- A HIGH-to-LOW transition at the $\overline{TCD}$ output,coinciding with the trailing edge of the sixth clock pulse, loads 0110 to the counter output.
- Therefore, immediately after the leading edge of the eighth clock pulse, the counter will be in the 0100 state.

11.11 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 2^N 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 2^N, 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 2^N 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.

11.11.1 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. The same for a *J-K* flip-flop and a *D* flip-flop are shown in Tables 11.7 and 11.8 respectively. Referring to Table 11.7, 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 0 or 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.

Table 11.7 Excitation table of a *J-K* flip-flop.

Present state (Q_n)	Next state (Q_{n+1})	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 11.8 Excitation table of a *D* flip-flop.

Present state (Q_n)	Next state (Q_{n+1})	*D*
0	0	0
0	1	1
1	0	0
1	1	1

11.11.2 State 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. 11.24 shows the state transition diagram of an MOD-8 binary counter.

11.11.3 Design Procedure

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

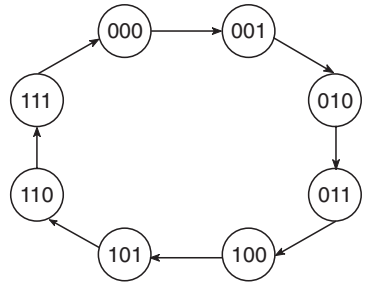

Figure 11.24 State transition diagram for an MOD-8 binary counter.

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 11.25 shows the state transition diagram

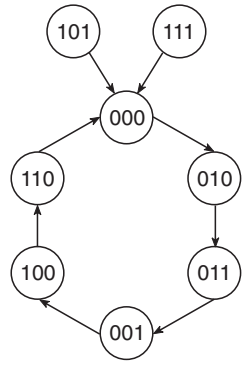

Figure 11.25 State transition diagram.

3. 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 11.9

Table 11.9 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 11.9). The counter is in the 000 state and is to go to 010 on application of a clock pulse. 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 J_A, K_A, J_B, K_B, J_C and K_C is as shown in the excitation table.

4. The next step is to design the logic circuits for generating J_A, K_A, J_B, K_B, J_C and K_C inputs from available A, $\overline{A}$, B, $\overline{B}$, C and $\overline{C}$ 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 J_A, K_A, J_B, K_B, J_C and K_C are respectively shown in Figs 11.26(a), (b), (c), (d), (e) and (f). The minimized Boolean expressions are as follows:

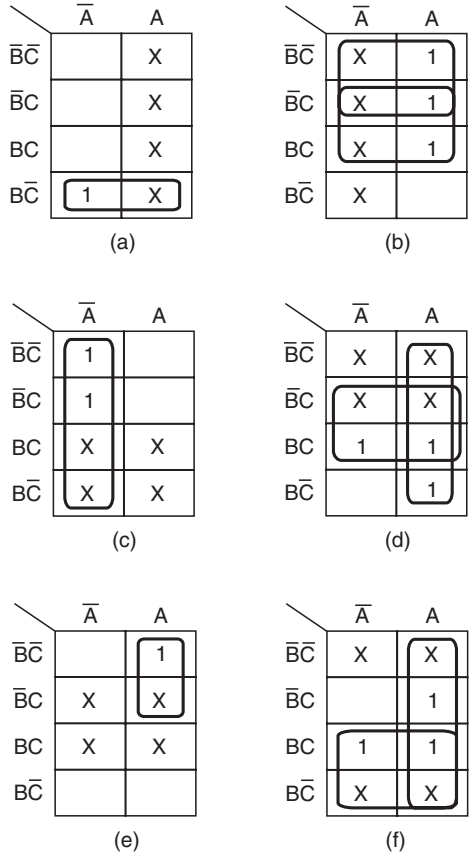

Figure 11.26 Karnaugh maps.

$$J_A = B.\overline{C} \tag{11.2}$$

$$K_A = \overline{B} + C \tag{11.3}$$

$$J_B = \overline{A} \tag{11.4}$$

$$K_B = A + C \tag{11.5}$$

$$J_C = A.\overline{B} \tag{11.6}$$

$$K_C = A + B \tag{11.7}$$

The above expressions can now be used to implement combinational circuits to generate J_A, K_A, J_B, K_B, J_C and K_C inputs. Figure 11.27 shows the complete counter circuit

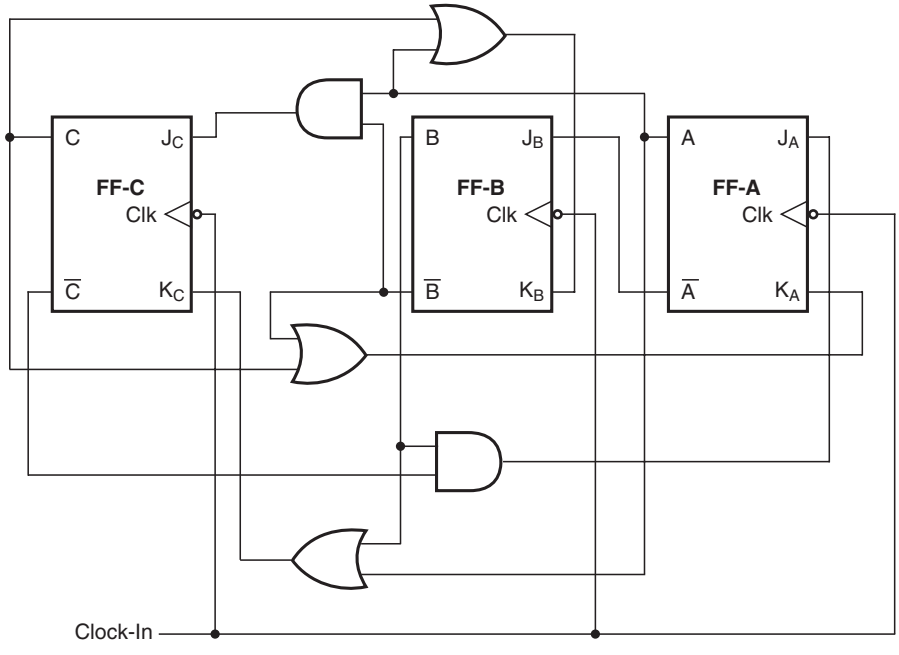

Figure 11.27 Counter with an arbitrary sequence.

The design procedure illustrated above can be used to design a synchronous counter for any given count sequence with the condition that no state occurs more than once in one complete cycle of the given count sequence as the design cannot handle a situation where a particular present state has more than one future state.

Table 11.10 Example 11.8.

Present state (Q_n)	Next state (Q_{n+1})	Inputs	
		X_1	X_2
0	0	0	0
0	1	0	1
1	0	1	X
1	1	X	1

X = don't care condition.

Example 11.8

Table 11.10 gives the excitation table of a certain flip-flop having X_1 and X_2 as its inputs. Draw the circuit excitation table of an MOD-5 synchronous counter using this flip-flop for the count sequence 000, 001, 011, 101, 110, 000, . . . If the present state is an undesired one, it should transit to 110 on application of a clock pulse. Design the counter circuit using the flip-flop whose excitation circuit is given in Table 11.10.

Solution

- The circuit excitation table is shown in Table 11.11.
- The number of flip-flops required is 3.
- X_1 (A) and X_2 (A) are the inputs of flip-flop A, which is also the LSB flip-flop.
- X_1 (B) and X_2 (B) represent the inputs to flip-flop B.
- X_1 (C) and X_2 (C) are the inputs to flip-flop C, which is also the MSB flip-flop.
- The next step is to draw Karnaugh maps, one each for different inputs to the three flip-flops.
- Figures 11.28(a) to (f) show the Karnaugh maps for X_1 (A), X_2 (A), X_1 (B), X_2 (B), X_1 (C) and X_2 (C) respectively.
- The minimized expressions are as follows:

$$X_1(A) = A \tag{11.8}$$

$$X_2(A) = A + \overline{B}.\overline{C} \tag{11.9}$$

$$X_1(B) = B \tag{11.10}$$

$$X_2(B) = A + B + C \tag{11.11}$$

$$X_1(C) = C \tag{11.12}$$

$$X_2(C) = B + C \tag{11.13}$$

- Figure 11.29 shows the circuit implementation.

Example 11.9

Design a synchronous counter that counts as 000, 010, 101, 110, 000, 010, . . . Ensure that the unused states of 001, 011, 100 and 111 go to 000 on the next clock pulse. Use J-K flip-flops. What will the counter hardware look like if the unused states are to be considered as 'don't care's.

Table 11.11 Example 11.8.

Present state			Next state			Inputs					
C	*B*	*A*	*C*	*B*	*A*	$X_1(A)$	$X_2(A)$	$X_1(B)$	$X_2(B)$	$X_1(C)$	$X_2(C)$
0	0	0	0	0	1	0	1	0	0	0	0
0	0	1	0	1	1	X	1	0	1	0	0
0	1	0	1	1	0	0	0	X	1	0	1
0	1	1	1	0	1	X	1	1	X	0	1
1	0	0	1	1	0	0	0	0	1	X	1
1	0	1	1	1	0	1	X	0	1	X	1
1	1	0	0	0	0	0	0	1	X	1	X
1	1	1	1	1	0	1	X	X	1	X	1

X = don't care condition.

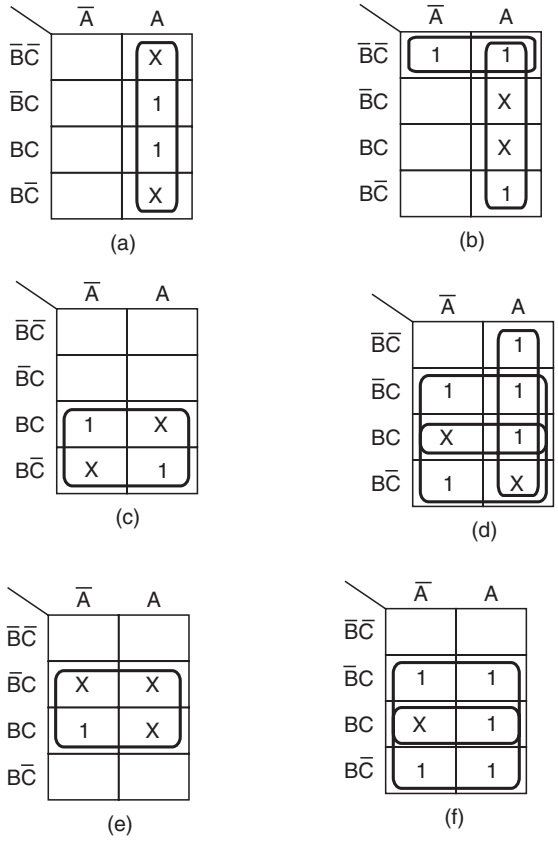

Figure 11.28 Karnaugh maps (example 11.8).

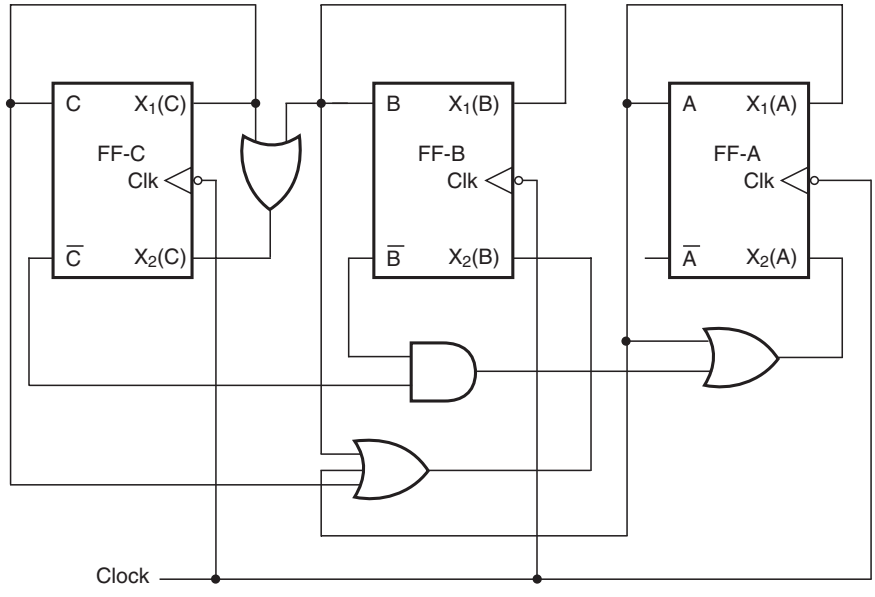

Figure 11.29 Counter circuit (example 11.8).

Table 11.12 Example 11.9.

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

Solution

- The number of flip-flops required is three.
- Table 11.12 shows the desired circuit excitation table.
- The Karnaugh maps for J_A, K_A, J_B, K_B, J_C and K_C are shown in Figs 11.30(a) to (f) respectively.
- The simplified Boolean expressions are as follows:

$$J_A = B.\overline{C} \tag{11.14}$$

$$K_A = 1 \tag{11.15}$$

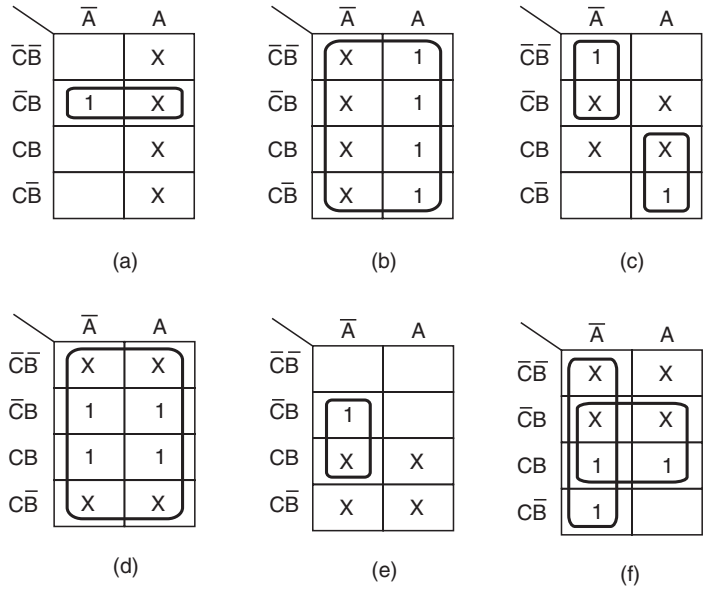

Figure 11.30 Karnaugh maps (example 11.9).

$$J_B = A.C + \overline{A}.\overline{C} \tag{11.16}$$

$$K_B = 1 \tag{11.17}$$

$$J_C = \overline{A}.B \tag{11.18}$$

$$K_C = \overline{A} + B \tag{11.19}$$

- The hardware implementation is shown in Fig. 11.31.
- In the case where the unused inputs are considered as 'don't cares', the circuit excitation table is modified to that shown in Table 11.13.
- Modified Karnaugh maps are shown in Fig. 11.32.
- The minimized Boolean expressions are derived from the Karnaugh maps of Figs 11.32(a) to (f).
- Minimized expressions for J_A, K_A, J_B, K_B, J_C and K_C respectively are as follows:

$$J_A = B.\overline{C} \tag{11.20}$$

$$K_A = 1 \tag{11.21}$$

$$J_B = 1 \tag{11.22}$$

$$K_B = 1 \tag{11.23}$$

$$J_C = B \tag{11.24}$$

$$K_C = \overline{A} \tag{11.25}$$

- Figure 11.33 shows the hardware implementation.

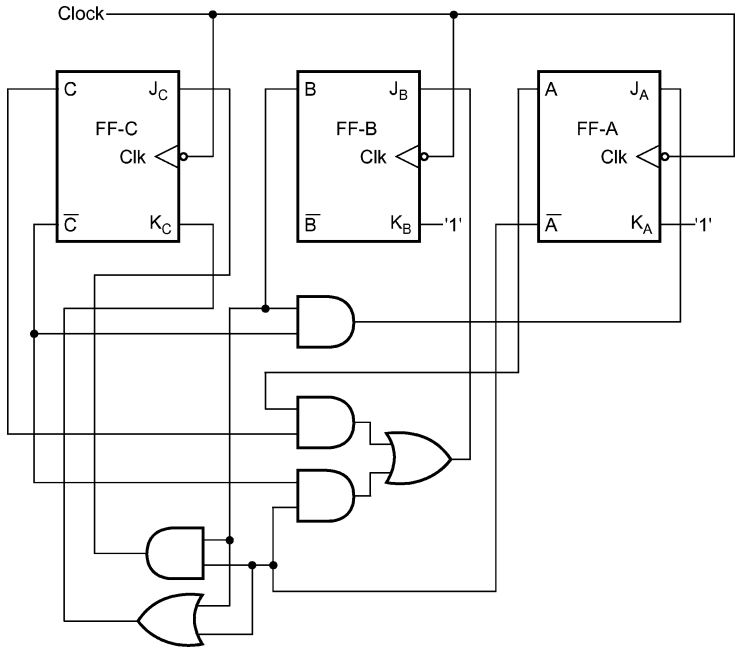

Figure 11.31 Hardware implementation of the counter circuit (example 11.9).

Table 11.13 Example 11.9.

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

11.12 Shift 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

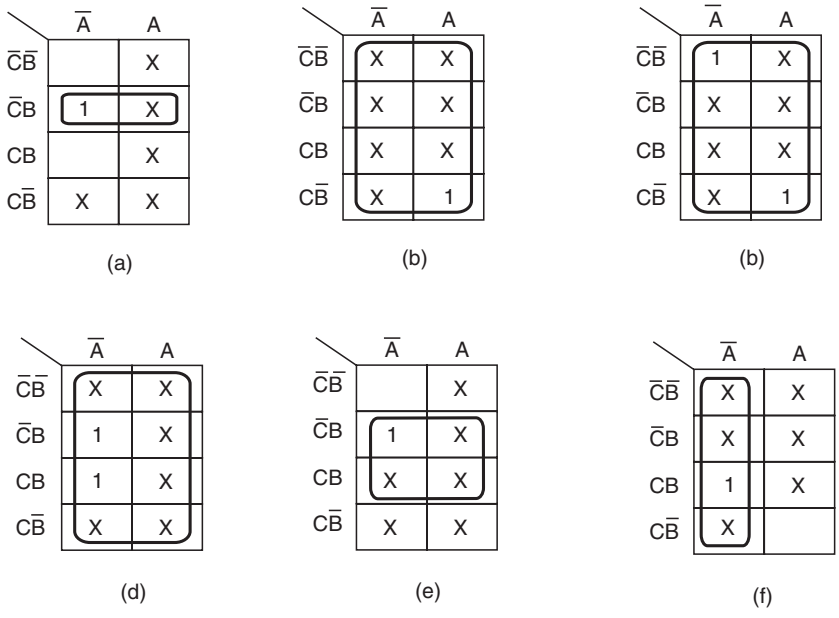

Figure 11.32 Modified Karnaugh maps (example 11.9).

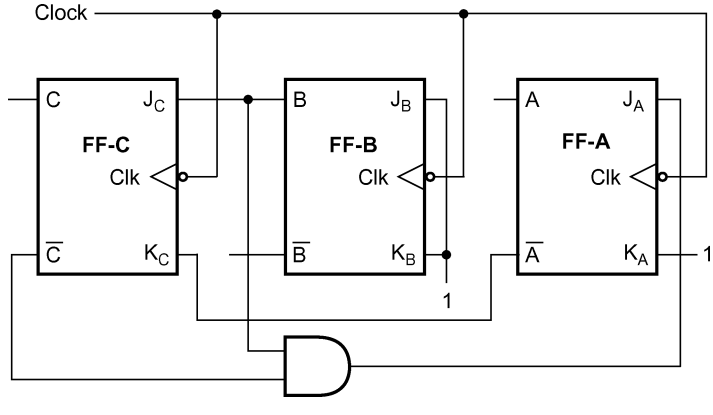

Figure 11.33 Hardware implementation of the counter circuit (example 11.9).

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 flip-flop, 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-flops have 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.

Figure 11.34 shows a circuit representation of the above-mentioned four types of shift register.

11.12.1 Serial-In Serial-Out Shift Register

Figure 11.35 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. 11.36. The waveforms shown include the clock pulse 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 Q_A output goes from logic '0' to logic '1'.

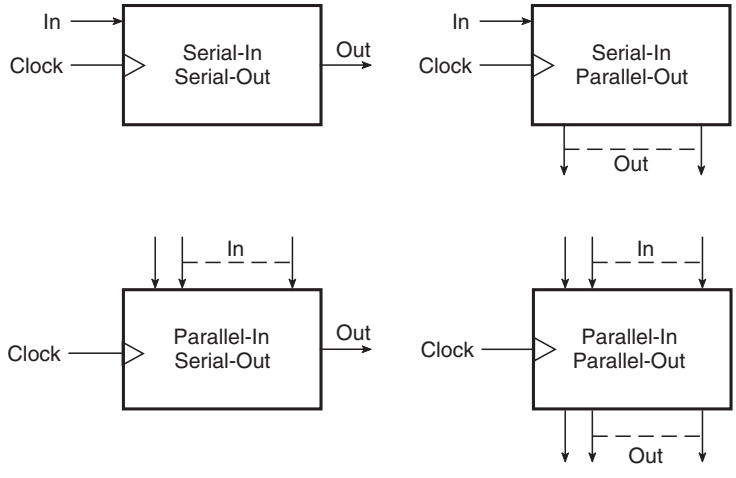

Figure 11.34 Circuit representation of shift registers.

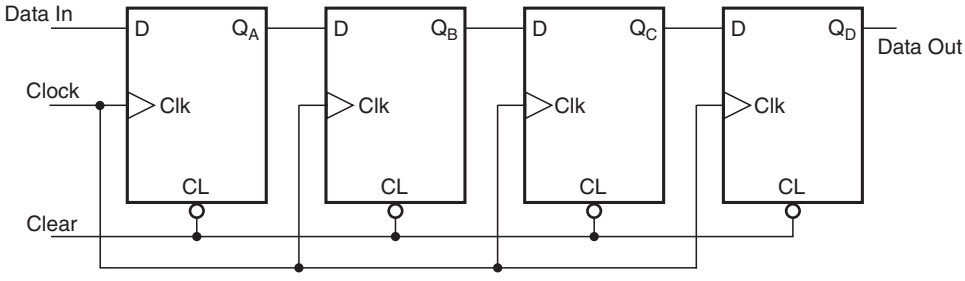

Figure 11.35 Serial-in, serial-out shift register.

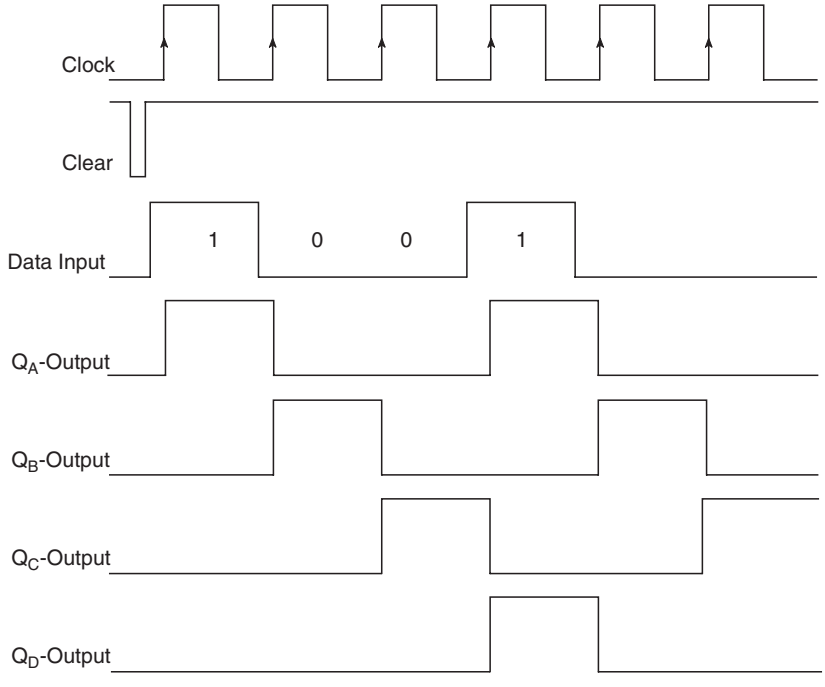

Figure 11.36 Timing waveforms for the shift register of Fig. 11.35.

The outputs of the other three flip-flops remain in the logic '0' state as their D inputs were in the logic '0' state at the time of clock transition. During the second clock transition, the Q_A output goes from logic '1' to logic '0' and the Q_B output goes from logic '0' to logic '1', again in accordance with the logic status of the D inputs at the time of relevant clock transition.

Thus, we have seen that a logic '1' that was present at the data input prior to the occurrence of the first clock transition has reached the Q_B output at the end of two clock transitions. This bit will reach the Q_D output at the end of four clock transitions. In general, in a four-bit shift register of the type

Table 11.14 Contents of four-bit serial-in serial-out shift register for the first eight clock cycles.

Clock	Q_A	Q_B	Q_C	Q_D
Initial contents	0	0	0	0
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	0	0	1	0
After fourth clock transition	**1**	**0**	**0**	**1**
After fifth clock transition	0	1	0	0
After sixth clock transition	0	0	1	0
After seventh clock transition	0	0	0	1
After eighth clock transition	**0**	**0**	**0**	**0**

shown in Fig. 11.35, a data bit present at the data input terminal at the time of the nth clock transition reaches the Q_D output at the end of the $(n+4)$th clock transition. During the fifth and subsequent clock transitions, data bits continue to shift to the right, and at the end of the eighth clock transition the shift register is again reset to all 0s. Thus, in a four-bit serial-in serial-out shift register, it takes four clock cycles to load the data bits and another four cycles to read the data bits out of the register. The contents of the register for the first eight clock cycles are summarized in Table 11.14. We can see that the register is loaded with the four-bit data in four clock cycles, and also that the stored four-bit data are read out in the subsequent four clock cycles.

IC 7491 is a popular eight-bit serial-in serial-out shift register. Figure 11.37 shows its internal functional diagram, which is a cascade arrangement of eight *R-S* flip-flops. Owing to the inverter between the *R* and *S* inputs of the data input flip-flop, it is functionally the same as a *D* flip-flop. The data to be loaded into the register serially can be applied either at *A* or *B* input of the NAND gate. The other input is then kept in the logic HIGH state to enable the NAND gate. In that case, data present at *A* or *B* get complemented as they appear at the NAND output. Another inversion provided by the inverter, however, restores the original status so that for a logic '1' at the data input there is a logic '1' at the SET input of the flip-flop and a logic '0' at the RESET input of the flip-flop, and for a logic '0' at the data input there is a logic '0' at the SET input and a logic '1' at the RESET input of the flip-flop. The NAND gate provides only a gating function, and, if it is not required, the two inputs of the NAND can be shorted to have a single-line data input. The shift register responds to the LOW-to-HIGH transitions of the clock pulses.

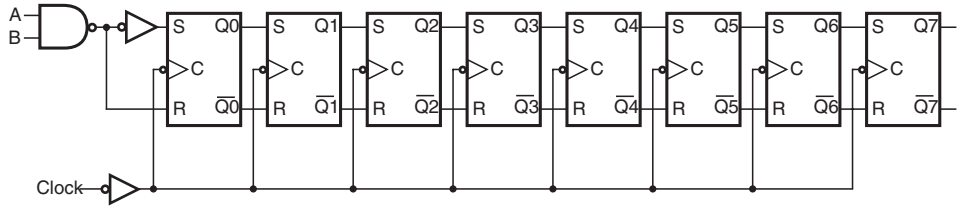

Figure 11.37 Logic diagram of IC 7491.

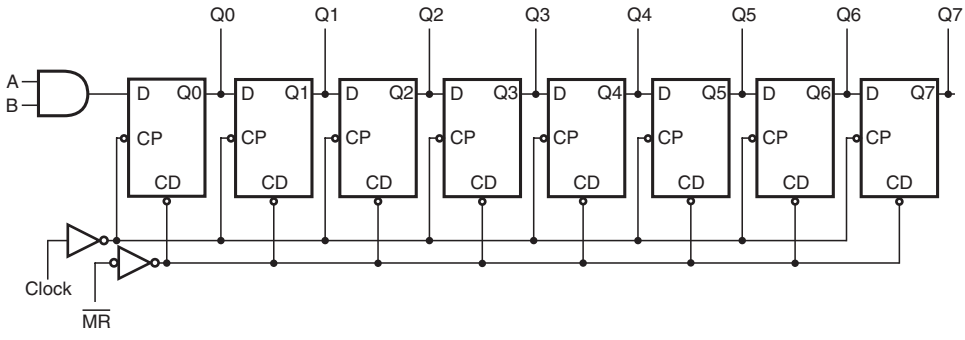

Figure 11.38 Logic diagram of IC 74164.

11.12.2 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. Figure 11.38 shows the logic diagram of a typical serial-in parallel-out shift register. In fact, the logic diagram shown in Fig. 11.38 is that of IC 74164, a popular eight-bit serial-in parallel-out shift register. 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 clock pulse. 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. Figure 11.39 shows the relevant timing waveforms.

11.12.3 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. Figure 11.40 shows the logic diagram of one such shift register. The logic diagram is that of IC 74166, which is an eight-bit parallel/serial-in, serial-out shift register belonging to the TTL family of devices.

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. The timing waveforms shown in Fig. 11.41 explain both serial-in, serial-out as well as parallel-in, serial-out operations.