UNIT III

SYNCHRONOUS SEQUENTIAL CIRCUITS

3.1FLIP-FLOPS

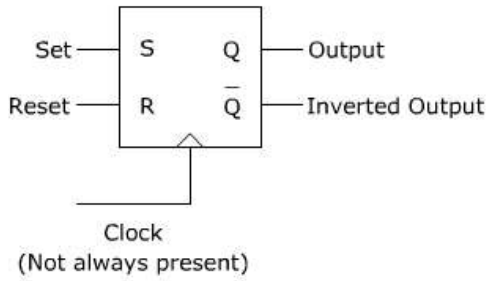
Flip-flops are the first stage in sequential logic design which incorporates memory (storage of previous states).

Flip-flops that we will look at include the following:

- SR type Flip-flop or Set / Reset
- T type Flip-flop or Triggered /Toggle
- D type Flip-flop or Data / Delay
- JK type Flip-flop

3.1.1SR Flip-flop - (Set / Reset)

This type of flip-flop has two inputs: Set and Reset. Two outputs: Q and Q' (Q' being the inverse of Q). The SR flip-flop can also have a clock input for a level driven circuit as opposed to a pulse driven circuit.



The operation of an SR flip-flop is as follows: The Set input will make Q go to 1 i.e. will 'set' the output. The Reset input will make the output Q go to 0 i.e. reset the output. The scenerio of having both Set and Reset at logic 1 is not allowed as this is not a logical pair of inputs.

Knowing the above, we can layout the operating characteristics and the state change table

Input		Circuit Action
S	R	$Q_{(Time\ t+1)}$
0	0	$Q_{(t)}$
1	0	1
0	1	0
1	1	Not Allowed

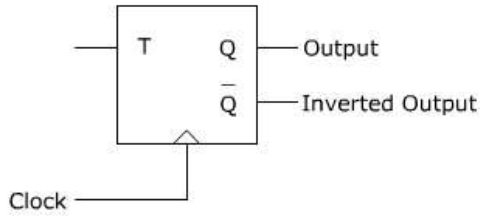
Present State	Next State	Inputs		Map Entry
		S	R	
0	1	1	0	S
1	0	0	1	R
1	1	X	0	s
0	0	0	X	r

There are a few different ways SR flip-flops can be made. They can be pulse driven or clock (and therefore level are used) driven. For the state change diagram above either a pulse or level input can be implied. When using the state change table to describe pulses, a '1' implies a pulse should be applied, where '0' implies that no pulse should exist at this state. For either a pulse driven circuit or a clock driven circuit, the following applies: An 'X' means a

pulse / level may or may not be applied. The reason behind this is because no matter of the input (0 or 1), the output will always go to the same value. It is because of this fact that this is considered a 'don't care' input. Hence 's' and 'r' are 'don't care' sets of 'S' and 'R' respectively ('s' leads to the same output as 'S' and that is why 's' is a subset of 'S').

3.1.2T flip-flop (Triggered / Toggle)

The T type flip-flop is a single input device: T (trigger). Two outputs: Q and Q' (where Q' is the inverse of Q).



The operation of the T type flip-flop is as follows: A '0' input to 'T' will make the next state the same as the present state (i.e. T = 0 present state = 0 therefore next state = 0). However a '1' input to 'T' will change the next state to the inverse of the present state (i.e. T = 1 present state = 0 therefore next state = 1).

Knowing the above, we can now formalize the operating characteristics and the state change table

Input T	Circuit Action $Q_{(Time\ t+1)}$
0	$Q_{(t)}$
1	$Q_{(t)}'$

Present State	Next State	Input T	Map Entry
0	0	0	0
0	1	1	T or 1
1	0	1	T or 1
1	1	0	0

The T type flip-flop is an edge driven device. Therefore you should not associate 1 and 0 with levels, but instead 1 should be considered as a pulse, and 0 as no pulse.

Notice that if a clock signal was tied to T, the output Q would be a clock signal at approximatly half the frequency of T. This property makes the T flip-flop a good candidate for applications such as frequency division.

3.1.3 D type flip-flop (Delay)

The D type flip-flop has one data input 'D' and a clock input. The circuit edge triggers on. the clock input. The flip-flop also has two outputs Q and Q' (where Q' is the reverse of Q).

The operation of the D type flip-flop is as follows: Any input appearing (present state) at the input D, will be produced at the output Q in time T+1 (next state). e.g. if in the present state we have D = 0 and Q = 1, the next state will be D = anything and Q = 0.

Knowing the above, we can now generate the state change table and the operating characteristics.

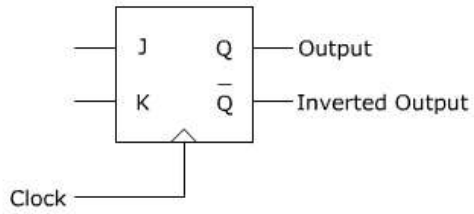
Input D	Circuit Action $Q_{(Time\ t+1)}$
0	0
1	1

Present State	Next State	Input D	Map Entry
0	0	0	0
0	1	1	D or 1
1	0	0	0
1	1	1	D or 1

The operation of the D type delays any input by exactly one clock cycle (given an instantanious response time i.e. a perfect flip-flop). Cascading several D type flip-flops together can produce delaying circuits, possible applications could be for matching time delays in digital television systems.

3.1.4JK flip-flop

The JK type flip-flop consists of two data inputs: J and K, and one clock input. There are again two outputs Q and Q' (where Q' is the reverse of Q).



The JK flip-flop operations are quite complicated to understand by text alone. So here we will simply see the operating characteristics diagram and then discuss it.

Input J	Input K	Circuit Action $Q_{(Time\ t+1)}$
0	0	$Q_{(t)}$
1	0	0
0	1	1
1	1	$Q_{(t)}'$

A. When J=K=0, the current output will carry through to the next state. e.g. Current state Q = Next state Q
B. When J=0 and K=1, the next state output will be put to 0. This happens regardless of the present state output.
C. When J=1 and K=0, the next state output will be asserted (put to 1). This happens regardless of the present state output.
D. When J=K=1, the next state output will be the inverse of the current state output. e.g. Current state Q' = Next state Q.

Knowing the above we can now construct the state change table:

Present State	Next State	Inputs J	Inputs K	Map Entry J	Map Entry K
0	0	0	X	0	X
0	1	1	X	1	X
1	0	X	1	X	1
1	1	X	0	X	0

Lets discuss this state change table with respect to the operating characteristics diagram. There actually exists two operating characteristics that satisfy every possible output combination. This means there should be some 'don't care' terms with each output combination. In the list below we shall see how each of the terms

i. Two conditions exist so that the next state is 0 while the present state is also 0. From the operating characteristics diagram, we can see that condition A and B would both satisfy this scenerio. The common term to make this scenerio true is J=0. We dont care about K, as K=1 or K=0 while J=0 will work. Hence the 'don't care' term is K,

ii. Operating characteristics C and D both satisfy this scenerio. The common term is again J, as the situation is solved by J=1 and either K=0 or K=1, therefore the 'don't care' term is K as shown on the state change table.

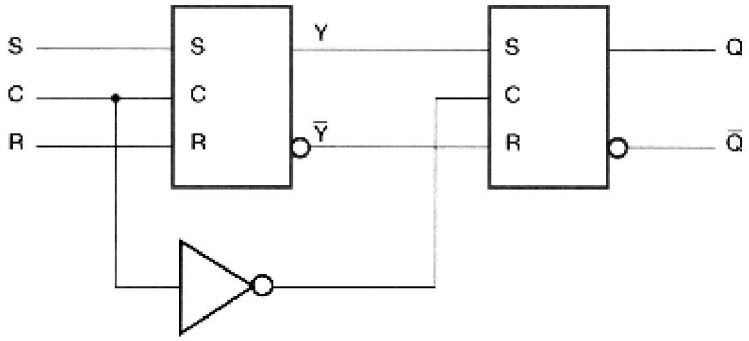
iii. When the output goes from 1 to 0, there are two characteristics that will allow this to happen; B and D. K=1 and J can be equal to 1 or 0. Therefore in this case, J is the 'don't care' term.

iv. When the JK flip-flop remains at logic, it means that either A or C of the four operating characteristics have been applied. K must equal 0 in either case, but J could have been equal to 1 (A) or 0 (C). Because of this, J is the 'don't care' term.

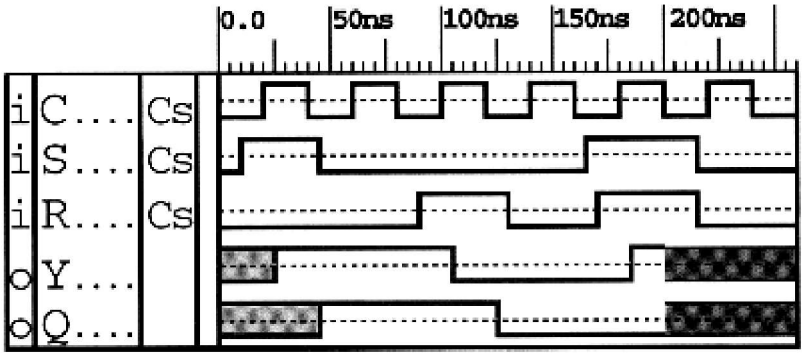
The JK flip-flop can actually be reconfigured so that it can perform the operation of some of the other flip-flops that are discussed above. For example, if the two inputs J and K are tied together, then the output characteristics are fixed to A and D. This precisely matches the characteristics of a T type flip flop. Also to note, because the way a JK is made, you may replace an SR flip-flop with a JK flip-flop without a change in operation. However you cannot replace a JK flip-flop with an SR flip-flop as a S=1 R=1 condition is not allowed, but a J=1 K=1 condition is permitted.

In the case of a latch with control input, the when the control is enabled, the latch is intransparent mode, that is its output changes its state according to the set and resetstates. During this transparent mode, it is still possible for the outputs to become unstable if both set and reset inputs change from asserted to de asserted states simultaneously.

We avoid this problem using master-slave flip-flop arrangement as illustrated below:



Note that it is impossible (well, almost) for both the set and reset inputs to be asserted for the slave latch on the right when its control is enabled because the control of the master latch on the left is disabled at this time.



3.1.5 Flip Flop Conversion

For the conversion of one flip flop to another, a combinational circuit has to be designed first. If a JK Flip Flop is required, the inputs are given to the combinational circuit and the output of the combinational circuit is connected to the inputs of the actual flip flop. Thus, the output of the actual flip flop is the output of the required flip flop. The following flip flop conversions will be explained.

- **SR Flip Flop to JK Flip Flop**
- **JK Flip Flop to SR Flip Flop**
- **SR Flip Flop to D Flip Flop**
- **D Flip Flop to SR Flip Flop**
- **JK Flip Flop to T Flip Flop**
- **JK Flip Flop to D Flip Flop**
- **D Flip Flop to JK Flip Flop**

3.1.5.1 SR Flip Flop to JK Flip Flop

As told earlier, J and K will be given as external inputs to S and R. As shown in the logic diagram below, S and R will be the outputs of the combinational circuit.

The truth tables for the flip flop conversion are given below. The present state is represented by Qp and Qp+1 is the next state to be obtained when the J and K inputs are applied.

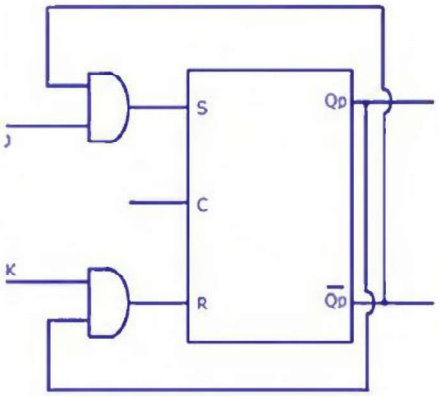
For two inputs J and K, there will be eight possible combinations. For each combination of J, K and Qp, the corresponding Qp+1 states are found. Q_{p+1} simply suggests the future values to be obtained by the JK flip flop after the value of Qp. The table is then completed by writing the values of S and R required getting each Q_{p+1} from the corresponding Qp. That is, the values of S and R that are required to change the state of the flip flop from Qp to Qp+1 are written.

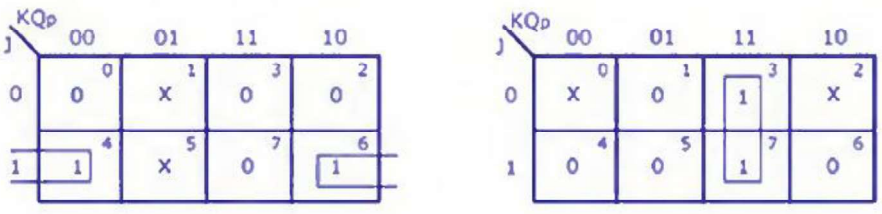
S-R Flip Flop to J-K Flip Flop

Conversion Table

J-K Inputs		Outputs		S-R Inputs	
J	K	Qp	Qp+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

Logic Diagram





$S = \overline{J}Qp$ $R = KQp$

K-Map

3.1.5.2 JK Flip Flop to SR Flip Flop

This will be the reverse process of the above explained conversion. S and R will be the external inputs to J and K. As shown in the logic diagram below, J and K will be the outputs of the combinational circuit. Thus, the values of J and K have to be obtained in terms of S, R and Qp. The logic diagram is shown below.

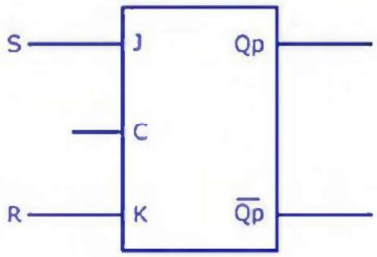
A conversion table is to be written using S, R, Qp, Qp+1, J and K. For two inputs, S and R, eight combinations are made. For each combination, the corresponding Qp+1 outputs are found ut. The outputs for the combinations of S=1 and R=1 are not permitted for an SR flip flop. Thus the outputs are considered invalid and the J and K values are taken as "don't cares".

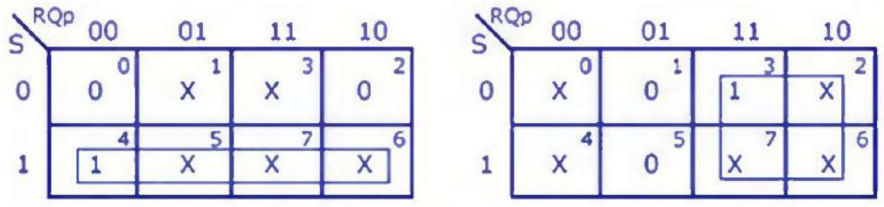
J-K Flip Flop to S-R Flip Flop

Conversion Table

S-R Inputs		Outputs		J-K Inputs	
S	R	Qp	Qp+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	Invalid		Dont care	
1	1	Invalid		Dont care	

Logic Diagram





J=S K-maps K=R

3.1.5.3 SR Flip Flop to D Flip Flop

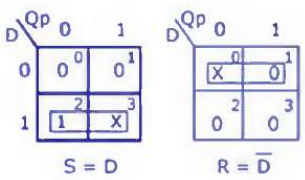
As shown in the figure, S and R are the actual inputs of the flip flop and D is the external input of the flip flop. The four combinations, the logic diagram, conversion table, and the K-map for S and R in terms of D and Qp are shown below.

S-R Flip Flop to D Flip Flop

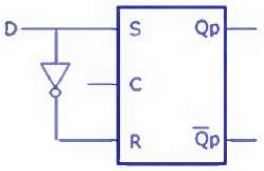
Conversion Table

D Input	Outputs Qp	Outputs Qp+1	S-R Inputs S	S-R Inputs R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

K-maps



Logic Diagram



3.1.5.4 D Flip Flop to SR Flip Flop

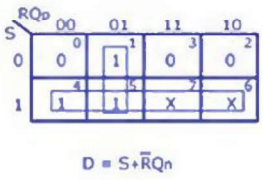
D is the actual input of the flip flop and S and R are the external inputs. Eight possible combinations are achieved from the external inputs S, R and Qp. But, since the combination of S=1 and R=1 are invalid, the values of Qp+1 and D are considered as "don't cares". The logic diagram showing the conversion from D to SR, and the K-map for D in terms of S, R and Qp are shown below.

D Flip Flop to S-R Flip Flop

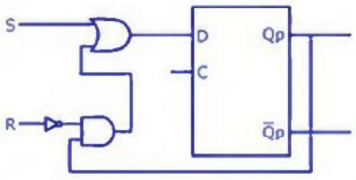
Conversion Table

S-R Inputs S	S-R Inputs R	Outputs Qp	Outputs Qp+1	D Input
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	Invalid		Dont care
1	1	Invalid		Dont care

K-map



Logic Diagram



3.1.5.5 JK Flip Flop to T Flip Flop

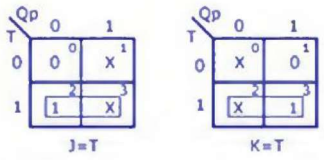
J and K are the actual inputs of the flip flop and T is taken as the external input for conversion. Four combinations are produced with T and Qp. J and K are expressed in terms of T and Qp. The conversion table, K-maps, and the logic diagram are given below.

J-K Flip Flop to T Flip Flop

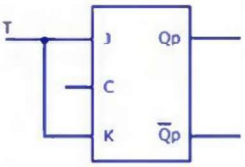
Conversion Table

T Input	Outputs Qp	Outputs Qp+1	J-K Inputs J	J-K Inputs K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

K-maps



Logic Diagram



3.1.5.6 JK Flip Flop to D Flip Flop

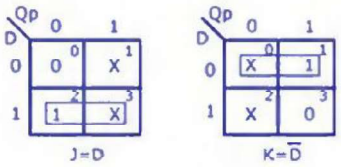
D is the external input and J and K are the actual inputs of the flip flop. D and Qp make four combinations. J and K are expressed in terms of D and Qp. The four combination conversion table, the K-maps for J and K in terms of D and Qp, and the logic diagram showing the conversion from JK to D are given below.

J-K Flip Flop to D Flip Flop

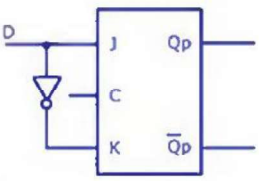
Conversion Table

D Input	Outputs Qp	Outputs Qp+1	J-K Inputs J	J-K Inputs K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0

K-maps



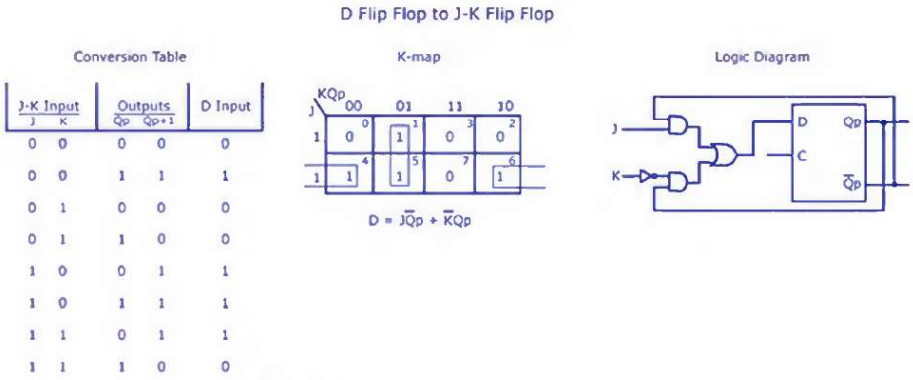
Logic Diagram



3.1.5.7 D Flip Flop to JK Flip Flop

In this conversion, D is the actual input to the flip flop and J and K are the external inputs. J, K and Qp make eight possible combinations, as shown in the conversion table below. D is expressed in terms of J, K and Qp.

The conversion table, the K-map for D in terms of J, K and Qp and the logic diagram showing the conversion from D to JK are given in the figure below.



3.2 COUNTERS

Counters and registers belong to the category of MSI sequential logic circuits. They have similar architecture, as both counters and registers comprise a cascaded arrangement of more than one flipflop with or without combinational logic devices. Both constitute very important building blocks of sequential logic, and different types of counter and register available in integrated circuit (IC) form are used in a wide range of digital systems. While counters are mainly used in counting applications, where they either measure the time interval between two unknown time instants or measure the frequency of a given signal, registers are primarily used for the temporary storage of data present at the output of a digital circuit before they are fed to another digital circuit. We are all familiar with the role of different types of register used inside a microprocessor, and also their use in microprocessor-based applications. Because of the very nature of operation of registers, they form the basis of a very important class of counters called shift counters. In this chapter, we will discuss different types of counter and register as regards their operational basics, design methodology and application-relevant aspects. Design aspectshave been adequately illustrated with the help of a large number of solved examples. A comprehensive functional index of a large number of integrated circuit counters and registers is given towards the end of the chapter.

3.2.1 Synchronous Counters

Synchronous counters are simple state machines made out of flip flops and logic gates. They have two parts, a register made out of flip flops and a decoder made out of logic gates. A register is a simple group of flip flops that are all clocked at the same time. In this way they can hold the counters output value until the next clock cycle. The decoder, decodes the current

count and generates the correct value for the next count to the flop flops. For example in a simple up counter the decoder would always output the current count plus one. The major advantage of Synchronous Counters is that all the bits of their output change at the same time.In a synchronous counter, also known as a parallel counter, all the flip-flops in the counter change

state at the same time in synchronism with the input clock signal. The clock signal in this case is simultaneously applied to the clock inputs of all the flip-flops. The delay involved in this case is equal to the propagation delay of one flip-flop only, irrespective of the number of flip-flops used to construct the counter. In other words, the delay is independent of the size of the counter.

3.2.1.1Binary 4-bit Synchronous Up Counter

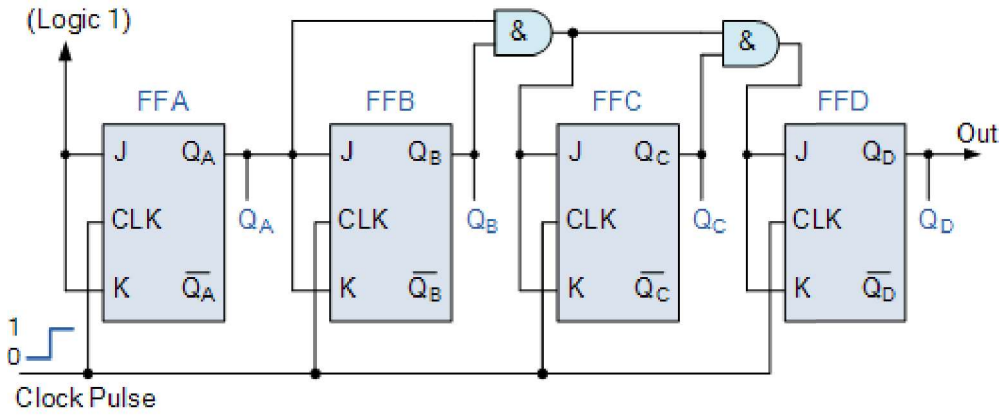


Figure 3.1 4-bit Synchronous Up Counter

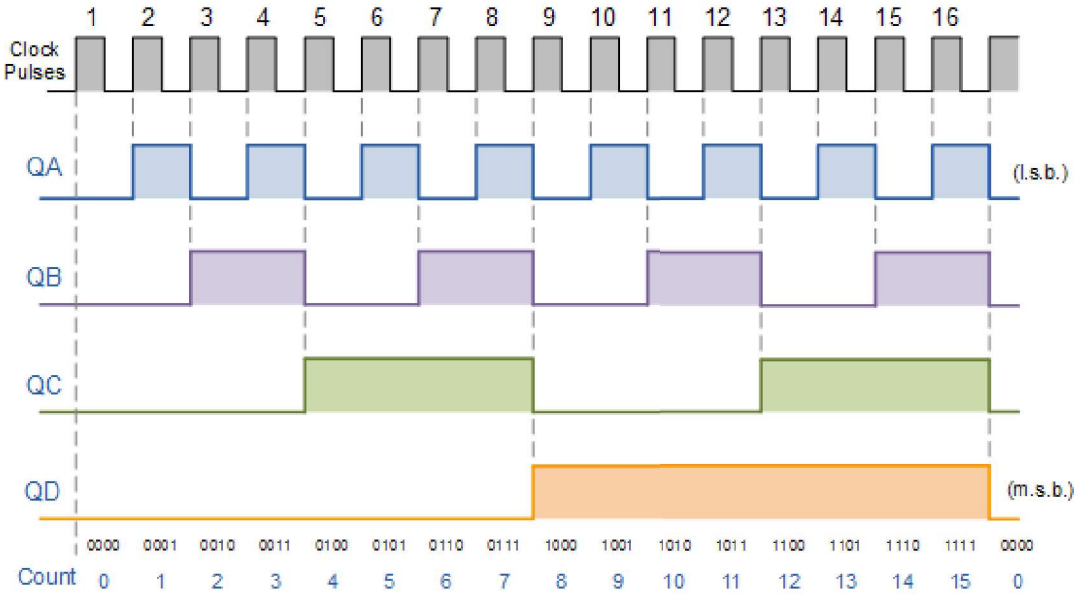
It can be seen above, that the external clock pulses (pulses to be counted) are fed directly to each of the J-K flip-flops in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop FFA (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.

The J and K inputs of flip-flop FFB are connected directly to the output Q_A of flip-flop FFA, but the J andK inputs of flip-flops FFC and FFD are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage. These additional AND gates generate the required logic for the JK inputs of the next stage.

If we enable each JK flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time.

Then as there is no inherent propagation delay in synchronous counters, because all the counter stages are triggered in parallel at the same time, the maximum operating frequency of this type of frequency counter is much higher than that for a similar asynchronous counter circuit.

4-bit Synchronous Counter Waveform Timing Diagram.



Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 (0000) to 15 (1111). Therefore, this type of counter is also known as a4-bit Synchronous Up Counter.

However, we can easily construct a 4-bit Synchronous Down Counter by connecting the AND gates to the Q output of the flip-flops as shown to produce a waveform timing diagram the reverse of the above. Here the counter starts with all of its outputs HIGH (1111) and it counts down on the application of each clock pulse to zero, (0000) before repeating again.

3.2.1.2Binary 4-bit Synchronous Down Counter

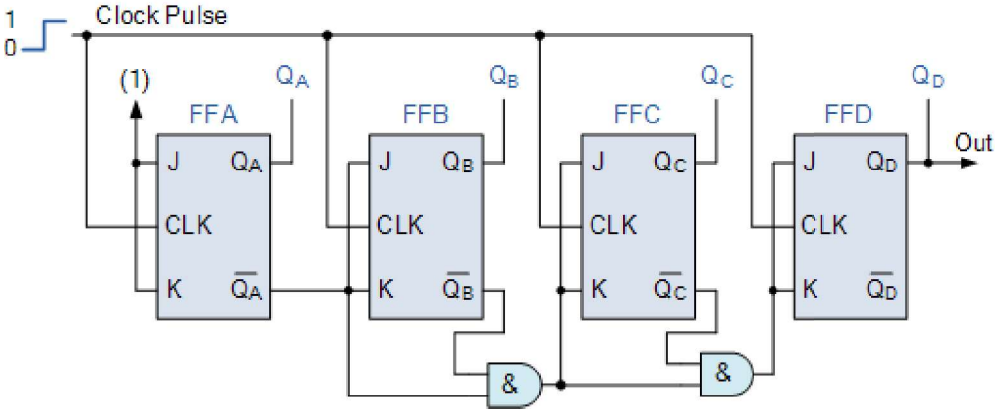


Figure 3.2 4-bit Synchronous Down Counter

As synchronous counters are formed by connecting flip-flops together and any number of flip-flops can be connected or "cascaded" together to form a "divide-by-n" binary counter, the modulo's or "MOD" number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to 2^n-1 can be built along with truncated sequences. All we need to increase the MOD count of an up or down synchronous counter is an additional flip-flop and AND gate across it.

3.2.1.3Decade 4-bit Synchronous Counter

A 4-bit decade synchronous counter can also be built using synchronous binary counters to produce a count sequence from 0 to 9. A standard binary counter can be converted to a decade (decimal 10) counter with the aid of some additional logic to implement the desired state sequence. After reaching the count of "1001", the counter recycles back to "0000". We now have a decade or Modulo-10counter.

Decade 4-bit Synchronous Counter

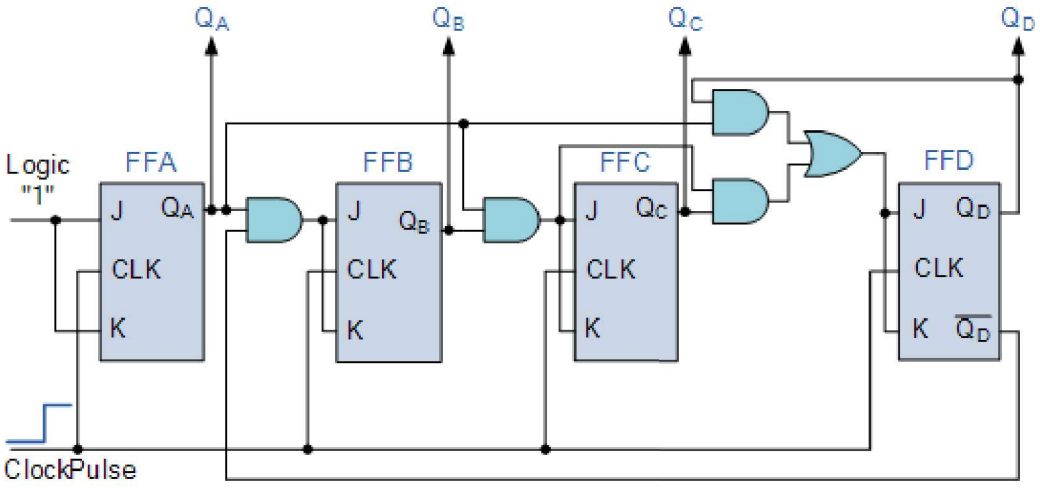


Figure 3.3 Decade 4-bit Synchronous Counter

The additional AND gates detect when the counting sequence reaches “1001”, (Binary 10) and causes flip-flop FF3 to toggle on the next clock pulse. Flip-flop FF0 toggles on every clock pulse. Thus, the count is reset and starts over again at “0000” producing a synchronous decade counter.

We could quite easily re-arrange the additional AND gates in the above counter circuit to produce other count numbers such as a Mod-12 counter which counts 12 states from”0000″ to “1011” (0 to 11) and then repeats making them suitable for clocks, etc.

This is achieved by forcing the counter to reset itself to zero at a pre-determined value producing a type of asynchronous counter that has truncated sequences. Then an n-bit counter that counts up to its maximum modulus (2n) is called a full sequence counter and a n-bit counter whose modulus is less than the maximum possible is called a **truncated counter**.

But why would we want to create an asynchronous truncated counter that is not a MOD-4, MOD-8, or some other modulus that is equal to the power of two. The answer is that we can by using combinational logic to take advantage of the asynchronous inputs on the flip-flop. If we take the modulo-16 asynchronous counter and modified it with additional logic gates it can be made to give a decade (divide-by-10) counter output for use in standard decimal counting and arithmetic circuits.

Such counters are generally referred to as Decade Counters. A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010 and to do this we need to feed this condition back to the reset input. A counter with a count sequence from binary “0000” (BCD = “0”) through to “1001” (BCD = “9”) is generally referred to as a BCD binary-coded-decimal counter because its ten state sequence is that of a BCD code but binary decade counters are more common.

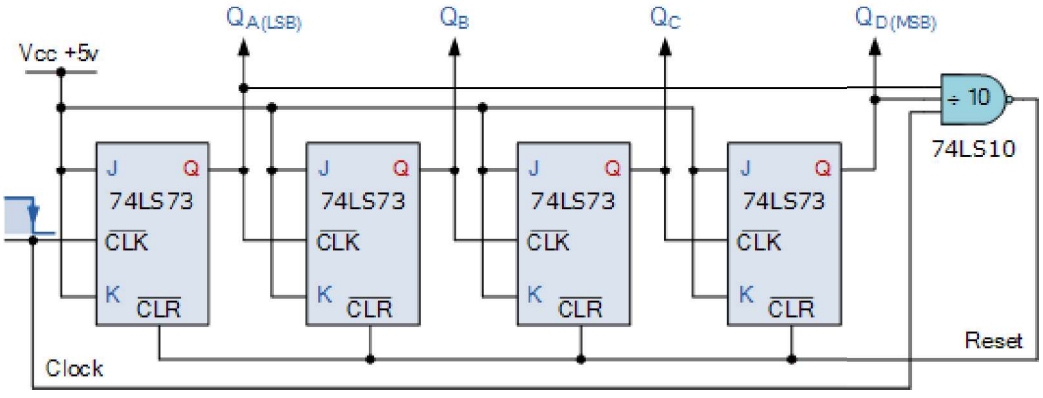


Figure 3.4 Asynchronous Decade Counter

This type of asynchronous counter counts upwards on each trailing edge of the input clock signal starting from 0000 until it reaches an output 1001 (decimal 9). Both outputs QA and QD are now equal to logic “1”. One the application of the next clock pulse, the output from the 74LS10 NAND gate changes state from logic “1” to a logic “0” level.

As the output of the NAND gate is connected to the CLEAR (CLR) inputs of all the 74LS73 J-K Flip-flops, this signal causes all of the Q outputs to be reset back to binary 0000 on the count of 10. As outputs QA and QD are now both equal to logic “0” as the flip-flop’s have just been reset, the output of the NAND gate returns back to a logic level “1” and the counter restarts again from 0000. We now have a decade or Modulo-10 up-counter.

3.2.2 Ripple (Asynchronous) Counter

Ripple counters are the simplest type of counters. They are nothing more than toggle flip flops connected in a chain to divide each others output frequency by two. The result is a binary count. They are called ripple counters because the new count ripples through them. The major disadvantage of ripple counters is that because of new count "rippling" through the flip flops all the bits of the count arrive at different times. A ripple counter is a cascaded arrangement of flip-flops where the output of one flip-flop drives the clock input of the following flip-flop. The number of flip-flops in the cascaded arrangement depends upon the number of different logic states that it goes through before it repeats the sequence, a parameter known as the modulus of the counter. In a ripple counter, also called an asynchronous counter or a serial counter, the clock input is applied only to the first flip-flop, also called the input flip-flop, in the cascaded arrangement. The clock input to any subsequent flip-flop comes from the output of its immediately preceding flip-flop. For instance, the output of the first flip-flop acts as the clock input to the second flip-flop, the output of the second flip-flop feeds the clock input of the third flip-flop and so on. In general, in an arrangement of n

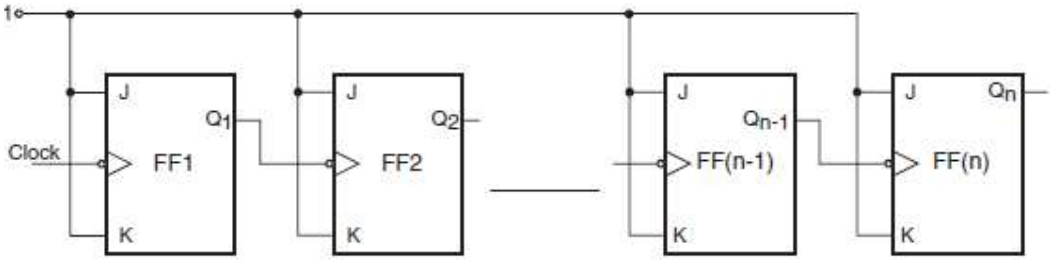


Figure 3.5Generalized block schematic of n-bit binary ripple counter.

flip-flops, the clock input to the nth flip-flop comes from the output of the (n−1)th flip-flop for n>1.Figure 11.1 shows the generalized block schematic arrangement of an n-bit binary ripple counter. As a natural consequence of this, not all flip-flops change state at the same time. The second flip-flop can change state only after the output of the first flip-flop has changed its state. That is, the second flip-flop would change state a certain time delay after the occurrence of the input clock pulse owing to the fact that it gets its own clock input from the output of the first flip-flop and not from the input clock. This time delay here equals the sum of propagation delays of two flip-flops, the first and the second flip-flops. In general, the nth flip-flop will change state only after a delay equal ton times the propagation delay of one flip-flop. The term ‘ripple counter’ comes from the mode in which the clock information ripples through the counter. It is also called an ‘asynchronous counter’ as different flip-flops comprising the counter do not change state in synchronization with the input clock.

In a counter like this, after the occurrence of each clock input pulse, the counter has to wait for a time period equal to the sum of propagation delays of all flip-flops before the next clock pulse can be applied. The propagation delay of each flip-flop, of course, will depend upon the logic family to which it belongs.

3.2.3 Modulus of a Counter

The modulus (MOD number) of a counter is the number of different logic states it goes through before it comes back to the initial state to repeat the count sequence. An n-bit counter that counts through all its natural states and does not skip any of the states has a modulus of 2n. We can see that such counters have a modulus that is an integral power of 2, that is, 2, 4, 8, 16 and so on. These can be modified with the help of additional combinational logic to get a modulus of less than 2n.To determine the number of flip-flops required to build a counter having a given modulus, identify the smallest integer m that is either equal to or greater than the desired modulus and is also equal to an integral power of 2. For instance, if the desired modulus is 10, which is the case in a decade counter, the smallest integer greater than or equal to 10 and which is also an integral power of 2 is16. The number of flip-flops in this case would be 4, as 16 = 24. On the same lines, the number of flip-flops required to construct counters with MOD numbers of 3, 6, 14, 28 and 63 would be 2, 3, 4, 5and 6 respectively. In general, the arrangement of a minimum number of N flip-flops can be used to construct any counter with a modulus given by the equation

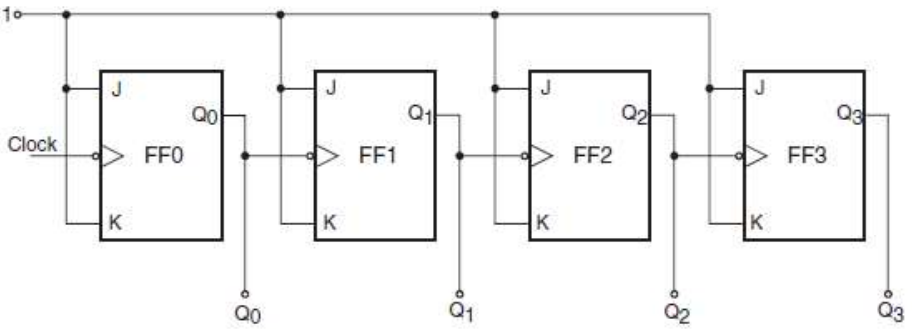
$$2N-1+1 \leq \text{modulus} \leq 2N$$

3.2.4 Binary Ripple Counter

The operation of a binary ripple counter can be best explained with the help of a typical counter of this type. Figure 11.2(a) shows a four-bit ripple counter implemented with negative edge-triggered J-K flip-flops wired as toggle flip-flops. The output of the first flip-flop feeds the clock input of the second, and the output of the second flip-flop feeds the clock input of the third, the output of which in turn feeds the clock input of the fourth flip-flop. The outputs of the four flip-flops are designated asQ0 (LSB flip-flop), Q1, Q2 and Q3 (MSB flip-flop). Figure 11.2(b) shows the waveforms appearing atQ0, Q1, Q2 and Q3 outputs as the clock signal goes through successive cycles of trigger pulses. The counter functions as follows.

Let us assume that all the flip-flops are initially cleared to the ‘0’ state. On HIGH-to-LOW transition of the first clock pulse, Q0 goes from ‘0’ to ‘1’ owing to the toggling action. As the flip-flops use dare negative edge-triggered ones, the ‘0’ to ‘1’ transition of Q0 does not trigger flip-flop FF1. FF1,along with FF2 and FF3, remains in its ‘0’ state. So, on the occurrence of the first negative-going clock transition, Q0= 1, Q1= 0, Q2= 0 and Q3= 0.

On the HIGH-to-LOW transition of the second clock pulse, Q0 toggles again. That is, it goes from‘1’ to ‘0’. This ‘1’ to ‘0’ transition at the Q0 output triggers FF1, the output Q1 of which goes from ‘0’



(a)

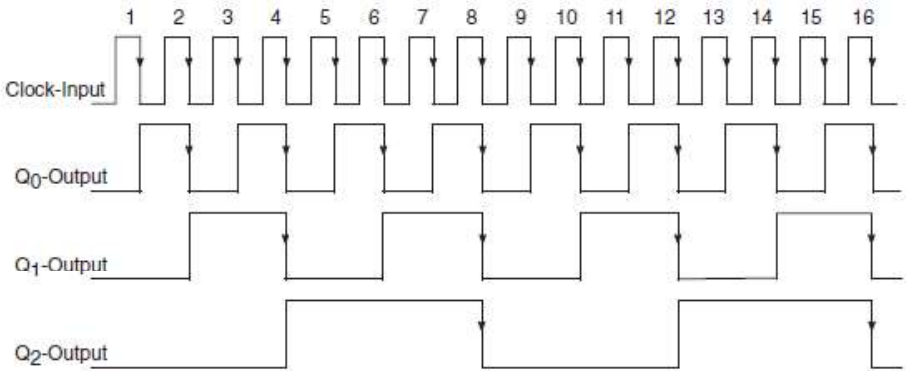


Figure 3.6 Four-bit binary ripple counter.

to '1'. The Q2 and Q3 outputs remain unaffected. Therefore, immediately after the occurrence of these cond HIGH-to-LOW transition of the clock signal, Q0= 0, Q1= 1, Q2= 0 and Q3= 0. On similar lines, we can explain the logic status of Q0, Q1, Q2 and Q3 outputs immediately after subsequent clock transitions.

Thus, we see that the counter goes through 16 distinct states from 0000 to 1111 and then, on the occurrence of the desired transition of the sixteenth clock pulse, it resets to the original state of 0000from where it had started. In general, if we had N flip-flops, we could count up to 2N pulses before the counter resets to the initial state. We can also see from the Q0, Q1, Q2 and Q3 waveforms, as shown in Fig.3.6 that the frequencies of the Q0, Q1, Q2 and Q3 waveforms are f/2,f/4, f/8 and f/16respectively. Here, f is the frequency of the clock input. This implies that a counter of this type can be used as a divide-by-2N circuit, where N is the number of flip-flops in the counter chain. In fact, such a counter provides frequency-divided outputs of f/2N , f/2N−1, f/2N−2, f/2N−3, _ _ _ , f/2 at the outputs of the Nth, (N− 1)th, (N − 2)th, (N− 3)th, _ _ _ , first flip-flops. In the case of a four-bit counter of the type shown in Fig. 3.6 outputs are available at f/2 from the Q0 output, at f/4 from the Q1 output, at f/8 from the Q2 output and at f/16 from the Q3 output. It may be noted that frequency division is one of the major applications of counters.

3.2.5 UP/DOWN Counters

Counters are also available in integrated circuit form as UP/DOWN counters, which can be made to operate as either UP or DOWN counters. An UP counter is one that counts upwards or in the forward direction by one LSB every time it is clocked. A four-bit binary UP counter will count as 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011,1100, 1101, 1110, 1111, 0000, 0001, _ _ _ and so on. A DOWN counter counts in the reverse direction or downwards by one LSB every time it is clocked. The four-bit binary DOWN counter will count as0000, 1111, 1110, 1101, 1100, 1011, 1010, 1001, 1000, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000, 1111, _ _ _ and so on. Some counter ICs have separate clock inputs for UP and DOWN counts, while others have a single clock input and an UP/DOWN control pin. The logic status of this control pin decides the counting mode. As an example, ICs 74190 and 74191 are four-bit UP/DOWN counters in the TTL family with a single clock input and an UP/DOWN control pin. While IC 74190 is a BCD decade counter, IC74191 is a binary counter. Also, ICs 74192 and 74193 are four-bit UP/DOWN counters in the TTL family, with separate clock input terminals for UP and DOWN counts. While IC 74192 is a BCD decade counter, IC 74193 is a binary counter.

Figure 3.8shows a three-bit binary UP/DOWN counter. This is only one possible logic arrangement. As we can see, the counter counts upwards when UP control is logic '1' and DOWN

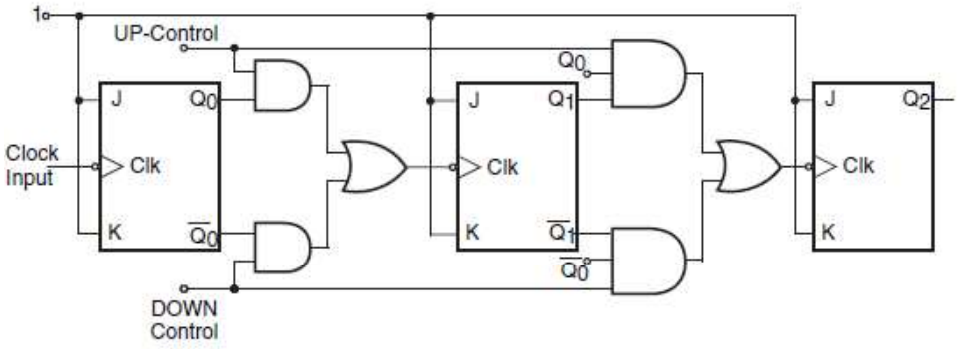


Figure 3.7 4 bit UP/DOWN Counter

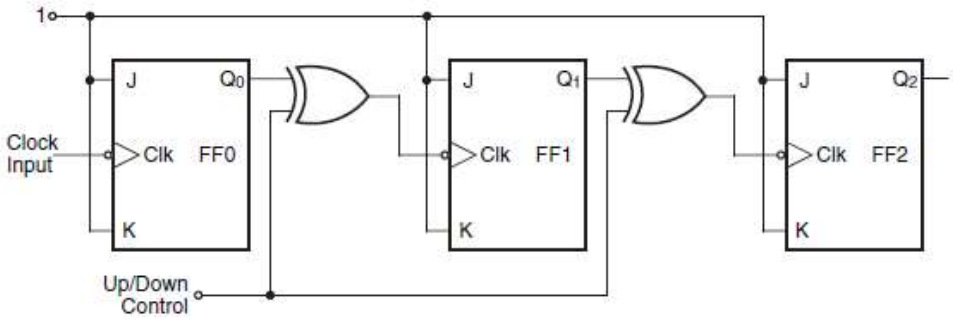


Figure3.8 3 bit UP/DOWN counter with a common clock input

control is logic ‘0’. In this case the clock input of each flip-flop other than the LSB flip-flop is fed from the normal output of the immediately preceding flip-flop. The counter counts downwards when the UP control input is logic ‘0’ and DOWN control is logic ‘1’. In this case, the clock input of each flip-flop other than the LSB flip-flop is fed from the complemented output of the immediately preceding flip-flop. Figure 3.8 shows another possible configuration for a three-bit binary ripple UP/DOWN counter. It has a common control input. When this input is in logic ‘1’ state the counter counts downwards, and when it is in logic ‘0’ state it counts upwards.

3.2.5 Designing Counters with Arbitrary Sequences

So far we have discussed different types of synchronous and asynchronous counters. A large variety of synchronous and asynchronous counters are available in IC form, and some of these have been mentioned and discussed in the previous sections. The counters discussed hitherto count in either the normal binary sequence with a modulus of 2N or with slightly altered binary sequences where one or more of the states are skipped. The latter type of counter has a modulus of less than 2N , N being the number of flip-flops used. Nevertheless, even these counters have a sequence that is either upwards or downwards and not arbitrary. There are applications where a counter is required to follow a sequence that is arbitrary and not binary. As an example, an MOD-10 counter may be required to follow the sequence 0000, 0010, 0101, 0001, 0111, 0011, 0100, 1010, 1000, 1111, 0000, 0010 and so on. In such cases, the simple and seemingly obvious feedback arrangement with a single NAND gate discussed in the earlier sections of this chapter for designing counters with a modulus of less than 2N cannot be used.

There are several techniques for designing counters that follow a given arbitrary sequence. In the present section, we will discuss in detail a commonly used technique for designing synchronous counters using J-K flip-flops or D flip-flops. The design of the counters basically involves designing a suitable combinational logic circuit that takes its inputs from the normal and complemented outputs of the flip-flops used and decodes the different states of the counter to generate the correct logic states for the inputs of the flip-flops such as J, K, D, etc. But before we illustrate the design procedure with the help of an example, we will explain what we mean by the excitation table of a flip-flop and the state transition diagram of a counter. An excitation table in fact can be drawn for any sequential logic circuit, but, once we understand what it is in the case of a flip-flop, which is the basic building block of sequential logic, it would be much easier for us to draw the same for more complex sequential circuits such as counters, etc.

3.2.5.1 Excitation Table of a Flip-Flop

The excitation table is similar to the characteristic table that we discussed in the previous chapter on flip-flops. The excitation table lists the present state, the desired next state and the flip-flop inputs(J, K, D, etc.) required to achieve that. if the output is in the logic ‘0’ state and it is desired that it goes to the logic ‘1’ state on occurrence of the clock pulse, the J input must be in the logic ‘1’ state and the K input can be either in the logic ‘0’ or logic ‘1’ state. This is true as, for a ‘0’ to ‘1’ transition, there are two possible input conditions that can achieve this. These are J = 1,K = 0 (SET mode) and J = K = 1 (toggle mode), which further leads to J = 1_ K = X (either 0or 1). The other entries of the excitation table can be explained on similar lines. In the case of a D flip-flop, the D input is the same as the logic status of the desired next state. This is true as, in the case of a D flip-flop, the D input is transferred to the output on the occurrence of the clock pulse, irrespective of the present logic status of the Q output.

3.2.5.2State Transition Diagram

The state transition diagram is a graphical representation of different states of a given sequential circuit and the sequence in which these states occur in response to a clock input. Different states are represented by circles, and the arrows joining them indicate the sequence in which different states occur. As an example, Fig. 3.8 shows the state transition diagram of an MOD-8 binary counter.

3.2.5.3 Design Procedure

We will illustrate the design procedure with the help of an example. We will do this for an MOD-6synchronous counter design, which follows the count sequence 000, 010, 011, 001, 100, 110, 000,010, _ _ _ :

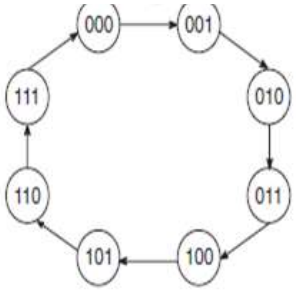


Figure 3.8 State transition diagram for an MOD 8 binary counter

1. Determine the number of flip-flops required for the purpose. Identify the undesired states. In the present case, the number of flip-flops required is 3 and the undesired states are 101 and 111

2. Draw the state transition diagram showing all possible states including the ones that are not desired. The undesired states should be depicted to be transiting to any of the desired states. We have chosen the 000 state for this purpose. It is important to include the undesired states to ensure that, if the counter accidentally gets into any of these undesired states owing to noise or power-up, the counter will go to a desired state to resume the correct sequence on application of the next clock pulse.

Figure 3.9 shows the state transition diagram

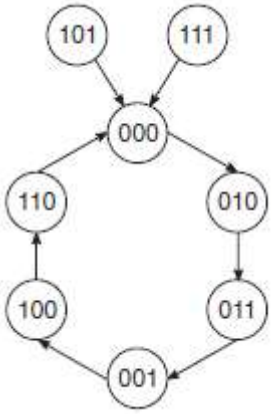


Figure 3.9 state transition diagram

Draw the excitation table for the counter, listing the present states, the next states corresponding to the present states and the required logic status of the flip-flop inputs (the J and K inputs if the counter is to be implemented with J-K flip-flops). The excitation table is shown in Table 3.10

Table 3.10 Excitation Table

Present state			Next state			Inputs					
C	B	A	C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	0	0	X	1	X	0	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	0	0	0	X	1	0	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

The circuit excitation table can be drawn very easily once we know the excitation table of the flip-flop to be used for building the counter. For instance, let us look at the first row of the excitation table Table 3.10. The counter is in the 000 state and is to go to 010 on application of a clockpulse. That is, the normal outputs of C, B and A flip-flops have to undergo '0' to '0', '0' to '1' and'0' to '0' transitions respectively. Referring to the excitation table of a J-K flip-flop, the desired transitions can be realized if the logic status of JA, KA, JB, KB, JC and KC is as shown in the excitation table.

4. The next step is to design the logic circuits for generating JA, KA, JB, KB, JC and KC inputs from available A, A, B, B, C and C outputs. This can be done by drawing Karnaugh maps for each one of the inputs, minimizing them and then implementing the minimized Boolean

expressions. The Karnaugh maps for JA, KA, JB, KB, JC and KC are respectively shown in Figs 3.11.The minimized Boolean expressions are as follows:

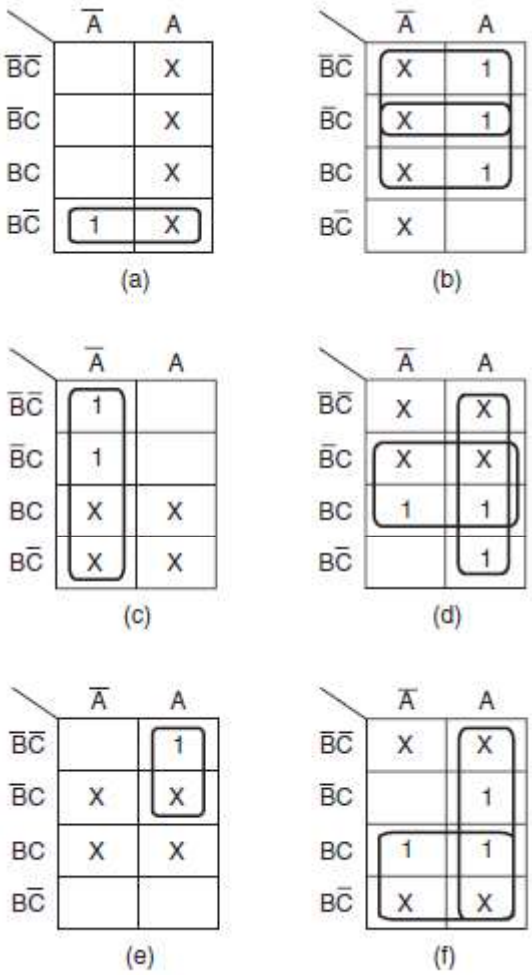


Figure 3.11 kmap simplification

$$J_A = B.\overline{C}$$
$$K_A = \overline{B} + C$$
$$J_B = \overline{A}$$
$$K_B = A + C$$
$$J_C = A.\overline{B}$$
$$K_C = A + B$$

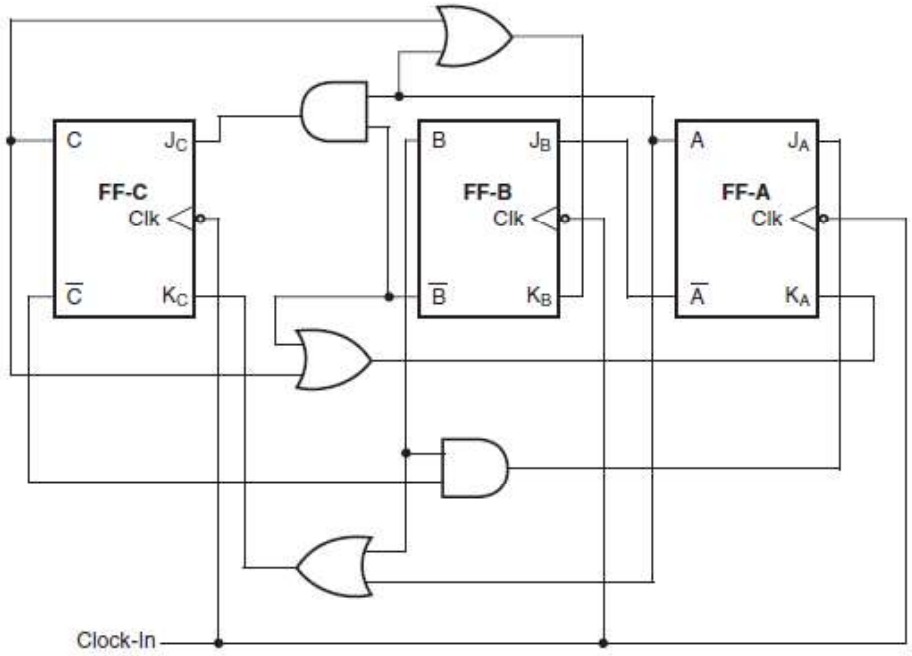


Figure 3.12 Counter with an arbitrary sequence

3.3Shift Register

A shift register is a digital device used for storage and transfer of data. The data to be stored could be the data appearing at the output of an encoding matrix before they are fed to the main digital system for processing or they might be the data present at the output of a microprocessor before they are fed to the driver circuitry of the output devices. The shift register thus forms an important link between the main digital system and the input/output channels. The shift registers can also be configured to construct some special types of counter that can be used to perform a number of arithmetic operations such as subtraction, multiplication, division, complementation, etc. The basic building block in all shift registers is the flipflop ,mainly a D-type flip-flop. Although in many of the commercial shift register ICs their internal circuit diagram might indicate the use of R-S flip-flops, a careful examination will reveal that these R-S flip-flop shave been wired as D flip-flops only. The storage capacity of a shift register equals the total number of bits of digital data it can store, which in turn depends upon the number of flip-flops used to construct the shift register. Since each flip-flop can store one bit of data, the storage capacity of the shift register equals the number of flip-flops used. As an example, the internal architecture of an eight-bit shift register will have a cascade arrangement of eight flip-flops. Based on the method used to load data onto and read data from shift registers, they are classified as serial-in serial-out (SISO) shift registers, serial-in parallel-out (SIPO) shift registers, parallel-in serial-out (PISO) shift registers and parallel-in parallel-out (PIPO) shift registers.

3.3.1 Serial-In Serial-Out Shift Register

Figure 3.14.shows the basic four-bit serial-in serial-out shift register implemented using D flip-flops. The circuit functions as follows. A reset applied to the CLEAR input of all the flip-flops resets their Q outputs to 0s. Refer to the timing waveforms of Fig. 3.15. The waveforms shown include the clockpulse train, the waveform representing the data to be loaded onto the shift register and the Q outputs of different flip-flops.

The flip-flops shown respond to the LOW-to-HIGH transition of the clock pulses as indicated by their logic symbols. During the first clock transition, the QA output goes from logic '0' to logic '1'.

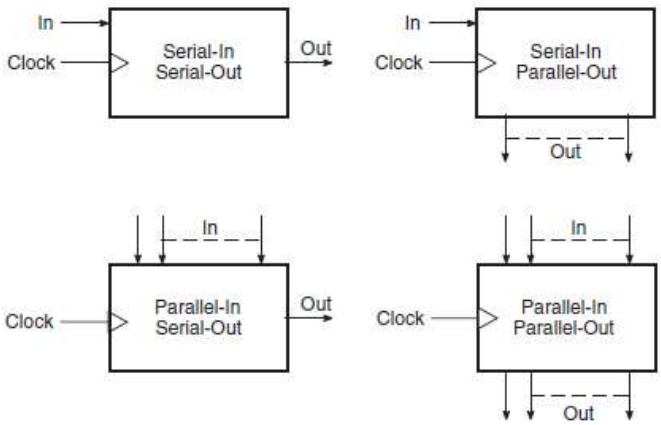


Figure 3.13 Circuit representation of shift registers

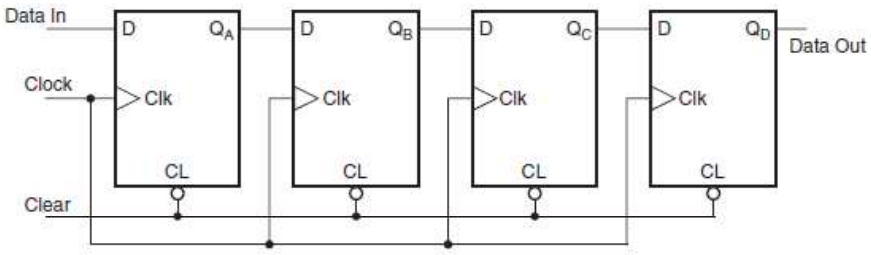


Figure 3.14 SISO shift register

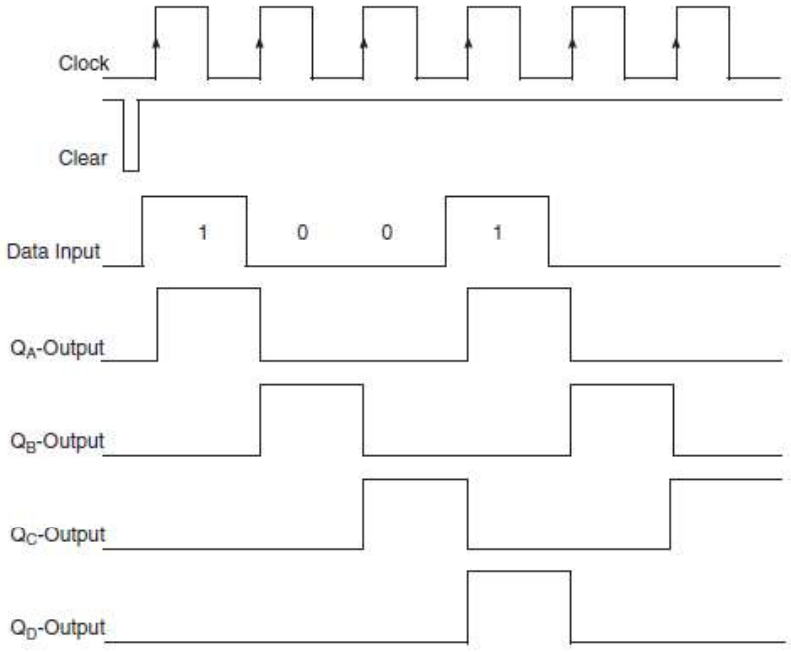


Figure 3.15Timing waveforms for the shift register

3.3.2 Serial-In Parallel-Out Shift Register

A serial-in parallel-out shift register is architecturally identical to a serial-in serial-out shift register except that in the case of the former all flip-flop outputs are also brought out on the IC terminals. The gated serial inputs A and B control the incoming serial data, as a logic LOW at either of the inputs inhibits entry of new data and also resets the first flip-flop to the logic LOW level at the next clockpulse. Logic HIGH at either of the inputs enables the other input, which then determines the state of the first flip-flop. Data at the serial inputs may be changed while the clock input is HIGH or LOW, and the register responds to LOW-to-HIGH transition of the ***clock.***

3.3.3 Parallel-In Serial-Out Shift Register

We will explain the operation of a parallel-in serial-out shift register with the help of the logic diagram of a practical device available in IC form. The parallel-in or serial-in modes are controlled by a SHIFT/LOAD input. When the SHIFT/LOAD input is held in the logic HIGH state, the serial data input AND gates are enabled and the circuit behaves like a serial-in serial-out shift register. When the SHIFT/LOAD input is held in the logic LOW state, parallel data input AND gates are enabled and data are loaded in parallel, in synchronism with the next clock pulse. Clocking is accomplished on the LOW-to-HIGH transition of the clock pulse via a two-input NOR gate. Holding one of the inputs of the NOR gate in the logic HIGH state inhibits the clock applied to the other input. Holding an input in the logic LOW state enables the clock to be applied to the other input. An active LOW CLEAR input overrides all the inputs, including the clock, and resets all flip-flops to the logic ‘0’ state.

3.3.4 Parallel-In Parallel-Out Shift Register

The hardware of a parallel-in parallel-out shift register is similar to that of a parallel-in serial-out shift register. If in a parallel-in serial-out shift register the outputs of different flip-flops are brought out, it becomes a parallel-in parallel-out shift register. In fact, the logic diagram of a parallel-in parallel-out shift register is similar to that of a parallel-in serial-out shift register. As an example, IC74199 is an eight-bit parallel-in parallel-out shift register.

3.3.5 Bidirectional Shift Register

A bidirectional shift register allows shifting of data either to the left or to the right. This is made possible with the inclusion of some gating logic having a control input. The control input allows shifting of data either to the left or to the right, depending upon its logic status.

3.3.6 Universal Shift Register

A universal shift register can be made to function as any of the four types of register discussed in previous sections. That is, it has serial/parallel data input and output capability, which means that it can function as serial-in serial-out, serial-in parallel-out, parallel-in serial out and parallel-in parallel-out shift registers.

3.4 Shift Register Counters

We have seen that both counters and shift registers are some kinds of cascade arrangement of flip-flops. A shift register, unlike a counter, has no specified sequence of states. However, if the serial output of the shift register is fed back to the serial input, we do get a circuit that exhibits a specified sequence of states. The resulting circuits are known as shift register counters. Depending upon the nature of the feedback, we have two types of shift register counter, namely the ring counter and the shift counter, also called the Johnson counter. These are briefly described in the following paragraphs.

3.4.1 Ring Counter

A ring counter is obtained from a shift register by directly feeding back the true output of the output flip-flop to the data input terminal of the input flip-flop. If D flip-flops are being used to construct the shift register, the ring counter, also called a circulating register, can be constructed by feeding back the Q output of the output flip-flop back to the D input of the input flip-flop. If J-K flip-flops are being used, the Q and Q outputs of the output flip-flop are respectively fed back to the J and Kinputs of the input flip-flop. Figure 3.16 shows the logic diagram of a four-bit ring counter. Let us assume that flip-flop FF0 is initially set to the logic ‘1’ state and all other flip-flops are reset to the logic ‘0’ state. The counter output is therefore 1000. With the first clock pulse, this ‘1’ gets shifted to the second flip-flop output and the counter output becomes 0100. Similarly, with the second and third clock pulses, the counter output will become 0010 and 0001. With the fourth clock pulse, the counter output will again become 1000. The count cycle repeats in the subsequent clock pulses. Circulating registers of this type find wide application in the control section of microprocessor-based systems where one event should follow the other. The timing waveforms for the circulating register of Figure3.16, as shown in Fig. 3.17, further illustrate their utility as a control element in a digital system to generate control pulses that must occur one after the other sequentially.

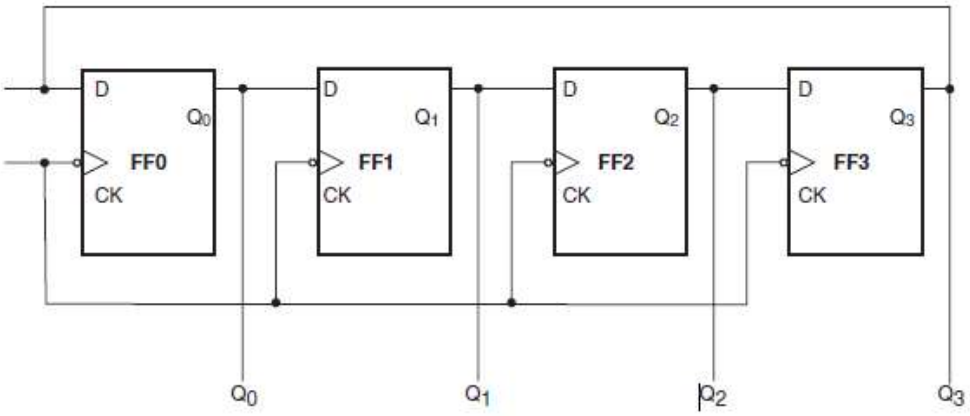


Figure 3.16 four bit ring counter

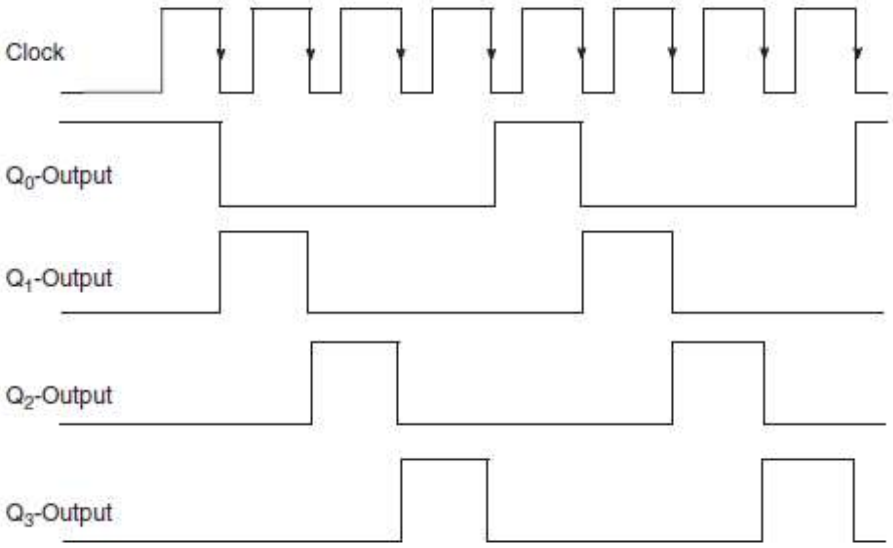


Figure 3.17 Timing waveforms of the four bit ring counter

3.4.2 Shift Counter

A shift counter on the other hand is constructed by having an inverse feedback in a shift register. For instance, if we connect the Q output of the output flip-flop back to the K input of the input flip-flop and the Q output of the output flip-flop to the J input of the input flip-flop in a serial shift register, the result is a shift counter, also called a Johnson counter. If the shift register employs D flip-flops,the Q output of the output flip-flop is fed back to the D input of the input flip-flop. If R-S flip-flops are used, the Q output goes to the R input and the Q output is connected to the S input. Figure 3.18shows the logic diagram of a basic four-bit shift counter. Let us assume that the counter is initially reset to all 0s. With the first clock cycle, the outputs will become 1000. With the second, third and fourth clock cycles, the outputs will respectively be 1100,1110 and 1111. The fifth clock cycle will change the counter output to 0111. The sixth,

seventh and eighth clock pulses successively change the outputs to 0011, 0001 and 0000. Thus, one count cycle

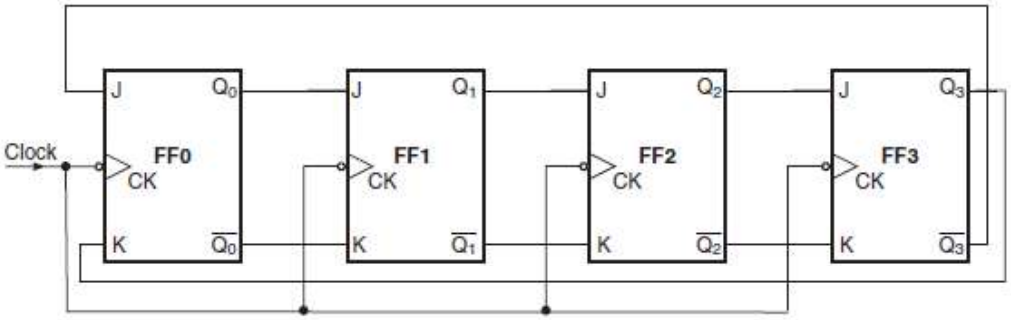


Figure 3.18 four bit shift counter

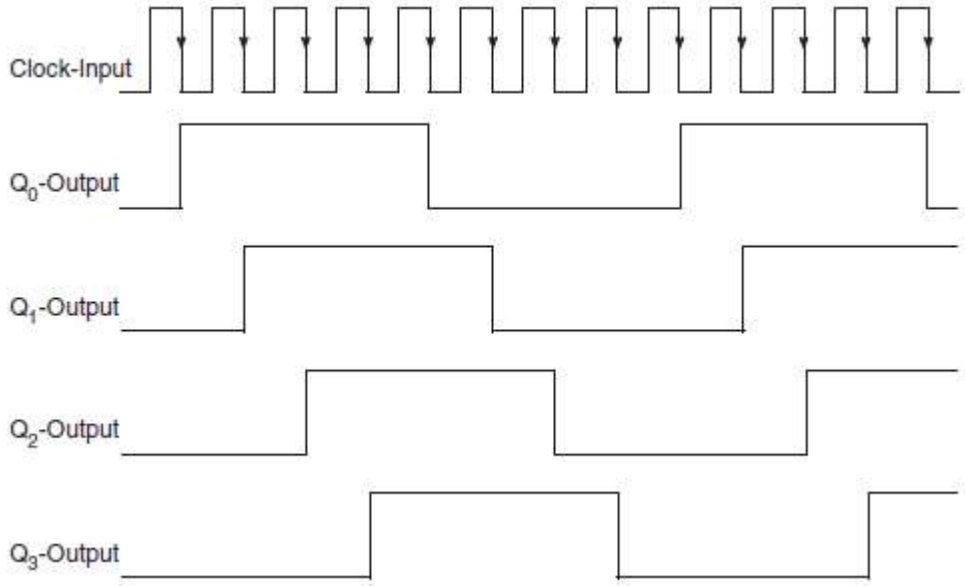


Figure 3.19 Timing waveforms of the shift counter

is completed in eight cycles. Figure 3.19 shows the timing waveforms. Different output waveforms are identical except for the fact that they are shifted from the immediately preceding one by one clock cycle. Also, the time period of each of these waveforms is 8 times the period of the clock waveform. That is, this shift counter behaves as a divide-by-8 circuit. In general, a shift counter comprising n flip-flops acts as a divide-by-2n circuit. Shift counters can be used very conveniently to construct counters having a modulus other than the integral power of 2.