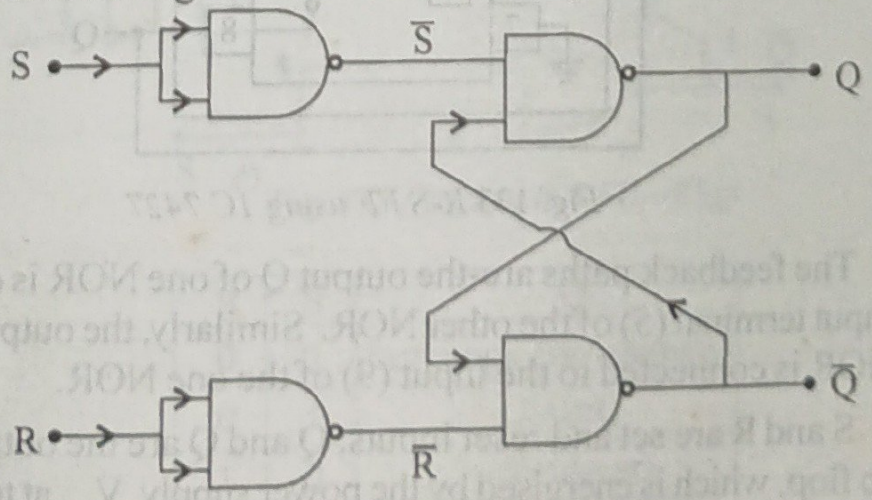


Fig. 132 R-S FF using NAND gates

R	S	Q
0	0	last state (no change)
0	1	1 (set)
1	0	0 (reset)
1	1	not allowed

RS Flip-flop using IC 7427

Figure shows the pin-out diagram for a 7427 triple 3-input NOR gate. Only two NOR gates are used to construct the RS flip-flop. Also we need only two inputs in each NOR gate. The third input is shorted to the second of the two NOR gates. The principle of wiring is the same as that of the logic diagram using NOR gates.

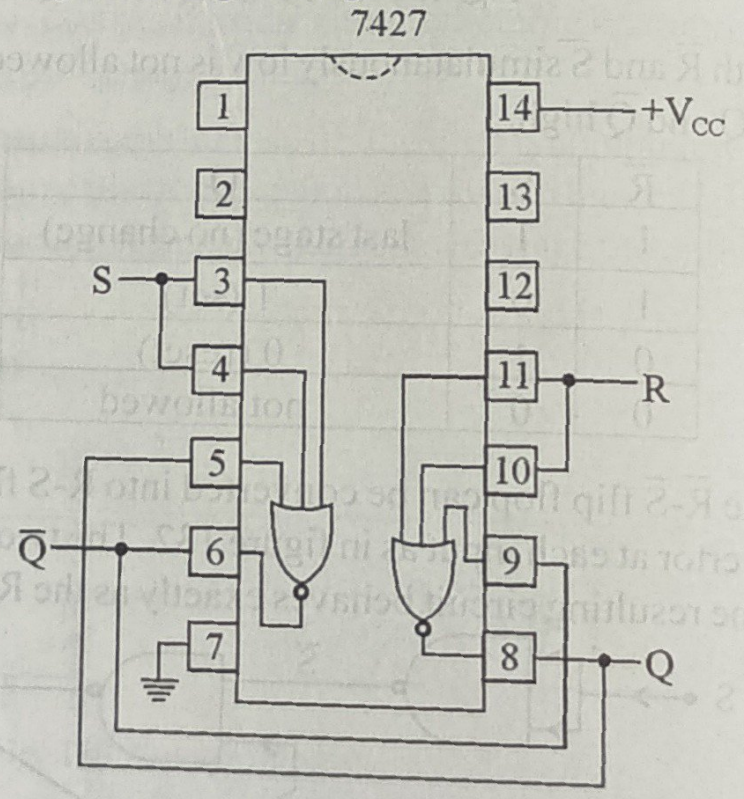


Fig. 133 *R-S FF using 1C 7427*

The feedback paths are-the output Q of one NOR is connected to the input terminal (5) of the other NOR. Similarly, the output Q of the other NOR is connected to the input (9) of the one NOR.

S and R are set and reset inputs; Q and Q are the outputs of the R.S. flip flop, which is energised by the power supply V_{CC} at terminal 14 and terminal 7 is grounded.

The timing diagram is the same as the one already studied with logic diagrams.

Clock Pulses

Computers use thousands of flip-flops. To coordinate the over all action a square wave signal, called the clock pulse, is sent to each FF. The train of square wave signal is produced by a crystal controlled oscillator, called master clock. This signal prevents the flip- flop from changing states until the right time comes. The frequency of the clock pulse determines the speed of the computer. The clock pulse wave form is shown in the figure 103. The clock pulse duration is typically 5 micro second for clock frequency is 0.2 megahertz. The clock pulse can be converted into sharp spikes by passing them through a differentiator (CR circuit).

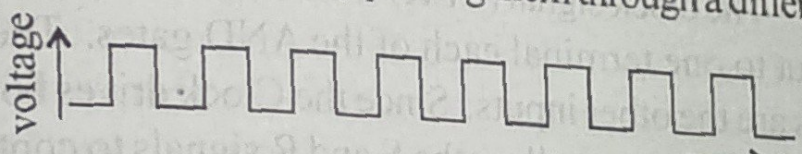


Fig. 134 Clock pulses

Clocked R-S Flip-flop (Latch)

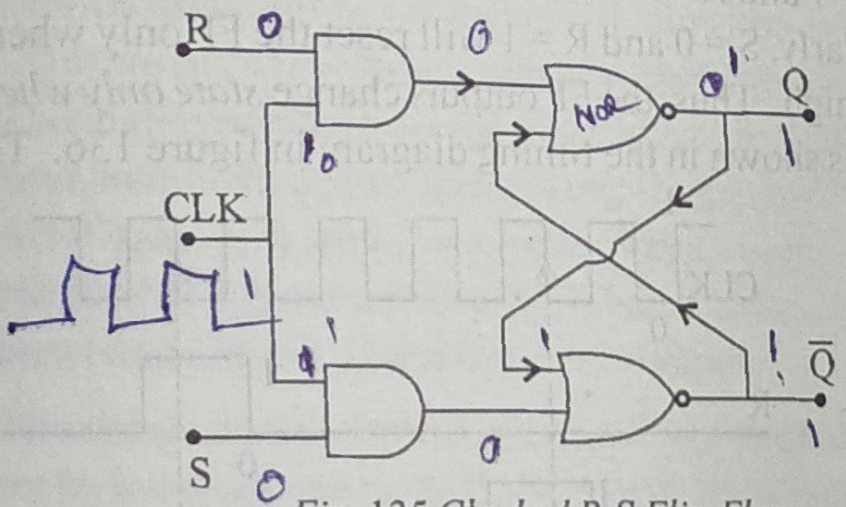


Fig. 135 *Clocked R-S Flip-Flop*

R	S	Q
0	0	last stage (no change)
0	1	1 (set)
1	0	0 (reset)
1	1	not allowed

In a digital system which involves many gates and flip flops, it is not possible to guarantee that the S and R control signals (data inputs) will arrive at exact times, as required for the logic operations. The difficulty can be overcome, by allowing the flip flop to change state only when the clock pulse is present. ie: the clock pulse has control over the operation of the FF. In this way, the output at Q is synchronised with the clock and it does not depend merely on the time of arrival of the S or R signal.

The clocked R-S flip has three inputs. Set (S), Reset (R) and Clock (CLK) and two outputs Q and Q. It has two additional AND gates along with the R-S flip flop.

The clock signal (CLK) which does the synchronisation is given as input to one terminal each of the AND gates. The *S* and *R* control signals are the other inputs. Since the Clock drives both the AND gates a low clock does not allow the S and R signals to control the FF (ie., to change the output state).

S = 1 and R= 0 will set the FF only when the clock pulse goes high. Similarly, S = 0 and R = 1 will reset the FF only when the clock pulse goes high. Thus, the FF outputs change *state only when the clock goes high* as shown in the timing diagram in figure 136. This is latch action.

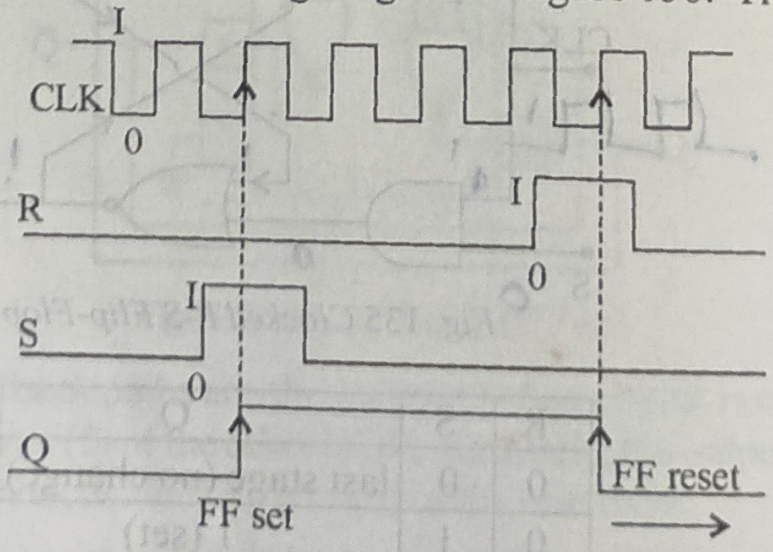


Fig. 136 Timing diagram of Clocked R - S Flip-Flop

The J-K Flip-Flop

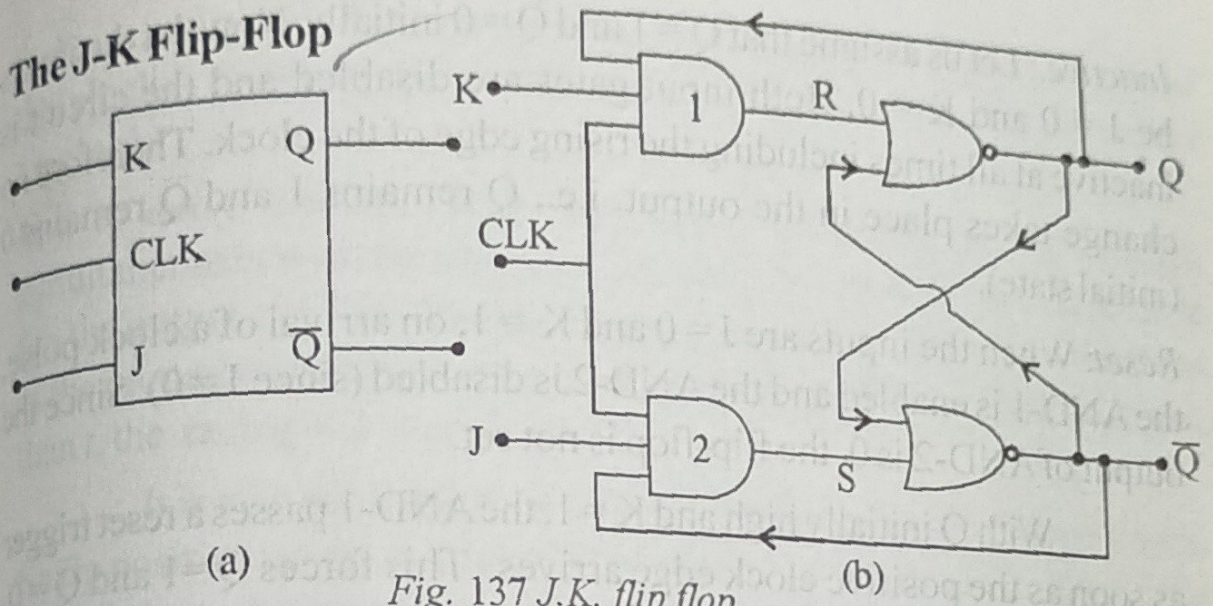


Fig. 137 J.K. flip flop

In an R-S flip flop the input R = 1 and S = 1 must not be used. The circuit of R-S flip flop can be modified so that even when R = 1 and S = 1, it is possible to make the outputs to be the complement of each other (i.e., if Q = 1 and $\overline{Q}$ = 0 and vice versa). A flip-flop, doing this, is called the J-K flip-flop.

The K input replaces the R input and the J input replaces the S input. The block diagram for a clocked J-K flip-flop is shown in the figure 137. It consists of two inter-connected NOR gates. Each NOR is further fed with the output of one of the AND gates. Each AND gates has three input terminals. Q is connected to one input of AND gate 1 and $\overline{Q}$ to the AND gate 2, K and J are other inputs to the AND gate as shown. Both the AND gates are driven by clock pulses. J and K are the control inputs (data inputs). Q and $\overline{Q}$ are the outputs.

The outputs are controlled not mearly by the presence of J and K inputs but by the clock pulse. With the presence of J and K, change of state in the output occurs only at the time of the rising edge (positive going edge) of the clock pulse (J, K is like S, R.)

Operation : Let us dicuss the operation of the J-K flip flop, by considering specific examples. When the clock pulses are absent, the two AND gates are disabled and so no change of state in the output takes place. What happens when the clock pulses are present?

Inactive : Let us assume that $Q = 1$ and $\overline{Q} = 0$ initially. Now, let the inputs be $J = 0$ and $K = 0$. Both input gates are disabled and the circuit is inactive at all times including the rising edge of the clock. Therefore no change takes place in the output. i.e., Q remains 1 and $\overline{Q}$ remains 0 (initial state).

Reset: When the inputs are $J = 0$ and $K = 1$, on arrival of a clock pulse, the AND-1 is enabled and the AND-2 is disabled (since $J = 0$). Since the output of AND-2 is 0, the flip-flop is not set.

With Q initially high and $K = 1$, the AND-1 passes a reset trigger as soon as the positive clock edge arrives. This forces $Q = 1$ and $\overline{Q} = 0$. Therefore, $J = 0$ and $K = 1$ means that the rising clock edge resets the FF. i.e, the state of the J K flip flop changes to 0. (if the initial state is already $Q = 0$, then it will continue to remain so, even in the presence of the clock pulse).

Set: When the inputs are $J = 1$ and $K = 0$, with $Q = 0$ and AND gate- 1 is disabled and AND-2 is enabled and the FF passes a set- trigger during the positive clock edge. This drives Q into the 1 state i.e, $Q = 1$ and $\overline{Q} = 0$ i.e.; $J = 1$ and $K = 0$ means that the next positive clock edge sets the FF i.e. the state of the FF changes to 1.

Toggle: When the inputs are $J = 1$ and $K = 1$, it is possible to set or reset the flip- flop, depending on the existing state of the outputs. If Q is 1 (existing state) and AND-1 passes a reset trigger on the next positive clock edge. This makes $Q = 0$ and $\overline{Q} = 1$. On the other hand, when the existing state is $Q = 0$, $\overline{Q} = 1$, AND-2 passes a set trigger on arrival of the next positive clock edge. This would make $Q = 1$ and $\overline{Q} = 0$. Either way Q changes from the existing stage to its complementary state. Therefore $J = 1$ and $K = 1$ means that the flip-flop will toggle on the next positive clock edge. (toggle means switching to opposite state).

Racing: With $J = 1$, $K = 1$ and high clock, the output will toggle. Accordingly, the new outputs are fed back to the input gates. Hence the output toggles again. Once more new outputs return to the input gates. In this way, the output can toggle repeatedly, as long as the clock remains

high, i.e., we get oscillations in the output during the positive half cycle of the clock. Toggling more than once during a clock cycle is called 'racing'.

How to avoid racing: In a flip- flop, the outputs do not change simultaneously with the change of inputs. The amount of time it takes for the output of FF to change states, after the input changes, is known as the propagation delay time (t). If the width of the clock pulse is less than t, the racing will be prevented.

It is also advisible to work with clock pulses of reduced width. Sharp positive going spikes could be produced at the rising edge of the clock pulse and fed as trigger to the clock input terminal. This is achieved by connecting CR circuit (as differentiator) at the clock pulse input as shown in figure 138. The flip flop is triggered only at the arrival of the positive spikes. Thus we get only one toggle in each clock cycle and racing is eliminated.

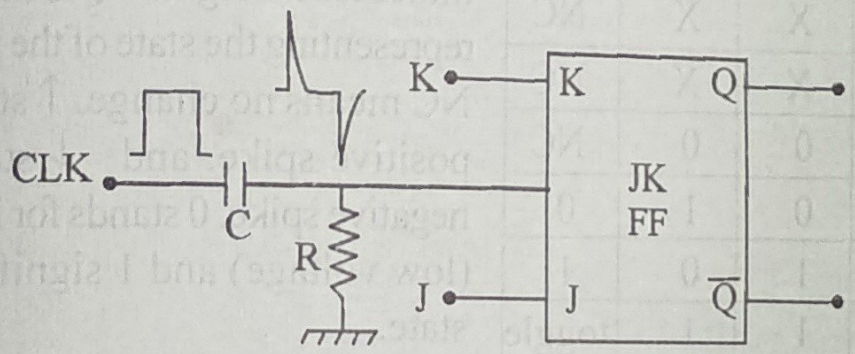


Fig. 138 *J.K. flip flop*

Another way to avoid racing problem is to build J-K master slave flip flop.

The flip flop, operated by positive spikes at the clock input terminal, is called positive edge triggered J-K flip flop. If the circuit arrangement is such that the output of the flip-flop changes at the trailing (falling) edge of the clock, the FF is known as negative edge triggered J-K flip flop. It is indicated by a bubble along with the arrow at the clock input terminal of the FF block diagram.

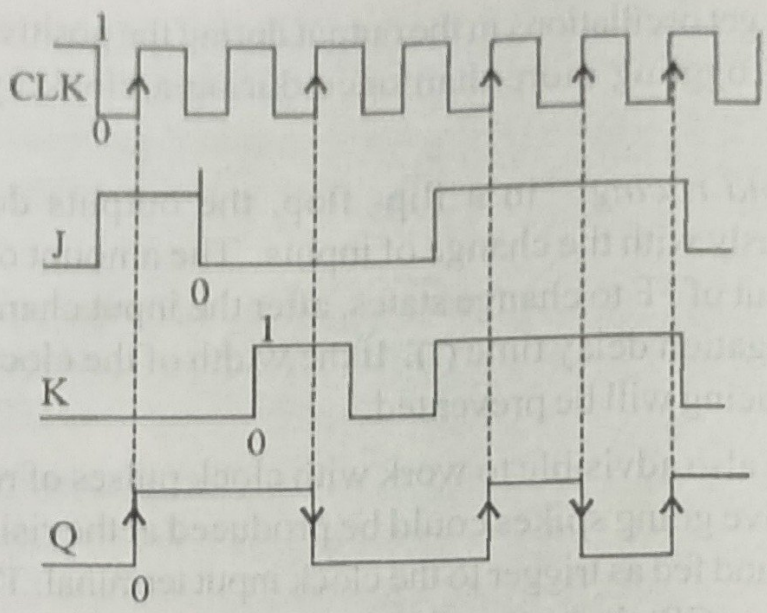


Fig. 139 J.K. flip flop timing diagram.

Truth table of positive edge triggered J-K flip flop

CLK	J	K	Q
0	X	X	NC
1	X	X	NC
↓	X	X	NC
X	0	0	NC
↑	0	1	0
↑	1	0	1
↑	1	1	toggle

In the truth table given, X indicates no signal. Q is the output representing the state of the flip flop. NC means no change. ↑ stands for positive spike. and ↓ stands for negative spike. 0 stands for low state (low voltage) and 1 signifies high state.

TTL -Clock (to produce square waves)

The TTL Clock can be constructed using two isolated NOT gates N_1, N_2 in a hex inverter (IC 7404). Each NOT gate acts as an amplifier, producing phase shift of 180^0. The two inverters are used to construct a two stage amplifier with an overall phase shift of 360^0 between pins 1 and 6. A part of the signal at pin 6 is fed back to pin 1 by means of a crystal. The feed back is positive.

The inverter amplifiers N_1 and N_2 have feedback resistors R_1 and R_2 respectively. The gain of amplifier $N_1 = \frac{V_0}{I}$ where

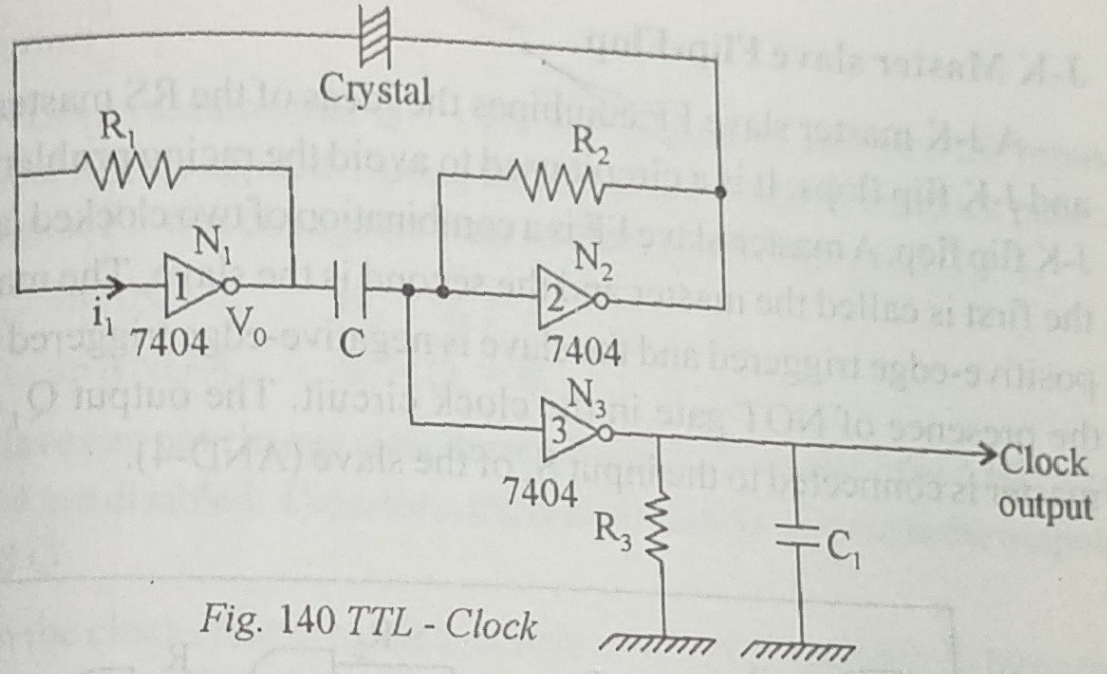


Fig. 140 TTL - Clock

(current-to-voltage amplifier)

I is current through the resistance and V_0 is the output voltage.

i.e; the gain $A_1 = -R_1$ (since $V_0 = -IR$) .....(1)

Similarly, the gain of amplifier N_2 is $A_2 = -R_2$ .....(2)

The two inverter amplifiers are coupled through a capacitor C to form an amplifier. The overall gain of this amplifier is $A = A_1 \times A_2$

i.e., $A = (-R_1) \times (-R_2) = R_1 R_2$

The overall gain is positive. The gain $R_1 R_2$ is sufficient to make the circuit oscillate, since Barkhausen criterion is satisfied.

The crystal in series mode in the feedback path acts as an RLC circuit. At resonant frequency, it appears as a low resistance and produces no phase shift by itself. Hence the oscillator frequency is the resonant frequency of the crystal. The oscillator frequency is stable since the feedback element is a crystal.

The output can be drawn at the input of N_2. This is amplified by a third inverter gate, which together with $R \| C_1$ acts as buffer amplifier, isolating the oscillator and output terminal. The output is a series of square wave pulses, called clock pulses. The output frequency is typically 5 megahertz.

J-K Master slave Flip-Flop

A J-K master slave FF combines the ideas of the RS master slave and J-K flip flops. It is a circuit used to avoid the racing problem with J-K flip flop. A master-slave FF is a combination of two clocked latches; the first is called the master and the second is the slave. The master is positive-edge triggered and the slave is negative-edge triggered due to the presence of NOT gate in the clock circuit. The output Q_1 of the master is connected to the input S_2 of the slave (AND-4).

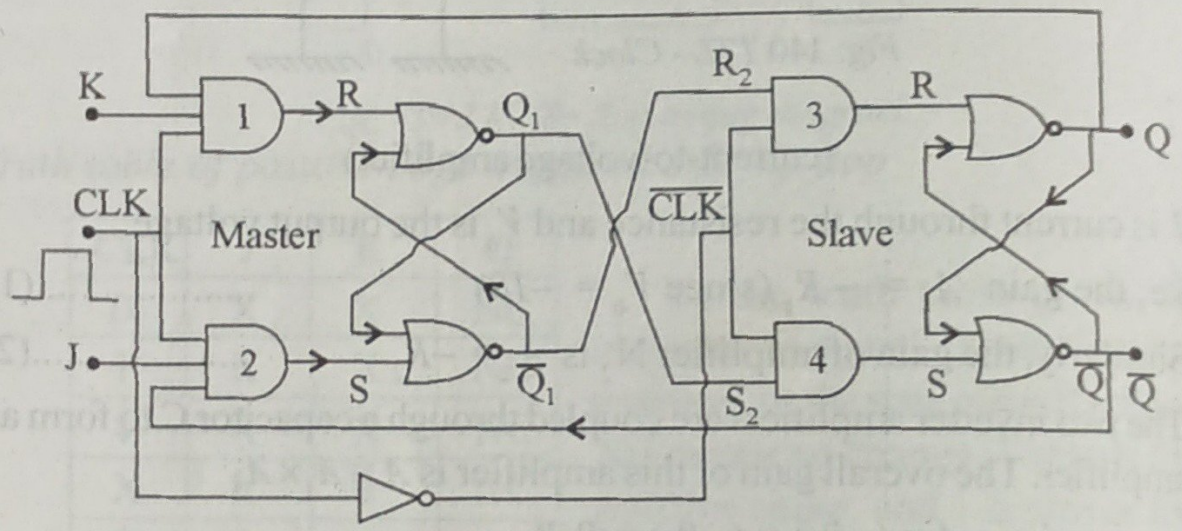


Fig. 141 *J.K Master-slave flip flop*

The output of $\overline{Q}_1$ is connected to R_2 of the slave (AND-3). The inputs are J, K and clock (CLK) to the master and the final outputs are at Q and $\overline{Q}$ of the slave.

We note, that the master is positively clocked and the slave is negatively clocked. This means the following:

(a) While the clock is high, the master is active and the slave is inactive.

(b) While the clock is low, the master is inactive and the slave is active.

Operation

Reset: Let us assume that Q = 1 and $\overline{Q} = 0$ initially and then the input J = 0 and K = 1 is applied. During the first positive half of the clock pulse, the master FF will change its state so that Q_1 become 0 and $\overline{Q}_1 = 1$.

In AND-3, $R_2 = \overline{Q}_1 = 1$

In AND-4, $S_2 = \overline{Q}_1 = 0$.

The slave can not change state during the high clock since the AND gate 3 and 4 are disabled. Therefore, there is no change of state at the outputs Q and $\overline{Q}$.

When the clock returns to the low state , the low S_2 and high R_2 force the slave to reset. This results in the output Q = 0 and $\overline{Q} = 1$ resetting the J-K master slave FF. i.e. when J = 0 and K = 1, the flip is reset when the clock goes low.

Set: Let Q = 0 and $\overline{Q} = 1$ in the initial state. For the input J = 1 and K = 0 and the high CLK, the master goes into the set state, producing $S_2 = 1$ and $R_2 = 0$. Nothing now happen to the $\overline{Q}$ and Q outputs since the slave is inactive while the clock is high. When the clock goes low, the high S_2 and low R_2 forces the slave into set state making Q = 1 and $\overline{Q} = 0$.

Toggle: When the inputs are J = 1 and K = 1 the master toggles once when the clock is high. When the clock goes low, the slave changes its state until the next positive clock pulse arrives. Thus, the final output will change once during each clock pulse

and the racing is avoided. The problem of instability in the output of J K flip flop with J = 1 and K = 1 is removed.

Also, with J = 1 and K = 1 the flip flop toggles each time the clock goes low. The wave form at Q has a period twice that of the clock period. That is, the frequency of wave form at Q is one half of that of clock input. Thus, the circuit acts as a frequency divider (divide-by 2).

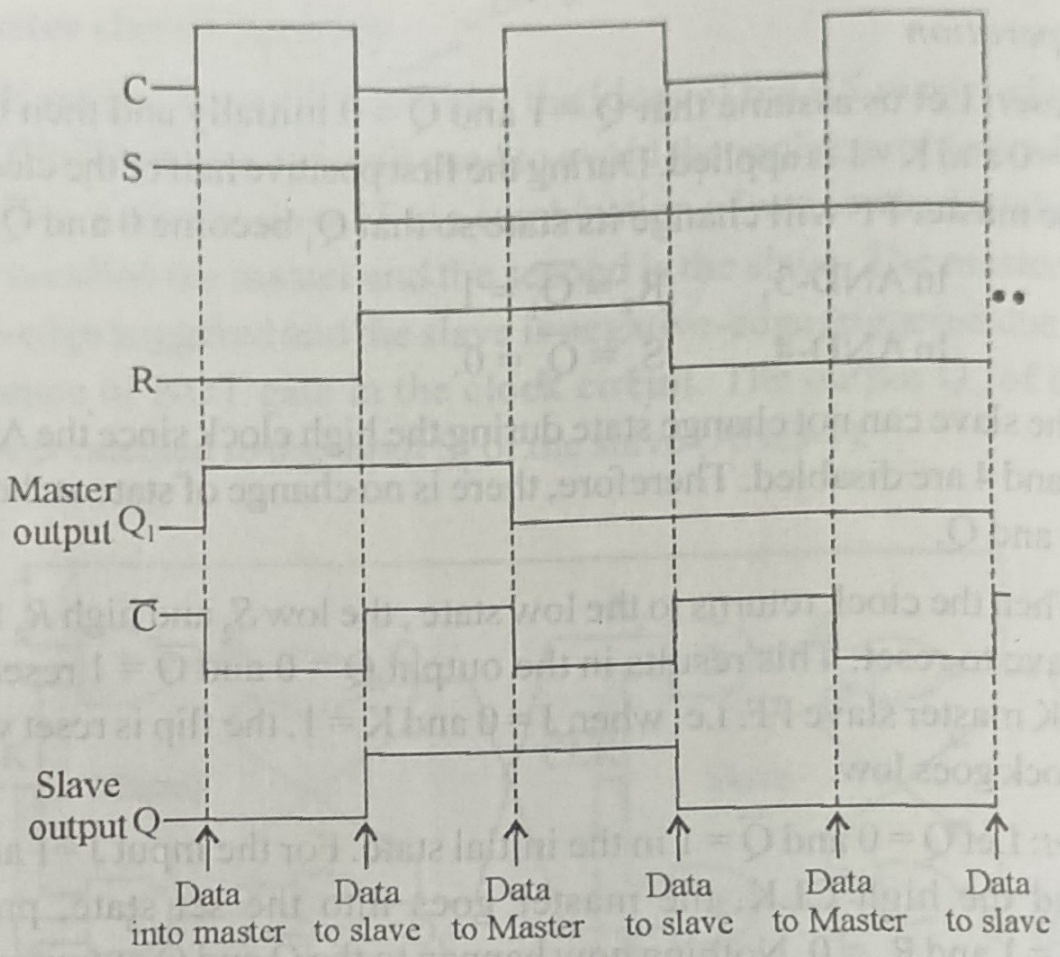


Fig. 142 *Timing diagram for master-slave flip flop.*

Figure 142 shows the symbol of the JK master slave FF with preset and clear functions. The bubble on the CLK input reminds us that the output changes when the clock goes low. The timing diagram of JK master slave flip flop is shown in Fig. 142(negative edge triggering). When a high input(1) is given to the Preset (PR) terminal, the FF is brought to set condition (namely Q = 1) state. When a high input (1) is supplied to clear (CLR) input, the FF is reset (Q = 0).

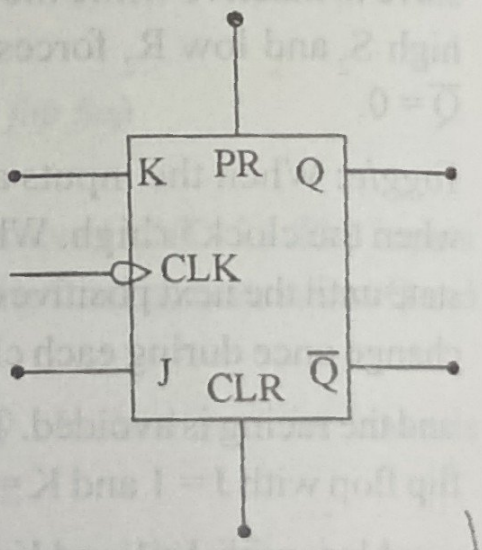


Fig. 143 *J-K Master slave FF.*

D-Flip Flop

The RS flip-flop has two data inputs: S and R. To store a high bit, we need a high S and a low R. To store a low bit, we need a low S and high R.

Generating two signals to drive a flip-flop is a disadvantage in many applications. It would be desirable to have a flip-flop that would operate with one-data input signal. Such a flip-flop is the D-flip-flop. The forbidden condition that R = 1 and S =1 does not arise in D-flip flop because the D-FF needs only a single data input. The D-flip flop receives the designation from its ability to transfer 'data' into a flip flop.

As it is clear from the figure 144, D drives the S input and $\overline{D}$ drives the R input. When the input D =1, the output Q = 1 and the flip-flop is set.

∴ high D sets the FF and a low D resets it, when the clock is high.

The truth table is given below.

D	Q	Condition
high (1)	1	Set
low (0)	0	Reset

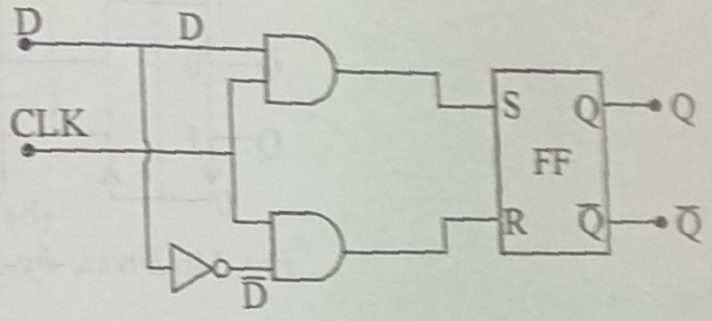


Fig. 144 D-Flip-Flop

We note that there are bubbles at the PR (preset) and CLR (clear) input terminals. The bubbles indicate that PR and CLR will be active only in the low state (0). To preset the FF (to make Q =1) the PR input must go low temporarily and then be returned to high. Similarly, to reset the flip flop (Q = 0) CLR input must go low and then come back to high. The PR and CLR can activate the flip flop independently of the clock. The D input is effective only when clock edge occurs.

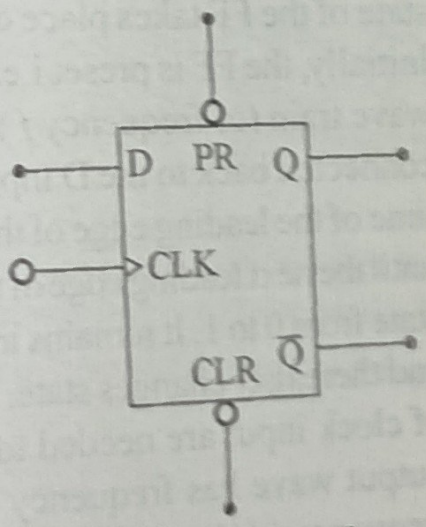


Fig. 145 D - flip - flop.

Applications of D-flip flop

(i) as storage register

and (ii) as divide-by-2-counter.

Divide-by-2 circuit with D-Flip-flop (Frequency divider)

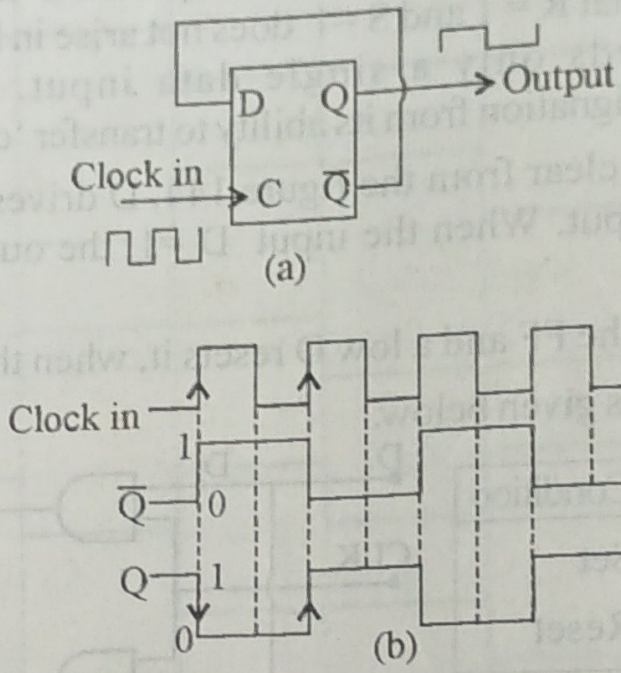
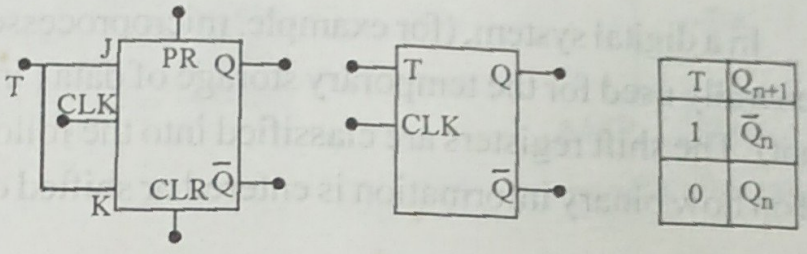


Fig. 146 *Divide-by-2 counter*

The flip flop chosen is positive edge triggered i.e., change of state of the FF takes place only at the leading edge of the clock pulse. Initially, the FF is preset i.e., Q = 1 (as in the timing diagram). Square wave train (of frequency f) is fed to the clock input. The $\overline{Q}$ output is connected back to the D input. The flip flop changes state (1→0) at the time of the leading edge of the first pulse arrived. It remains in this state until the next leading edge of the clock pulse arrives. Now the FF changes state from 0 to 1. It remains in this state until the arrival of the next pulse and thereafter changes state. This process is repeated. Thus, two cycles of clock input are needed to produce one cycle at the Q output. The output wave has frequency (f/2). To divide by 4, i.e., to obtain the frequency (f/4) of the input wave, another D flip flop is connected in sequence. The output Q of the FF, is connected as the clock input of FF2. The output Q_2 of $\overline{FF2}$ gives out a wave of frequency (f/4). The resulting wave form of the output is seen in the figure 146 (b).

T- Flip Flop

It is a single input version of JK flip flop.



T	Q_{n+1}
1	$\bar{Q}_n$
0	Q_n

Fig. 147 *T- Flip Flop*

CLK	T	Q_{n+1}
high	1	$\overline{Q}_n$
high	0	Q_n

To construct a T-flip flop, a JK flop can be used. The J and K terminals are held together at 1 state (high state). The state of the flip-flop (indicated by the state of the output Q) can be made to change from 1 to 0 or from 0 to 1 (toggle), by applying clock pulse at the clock input (CLK) state. This is indicated in the truth table as $Q_{n+1} = \overline{Q}_n$ when $T = 1$. This kind of flip flop is known as toggle (T) or complementing flip-flop.

However, when $T = 0$ state, the flip-flop will not change its state even when the clock pulse is applied. The FF does not work for $T = 0$. This is indicated in the truth table as $Q_{n+1} = Q_n$ for $T = 0$. The T flip-flop can be used as a frequency divider.

Shift Register

A flip-flop can store one bit of digital information (1 or 0). It is known as one-bit register. An array of flip-flops may be used (in cascade form) to store binary information. The number of flip-flops required is equal to the number of bits in the binary word. This is called a register.

Thus, a register is a device which is used to store and (or) shift data entered from external sources. The stored information is in the form of binary numbers.

In a digital system, (for example, microprocessor) such registers are generally used for the temporary storage of data (in relatively small amount). The shift registers are classified into the following four types, based on how binary information is entered or shifted out.

1. Serial-in, serial-out (SISO) - 54 / 74 LS 91
2. Serial-in, parallel out (SIPO) - 54 / 74164
3. Parallel-in, serial out (PISO)-54 / 74165
4. Parallel-in, parallel out (PIPO) 54 / 74198

A register is referred to as a universal register if it can be operated in all the four possible modes and also as a bi-directional register. IC 74194 is a universal register.

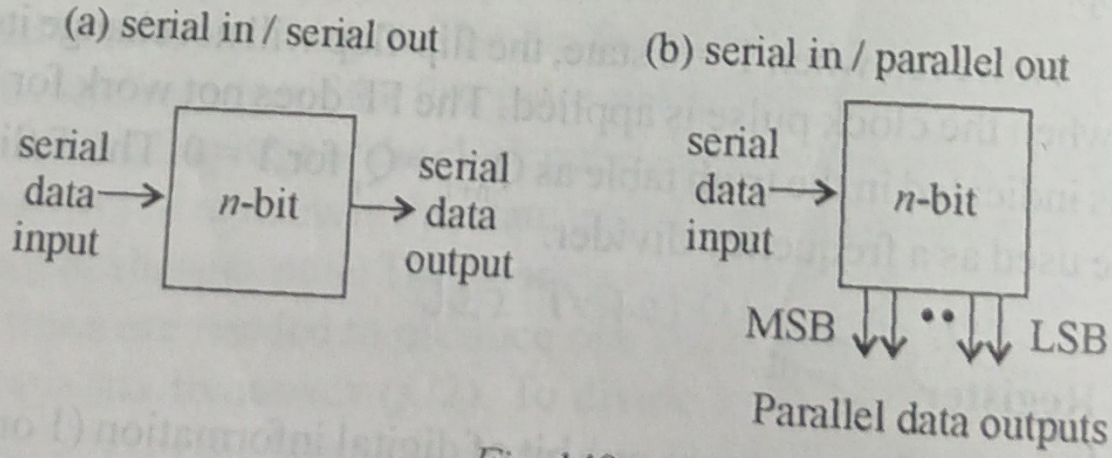


Fig. 148

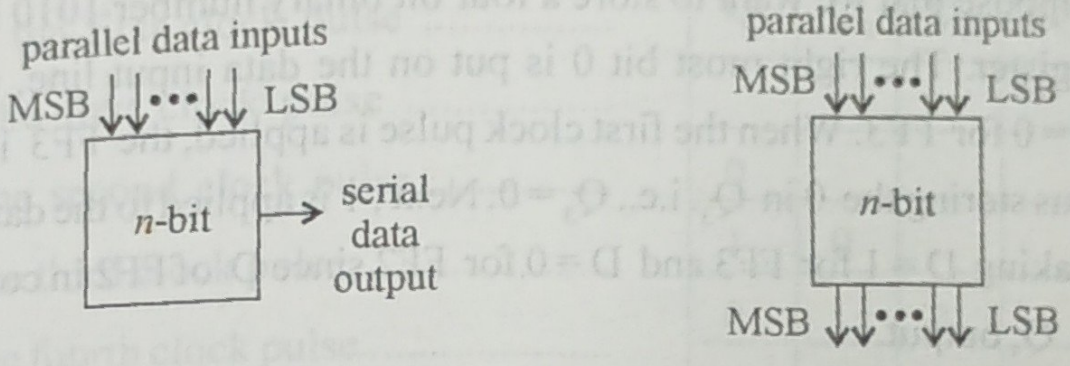


Fig. 149

1. Serial-in, serial-out shift register

A shift register is a device, capable of shifting a binary word to the left or right. We consider a shift register that accepts data serially. That is, one bit at a time on a single line. It will produce the stored information on its outputs also in serial form. The arrangement for four bit serial shift register is shown in the figure. Four D-type flip-flops are interconnected.

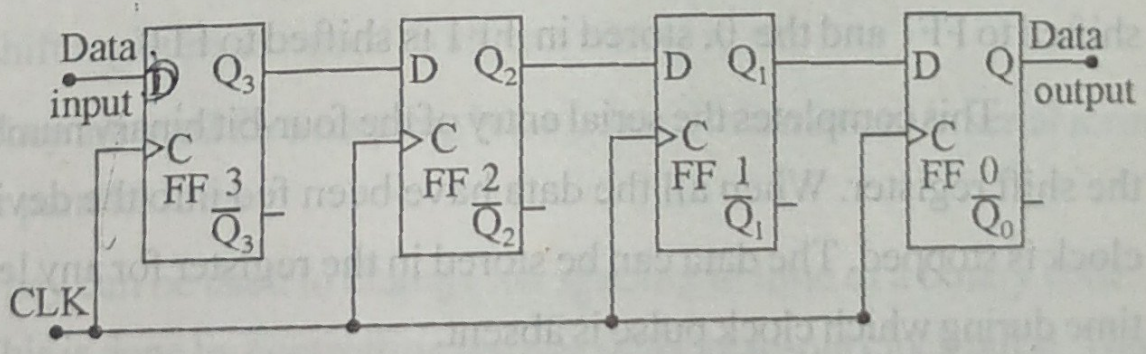


Fig. 150 *Shift register*

The clock pulse line is connected to all the CLK terminals. The data input (to be stored) is fed bit by bit to the D input terminal of the flip-flop FF3, whose output Q_3 is given to the input of FF2 and so on. By this arrangement, four bit data can be stored i.e., its storage capacity is four bits. The process of entering the data is known as writing into the register.

Suppose that we want to store a four bit binary number 1010 into the register. The right most bit 0 is put on the data input line, making $D = 0$ for FF3. When the first clock pulse is applied, the FF3 is re-set, thus storing the 0 in Q_3. i.e., $Q_3 = 0$. Next, 1 is applied to the data input, making $D = 1$ for FF3 and $D = 0$ for FF2 since D of FF2 in connected to Q_3 output.

When the second clock pulse occurs, the 1 on the data input is shifted into FF3 (FF3 sets), shifting the 0 that was in FF3 into the output of FF2.

The next 0 bit in the binary number is now put on into the data input line and a clock pulse is applied. The 0 is entered into FF3. The 1 stored in FF3 is shifted to FF2 and the 0 that was in FF2 is shifted into FF1 output. Similarly, the last bit in the binary number 1 is now applied to the data input and clock pulse is applied. This time, the 1 is entered into FF3. The 0 that was in FF3 is shifted to FF2. The 1 stored in FF2 is shifted to FF1 and the 0, stored in FF1 is shifted to FF0.

This completes the serial entry of the four-bit binary number into the shift register. When all the data have been fed into the device, the clock is stopped. The data can be stored in the register for any length of time during which clock pulse is absent.

To bring the data out of the register, they must be shifted serially and taken off the Q_0 output. On application of the fifth clock pulse, the right most bit 0 comes out from Q_0 and the second bit 1 appears on the Q_0 output. On arrival of each clock pulse, the bits are shifted right and drawn out in sequence one after another and are used. The information stored in a four-bit shift register on successive clock pulse is shown below :

	Q_3	Q_2	Q_1	Q_0
Before the first clock pulse				
After the first clock pulse	0			
After the second clock pulse	1	0		
After the third clock pulse	0	1	0	
After the fourth clock pulse.....	1	0	1	0
.....				
After the fifth clock pulse	1	0	1	

It is also possible to read the stored data in the parallel form, directly from $Q_3Q_2Q_1Q_0$. No clock pulse need be applied to read the output. The contents of the register can be read any number of times, until new data are stored in the register.

Uses

(1) Shift registers are used to store a binary word.

(2) They are used for accepting binary data (information) in serial form or parallel form.

(3) They can be used to change the spacing in time of a binary code. This is done by controlling the rate of the read-out clock pulse. This phenomenon is called buffering.

(4) Shift registers are used for aligning decimal places for shifting multiplicand in binary multiplication.

(5) A number can be divided by two, by producing one shift-right and a number can be multiplied by two by producing one shift-left.

Number stored			1	1	=	3
After one shift-left		1	1	0	=	6
After further one-shift right			1	1	=	3

(6) Another application of the shift register is the conversion of data from serial form into parallel form (receiving information serially and reading the same in parallel). This application is most useful for converting serial digital information telemetered from a remote source to a parallel output, suitable for printing of computer input.

The basic difference between a register and a counter is that a register has no specified sequence of states. A register is used for storing and shifting data (1 s or 0s), entered into it from external source and possess no characteristic internal *sequence of states.*

Since data entered in to the register can remain stationary until a clock pulse is used to shift, the shift register we have studied so far is known as *static register*.

2. Serial-in, parallel out shift-register

In this type of register, the data is shifted in serially but shifted out in parallel. In order to shift the data out in parallel, it is necessary to have all the data bits available as outputs at the same time. This can be achieved by connecting the output of each flip-flop to an output pin, from where output can be taken (say Q_3, Q_2, Q_1, Q_0). The total parallel output will be $Q_3\ Q_2\ Q_1\ Q_0$ taken simultaneously.

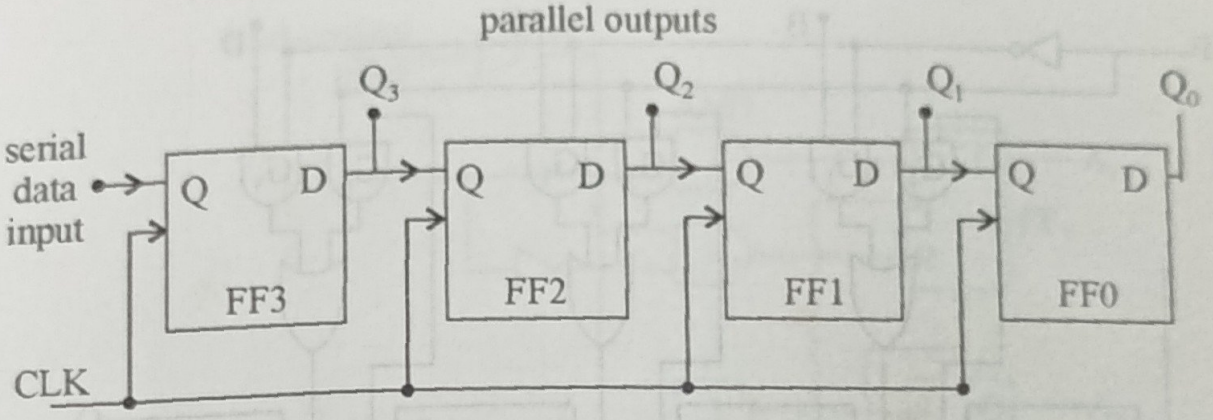


Fig. 151

The circuit arrangement is shown in the figure. The working of the circuit is the same as that of serial-in, serial-out shift register.

The register shifts one bit of information from one FF to the next (say first FF → second FF) for each clock pulse. Let there be four bits of information to be stored in the register. LSB enters into the FF-3 at the end of the first clock pulse. At the end of the second clock pulse, this bit is shifted to FF-2 and the next bit of information enters FF-3. For successive four clock pulses, all the four bits are stored as Q_3, Q_2, Q_1, Q_0 in the four flip-flop outputs, with the MSB in FF3.

When all the data has been fed into the register, the clock is stopped. The data can be stored for any length of time, during which the clock pulse is absent. When a reset pulse is given to all the flip-flops, the register is cleared off.

3. Parallel-in, serial out shift register

We are familiar with shifting a data bit into and out of a register in serial mode. Here we follow a different method for the parallel entry of data bits. FF_1, FF_2, FF_3, FF_4 are the shift registers and A, B, C, D are the inputs (data bits), available at the input terminals. The bits are to be simultaneously entered into the respective flips. The bits B, C, D can be entered into the respective flip flops by the use of the AND gates G_1, G_2 and G_3 and OR gates, as shown in the figure. The input A is given directly to FF_1.

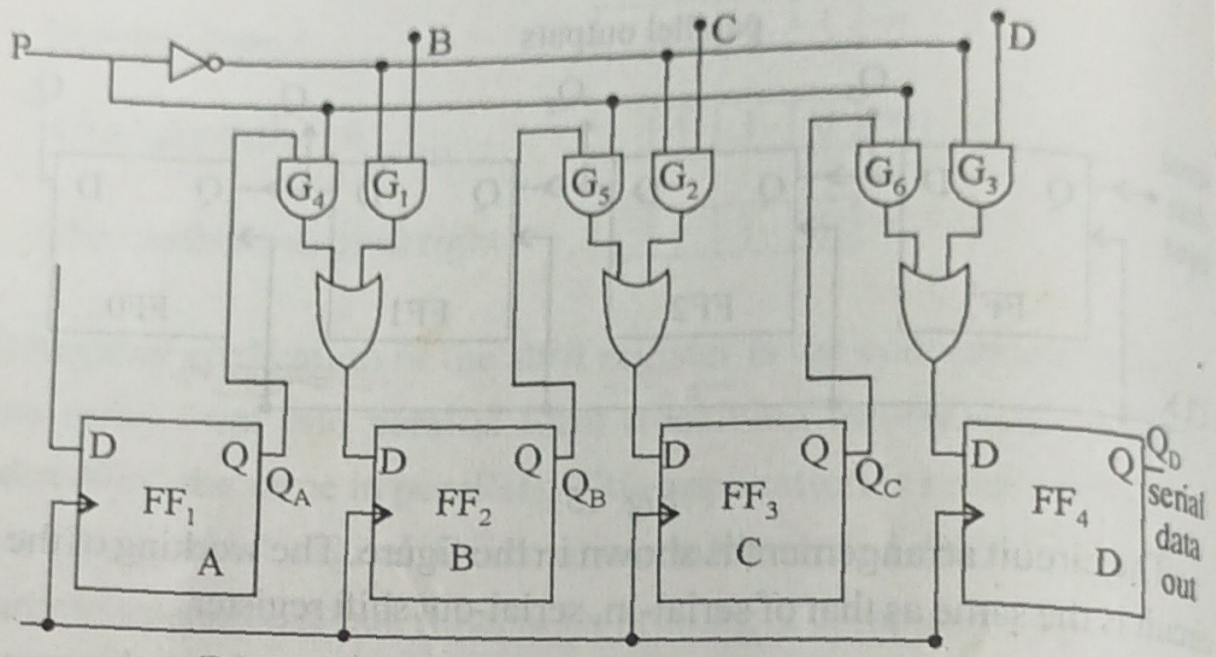


Fig. 152 A 4 - bit parallel-in-serial-out shift register

To load the flip-flops FF_2, FF_3 and FF_4, the shift / load point P is given 0 state (low), AND gates G_1, G_2, and G_3 are enabled, allowing the data to appear as parallel inputs. i.e., the inputs B, C and D are at the input terminals of the respective flip flops simultaneously. The inprit A is already at the input terminal of FF1. When a clock pulse is applied, the inputs (four bits) are stored simultaneously in the respective flip-flops.

Now the point P (shift/load) is kept high (1 state), the AND gates G_4, G_5 and G_6 are enabled, allowing the data bits shift right from one stage to the next stage. The shifting takes place bit by bit for each clock pulse. At the end of four clock pulses, all the data are serially available at the output terminal of FF_4 one by one. Thus, the circuit works as a parallel-in, serial-out shift register.

Parallel -in, Parallel-out shift register

A register is a group of binary storage cells (Flip Flops), suitable for holding binary information. The transfer of a new information into a register is known as loading the register. If all the bits of the register are loaded simultaneously with a single clock pulse, we say that loading is done in parallel.

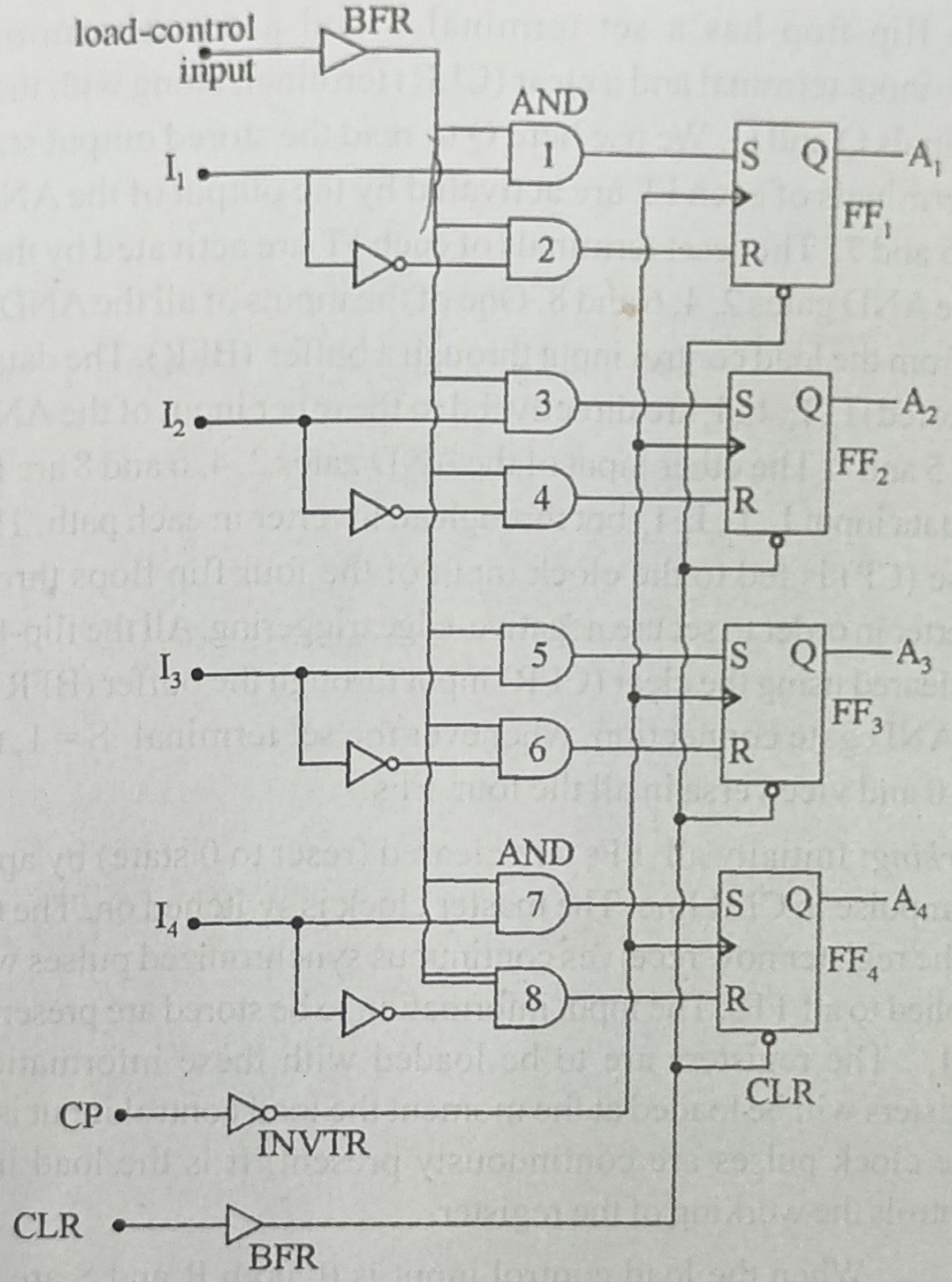


Fig. 153 *Registers with parallel load*

A 4-bit register with parallel loading using four R-S flip flop and a load control input is shown in figure 153. I_1, I_2, I_3, I_4 are the input bits (of information) to be stored in the register. A_1, A_2, A_3, A_4 are the corresponding bits stored in the flip-flops and these are the parallel outputs simultaneously available at the register.

Each flip-flop has a set terminal S and a re-set terminal R, a clock-input terminal and a clear (CLR) terminal, along with the output terminals Q and $\overline{Q}$. We use here Q to read the stored output state. The set terminals of each FF are activated by the output of the AND gates 1,3, 5 and 7. The reset terminals of each FF are activated by the output of the AND gates 2, 4, 6 and 8. One of the inputs of all the AND gates is fed from the load control input through a buffer (BFR). The data bits (to be stored) I_1, I_2, I_3, I_4 are directly fed to the other input of the AND gates 1, 3, 5 and 7. The other input of the AND gates 2, 4, 6 and 8 are fed from the data input I_1, I_2, I_3, I_4 but through an inverter in each path. The clock pulse (CP) is fed to the clock input of the four flip flops through the inverter in order to secure negative-edge triggering. All the flip-flops can be cleared using the clear (CLR) input through the buffer (BFR). Due to the AND gate connection, whenever the set terminal S = 1, the reset R = 0 and vice versa in all the four FFs.

Working: Initially all FFs are cleared (reset to 0 state) by applying a clear pulse in CLR line. The master clock is switched on. The CP input of the register now receives continuous synchronized pulses which are applied to all FFs. The input information to be stored are present as I_1, I_2, I_3, I_4 . The registers are to be loaded with these informations. The registers will be loaded at the moment the load control input is applied. The clock pulses are continuously present. It is the load input that controls the working of the register.

When the load control input is 0, both R and S are 0 and no change of state occurs with any clock pulse. Thus, the load input is a control variable, which can prevent any information change in the register as long as its input is 0.

When the load control input goes to 1, input I_1, I_2, I_3, and I_4 determine what binary information is loaded into the register on the next clock pulse. When any input is present as I = 1, the corresponding Flip Flop's S = 1 and R = 0. The corresponding flip flop is set and it stores the information bit I = 1.

For each input I = 0, the corresponding Flip-Flop S = 0 and R = 1 and the input I = 0 is stored in the corresponding Flip-Flop. Thus, the input value is transfered into the register only if

(i) the load input is 1

(ii) the clear input is 1

and (iii) a clock pulse goes from 1 to 0 (negative edge triggering).

This type of transfer is called a parallel-load transfer because all bits of the register are loaded simultaneously. Also, the output A_1, A_2, A_3, A_4 are available simultaneously. Hence the register is parallel-in parallel-out shift register.

The need for the buffer (BFR) is to decrease the loading of the load control input pulse generator due to the (on) AND gates. The purpose of the inverter (INVTR) is to reduce the loading of the master clock generator due to the Flip-Flops.