Counters

We have studied earlier that flip flops can be used to store informaton. The state of the flip-flop can be made to change (from 1 to 0 or from 0 to 1) by applying clock pulse at the clock input (CLK) of the flip-flop. The change occurs in flip-flops during the positive going edge of the clock, provided the J and K terminals are held together at 1 state (high state). However, if the clock pulse is applied through a NOT gate (as represented by a bubble) the change of state of the flip-flop will take place during the negative going edge of the clock pulse. The flip flop is obviously T-flip-flop.

By producing such changes of state in (three or) four flip-flops (FF) connected in cascade, the number of clock pulse applied as the input to the first FF can be counted in binary form. Thus the flip-flops can be used to construct digital counters that counts and records the input pulses.

Digital counters are often needed to count events. For example, counting the numbers of bottles moving in a conveyor belt. Electrical pulses corresponding to the event are produced using a transducer and then the pulses acting as clock pulse, are counted using a counter.

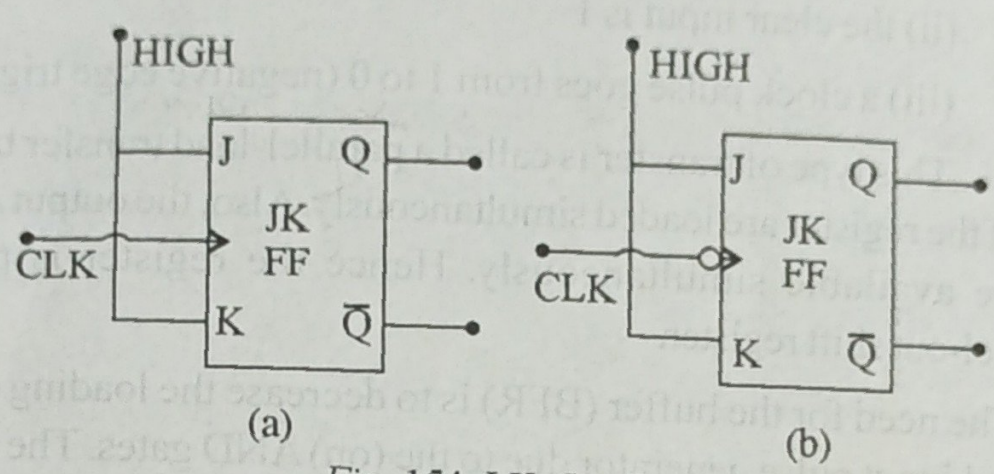

Fig. 154 *J.K. flip flop*

Binary counter (Shift counter)

Counters are composed of flip-flops. A three bit counter consisting of three interconnected flip-flops is shown in the figure 155. The three bit counter can count from decimal 0 to 7 (binary 000 to 111).

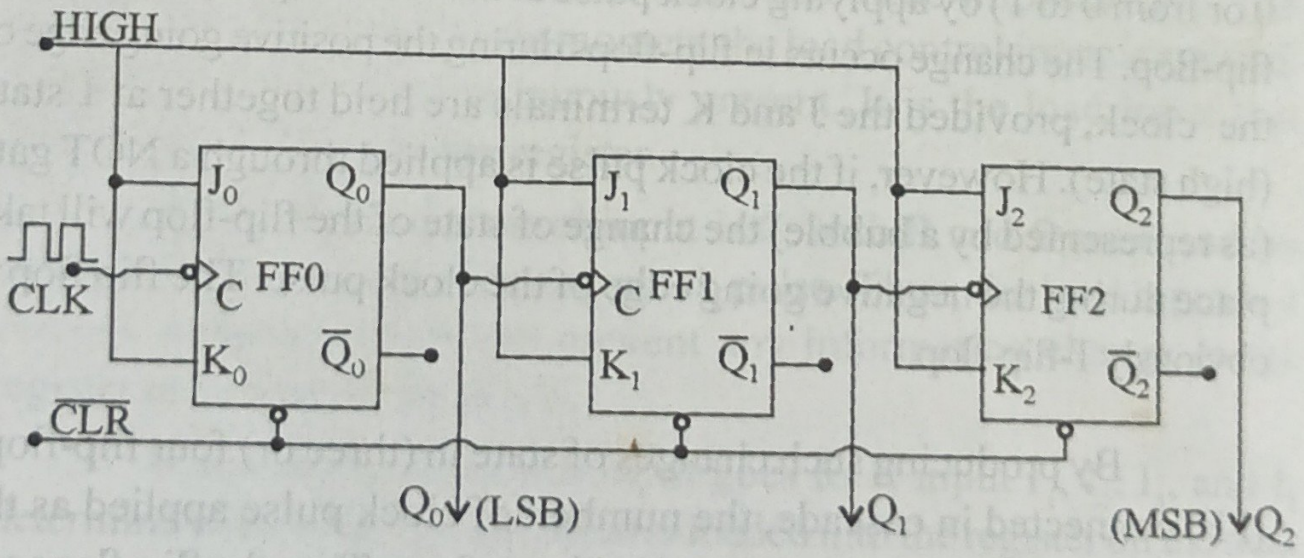

Fig. 155 *Binary counter*

The J and K terminals of each flip-flop are connected together to make J = K = 1 (This is for toggle operation, on arrival of each clock pulse). When J = K = 1, the counter is set ready to count clock pulse. If J = K = 0, the counter stops counting. The pulses to be counted can be applied to the CLK input of the first flip-flop. The Q_0 output is connected to the CLK input of the second flip-flop and so on.

At any instant, the counter output is available at the Q_2, Q_1, Q_0 with Q_0 for the least significant bit (LSB) and Q_2 for the most significant bit (MSB).

Working The flip-flop used are negative-edge trigged type. So, the state of each FF will change at the negative going edge of the clock pulse, applied to the flip-flop as the input.

Initially all the FFs are cleared (reset to 0 state) by applying a clear pulse - on the CLR line. Now $Q_0 = 0$, $Q_1 = 0$, $Q_2 = 0$, i.e., the count is 000. Now the clock input is applied to the first FF. After one clock pulse, at the negative edge of the pulse, the output of FF_0 changes state and Q_0 goes high (1).

$Q_0 = 1$; $Q_1 = 0$; $Q_2 = 0$

The count is $Q_2 Q_1 Q_0 = 001$.

When the second clock pulse arrives, FF0 toggles from 1 to 0 state. The output Q_0 goes from 1 to 0. At this negative going edge of the output pulse, the flip-flop FFI changes stage. i.e., the output Q_1 goes from 0 to 1. At the end of the second clock pulse,

$Q_0 = 0$; $Q_1 = 1$; $Q_2 = 0$ and

the count is $Q_2 Q_1 Q_0 = 010$, which corresponds to count 2. Similarly, when the third clock pulse arrives, $Q_0 = 1$, $Q_1 = 1$ and $Q_2 = 0$ due to change of state in first FF at the falling edge of the clock pulse. The count now is

$Q_2 Q_1 Q_0 = 011$.

This binary output corresponds to count 3.

The counting continues to take place for the arrival of each clock pulse as input. The output states for successive pulses are shown in the table below:

Clock pulse	Otput state of FFs		
	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

At the negative going of the 7th clock pulse, all the FF outputs are high

i.e., $Q_2 Q_1 Q_0 = 111$.

When the eighth clock pulse arrives, it resets FF0, and consequently FF0 resets FF1 and FF1 resets FF2 because of the falling outputs from 1 to 0 state in each flip-flop.

i.e., $Q_2 Q_1 Q_0 = 000$.

The counter is now recycled back to its initial state. After this for each further clock pulse, the circuit starts to count once more.

Thus with a binary having three flip-flop, we can count from decimal 0 to 7 up to 8 pulses (2^3). With four flip flops, the counter can count up to 16 pulses (2^4). This counter is said to have a modulus of

16 ($=2^4$). *The modulus of a counter* is the number of unique combinations of outputs possible with the flip-flops used in the counter. In general, n flips-flops would be required to count number up to 2^n counts. Since the count goes through a binary sequence, the counter is binary counter. The maximum decimal number that can be counted by a counter having n flip-flops is $(2^n - 1)$.

The wave forms of Q_0, Q_1 and Q_2 outputs in the flip-flops are shown relative to the pulse in figure 156. This wave form relationship is called the timing diagram.

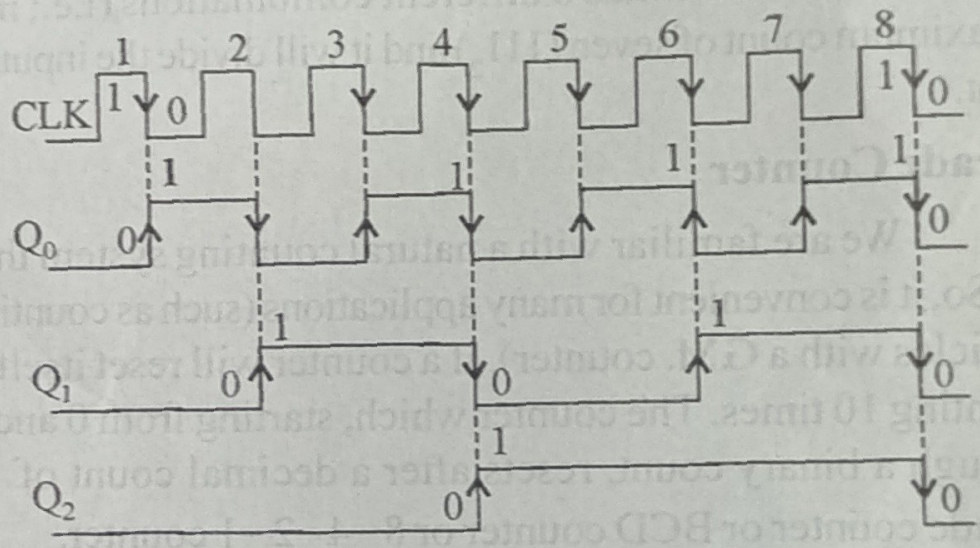

Fig. 156 *Timing diagram of binary counter*

Uses:

1. The binary counter is used to count events. The outputs $Q_2 Q_1 Q_0$ can be decoded by means of a logic gate and the count can be displayed in a seven- segment display in decimal number.

2. The timing diagram reveals that frequancy division has taken place. The frequencies of the output Q_0 is one half of the CLK frequency. i.e., the frequency of the clock pulse is divided by 2 and one FF acts as a "divided-by-two" counter. By suitable modification and sufficient flip-flops, "divided-by 12" and "divided by 60" counter may be constructed. Such counters are used in digital clocks.

Example

Calculate the following for a 3-bit asynchronous binary counter. (a) maximum count (b) modulas (c) division factor.

(a) maximum count $= 2^n - 1$ $n = 3$

$= 2^3 - 1 = 8 - 1 = 7$

(b) modulus $= 2^n = 2^3 = 8$

(c) division factor $= 2^n = 2^3 = 8$

Thus, a 3-bit counter has 8 different combinations (i.e.; modulus = 8), a maximum count of seven (111_2) and it will divide the input frequency by eight.

Decade Counter

We are familiar with a natural counting system that has a base 10. So, it is convenient for many applications (such as counting the charge particles with a G.M. counter), if a counter will reset itself to zero after counting 10 times. The counter which, starting from 0 and progressing through a binary count, resets after a decimal count of 9 is called a decade counter or BCD counter or 8–4–2–1 counter.

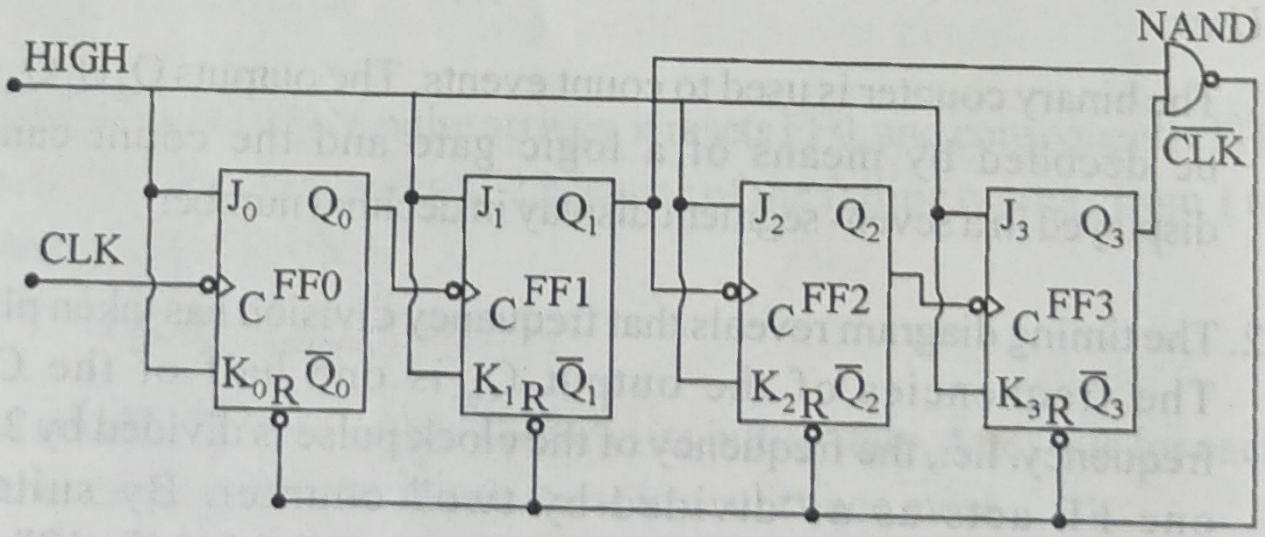

Fig. 157 Decade counter

Regular binary counters do have a maximum modulus i.e., they progress through all their possible states. The maximum possible number of states of a counter is 2^n where n is number of flip-flops in the counter.

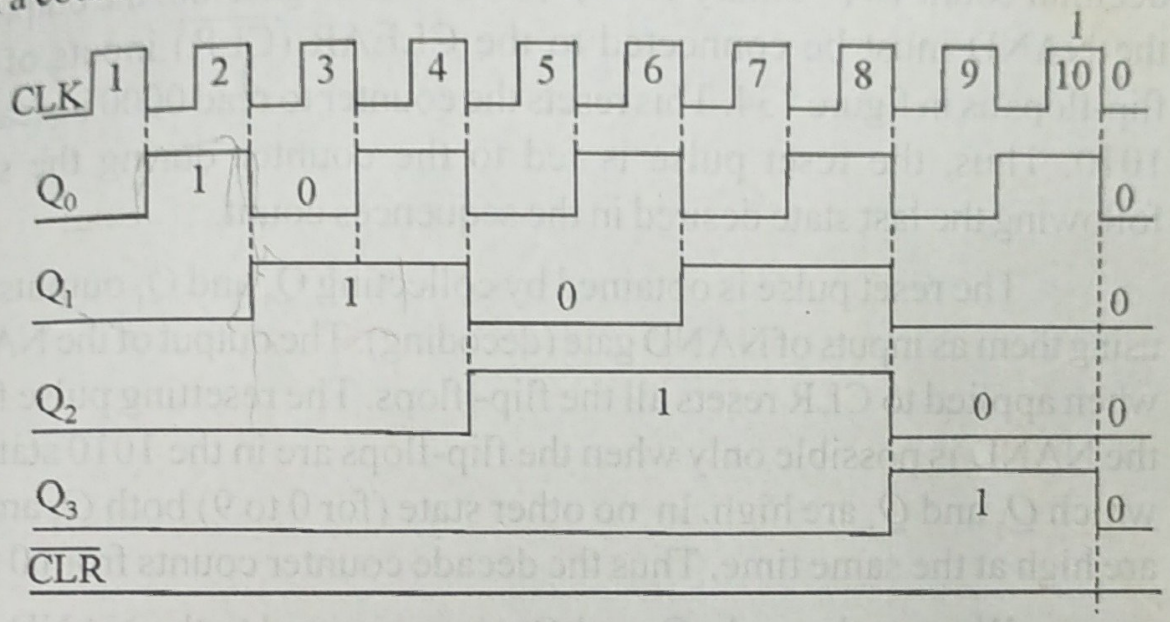

Fig. 158 *Timing diagram of decade counter*

Counters can also be designed to have a desired number of states in their sequence that is less than 2^n. The resulting sequences is called truncated sequence (shortened modulus). A decade counter is a common example of the counter with truncated sequence. The decade counter has ten states in its seqeance.

Construction

To construct a decade counter, it is necessary to force the counter to recycle before going through all of its normal states. The decade counter will recycle back to the 0000 state after the 1001 state.

The decade counter consists of four flip-flops connected in cascade as in figure. (Three flip flops are insufficient since $2^3 = 8$ and the counter with three FF can count only upto 8). The decade counter is required to have 10 states in its sequence. It has to recycle back to the 0000 state after the 1001 state (decimal 9) For every 10 clock pulse, there is one distinct output or recycling. Therefore, the output frequency of a decade counter is $\frac{1}{10}$ of the clock frequency.

Working

To recycle after the count 9 (binary 1001), we have to decode the decimal count 10 (= binary 1010) with a NAND gate and the output of the NAND must be connected to the CLEAR ($\overline{CLR}$) inputs of the flip-flops as in figure 134. This resets the counter to read 0000 instead of 1010. Thus, the reset pulse is fed to the counter during the state following the last state desired in the sequences count.

The reset pulse is obtained by collecting Q_3 and Q_1 outputs and using them as inputs of NAND gate (decoding). The output of the NAND when applied to $\overline{CLR}$ resets all the flip- flops. The resetting pulse from the NAND is possible only when the flip-flops are in the 1010 state, in which Q_3 and Q_1 are high. In no other state (for 0 to 9) both Q_3 and Q_1 are high at the same time. Thus the decade counter counts from 0 to 9.

We note that only Q_1 and Q_3 are connected to the NAND gate inputs. This is an example of partial decoding in which the two unique states are sufficient to decode the count of decimal 10.

The resulting timing diagram is shown in figure. There is a glitch (undesired voltage spike of short duration) on the Q_1 wave form. The reason for this glitch is that Q_1 must first go high before decoding count 10 (decimal) is effected. The counter is in the 1010 state for a short time before it is reset to 0000, thus producing a glitch on Q_1.

Example

Construct a modulo-12 counter using 4 bit binary counter.

$2^3 = 8$; $2^4 = 16$

Hence to construct a modulus-12 counter, 4 flip flops are needed.

The starting condition of the flip-flops is $Q_3Q_2Q_1Q_0 = 0000$.

The modulus of the counter = 12. i.e., the counter has to have sequence from 0 to 11.

The last desirable count is

$Q_3, Q_2, Q_1, Q_0 = 1011$ ..... (for 11).

Recycling has to take place when the count is

$Q_3, Q_2, Q_1, Q_0 = 1100$ ..... (for 12).

The high Q_s are identified now as Q_2 and Q_3. These are the inputs to the decoding NAND gate. The ouput of the NAND is fed as $\overline{CLR}$ to the Q_2 and Q_3 flip-flops. Thus, the counter is designed to count from 0 to 11. On arrival of 12th pulse at the input, the outputs of the FFs are $Q_3, Q_2, Q_1, Q_0 = 0000$. Thus, the counter counts from 1 to 11.

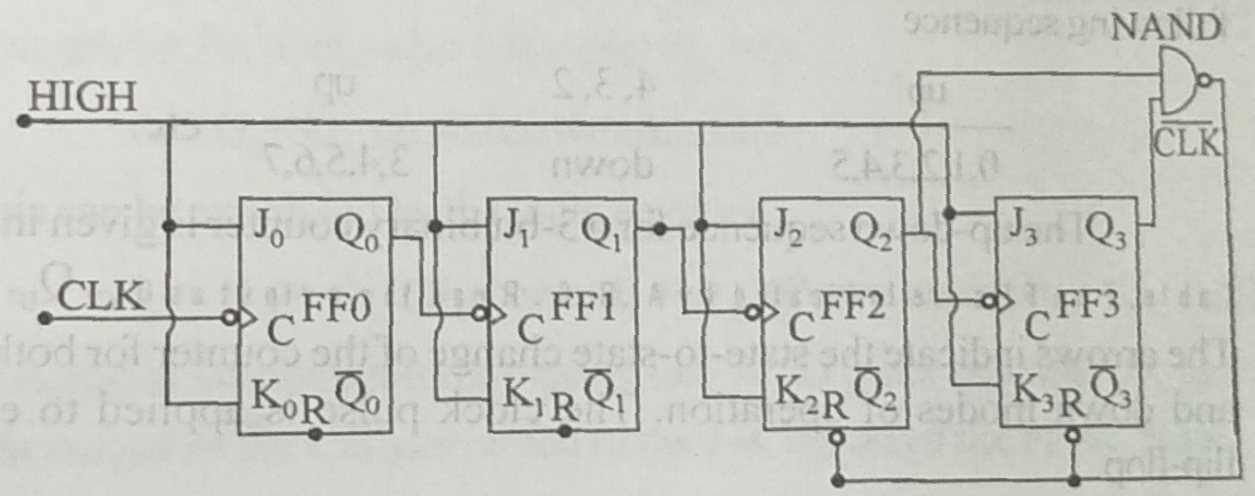

Fig. 159 *Modulo-12 counter*

Problem: Construct a modulo-5 counter.

Up-down counter (Bi-directional counter): Synchronous counter

Up- counter is one which counts in the up-direction, i.e., decimal equivalent of the counter output increases with the arrival of successive clock pulse. At times, we want to count in the reverse direction (as count-down during rocket-launching). The counter in which this happens, and the decimal equivalent of the counter output decreases with the application of successive clock pulse (the counter progresses-in the down direction) is called down-counter or count-down counters.

An up-down counter is one that is capable of counting in either

direction through a certain sequence, depending upon the control input signal. If the control signal is 1, then the counter functions as an up-counter. If the control signal is 0, then the counter functions as a down-counter.

A three-bit-binary up-down counter can advance upwards through a sequance (0,1, 2, 3, 4, 5, 6,7). Then it can be reversed so that it goes through the sequence in the opposite direction (7, 6, 5, 4, 3, 2,1,0).

An up-down counter can be reversed at any point in their sequence (as in the control of various types of computer operations). For example, the 3-bit binary counter can be made to go through the following sequence

$$\underset{0,1,2,3,4,5}{\overset{\text{up}}{\rightarrow}} \quad \underset{\text{down}}{\overset{4,3,2}{\rightarrow}} \quad \underset{3,4,5,6,7}{\overset{\text{up}}{\rightarrow}} \quad \text{etc}$$

The up-down sequence for a 3-bit binary counter is given in the Table. The FFs are indicated by A ,B ,C . Read the output as Q_C, Q_B, Q_A. The arrows indicate the state-to-state change of the counter for both up and down modes of operation. The clock pulse is applied to each flip-flop.

Clock pulse	up	Q_C	Q_B	Q_A	down
0	→ ↓	0	0	0	↑
1	↓	0	0	1	↑
2	↓	0	1	0	↑
3	↓	0	1	1	↑
4	↓	1	0	0	↑
5	↓	1	0	1	↑
6	↓	1	1	0	↑
7	↓	1	1	1	↑ ←

In either mode of operation, the output FF-A toggles (changes state) on each clock pulse at the leading edge of the pulse. So, the J and K inputs of the flip-flop A are kept high.

i.e., $J_A = K_A = 1$ . ..... (1)

For up-counting (when control input is 1), we find from the table that Q_B changes state on the next clock pulse whenever $Q_A = 1$.

i.e., $Q_B = Q_A$. (AND up control.)

Also, for the down - counting (when control input is zero) Q_B changes on the next pulse whenever $Q_A = 0$.

i.e., $Q_B = \overline{Q_A}$ (AND down control.)

This can be expressed by the logic equation:

$J_B = K_B = Q_A$. (Up control) + $\overline{Q_A}$. (Down control) ..... (2)

This can be implemented by using two AND gates and one OR gate. The output of the OR gate is fed to the J-K inputs of the FF-B. Now

$J_B = K_B = 1$

Next, looking at the table, we find that for up - counting, FF - C changes state on the next clock pulse when $Q_A = Q_B = 1$. For the downcount, FF-C changes state on the next clock pulse when $Q_A = Q_B = 0$.

That is $\overline{Q_A} = \overline{Q_B} = 1$

To keep the counter C to change state on application of clock pulse, its J-K inputs are to be 1. To obtain this 1, the logic can be expressed as

$J_C = K_C = Q_A \cdot Q_B$ (up control) + $\overline{Q_A} \cdot \overline{Q_B}$ (down control).

All the above three logic conditions can be implemented as shown in the figure 160.

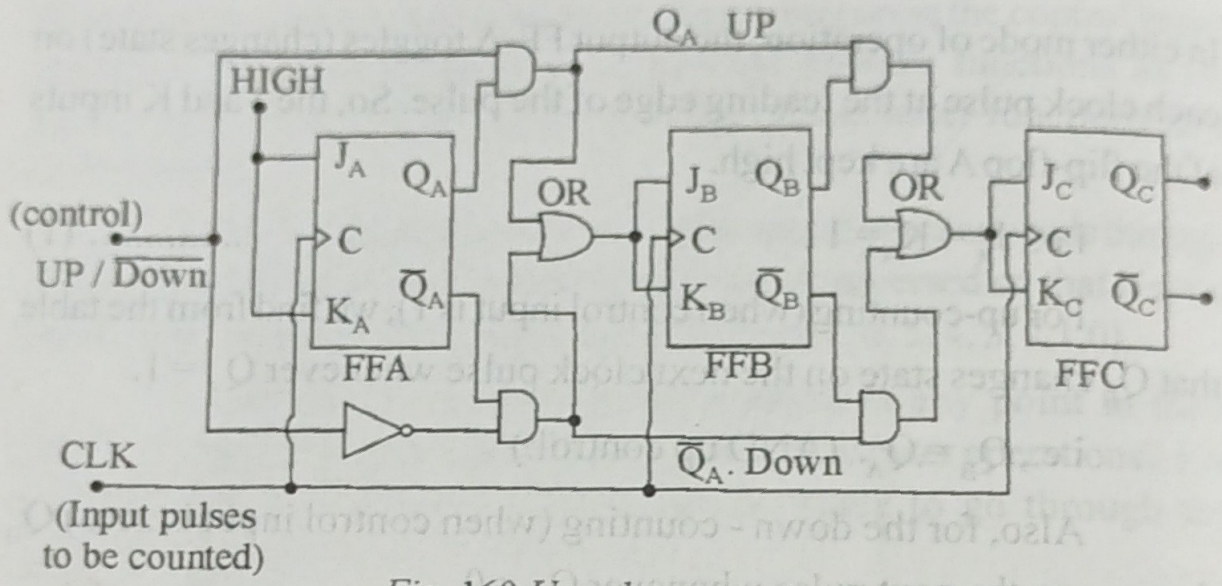

Fig. 160 *Up - down counter*

The clock-pulses applied to CLK line can be counted in the up direction when the control input - UP is high (1). This enables the AND gates in the upper section of the circuit. The pulses can be counted in the down - direction by keeping the control input-UP low (0). The low input is inverted by the NOT gate and it keeps the DOWN inputs as 1. This enables AND gates in the bottom section of the circuit. Thus the circuit functions as a 3-bit up/down counter. The output $Q_C Q_B Q_A$ at any instant gives the count at that instant.

Since all the three FFs change state simultancously on arrival of the clock pulse, the counter is an up-down synchronous counter.

Synchronous and asynchronous counters

Counters are classified into two categories according to the way in which they are clocked. They are synchronous counters and asynchronous counters.

In asynchronous counters (called ripple counters), the flip-flops are inter-connected and the clock pulse is applied to the first flip-flop. Then, each successive flip-flop is clocked by the Q or Q output of the previous flip-flop. Therefore, in asynchronous counter the FFs are not clocked simultaneously. Thus, an asynchronous counter is one, in which an action starts in response to a signal generated by a previous operation, rather than in response to a clock signal.

In synchronous counter (example, up-down counter) the clock input is connected to all FFs and thus simultaneously triggered from the clock input. The propagation delay is limited by the propagation delay of control gate (AND) and a flip flop.

Ripple counter (Asynchronous counter)

We have already studied 3-bit binary counters in which the external clock pulse is applied to the first flip flop. Each successive flip-flop is clocked by Q or $\overline{Q}$ of the previous FF, i.e., the binary counter is an asynchronous counter. Asynchronous counters are commonly known as Ripple counters for the following reason (circuit details and operation are the same as for 3 bit binary counter).

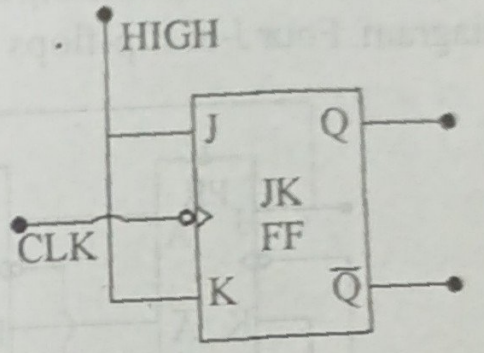

Fig. 161

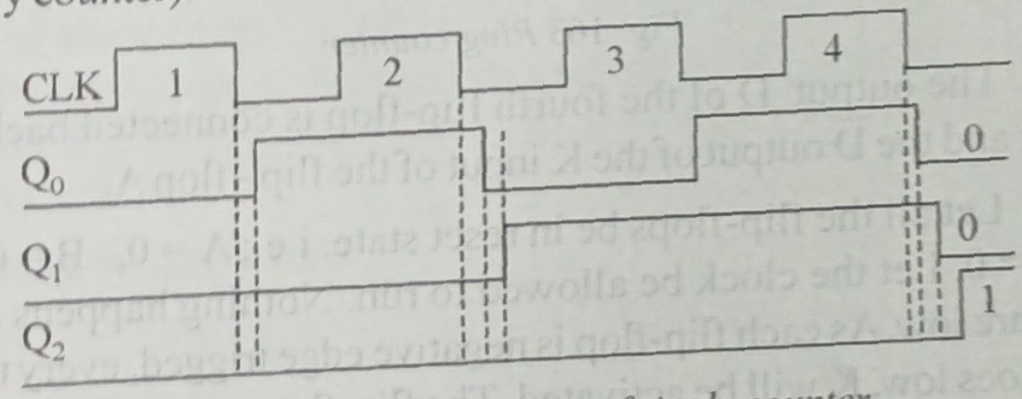

Fig. 162 *Timing diagram of ripple counter*

The effect of the input clock pulse is first felt by FF0. Thus the effect cannot get into FF_1 immediately due to propagation delay time through FF_1 before FF_2 can be triggered. Thus, the effect of an input clock pulse 'ripples' (as water waves) through the counters, taking some time due to propagation delays to reach the last flip-flop. In a 3-bit counter, it makes 3 propagation delay times for the effect of the third clock pulse to ripple through the counter and change Q_3 from low to high.

The cumulative delay limits the rate at which the counter can be clocked. It also creates a decoding problem (glitch) while withdrawing

the numbers for display,. This is a disadvantage. However, ripple counters are less complex and less expensive. (Glitch is an undesired positive or negative pulse appearing at the output logic gate.)