11

Counters and Registers

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 flip-flop 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 aspects have 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.

11.1 Ripple (Asynchronous) Counter

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

Digital Electronics: Principles, Devices and Applications Anil K. Maini
 ISBN: 978-0-470-03214-5

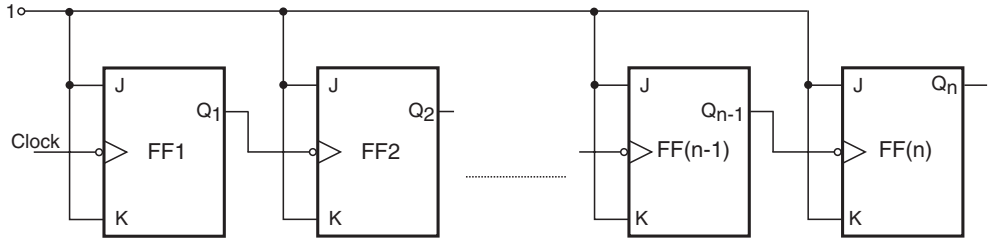


Figure 11.1 Generalized block schematic of n-bit binary ripple counter.

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 to n 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.

11.1.1 Propagation Delay in Ripple Counters

A major problem with ripple counters arises from the propagation delay of the flip-flops constituting the counter. As mentioned in the preceding paragraphs, the effective propagation delay in a ripple counter is equal to the sum of propagation delays due to different flip-flops. The situation becomes worse with increase in the number of flip-flops used to construct the counter, which is the case in larger bit counters. Coming back to the ripple counter, an increased propagation delay puts a limit on the maximum frequency used as clock input to the counter. We can appreciate that the clock signal time period must be equal to or greater than the total propagation delay. The maximum clock frequency therefore corresponds to a time period that equals the total propagation delay. If t_{pd} is the propagation delay in each flip-flop, then, in a counter with N flip-flops having a modulus of less than or equal to 2^N, the maximum usable clock frequency is given by $f_{max} = 1/(N \times t_{pd})$. Often, two propagation delay times are specified in the case of flip-flops, one for LOW-to-HIGH transition (t_{pLH}) and the other for HIGH-to-LOW transition (t_{pHL}) at the output. In such a case, the larger of the two should be considered for computing the maximum clock frequency.

As an example, in the case of a ripple counter IC belonging to the low-power Schottky TTL (LSTTL) family, the propagation delay per flip-flop typically is of the order of 25 ns. This implies that a four-bit

ripple counter from this logic family can not be clocked faster than 10 MHz. The upper limit on the clock frequency further decreases with increase in the number of bits to be handled by the counter.

11.2 Synchronous Counter

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.

11.3 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 2^n. 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 2^n.

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 is 16. The number of flip-flops in this case would be 4, as $16 = 2^4$. 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, 5 and 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

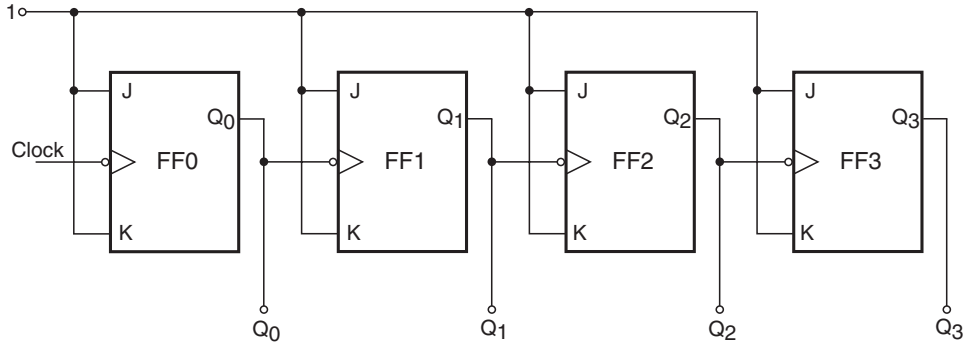
$$(2^{N-1} + 1) \leq modulus \leq 2^N \tag{11.1}$$

11.4 Binary Ripple Counter – Operational Basics

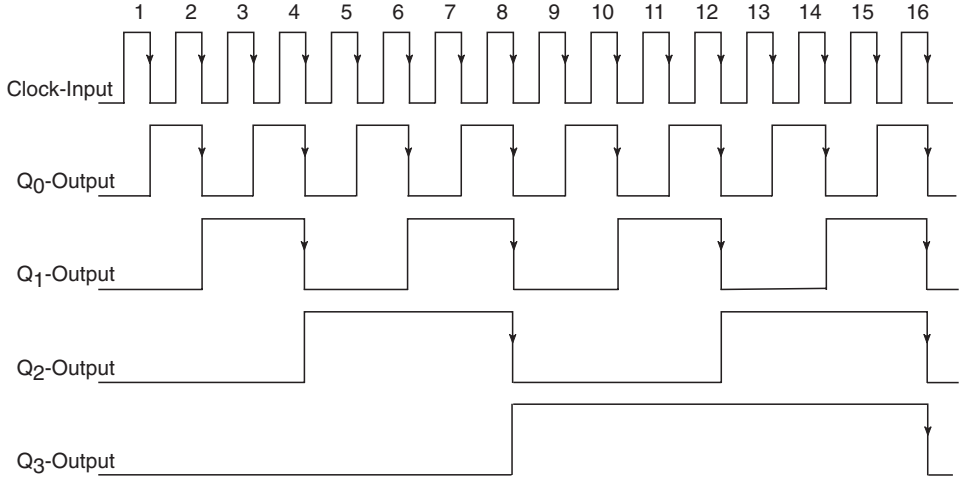
The operation of a binary ripple counter can be best explained with the help of a typical counter of this type. Figure 11.2(a) shows a four-bit ripple counter implemented with negative edge-triggered *J-K* flip-flops wired as toggle flip-flops. The output of the first flip-flop feeds the clock input 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 as Q_0 (LSB flip-flop), Q_1, Q_2 and Q_3 (MSB flip-flop). Figure 11.2(b) shows the waveforms appearing at Q_0, Q_1, Q_2 and Q_3 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, Q_0 goes from '0' to '1' owing to the toggling action. As the flip-flops used are negative edge-triggered ones, the '0' to '1' transition of Q_0 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, $Q_0 = 1$, $Q_1 = 0$, $Q_2 = 0$ and $Q_3 = 0$.

On the HIGH-to-LOW transition of the second clock pulse, Q_0 toggles again. That is, it goes from '1' to '0'. This '1' to '0' transition at the Q_0 output triggers FF1, the output Q_1 of which goes from '0'



(a)



(b)

Figure 11.2 Four-bit binary ripple counter.

to '1'. The Q_2 and Q_3 outputs remain unaffected. Therefore, immediately after the occurrence of the second HIGH-to-LOW transition of the clock signal, $Q_0 = 0$, $Q_1 = 1$, $Q_2 = 0$ and $Q_3 = 0$. On similar lines, we can explain the logic status of Q_0, Q_1, Q_2 and Q_3 outputs immediately after subsequent clock transitions. The logic status of outputs for the first 16 relevant (HIGH-to-LOW in the present case) clock signal transitions is summarized in Table 11.1.

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 0000 from where it had started. In general, if we had N flip-flops, we could count up to 2^N pulses before the counter resets to the initial state. We can also see from the Q_0, Q_1, Q_2 and Q_3 waveforms, as shown

Table 11.1 Output logic states for different clock signal transitions for a four-bit binary ripple counter.

Clock signal transition number	Q_0	Q_1	Q_2	Q_3
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	1	1	0	0
After fourth clock transition	0	0	1	0
After fifth clock transition	1	0	1	0
After sixth clock transition	0	1	1	0
After seventh clock transition	1	1	1	0
After eighth clock transition	0	0	0	1
After ninth clock transition	1	0	0	1
After tenth clock transition	0	1	0	1
After eleventh clock transition	1	1	0	1
After twelfth clock transition	0	0	1	1
After thirteenth clock transition	1	0	1	1
After fourteenth clock transition	0	1	1	1
After fifteenth clock transition	1	1	1	1
After sixteenth clock transition	0	0	0	0

in Fig. 11.2(b), that the frequencies of the Q_0, Q_1, Q_2 and Q_3 waveforms are $f/2$,$f/4$, $f/8$ and $f/16$ respectively. Here, f is the frequency of the clock input. This implies that a counter of this type can be used as a divide-by-2^N circuit, where N is the number of flip-flops in the counter chain. In fact, such a counter provides frequency-divided outputs of $f/2^N$, $f/2^{N-1}$, $f/2^{N-2}$, $f/2^{N-3}$, . . . , $f/2$ 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. 11.2(a), outputs are available at $f/2$ from the Q_0 output, at $f/4$ from the Q_1 output, at $f/8$ from the Q_2 output and at $f/16$ from the Q_3 output. It may be noted that frequency division is one of the major applications of counters.

Example 11.1

A four-bit binary ripple counter of the type shown in Fig. 11.2(a) is initially in the 0000 state before the clock input is applied to the counter. The clock pulses are applied to the counter at some time instant t_1 and then again removed some time later at another time instant t_2. The counter is observed to read 0011. How many negative-going clock transitions have occurred during the time the clock was active at the counter input?

Solution

It is not possible to determine the number of clock edges – it could have been 3, 19, 35, 51, 67, 83 . . . – as there is no means of finding out whether the counter has recycled or not from the given data. Remember that this counter would come back to the 0000 state after every 16 clock pulses.

Example 11.2

It is desired to design a binary ripple counter of the type shown in Fig. 11.1 that is capable of counting the number of items passing on a conveyor belt. Each time an item passes a given point, a pulse is generated that can be used as a clock input. If the maximum number of items to be counted is 6000, determine the number of flip-flops required.

Solution

- The counter should be able to count a maximum of 6000 items.
- An N-flip-flop would be able to count up to a maximum of $2^N - 1$ counts.
- On the 2^Nth clock pulse, it will get reset to all 0s.
- Now, $2^N - 1$ should be greater than or equal to 6000.
- That is, $2^N - 1 \geq 6000$, which gives $N \geq \log 6001/\log 2 \geq 3.778/0.3010 \geq 12.55$.
- The smallest integer that satisfies this condition is 13.
- Therefore, the minimum number of flip-flops required = 13

11.4.1 Binary Ripple Counters with a Modulus of Less than 2^N

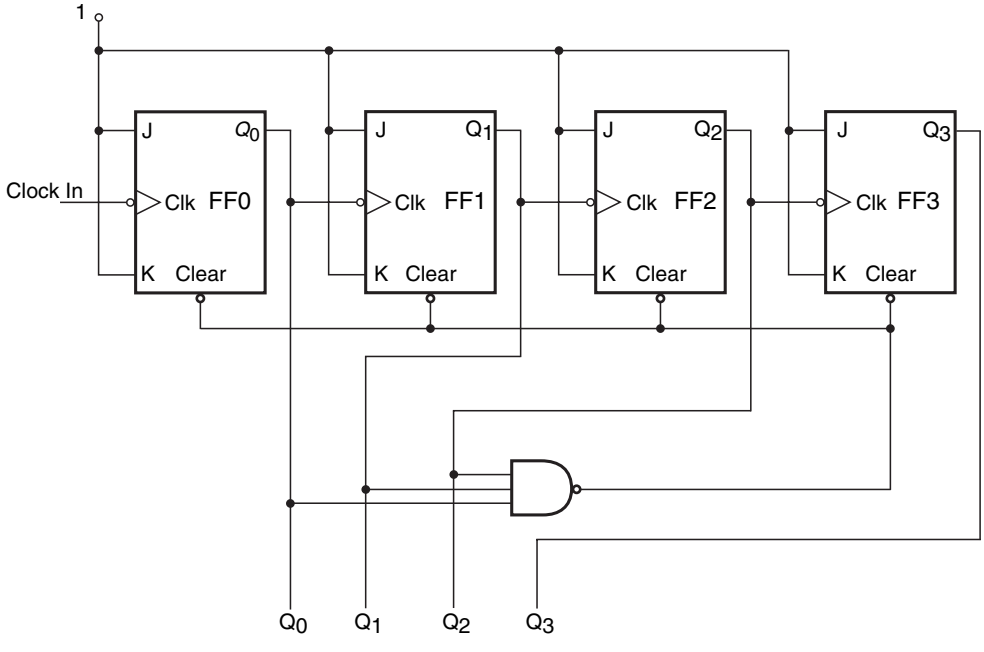
An N-flip-flop binary ripple counter can be modified, as we will see in the following paragraphs, to have any other modulus less than 2^N with the help of simple externally connected combinational logic. We will illustrate this simple concept with the help of an example.

Consider the four-flip-flop binary ripple counter arrangement of Fig. 11.3(a). It uses *J-K* flip-flops with an active LOW asynchronous CLEAR input. The NAND gate in the figure has its output connected to the CLEAR inputs of all four flip-flops. The inputs to this three-input NAND gate are from the Q outputs of flip-flops FF0, FF1 and FF2. If we disregard the NAND gate for some time, this counter will go through its natural binary sequence from 0000 to 1111. But that is not to happen in the present arrangement. The counter does start counting from 0000 towards its final count of 1111. The counter keeps counting as long as the asynchronous CLEAR inputs of the different flip-flops are inactive. That is, the NAND gate output is HIGH. This is the case until the counter reaches 0110. With the seventh clock pulse it tends to go to 0111, which makes all NAND gate inputs HIGH, forcing its output to LOW. This HIGH-to-LOW transition at the NAND gate output clears all flip-flop outputs to the logic '0' state, thus disallowing the counter to settle at 0111. From the eighth clock pulse onwards, the counter repeats the sequence. The counter thus always counts from 0000 to 0110 and resets back to 0000. The remaining nine states, which include 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 and 1111, are skipped, with the result that we get an MOD-7 counter.

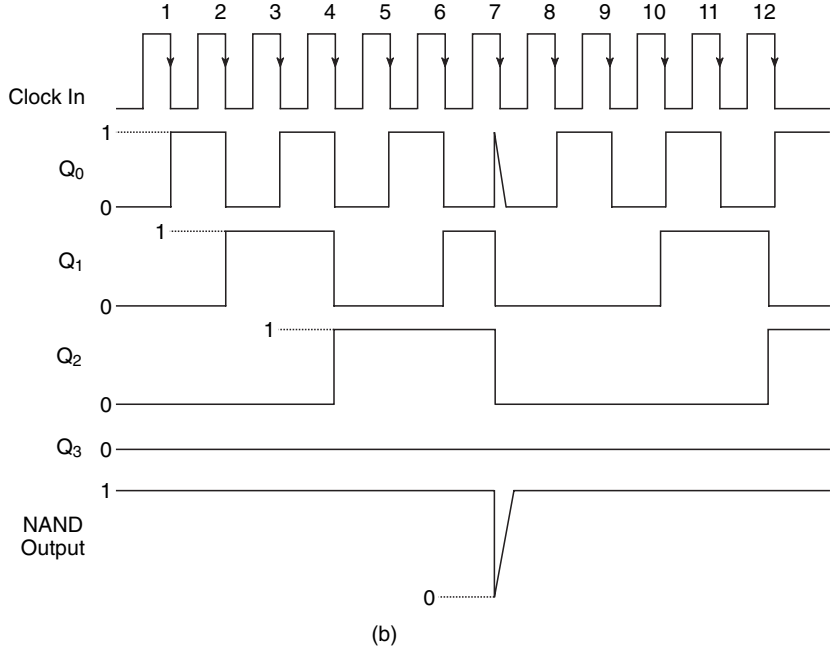
Figure 11.3(b) shows the timing waveforms for this counter. By suitably choosing NAND inputs, one can get a counter with any MOD number less than 16. Examination of timing waveforms also reveals that the frequency of the Q_2 output is one-seventh of the input clock frequency.

The waveform at the Q_2 output is, however, not symmetrical as it would be if the counter were to go through its full binary sequence. The Q_3 output stays in the logic LOW state. It is expected to be so because an MOD-7 counter needs a minimum of three flip-flops. That is why the fourth flip-flop, which was supposed to toggle on the HIGH-to-LOW transition of the eighth clock pulse, and on every successive eighth pulse thereafter, never gets to that stage. The counter is cleared on the seventh clock pulse and every successive seventh clock pulse thereafter.

As another illustration, if the NAND gate used in the counter arrangement of Fig. 11.3(a) is a two-input NAND and its inputs are from the Q_1 and Q_3 outputs, the counter will go through 0000 to 1001 and then reset to 0000 again, as, the moment the counter tends to switch from the 1001 to the 1010 state, the NAND gate goes from the '1' to the '0' state, clearing all flip-flops to the '0' state.



(a)



(b)

Figure 11.3 Binary ripple counter with a modulus of less than 2^N.

Steps to be followed to design any binary ripple counter that starts from 0000 and has a modulus of X are summarized as follows:

1. Determine the minimum number of flip-flops N so that $2^N \geq X$. Connect these flip-flops as a binary ripple counter. If $2^N = X$, do not go to steps 2 and 3.
2. Identify the flip-flops that will be in the logic HIGH state at the count whose decimal equivalent is X. Choose a NAND gate with the number of inputs equal to the number of flip-flops that would be in the logic HIGH state. As an example, if the objective were to design an MOD-12 counter, then, in the corresponding count, that is, 1100, two flip-flops would be in the logic HIGH state. The desired NAND gate would therefore be a two-input gate.
3. Connect the Q outputs of the identified flip-flops to the inputs of the NAND gate and the NAND gate output to asynchronous clear inputs of all flip-flops.

11.4.2 Ripple Counters in IC Form

In this section, we will look at the internal logic diagram of a typical binary ripple counter and see how close its architecture is to the ripple counter described in the previous section. Let us consider binary ripple counter type number 74293. It is a four-bit binary ripple counter containing four master–slave-type *J-K* flip-flops with additional gating to provide a divide-by-2 counter and a three-stage MOD-8 counter. Figure 11.4 shows the internal logic diagram of this counter. To get the full binary sequence of 16 states, the Q output of the LSB flip-flop is connected to the B input, which is the clock input of the next higher flip-flop. The arrangement then becomes the same as that shown in Fig. 11.2(a), with the exception of the two-input NAND gate of Fig. 11.4, which has been included here for providing the clearing features. The counter can be cleared to the 0000 logic state by driving both RESET inputs to the logic HIGH state. Tables 11.2 and 11.3 respectively give the functional table and the count sequence.

Example 11.3

Refer to the binary ripple counter of Fig. 11.5. Determine the modulus of the counter and also the frequency of the flip-flop Q_3 output.

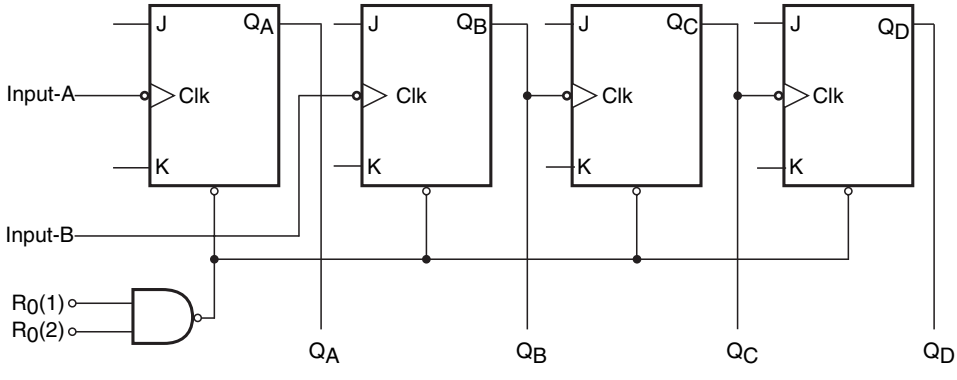


Figure 11.4 Logic diagram of IC 74293.

Table 11.2 Functional table for binary ripple counter, type number 74293.

RESET inputs		Outputs			
$R_0(1)$	$R_0(2)$	Q_D	Q_C	Q_B	Q_A
H	H	L	L	L	L
L	X		Count		
X	L		Count		

Table 11.3 Count sequence for binary ripple counter, type number 74293.

Count	Outputs			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

Solution

- The counter counts in the natural sequence from 0000 to 1011.
- The moment the counter goes to 1100, the NAND output goes to the logic '0' state and immediately clears the counter to the 0000 state.
- Thus, the counter is not able to stay in the 1100 state. It has only 12 stable states from 0000 to 1011.
- Therefore, the modulus of the counter = 12.
- The Q_3 output is the input clock frequency divided by 12.
- Therefore, the frequency of the Q_3 output waveform $= 1.2 \times 10^3/12 = 100$ kHz.

Example 11.4

Design a binary ripple counter that counts 000 and 111 and skips the remaining six states, that is, 001, 010, 011, 100, 101 and 110. Use presentable, clearable negative edge-triggered J-K flip-flops with active LOW PRESET and CLEAR inputs. Also, draw the timing waveforms and determine the frequency of different flip-flop outputs for a given clock frequency, f_c.

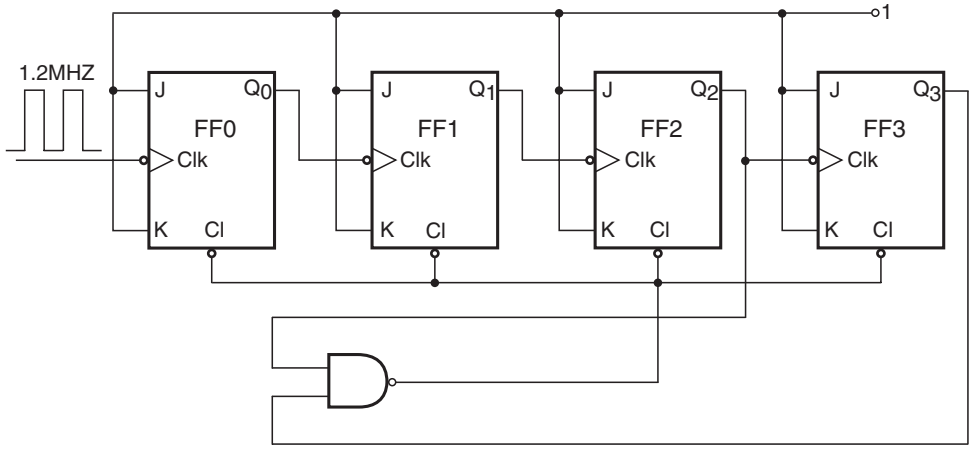


Figure 11.5 Example 11.3.

Solution

The counter is required to go to the 111 state from the 000 state with the first relevant clock transition. The second transition brings it back to the 000 state. That is, the three flip-flops toggle from logic '0' state to logic '1' state with every odd-numbered clock transition, and also the three flip-flops toggle from logic '1' state to logic '0' state with every even-numbered clock transition. Figure 11.6(a) shows the arrangement. The PRESET inputs of the three flip-flops have been tied to the NAND output whose inputs are $Q_A, \overline{Q_B}$ and $\overline{Q_C}$. Every time the counter is in the 000 state and is clocked, the NAND output momentarily goes from logic '1' state to logic '0' state, thus presetting the Q_A, Q_B and Q_C outputs to the logic '1' state. The timing waveforms as shown in Fig. 11.6(b) are self-explanatory.

The Q_A, $Q_B$and $Q_C$waveforms are identical, and each of them has a frequency of $f_c/2$, where f_c is the clock frequency.

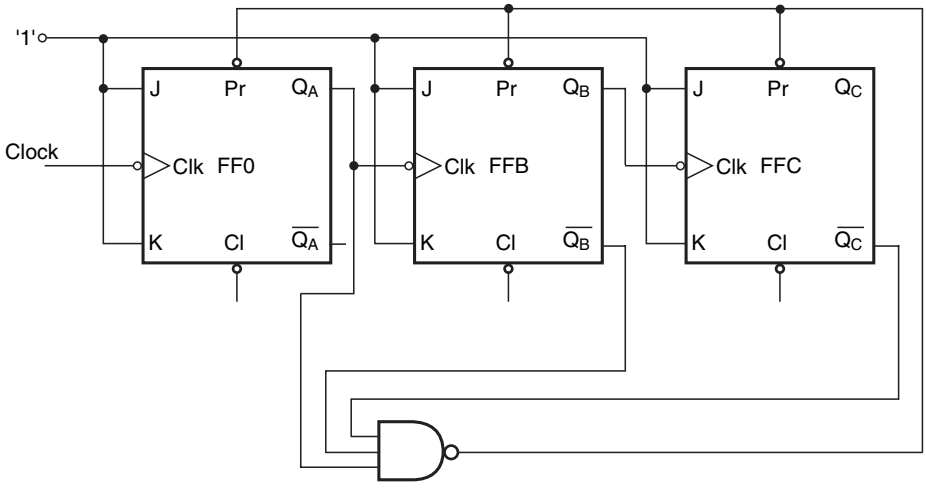
Example 11.5

Refer to the binary ripple counter arrangement of Fig. 11.7. Write its count sequence if it is initially in the 0000 state. Also draw the timing waveforms.

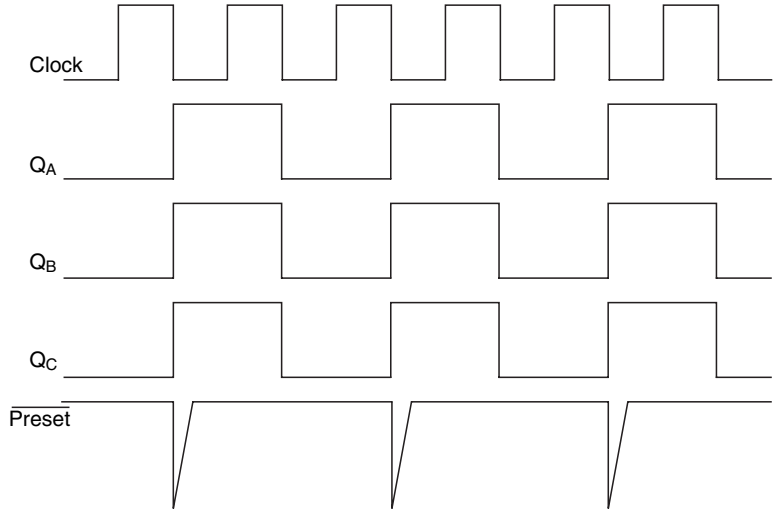
Solution

The counter is initially in the 0000 state. With the first clock pulse, Q_0 toggles from the '0' to the '1' state, which means $\overline{Q_0}$ toggles from '1' to '0'. Since $\overline{Q_0}$ here feeds the clock input of next flip-flop, flip-flop FF1 also toggles. Thus, Q_1 goes from '0' to '1'. Since flip-flops FF2 and FF3 are also clocked from complementary outputs of their immediately preceding flip-flops, they also toggle. Thus, the counter moves from the 0000 state to the 1111 state with the first clock pulse.

With the second clock pulse, Q_0 toggles again, but the other flip-flops remain unaffected for obvious reasons and the counter is in the 1110 state. With subsequent clock pulses, the counter keeps counting downwards by one LSB at a time until it reaches 0000 again, after which the process repeats. The count sequence is given as 0000, 1111, 1110, 1101,1100, 1011, 1010, 1001, 1000,



(a)



(b)

Figure 11.6 Example 11.4.

0111, 0110, 0101, 0100, 0011, 0010, 0001 and 0000. The timing waveforms are shown in Fig. 11.8. Thus, we have a four-bit counter that counts in the reverse sequence, beginning with the maximum count. This is a DOWN counter. This type of counter is discussed further in the subsequent paragraphs.

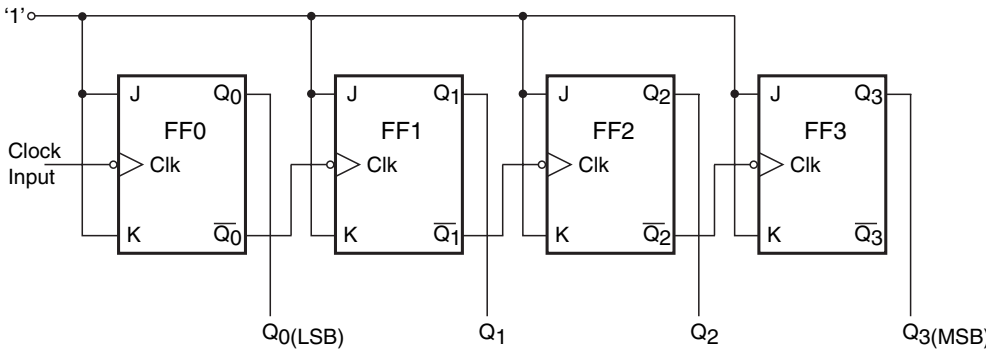


Figure 11.7 Counter schematic, example 11.5.

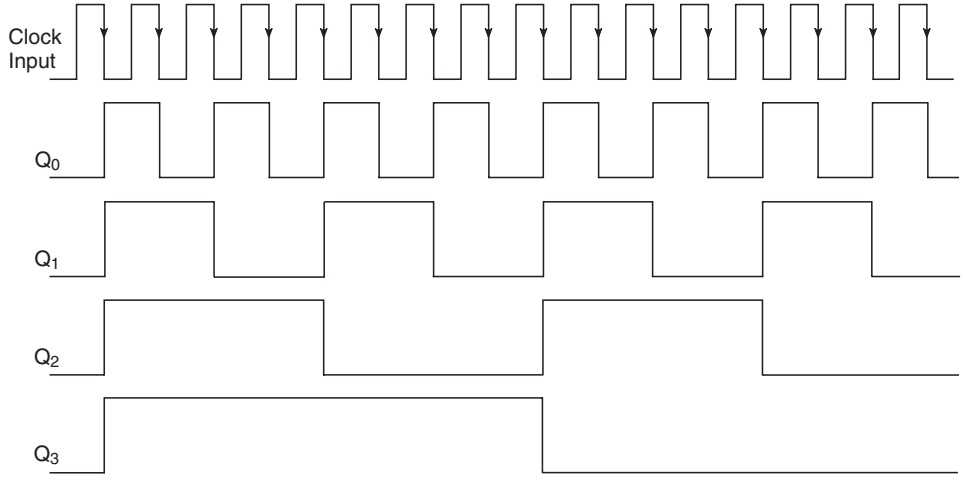


Figure 11.8 Timing waveforms, example 11.5.

From what we have discussed for a binary ripple counter, including the solved examples given to supplement the text, we can make the following observations:

1. If the flip-flops used to construct the counter are negative (HIGH-to-LOW) edge triggered and the clock inputs are fed from Q outputs, the counter counts in the normal upward count sequence.
2. If the flip-flops used to construct the counter are negative edge triggered and the clock inputs are fed from $\overline{Q}$ outputs, the counter counts in the reverse or downward count sequence.
3. If the flip-flops used to construct the counter are positive (LOW-to-HIGH) edge triggered and the clock inputs are fed from Q outputs, the counter counts in the reverse or downward count sequence.
4. If the flip-flops used to construct the counter are positive edge triggered and the clock inputs are fed from the $\overline{Q}$ outputs, the counter counts in the normal upward count sequence.

11.5 Synchronous (or Parallel) Counters

Ripple counters discussed thus far in this chapter are asynchronous in nature as the different flip-flops comprising the counter are not clocked simultaneously and in synchronism with the clock pulses. The total propagation delay in such a counter, as explained earlier, is equal to the sum of propagation delays due to different flip-flops. The propagation delay becomes prohibitively large in a ripple counter with a large count. On the other hand, in a synchronous counter, all flip-flops in the counter are clocked simultaneously in synchronism with the clock, and as a consequence all flip-flops change state at the same time. The propagation delay in this case is independent of the number of flip-flops used.

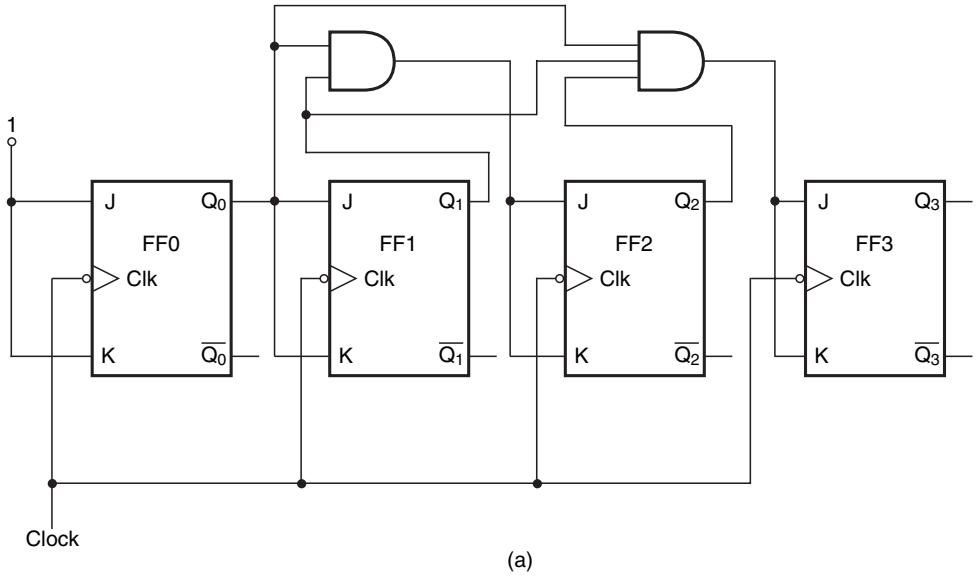
Since the different flip-flops in a synchronous counter are clocked at the same time, there needs to be additional logic circuitry to ensure that the various flip-flops toggle at the right time. For instance, if we look at the count sequence of a four-bit binary counter shown in Table 11.4, we find that flip-flop FF0 toggles with every clock pulse, flip-flop FF1 toggles only when the output of FF0 is in the '1' state, flip-flop FF2 toggles only with those clock pulses when the outputs of FF0 and FF1 are both in the logic '1' state and flip-flop FF3 toggles only with those clock pulses when Q_0, Q_1 and Q_2 are all in the logic '1' state. Such logic can be easily implemented with AND gates. Figure 11.9(a) shows the schematic arrangement of a four-bit synchronous counter. The timing waveforms are shown in Fig. 11.9(b). The diagram is self-explanatory. As an example, ICs 74162 and 74163 are four-bit synchronous counters, with the former being a decade counter and the latter a binary counter.

A synchronous counter that counts in the reverse or downward sequence can be constructed in a similar manner by using complementary outputs of the flip-flops to drive the J and K inputs of the following flip-flops. Refer to the reverse or downward count sequence as given in Table 11.5. As is evident from the table, FF0 toggles with every clock pulse, FF1 toggles only when Q_0 is logic '0', FF2 toggles only when both Q_0 and Q_1 are in the logic '0' state and FF3 toggles only when Q_0, Q_1 and Q_2 are in the logic '0' state.

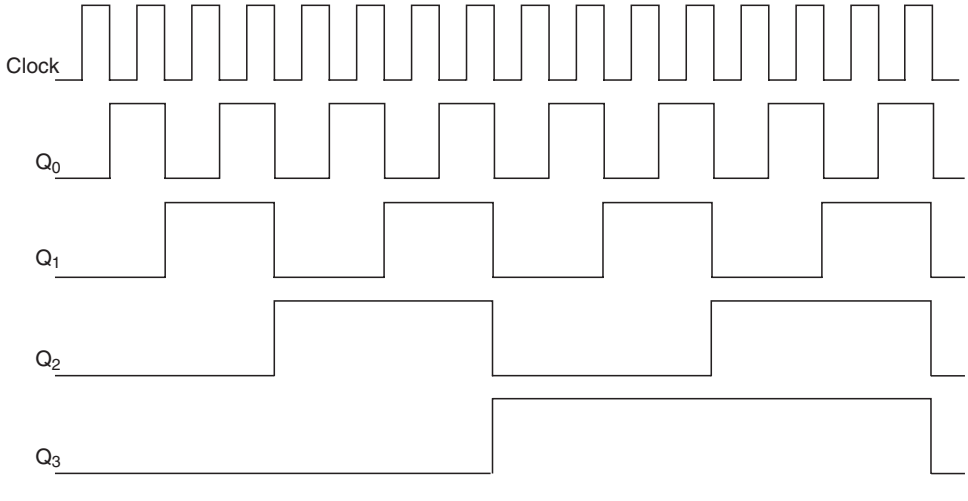
Referring to the four-bit synchronous UP counter of Fig. 11.9(a), if the J and K inputs of flip-flop FF1 are fed from the $\overline{Q_0}$ output instead of the Q_0 output, the inputs to the two-input AND gate are $\overline{Q_0}$ and $\overline{Q_1}$ instead of Q_0 and Q_1, and the inputs to the three-input AND gate are $\overline{Q_0}$, $\overline{Q_1}$ and $\overline{Q_2}$ instead of Q_0, Q_1 and Q_2, we get a counter that counts in reverse order. In that case it becomes a four-bit synchronous DOWN counter.

Table 11.4 Count sequence of a four-bit binary counter.

Count	Q_3	Q_2	Q_1	Q_0	Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1



(a)



(b)

Figure 11.9 Four-bit synchronous counter.

Table 11.5 Reverse or downward count sequence synchronous counter.

Count	Q_3	Q_2	Q_1	Q_0	Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	8	1	0	0	0
1	1	1	1	1	9	0	1	1	1
2	1	1	1	0	10	0	1	1	0
3	1	1	0	1	11	0	1	0	1
4	1	1	0	0	12	0	1	0	0
5	1	0	1	1	13	0	0	1	1
6	1	0	1	0	14	0	0	1	0
7	1	0	0	1	15	0	0	0	1

11.6 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. As outlined in Section 11.5, 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 as 0000, 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, IC 74191 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 11.10 shows 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

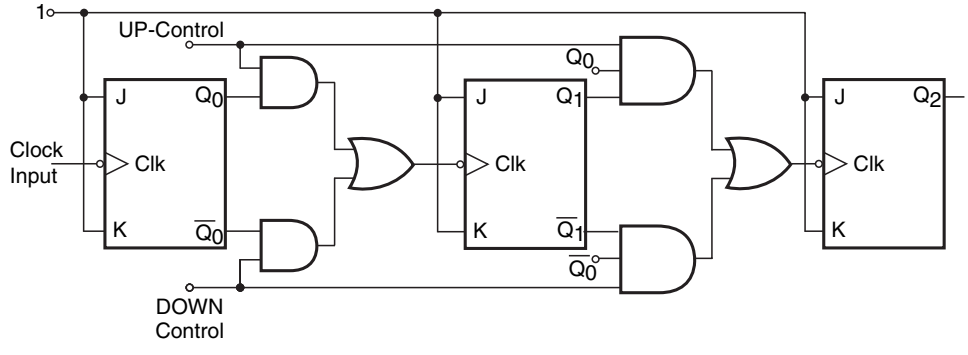


Figure 11.10 Four-bit UP/DOWN counter.

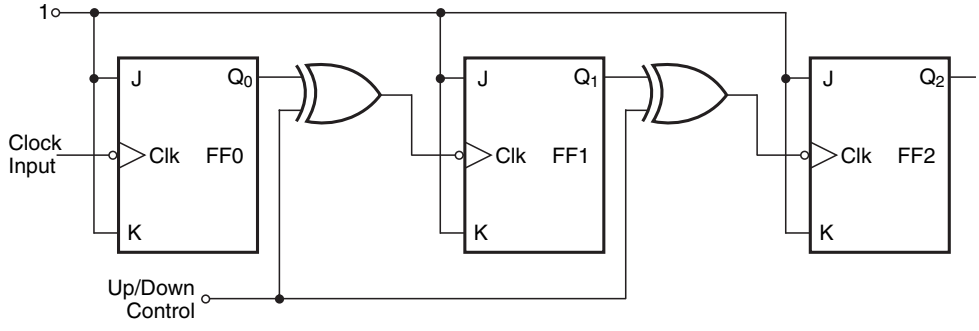


Figure 11.11 Three-bit UP/DOWN counter with a common clock input.

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 11.11 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.

11.7 Decade and BCD Counters

A *decade counter* is one that goes through 10 unique output combinations and then resets as the clock proceeds further. Since it is an MOD-10 counter, it can be constructed with a minimum of four flip-flops. A four-bit counter would have 16 states. By skipping any of the six states by using some kind of feedback or some kind of additional logic, we can convert a normal four-bit binary counter into a decade counter. A decade counter does not necessarily count from 0000 to 1001. It could even count as 0000, 0001, 0010, 0101, 0110, 1001, 1010, 1100, 1101, 1111, 0000, . . . In this count sequence, we have skipped 0011, 0100, 0111, 1000, 1011 and 1110.

A *BCD counter* is a special case of a decade counter in which the counter counts from 0000 to 1001 and then resets. The output weights of flip-flops in these counters are in accordance with 8421-code. For instance, at the end of the seventh clock pulse, the counter output will be 0111, which is the binary equivalent of decimal 7. In other words, different counter states in this counter are binary equivalents of the decimal numbers 0 to 9. These are different from other decade counters that provide the same count by using some kind of forced feedback to skip six of the natural binary counts.

11.8 Presettable Counters

Presettable counters are those that can be preset to any starting count either asynchronously (independently of the clock signal) or synchronously (with the active transition of the clock signal). The presetting operation is achieved with the help of PRESET and CLEAR (or MASTER RESET) inputs available on the flip-flops. The presetting operation is also known as the 'preloading' or simply the 'loading' operation.

Presettable counters can be UP counters, DOWN counters or UP/DOWN counters. Additional inputs/outputs available on a presettable UP/DOWN counter usually include PRESET inputs, from where any desired count can be loaded, parallel load (*PL*) inputs, which when active allow the PRESET inputs to be loaded onto the counter outputs, and terminal count (*TC*) outputs, which become active when the counter reaches the terminal count.

Figure 11.12 shows the logic diagram of a four-bit presettable synchronous UP counter. The data available on P_3, P_2, P_1 and P_0 inputs are loaded onto the counter when the parallel load ($\overline{PL}$) input goes LOW.

When the $\overline{PL}$ input goes LOW, one of the inputs of all NAND gates, including the four NAND gates connected to the PRESET inputs and the four NAND gates connected to the CLEAR inputs, goes to the logic '1' state. What reaches the PRESET inputs of FF3, FF2, FF1 and FF0 is $\overline{P_3}$, $\overline{P_2}$, $\overline{P_1}$ and $\overline{P_0}$ respectively, and what reaches their CLEAR inputs is P_3, P_2, P_1 and P_0 respectively. Since PRESET and CLEAR are active LOW inputs, the counter flip-flops FF3, FF2, FF1 and FF0 will respectively be loaded with P_3, P_2, P_1 and P_0. For example, if $P_3 = 1$, the PRESET and CLEAR inputs of FF3 will be in the '0' and '1' logic states respectively. This implies that the Q_3 output will go to the logic '1' state. Thus, FF3 has been loaded with P_3. Similarly, if $P_3 = 0$, the PRESET and CLEAR inputs of flip-flop FF3 will be in the '1' and '0' states respectively. The flip-flop output (Q_3 output) will be cleared to the '0' state. Again, the flip-flop is loaded with P_3 logic status when the $\overline{PL}$ input becomes active.

Counter ICs 74190, 74191, 74192 and 74193 are asynchronously presettable synchronous UP/DOWN counters. Many synchronous counters use synchronous presetting whereby the counter is preset or loaded with the data on the active transition of the same clock signal that is used for counting. Presettable counters also have terminal count ($\overline{TC}$) outputs, which allow them to be cascaded together to get counters with higher MOD numbers. In the cascade arrangement, the terminal count output of the lower-order counter feeds the clock input of the next higher-order counter. Cascading of counters is discussed in Section 11.10.

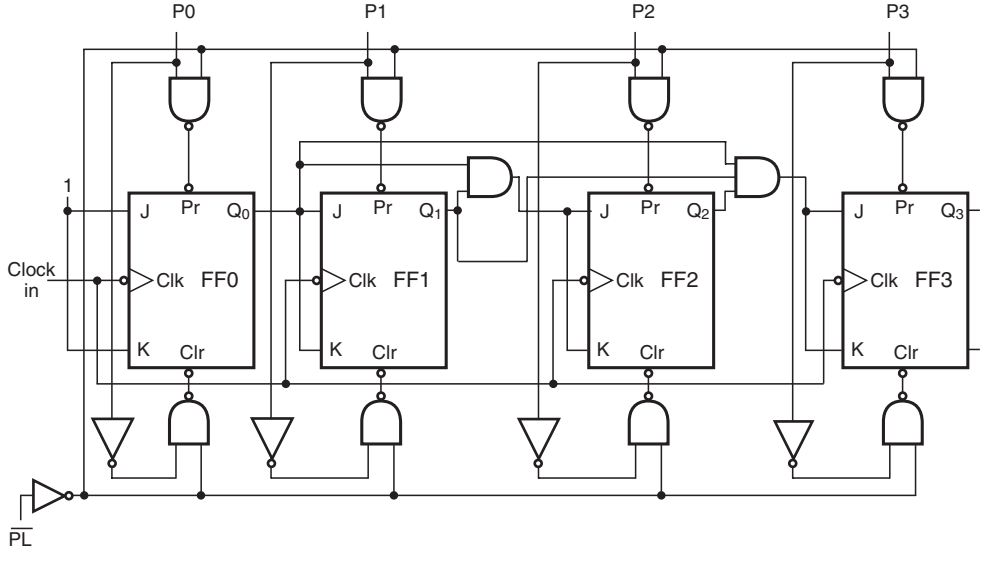


Figure 11.12 Four-bit presettable, clearable counter.

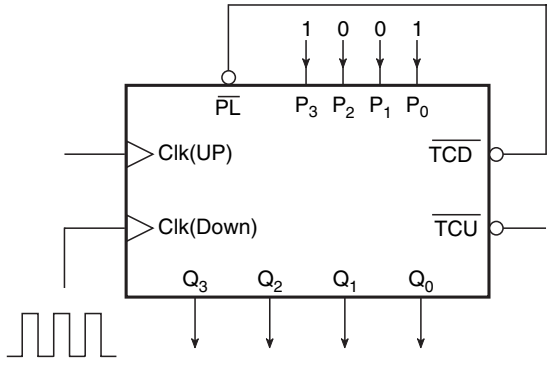


Figure 11.13 Presettable four-bit counter.

11.8.1 Variable Modulus with Presettable Counters

Presettable counters can be wired as counters with a modulus of less than 2^N without the need for any additional logic circuitry. When a presettable counter is preset with a binary number whose decimal equivalent is some number 'X', and if this counter is wired as a DOWN counter, with its terminal count (DOWN mode) output, also called borrow-out (B_o), fed back to the parallel load (PL) input, it works like an MOD-X counter.

We will illustrate this with the help of an example. Refer to Fig. 11.13. It shows a presettable four-bit synchronous UP/DOWN binary counter having separate clock inputs for UP and DOWN counting (both positive edge triggered), an active LOW parallel load input ($\overline{PL}$) and active LOW terminal count UP ($\overline{TCU}$) and terminal count DOWN ($\overline{TCD}$) outputs. This description is representative of IC counter type 74193. Let us assume that the counter is counting down and is presently in the 1001 state at time instant t_0. The $\overline{TCD}$ output is in the logic '1' state, and so is the $\overline{PL}$ input. That is, both are inactive. The counter counts down by one LSB at every positive-going edge of the clock input. Immediately after the ninth positive-going trigger (at time instant t_9), the counter is in the 0000 state, which is the terminal count. Coinciding with the negative-going edge of the same clock pulse, the $\overline{TCD}$ output goes to the logic '0' state, and so does the $\overline{PL}$ input. This loads the counter with 1001 at time instant t_{10}, as shown in the timing waveforms of Fig. 11.14. With the positive-going edges of the tenth clock pulse and thereafter, the counter repeats its DOWN count sequence. Examination of the Q_3 output waveform tells that its frequency is one-ninth of the input clock frequency. Thus, it is an MOD-9 counter. The modulus of the counter can be varied by varying the data loaded onto the parallel PRESET/LOAD inputs.

11.9 Decoding a Counter

The output state of a counter at any time instant, as it is being clocked, is in the form of a sequence of binary digits. For a large number of applications, it is important to detect or decode different states of the counter whose number equals the modulus of the counter. One typical application could be a need to initiate or trigger some action after the counter reaches a specific state. The decoding network therefore is going to be a logic circuit that takes its inputs from the outputs of the different flip-flops constituting the counter and then makes use of those data to generate outputs equal to the modulus or MOD-number of the counter.

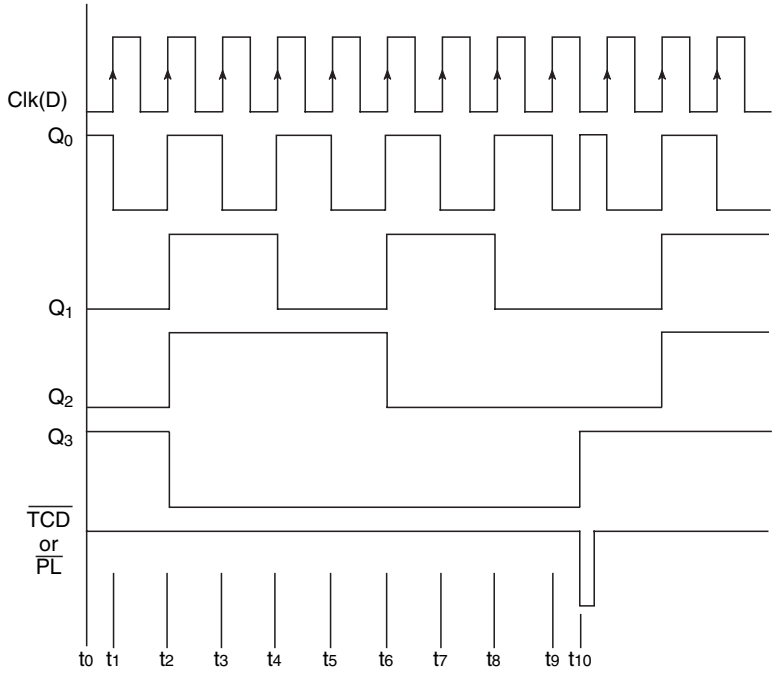
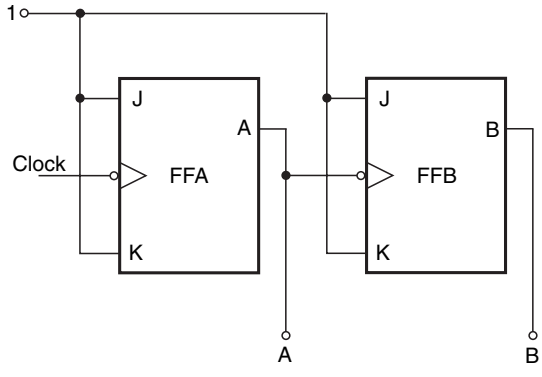


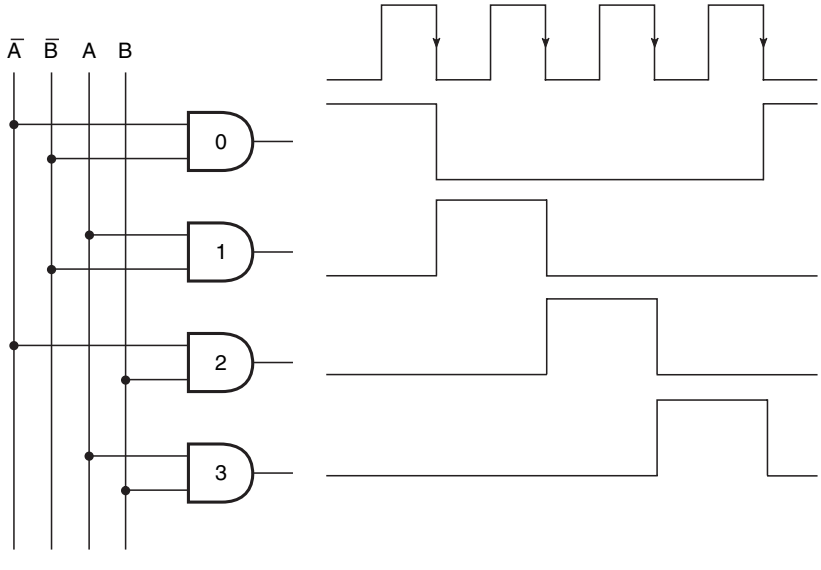
Figure 11.14 Timing waveforms for the counter of Fig. 11.13.

Depending upon the logic status of the decoded output, there are two basic types of decoding, namely *active HIGH* decoding and *active LOW* decoding. In the case of the former the decoder outputs are normally LOW, and for a given counter state the corresponding decoder output goes to the logic HIGH state. In the case of active LOW decoding, the decoder outputs are normally HIGH and the decoded output representing the counter state goes to the logic LOW state.

We will further illustrate the concept of decoding a counter with the help of an example. Consider the two-stage MOD-4 ripple counter of Fig. 11.15(a). This counter has four possible logic states, which need to be decoded. These include 00, 01, 10 and 11. Let us now consider the arrangement of four two-input AND gates as shown in Fig. 11.15(b) and what their outputs look like as the counter clock goes through the first four pulses. Before we proceed further, we have two important observations to make. Firstly, the number of AND gates used in the decoder network equals the number of logic states to be decoded, which further equals the modulus of the counter. Secondly, the number of inputs to each AND gate equals the number of flip-flops used in the counter. We can see from the waveforms of Fig. 11.15(b) that, when the counter is in the 00 state, the AND gate designated '0' is in the logic HIGH state and the outputs of the other gates designated '1', '2' and '3' are in the logic LOW state. Similarly, for 01, 10 and 11 states of the counter, the outputs of gates 1, 2 and 3 are respectively in the logic HIGH state. This is incidentally active HIGH decoding. We can visualize that, if the AND gates were replaced with NAND gates, with the inputs to the gates remaining the same, we would get an active LOW decoder. For a counter that uses N flip-flops and has a modulus of 'X', the decoder will have 'X' number of N-input AND or NAND gates, depending upon whether we want an active HIGH or active LOW decoder.



(a)



(b)

Figure 11.15 MOD-4 ripple counter with decoding logic.

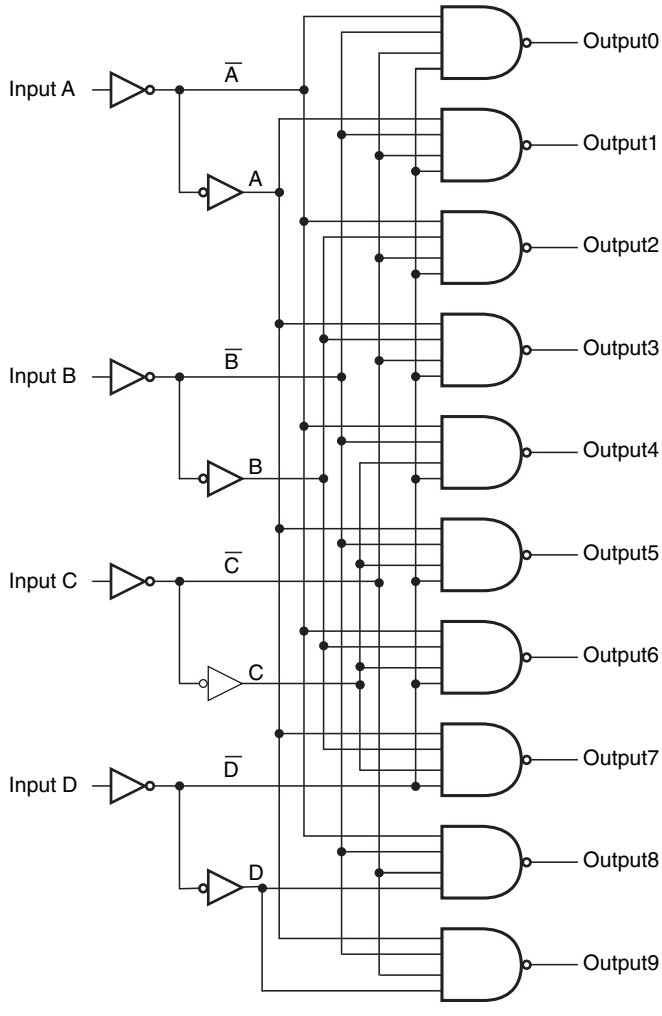


Figure 11.16 Logic diagram of four-line BCD-to-decimal decoder (IC 7442).

Figure 11.16 shows the logic diagram of a four-line BCD to decimal decoder with active low outputs. Full decoding of valid input logic states ensures that all outputs remain off or inactive for all invalid input conditions. Table 11.6 gives the functional table of the decoder of Fig. 11.16. The logic diagram shown in Fig. 11.16 is the actual logic diagram of IC 7442, which is a four-line BCD to decimal decoder in the TTL family.

The decoding gates used to decode the states of a ripple counter produce glitches (or spikes) in the decoded waveforms. These glitches basically result from the cumulative propagation delay as we move from one flip-flop to the next in a ripple counter. It can be best illustrated with the help of the MOD-4 counter shown in Fig. 11.17. The timing waveforms are shown in Fig. 11.18 and are self-explanatory.

Table 11.6 Functional table of the decoder of Fig. 11.16.

Decimal number	BCD input				Decimal output									
	D	*C*	*B*	*A*	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
Invalid	H	L	H	L	H	H	H	H	H	H	H	H	H	H
Invalid	H	L	H	H	H	H	H	H	H	H	H	H	H	H
Invalid	H	H	L	L	H	H	H	H	H	H	H	H	H	H
Invalid	H	H	L	H	H	H	H	H	H	H	H	H	H	H
Invalid	H	H	H	L	H	H	H	H	H	H	H	H	H	H
Invalid	H	H	H	H	H	H	H	H	H	H	H	H	H	H

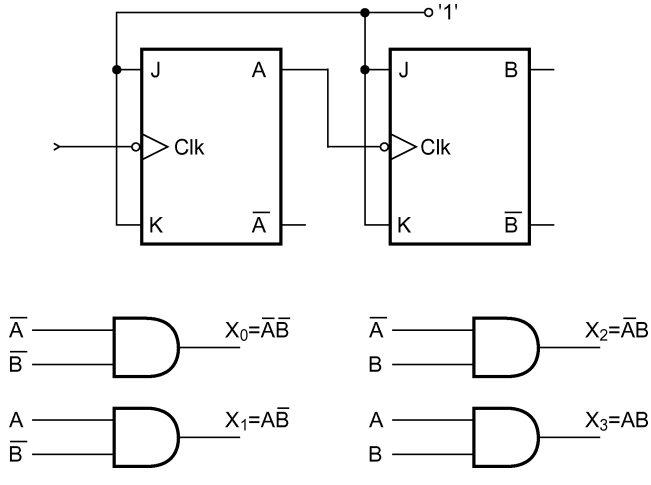


Figure 11.17 MOD-4 counter with decoding gates.

We can see the appearance of glitches at the output of decoding gates that decode X_0 and X_2 states.This problem for all practical purposes is absent in synchronous counters. Theoretically, it can even exist in a synchronous counter if the flip-flops used have different propagation delays.

One way to overcome this problem is to use a strobe signal which keeps the decoding gates disabled until all flip-flops have reached a stable state in response to the relevant clock transition. To implement

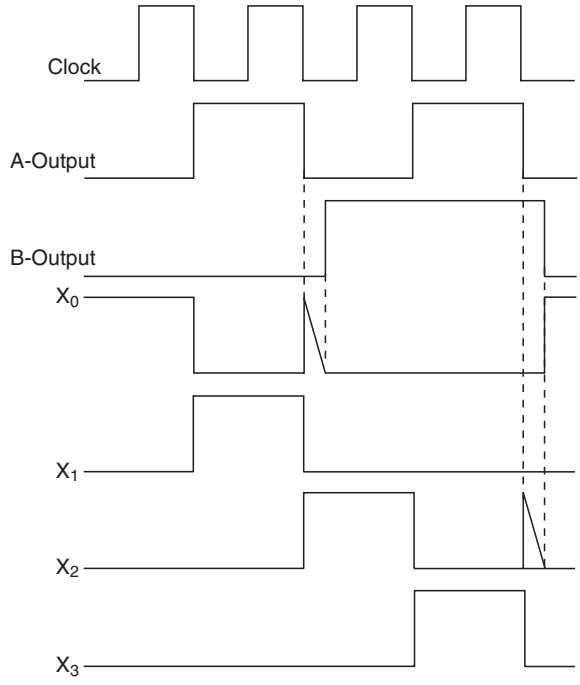


Figure 11.18 Glitch problem in decoders.

this, each of the decoding gates will have an additional input. This additional input of all decoding gates is tied together and the strobe signal applied to the common point.

One such decoder with additional strobe inputs to take care of glitch-related problems is IC 74154, which is a four-line to 16-line decoder in the TTL family. Figure 11.19 shows the internal logic diagram of IC 74154. We can see all NAND gates having an additional input line, which is controlled by strobe inputs $\overline{G}_1$ and $\overline{G}_2$.

11.10 Cascading Counters

A cascade arrangement allows us to build counters with a higher modulus than is possible with a single stage. The terminal count outputs allow more than one counter to be connected in a cascade arrangement. In the following paragraphs, we will examine some such cascade arrangements in the case of binary and BCD counters.

11.10.1 Cascading Binary Counters

In order to construct a multistage UP counter, all counter stages are connected in the count UP mode. The clock is applied to the clock input of a lowest-order counter, the terminal count UP (*TCU*), also called the carry-out (C_o), of this counter is applied to the clock input of the next higher counter stage

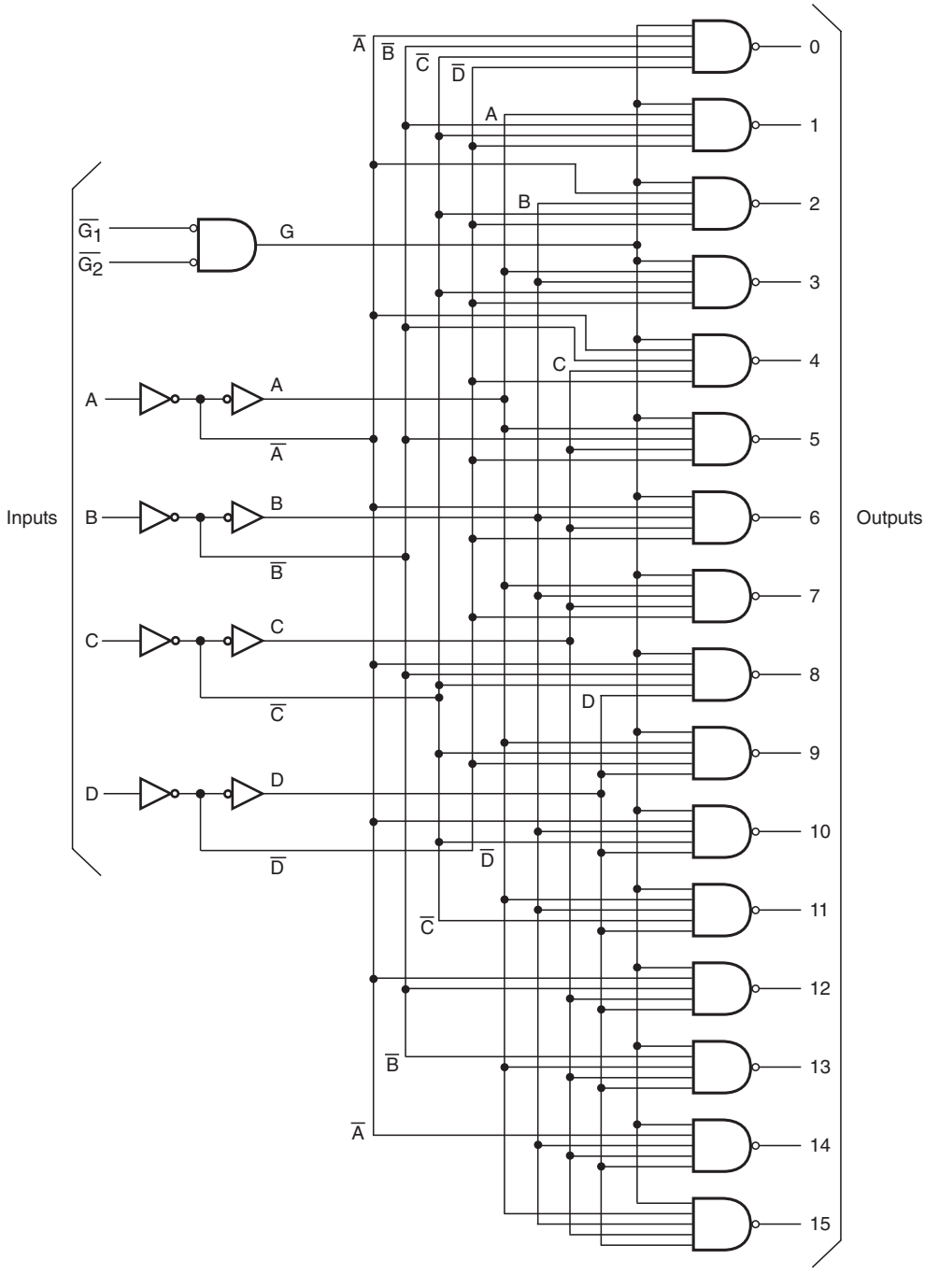


Figure 11.19 Logic diagram of IC 74154.

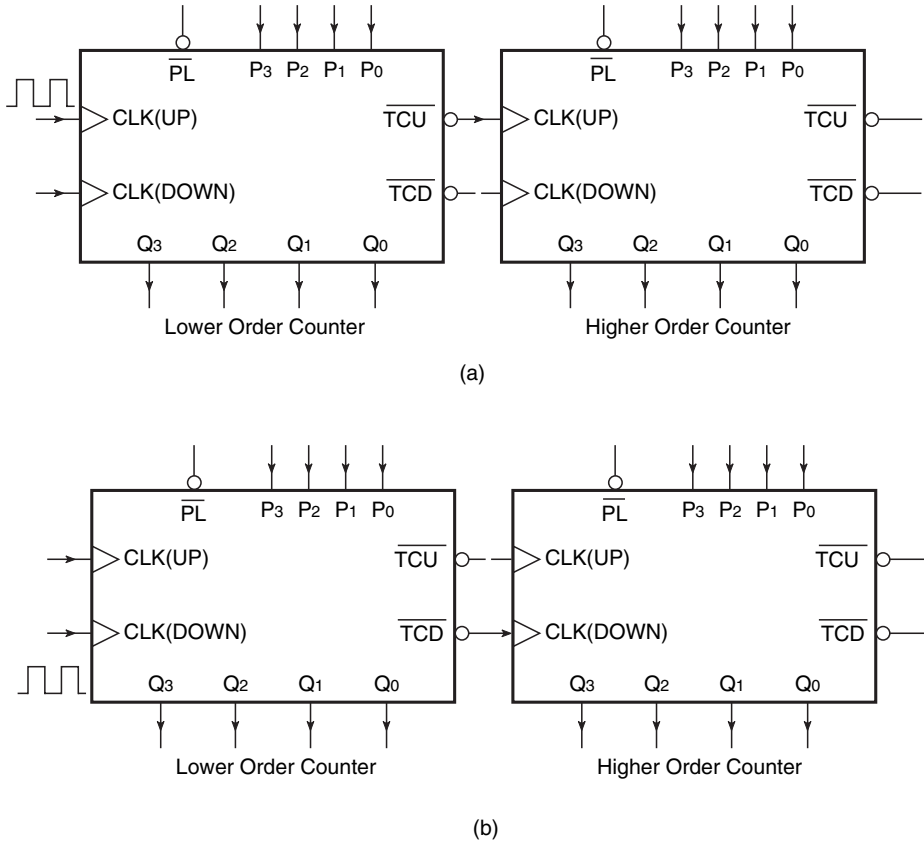


Figure 11.20 Cascading binary counters.

and the process continues. If it is desired to build a multistage DOWN counter, all counters are wired as DOWN counters, the clock is applied to the clock input of the lowest-order counter and the terminal count DOWN (*TCD*), also called the borrow-out (B_o), of the lowest-order counter is applied to the clock input of the next higher counter stage. The process continues in the same fashion, with the TCD output of the second stage feeding the clock input of the third stage and so on. The modulus of the multistage counter arrangement equals the product of the moduli of individual stages. Figures 11.20(a) and (b) respectively show two-stage arrangements of four-bit synchronous UP and DOWN counters respectively.

11.10.2 Cascading BCD Counters

BCD counters are used when the application involves the counting of pulses and the result of counting is to be displayed in decimal. A single-stage BCD counter counts from 0000 (decimal equivalent '0') to 1001 (decimal equivalent '9') and thus is capable of counting up to a maximum of nine pulses. The output in a BCD counter is in binary coded decimal (BCD) form. The BCD output needs

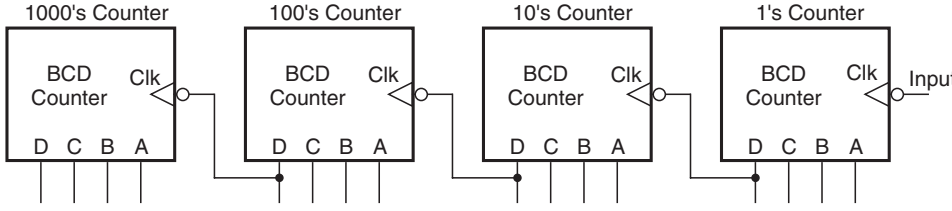


Figure 11.21 Cascading BCD counters.

to be decoded appropriately before it can be displayed. Decoding a counter has been discussed in the previous section. Coming back to the question of counting pulses, more than one BCD counter stage needs to be used in a cascade arrangement in order to be able to count up to a larger number of pulses. The number of BCD counter stages to be used equals the number of decimal digits in the maximum number of pulses we want to count up to. With a maximum count of 9999 or 3843, both would require a four-stage BCD counter arrangement with each stage representing one decimal digit.

Figure 11.21 shows a cascade arrangement of four BCD counter stages. The arrangement works as follows. Initially, all four counters are in the all 0s state. The counter representing the decimal digit of 1's place is clocked by the pulsed signal that needs to be counted. The successive flip-flops are clocked by the MSB of the immediately previous counter stage. The first nine pulses take 1's place counter to 1001. The tenth pulse resets it to 0000, and '1' to '0' transition at the MSB of 1's place counter clocks 10's place counter. 10's place counter gets clocked on every tenth input clock pulse. On the hundredth clock pulse, the MSB of 10's counter makes a '1' to '0' transition which clocks 100's place counter. This counter gets clocked on every successive hundredth input clock pulse. On the thousandth input clock pulse, the MSB of 100's counter makes 'l' to '0' transition for the first time and clocks 1000's place counter. This counter is clocked thereafter on every successive thousandth input clock pulse. With this background, we can always tell the output state of the cascade arrangement. For example, immediately after the 7364th input clock pulse, the state of 1000's, 100's, 10's and 1's BCD counters would respectively be 0111, 0011, 0110 and 0100.

Example 11.6

Figure 11.22 shows a cascade arrangement of two 74190s. Both the UP/DOWN counters are wired as UP counters. What will be the logic status of outputs designated as A, B, C, D, E, F, G and H after the 34th clock pulse?

Solution

The cascade arrangement basically constitutes a two-stage BCD counter that can count from 0 to 99. The counter shown on the left forms 1's place counter, while the one on the right is 10's place counter. The ripple clock ($\overline{RC}$) output internally enabled by the terminal count ($\overline{\text{TC}}$) clocks 10's place counter on the tenth clock pulse and thereafter on every successive tenth clock pulse. At the end of the 34th clock pulse, 1's counter stores the binary equivalent of '4' and 10's counter stores the binary equivalent of '3'. Therefore, the logic status of A, B, C, D, E, F, G and H outputs will be 0, 0, 1, 0, 1,1, 0 and 0 respectively.

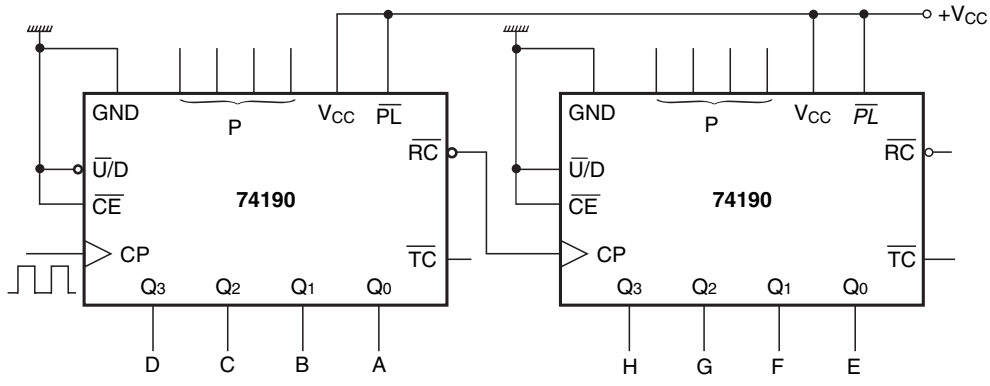


Figure 11.22 Cascade arrangement of two 74190s (example 11.6).

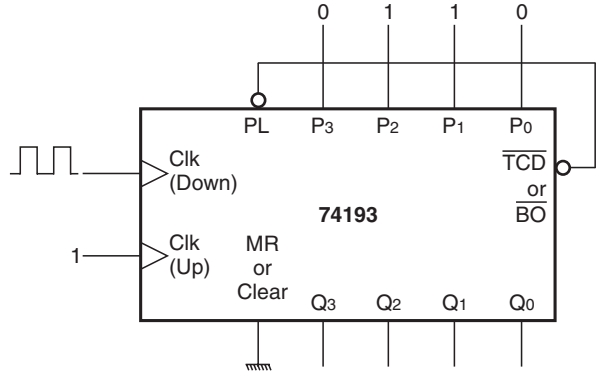


Figure 11.23 Presettable counter (example 11.7).

Example 11.7

Determine the modulus of the presettable counter shown in Fig. 11.23. If the counter were initially in the 0110 state, what would be the state of the counter immediately after the eighth clock pulse be?

Solution

- This presettable counter has been wired as a DOWN counter.
- The preset data input is 0110.
- Therefore, the modulus of the counter is 6 (the decimal equivalent of 0110).
- Now, the counter is initially in the 0110 state.
- Therefore, at the end of the sixth clock pulse, immediately after the leading edge of the sixth clock pulse, the counter will be in the 0000 state.

- 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, . . . :

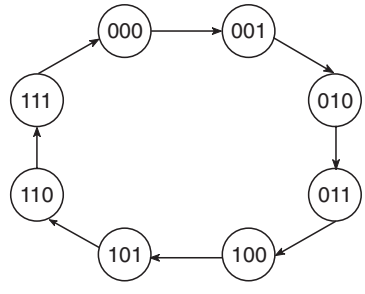


Figure 11.24 State transition diagram for an MOD-8 binary counter.

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

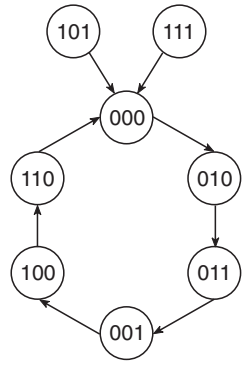


Figure 11.25 State transition diagram.

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:

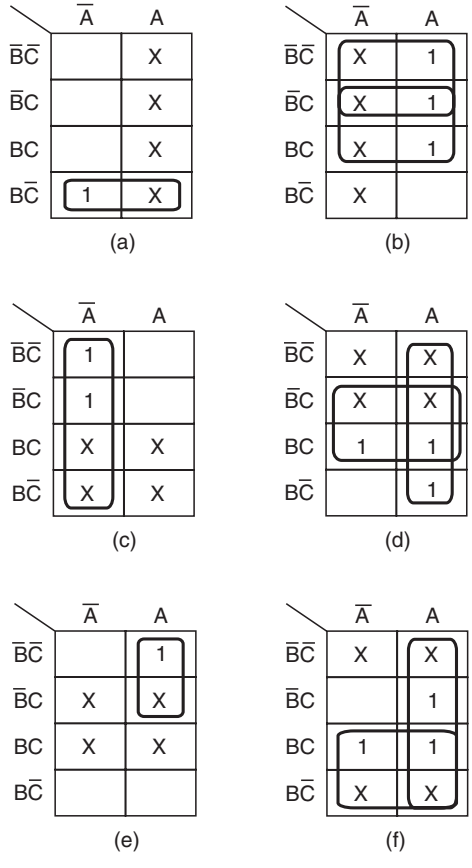


Figure 11.26 Karnaugh maps.

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

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

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

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

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

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

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

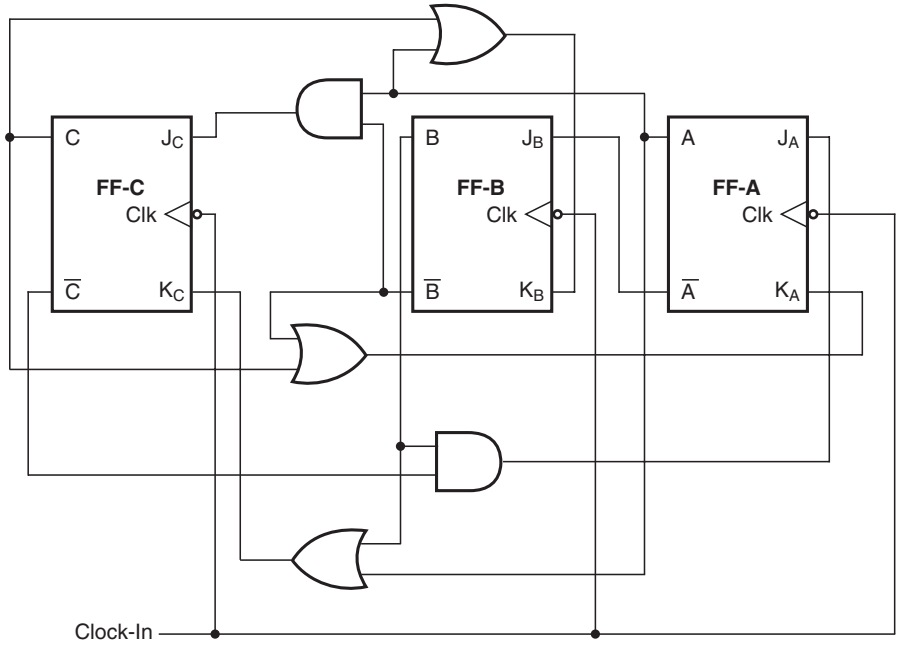


Figure 11.27 Counter with an arbitrary sequence.

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.

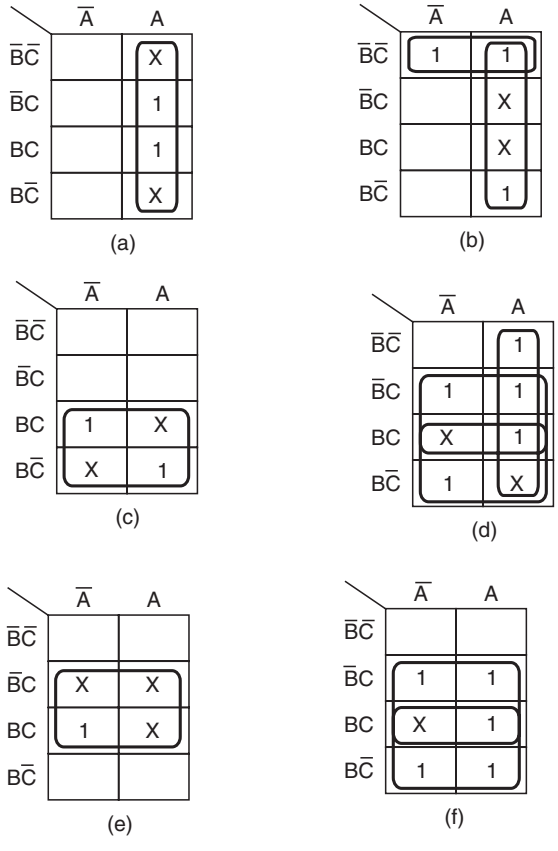


Figure 11.28 Karnaugh maps (example 11.8).

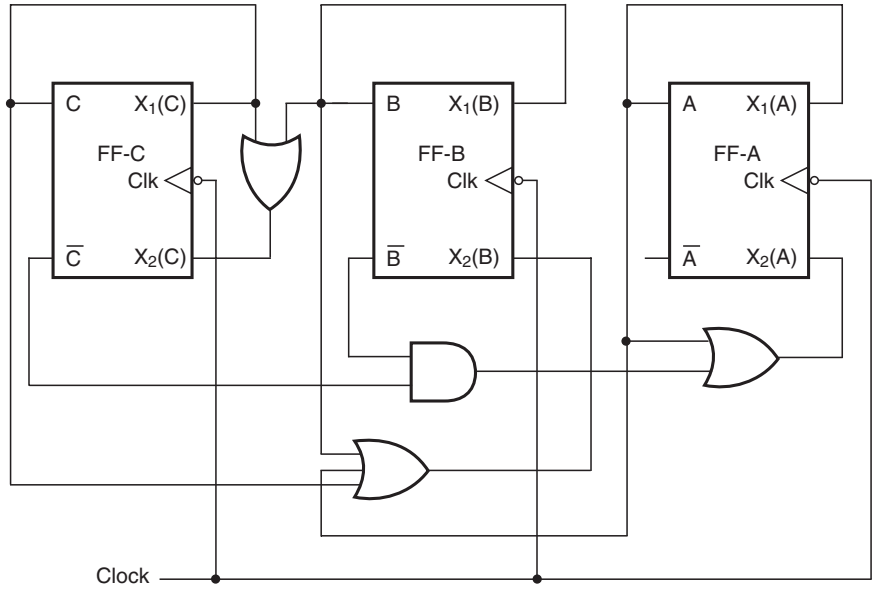


Figure 11.29 Counter circuit (example 11.8).

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:

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

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

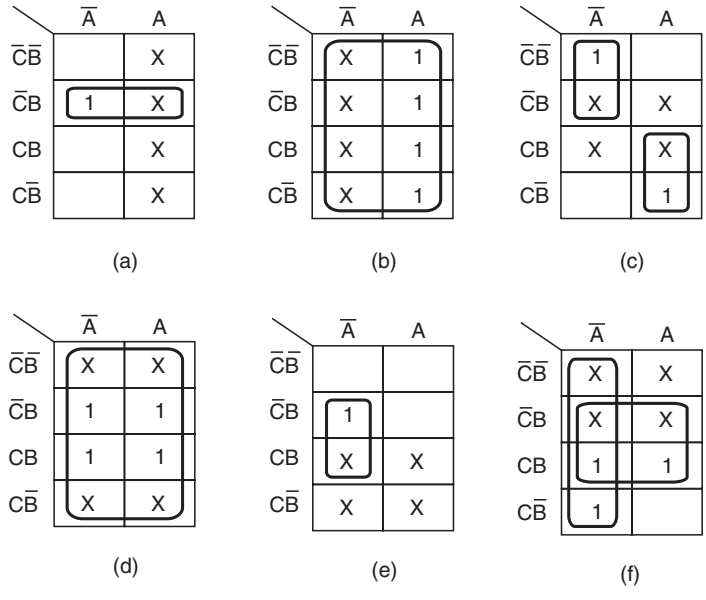


Figure 11.30 Karnaugh maps (example 11.9).

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

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

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

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

- 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.

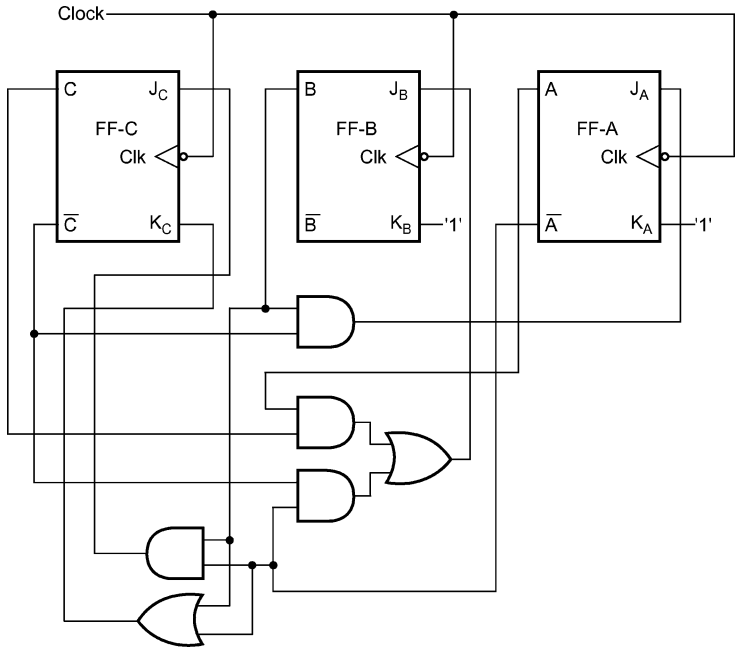


Figure 11.31 Hardware implementation of the counter circuit (example 11.9).

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

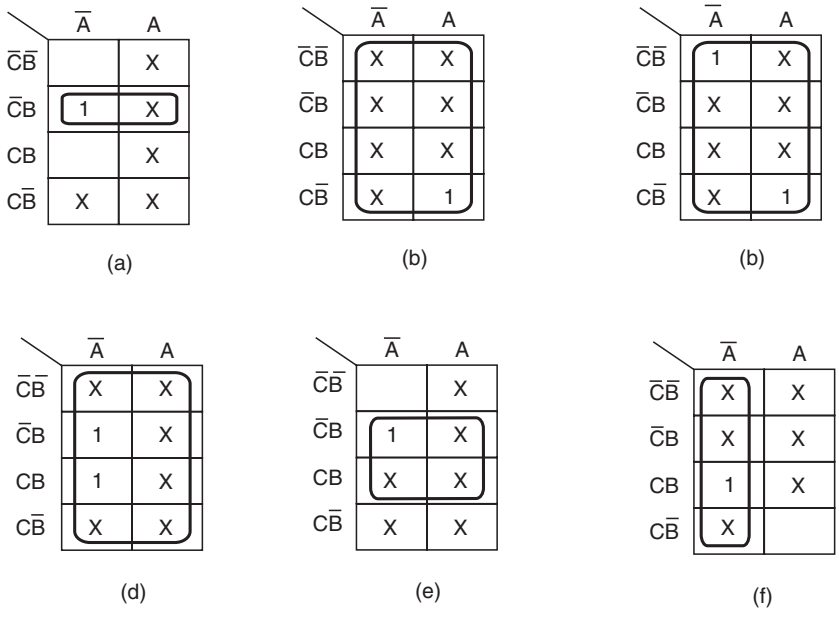


Figure 11.32 Modified Karnaugh maps (example 11.9).

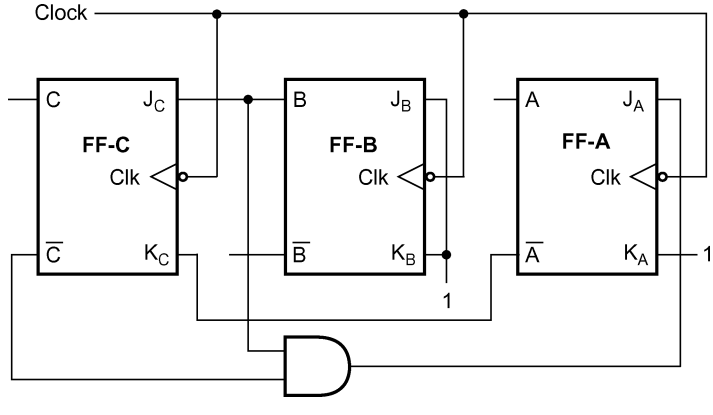


Figure 11.33 Hardware implementation of the counter circuit (example 11.9).

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'.

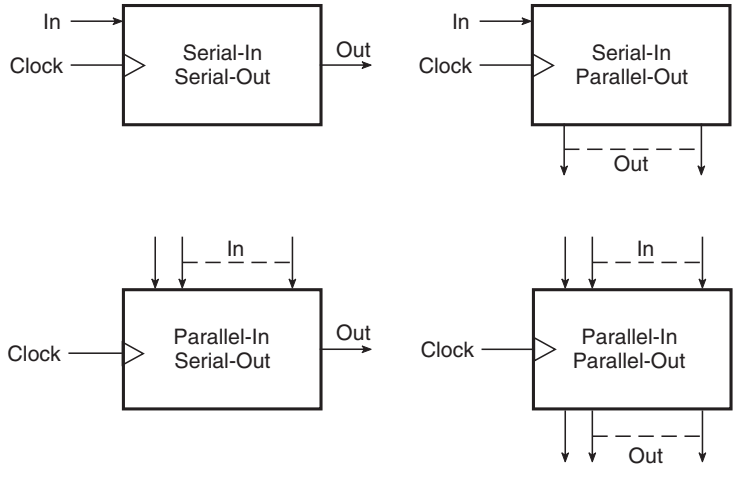


Figure 11.34 Circuit representation of shift registers.

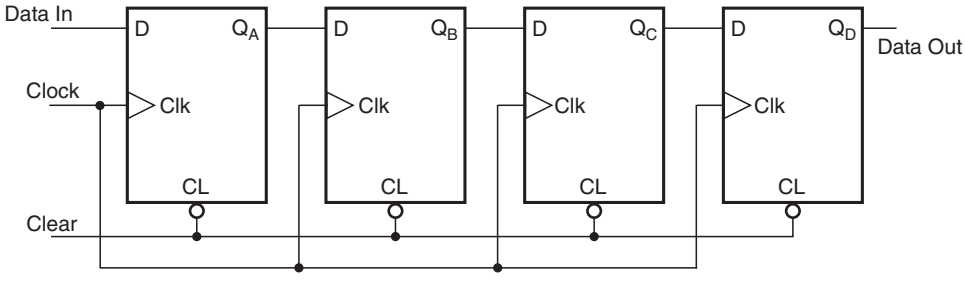


Figure 11.35 Serial-in, serial-out shift register.

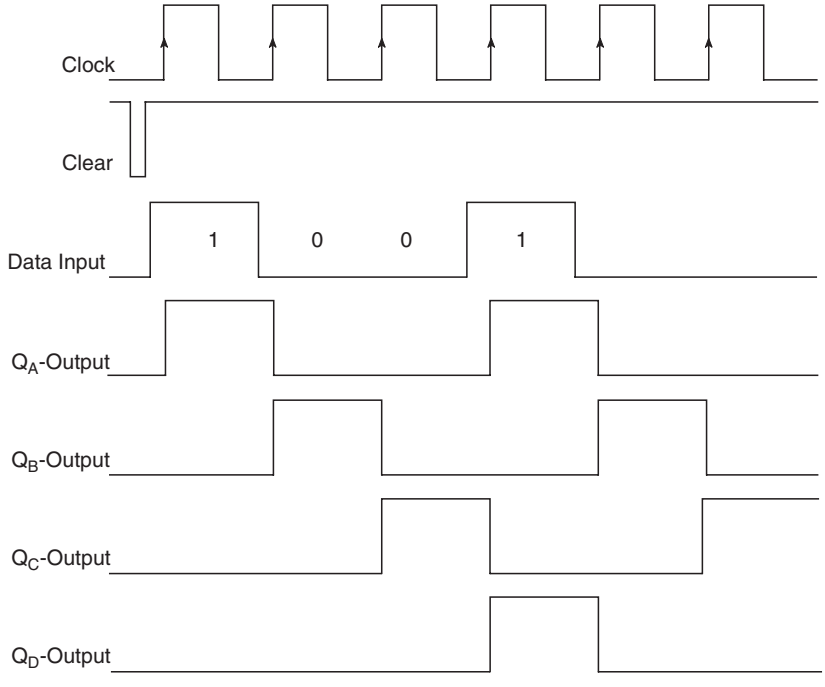


Figure 11.36 Timing waveforms for the shift register of Fig. 11.35.

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.

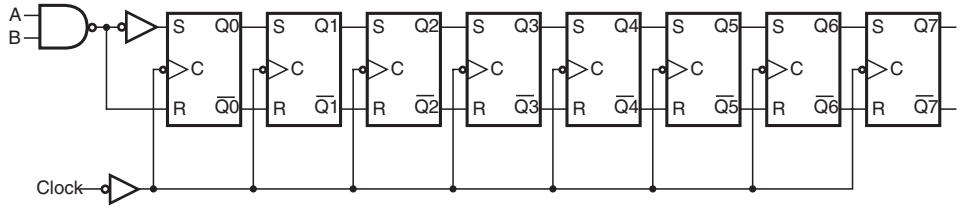


Figure 11.37 Logic diagram of IC 7491.

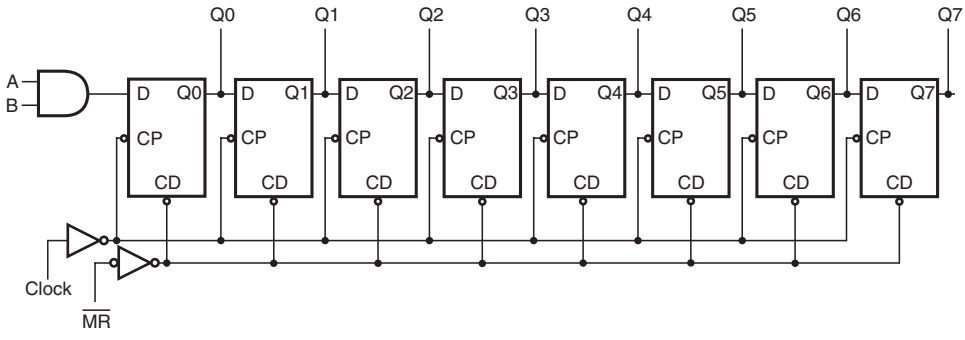


Figure 11.38 Logic diagram of IC 74164.

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.

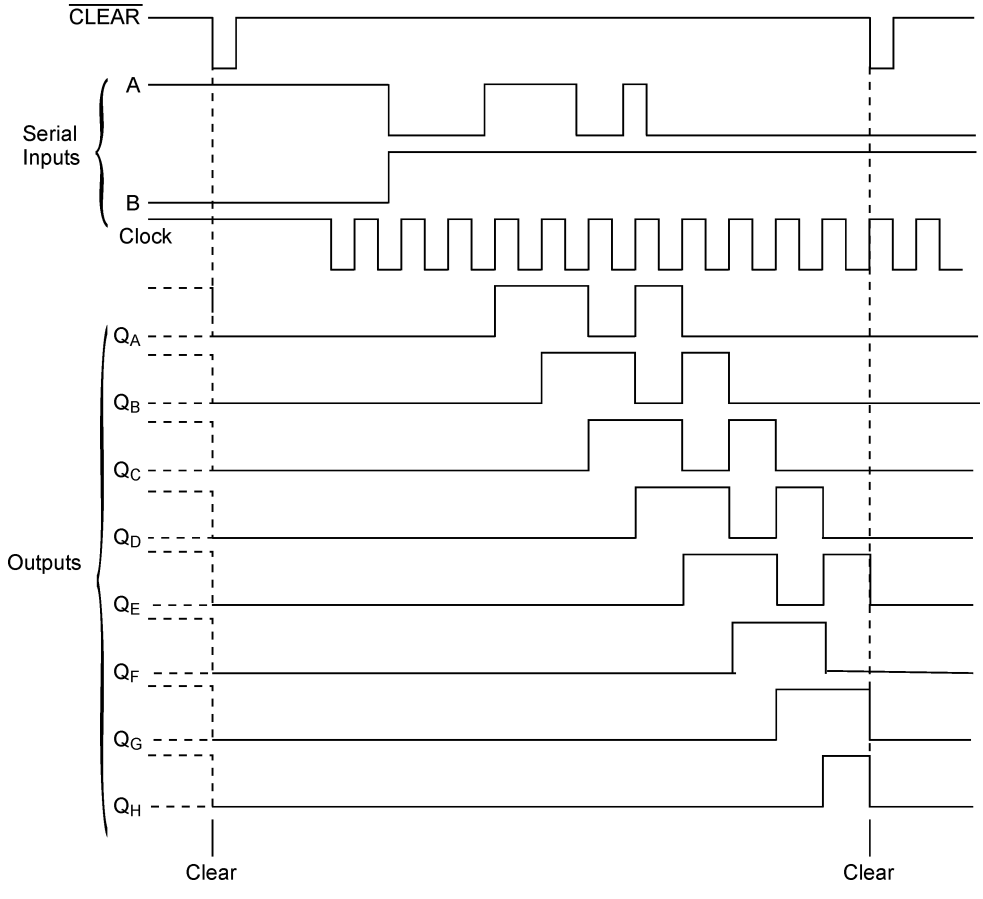


Figure 11.39 Timing waveforms of IC 74164.

11.12.4 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, IC 74199 is an eight-bit parallel-in parallel-out shift register. Figure 11.42 shows its logic diagram. We can see that the logic diagram of IC 74199 is similar to that of IC 74166 mentioned in the previous section, except that in the case of the former the flip-flop outputs have been brought out on the IC terminals.

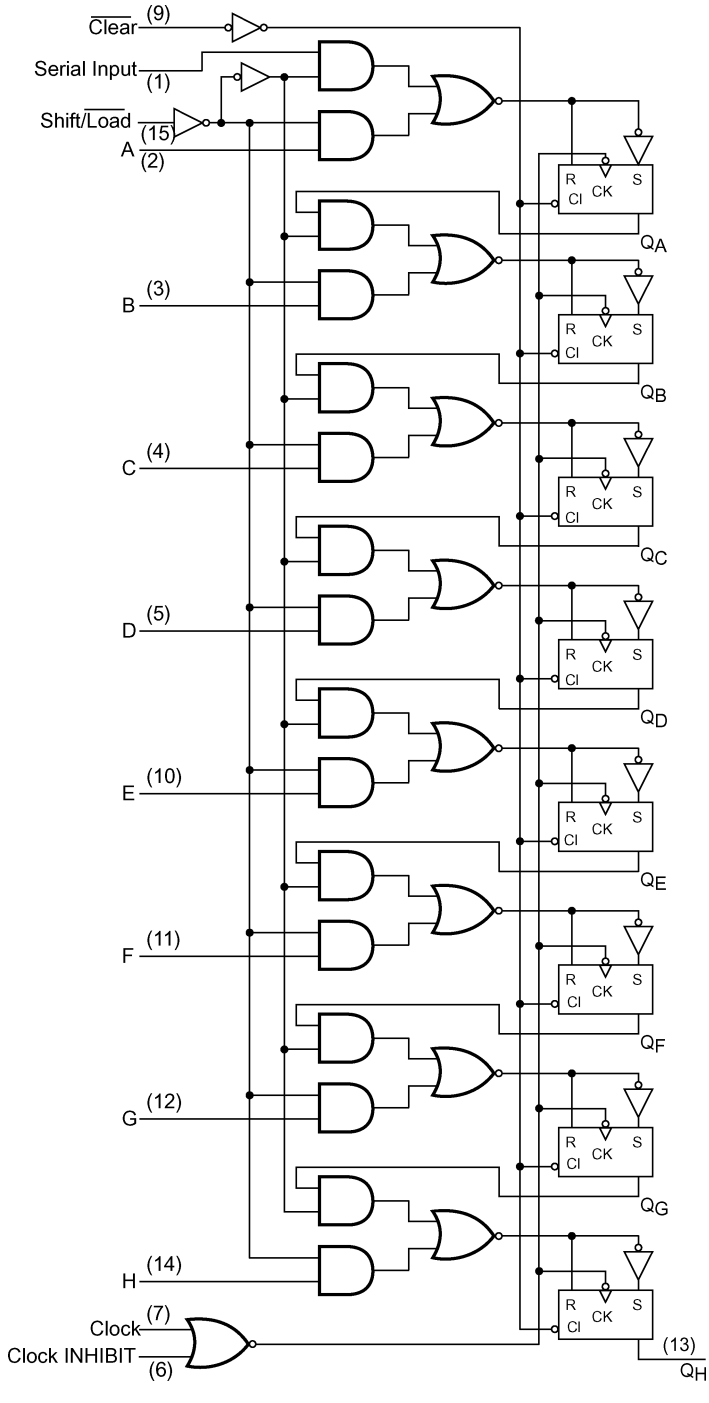


Figure 11.40 Logic diagram of 74166.

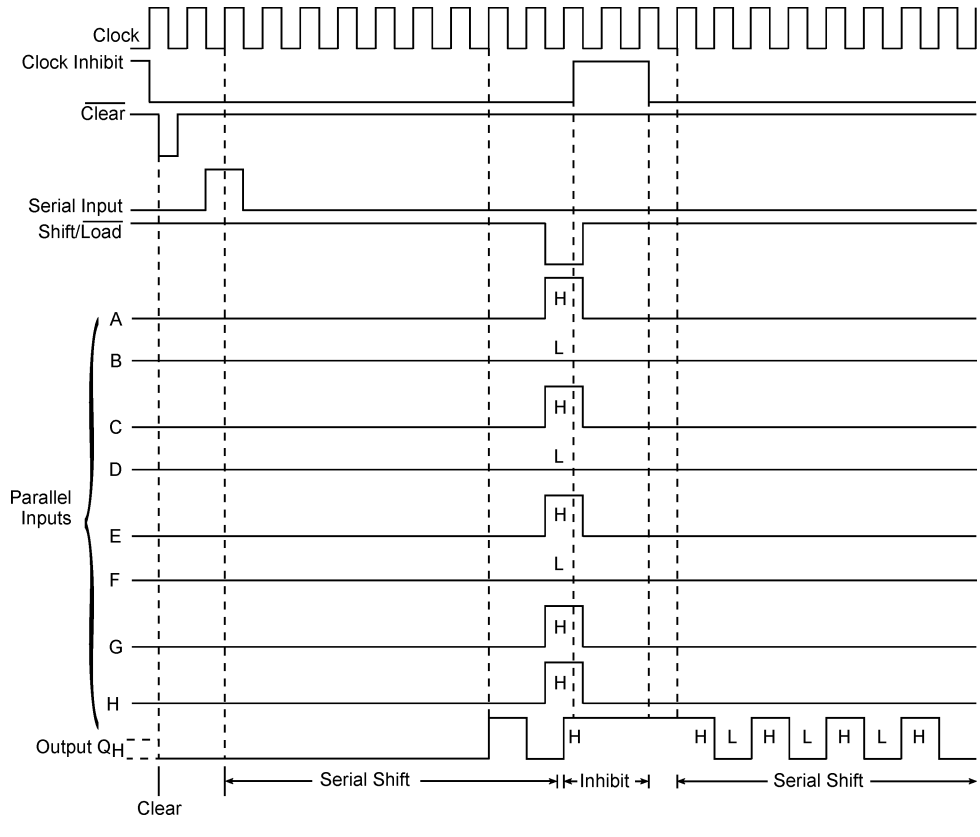


Figure 11.41 Timing waveforms of IC 74166.

11.12.5 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.

11.12.6 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.

IC 74194 is a common four-bit bidirectional universal shift register. Figure 11.43 shows the logic diagram of Ic 74194. the device offers four modes of operation, namely (a) inhibit clock, (b) shift right, (c) shift left and (d) parallel load. Clocking of the device is inhibited when both the mode control inputs S_1 and S_0 are in the logic LOW state. shift right and shift left operations are accomplished

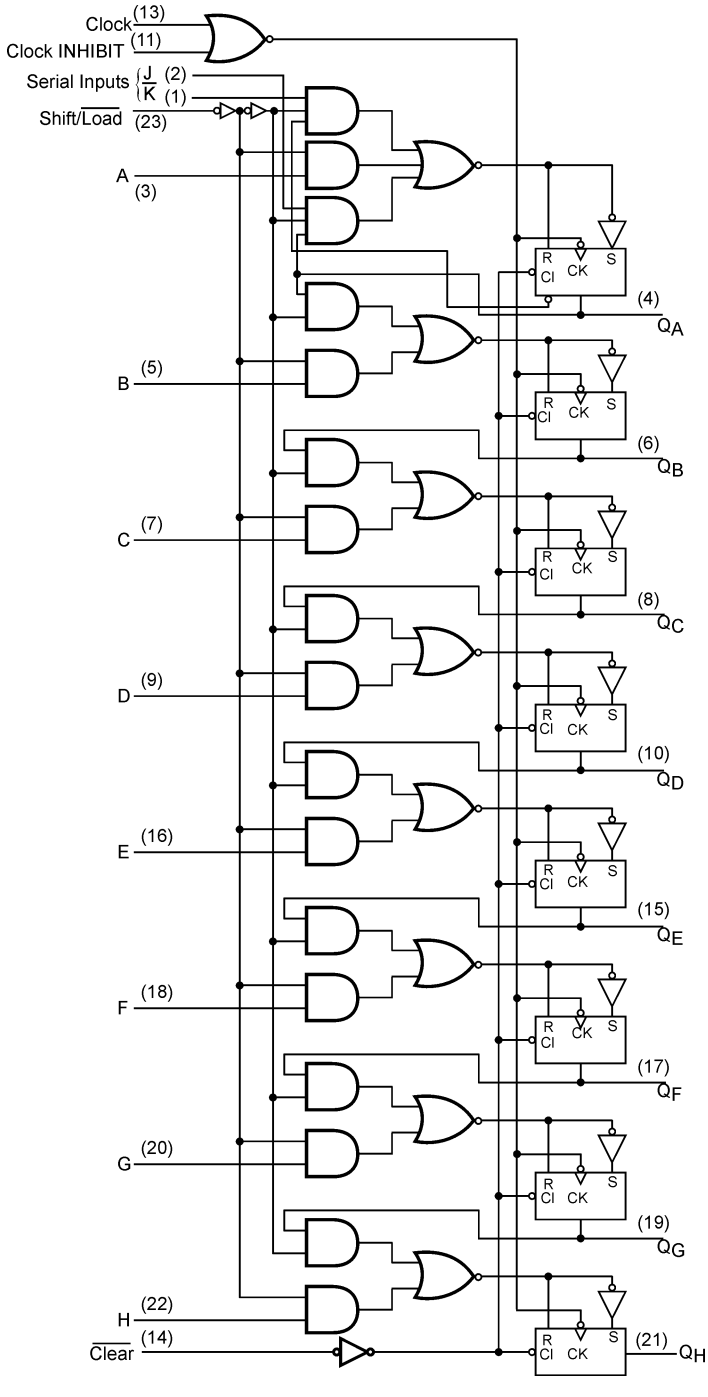


Figure 11.42 Logic diagram of IC 74199.

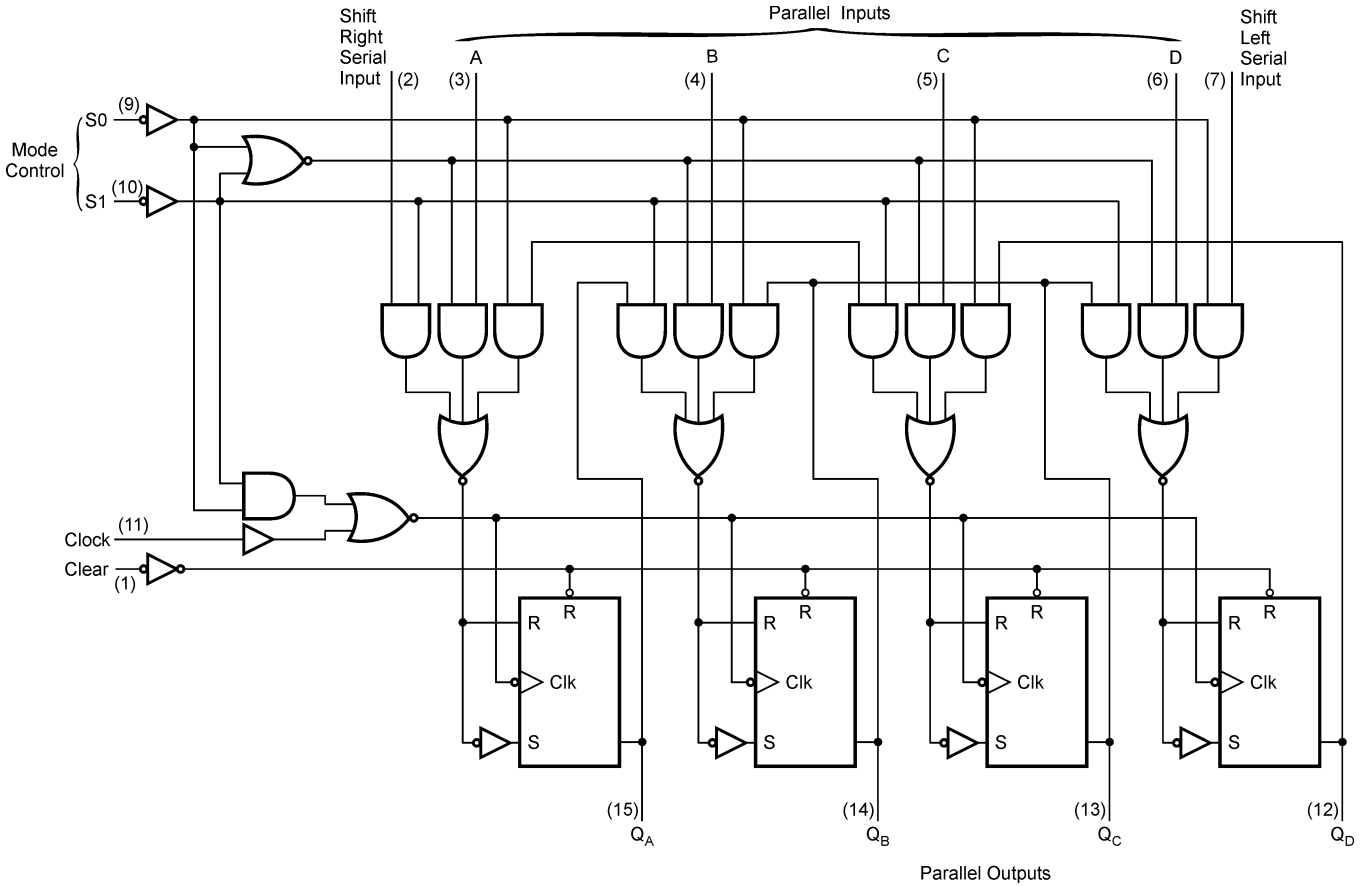


Figure 11.43 Logic diagram of IC 74194.

synchronously with LOW-to-HIGH transition of the clock with S_1 LOW and S_0 HIGH (for shift right) and S_1 HIGH and S_0 LOW (for shift left). Serial data are entered in the case of shift right and shift left operations at the corresponding data input terminals. Parallel loading is also accomplished synchronously with LOW-to-HIGH clock transitions by applying four bits of data and then driving the mode control inputs S_1 and S_0 to the logic HIGH state. Data are loaded into corresponding flip-flops and appear at the outputs with LOW-to-HIGH clock transition. Serial data flow is inhibited during parallel loading. Different modes of operation are apparent in the timing waveforms of Fig. 11.44.

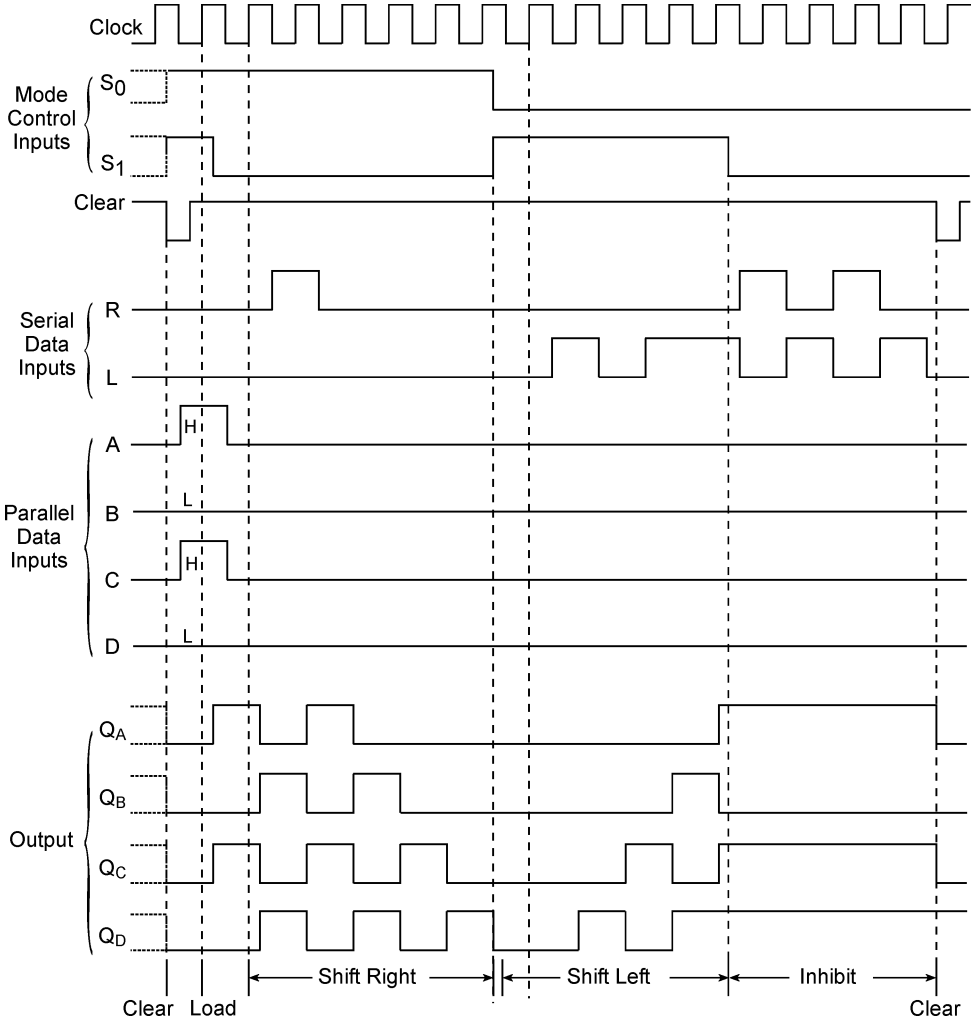


Figure 11.44 Timing waveforms of IC 74194.

11.13 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.

11.13.1 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 $\overline{Q}$ outputs of the output flip-flop are respectively fed back to the *J* and *K* inputs of the input flip-flop. Figure 11.45 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 Figure 11.45, as shown in Fig. 11.46, further illustrate their utility as a control element in a digital system to generate control pulses that must occur one after the other sequentially.

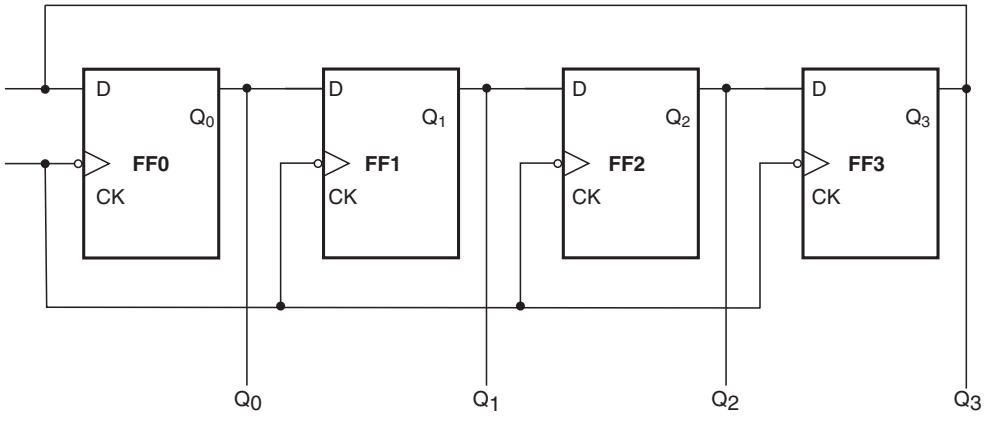


Figure 11.45 Four-bit ring counter.

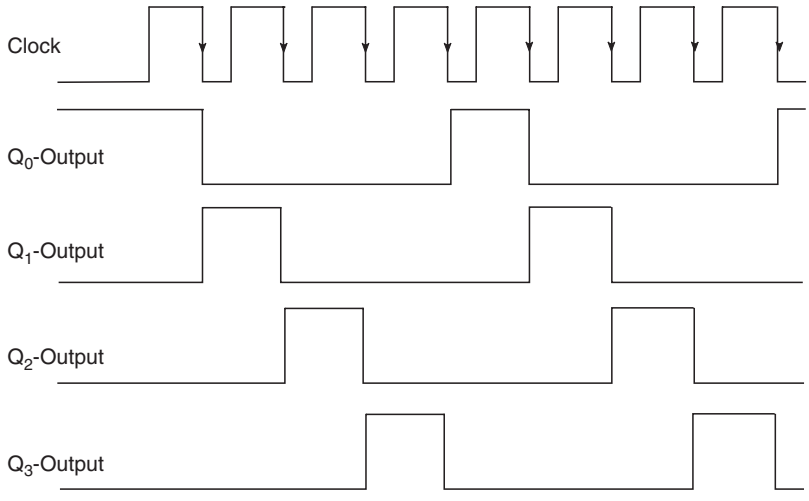


Figure 11.46 Timing waveforms of the four-bit ring counter.

11.13.2 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 $\overline{Q}$ 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 $\overline{Q}$ 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 $\overline{Q}$ output is connected to the S input. Figure 11.47 shows 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

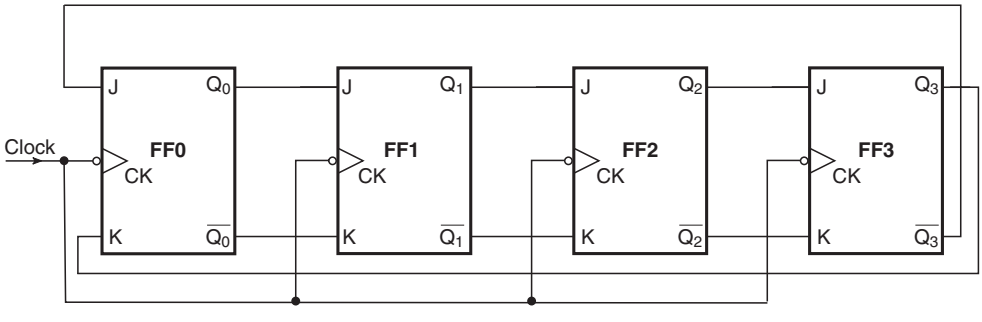


Figure 11.47 Four-bit shift counter.

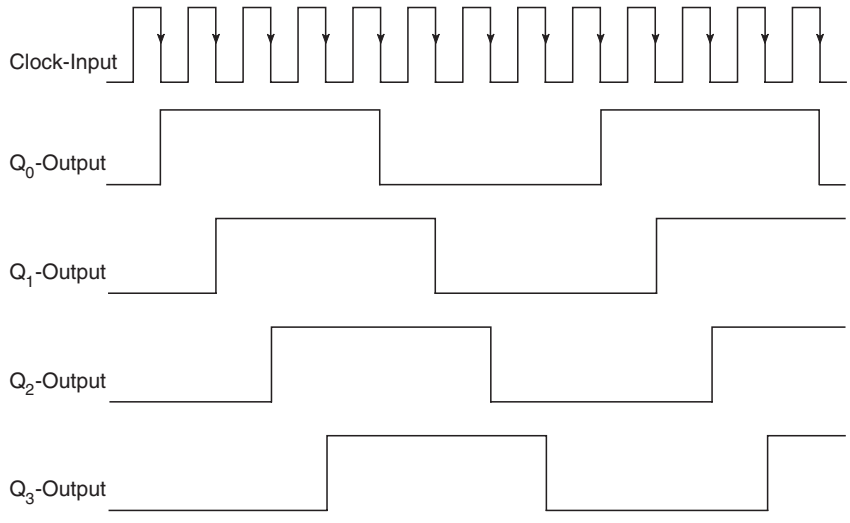


Figure 11.48 Timing waveforms of the shift counter.

is completed in eight cycles. Figure 11.48 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.

Example 11.10

Refer to Fig. 11.49, which shows an application circuit of eight-bit serial-in serial-out shift register type IC 7491 along with the waveform applied at the shorted A and B inputs:

(a) What will be the data bit present at the output at the end of the eleventh LOW-to-HIGH transition of the clock waveform?
(b) If there is a logic '1' at the end of the nth LOW-to-HIGH clock transition at the Q_3 output, what will the Q_5 output at the end of the $(n+2)$th transition be?

Solution

(a) At the end of the eighth LOW-to-HIGH clock transition, the data bits loaded into the register will be 10110010, with the '0' on the extreme right appearing at the Q_7 output (refer to the logic diagram of IC 7491 shown in Fig. 11.37). The ninth clock transition will shift this '0' out of the register, and the next adjacent bit (that is, '1') will take its place on the Q_7 output. Each subsequent clock pulse will shift the bits one step towards the right, with the result that at the end of the eleventh clock transition the Q_7 output will be a logic '0'.
(b) It will be a logic '1' only. The Q_3 output will be shifted two bit positions to the right by two clock transitions.

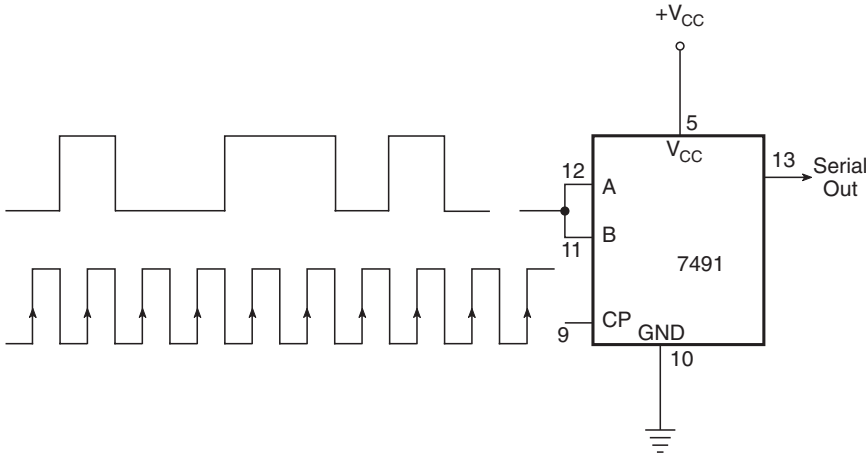


Figure 11.49 Example 11.10.

Example 11.11

Determine the number of flip-flops required to construct (a) a MOD-10 ring counter and (b) a MOD-10 Johnson counter. Also, write the count sequence in the two cases.

Solution

(a) The modulus of a ring counter is the same as the number of bits (or flip-flops). Therefore, the number of flip-flops required = 10. The count sequence is 1000000000, 0100000000, 0010000000, 0001000000, 0000100000, 0000010000, 0000001000, 0000000100, 0000000010, 0000000001 and back to 1000000000.

(b) The modulus of a Johnson counter is twice the number of flip-flops. Therefore, the number of flip-flops = 5. The count sequence is 00000, 10000, 11000, 11100, 11110, 11111, 01111, 00111, 00011, 00001 and back to 00000.

Example 11.12

Refer to the logic circuit of Fig. 11.50. Determine the modulus of this counter and write its counting sequence.

Solution

The LSB of the five-bit ring counter feeds the clock input of the *J-K* flip-flop that has been wired as a toggle flip-flop. The ring counter has a modulus of 5, and the *J-K* flip-flop works like a divide-by-2 circuit. The modulus of the counter circuit obtained by the cascade arrangement of the two is therefore 10. The counting sequence of this arrangement is given in Table 11.15.

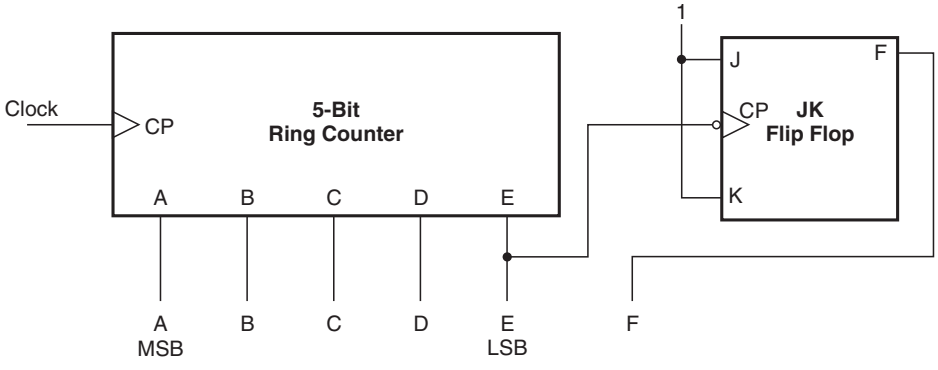


Figure 11.50 Example 11.12.

Table 11.15 Example 11.11.

Clock pulse	Outputs					
	A	*B*	*C*	*D*	*E*	*F*
1	1	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	1	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	1	0
6	1	0	0	0	0	1
7	0	1	0	0	0	1
8	0	0	1	0	0	1
9	0	0	0	1	0	1
10	0	0	0	0	1	1
11	1	0	0	0	0	0

It is very simple to write the count sequence. Firstly, we write the first 10 states of the ring counter output (designated by *A*, *B*, *C*, *D* and *E*). The logic status of *F* can be written by examining the logic status of *E*. *F* toggles whenever *E* undergoes '1' to '0' transition.

Example 11.13

Refer to the logic circuit arrangement of Fig. 11.51 built around an eight-bit serial-in/parallel-out shift register, type number 74164. A and B are the data inputs. The serial data feeding the register are obtained by an ANDing operation of A and B inputs inside the IC. $\overline{MR}$ *is an active LOW master reset. Write the logic status of register outputs for the first eight clock pulses.* Q_0 *represents the first flip-flop in this serial shift register.*

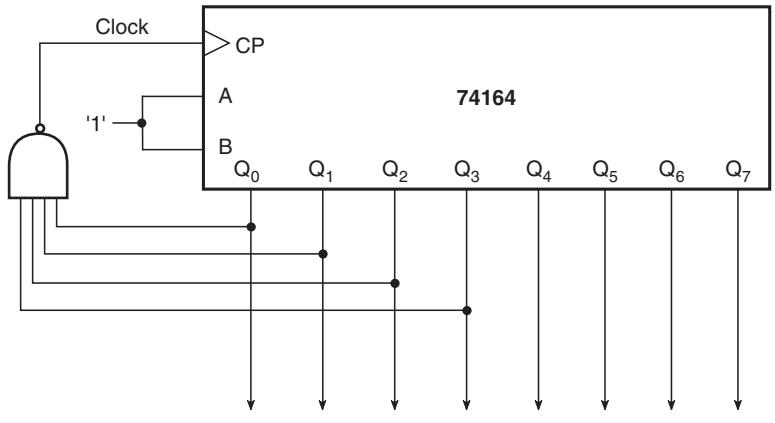


Figure 11.51 Example 11.13.

Solution

Initially, all outputs are in the logic '0' state. Since $A = B = 1$, the serial input to the shift register is a logic '1'. The $\overline{\text{MR}}$ input is initially inactive. For the first three clock pulses, the output status is 10000000, 11000000 and 11100000. With the fourth clock pulse, the output tends to go to 11110000, but it cannot be stable state as the NAND output goes from '1' to '0'. This resets the register to 00000000. Thus, the register transits from 11100000 to 00000000. With the fifth, sixth and seventh clock pulses, the circuit goes through 10000000, 11000000 and 11100000. The eight clock pulse again resets it to 00000000.

11.14 IEEE/ANSI Symbology for Registers and Counters

We introduced IEEE/ANSI symbology for digital integrated circuits as contained in IEEE/ANSI Standard 91-1984 in Section 4.22 of Chapter 4 on logic gates and related devices. A brief description of salient features of this symbology and its particular significance to sequential logic devices such as flip-flops, counters, registers, etc., was given, highlighting the use of dependency notation to provide almost complete functional information of the device. In this section, we will illustrate IEEE/ANSI symbology for counters and registers with the help of IEEE/ANSI symbols of some popular devices.

11.14.1 Counters

As an illustration, we will consider IEEE/ANSI symbols of a decade counter, type number 7490, and a presettable four-bit binary UP/DOWN counter, type number 74193. The IEEE/ANSI notation for IC 7490 and IC 74193 is shown in Figs 11.52(a) and (b) respectively.

The upper portion of the notation represents the common control block that affects all flip-flops constituting the counter. The lower portion represents individual flip-flops. Before we interpret different labels and inputs/outputs for the two counter ICs, we should know the following:

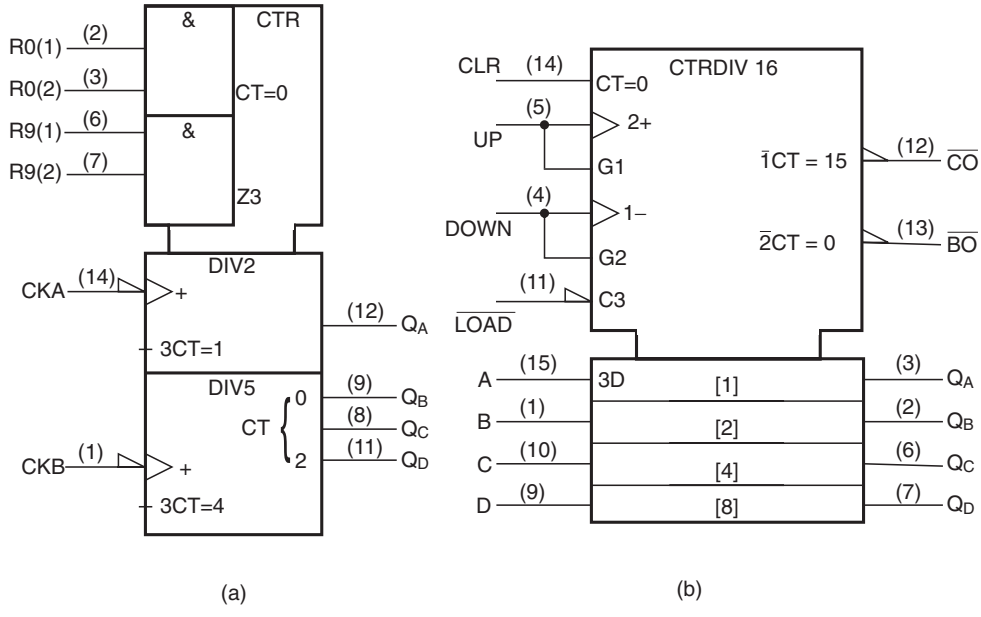


Figure 11.52 IEEE/ANSI notation for (a) IC 7490 and (b) IC 74193.

1. Letter 'C' represents control dependency. Use of the letter 'C' in the label of a certain input means that that particular input controls the entry of data into a storage element such as a flip-flop. The storage element or elements that are controlled by this input are indicated by a digit used as a suffix to the letter 'C'. The same digit appears as a prefix in the labels of all those storage elements that are controlled by this input.
2. Letter 'G' represents an AND dependency. The use of the letter 'G' followed by a digit in the label of an input means that this input is internally ANDed with another input or output and that the input or output will have the same digit as a prefix in its label.
3. Plus (+) and minus (−) signs in the labels indicate the count direction, with the former implying an UP count sequence and the latter implying a DOWN count sequence. These signs are used with clock inputs.

We will now interpret different inputs and outputs for the two counters. We will begin with IC 7490. Reset inputs R_0 (1) and R_0 (2) have an AND dependency, and when both of them are driven to the logic HIGH state the counter is reset to all 0s. Reset inputs R_9 (1) and R_9 (2) also have an AND dependency when both of them are driven to the logic HIGH state, the divide-by-2 portion of the counter is reset to count '1' (which is also the logic '1' state for the flip-flop true output) and the divide-by-5 portion of the counter is reset to count '4' (which is the 100 state for the counter outputs). If the two portions were used in cascade, the counter output would become 1001, which would mean that the counter is reset to count '9'. Clock A (CKA) and clock B (CKB) inputs allow the two portions of the counter to count in the upward sequence as indicated by the (+) sign.

We will now look at the IEEE/ANSI symbol of the other counter, that is, the counter IC type number 74193. Label CTR DIV16 means that IC 74193 is a divide-by-16 counter. Label CT = 0 with master

reset (*MR*) input implies that the counter is reset to all 0s when the *MR* input is in the logic HIGH state. Label C3 with parallel load (*PL*) input means that the data on parallel load inputs P_0, P_1, P_2 and P_3 are loaded onto the corresponding flip-flops when the *PL* input is in the logic LOW state. We can see the prefix 3 in the labels of the flip-flops. The *CPU* input has an AND dependency with the *TCU* output and *CPD* input. In the case of the former, the *TCU* output goes to the logic LOW state when the *CPU* is LOW and the count reaches '15'. In the case of the latter, the *CPU* input should be in the logic HIGH state in order to allow the *CPD* to perform the count DOWN function. Similarly, the *CPD* input has an AND dependency with the *TCD* output and *CPU* input. In the case of the former, the *TCD* output goes to the logic LOW state when the *CPD* is LOW and the count reaches '0'. In the case of the latter, the *CPD* input should be in the logic HIGH state in order to allow the *CPU* to perform the count UP function.

11.14.2 Registers

As an illustration, we will consider IEEE/ANSI symbols of a serial-in serial-out shift register, type number 7491, and a serial-in parallel-out shift register, type number 74164. Figures 11.53(a) and (b) show the IEEE/ANSI notations for IC 7491 and IC 74164 respectively.

We will begin with shift register type number 7491. Label SRG8 stands for eight-bit shift register. Label C1/→ with the clock input means that the relevant clock transition performs two functions. Firstly, it loads data onto the data input as indicated by prefix '1' with the *D* input. Secondly, it performs a right shift operation. The *A* and *B* inputs have an AND dependency. When data are entered through either of the two inputs, the other input must be held in the logic HIGH state to allow the data bit to be loaded onto the data input terminal.

We will now consider shift register type number 74164. Label 'R' stands for reset operation. Whenever the *MR* input is driven to the logic LOW state, the shift register is reset to all 0s. The rest of the notations have already been explained in the case of register type number 7491.

11.15 Application-Relevant Information

Table 11.16 lists the commonly used IC counters and registers belonging to the TTL, CMOS and ECL logic families. Application-relevant information on more popular type numbers is given in the companion website. The information includes the pin configuration diagram, functional table and timing waveforms in some cases.

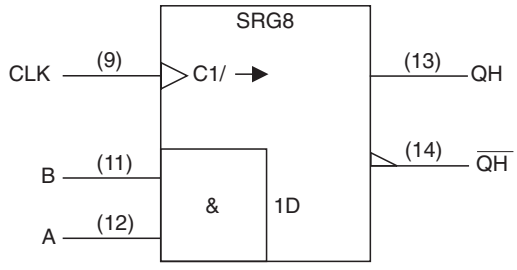
Review Questions

1. Differentiate between:

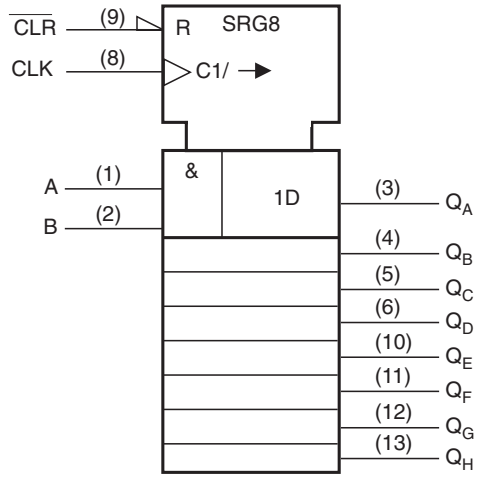
   (a) asynchronous and synchronous counters;
   (b) UP, DOWN and UP/DOWN counters;
   (c) presettable and clearable counters;
   (d) BCD and decade counters.

2. Indicate the difference between the counting sequences of:

   (a) a four-bit binary UP counter and a four-bit binary DOWN counter;
   (b) a four-bit ring counter and a four-bit Johnson counter.



(a)



(b)

Figure 11.53 IEEE/ANSI notation for (a) IC 7491 and (b) IC 74164.

3. Briefly describe:

   (a) how the architecture of an asynchronous UP counter differs from that of a DOWN counter;
   (b) how the architecture of a ring counter differs from that of a shift counter.

4. Briefly explain why the maximum usable clock frequency of a ripple counter decreases as more flip-flops are added to the counter to increase its MOD-number.
5. Why is the maximum usable clock frequency in the case of a synchronous counter independent of the size of counter?
6. How can presettable counters be used to construct counters with variable modulus?

7. Indicate the type of shift register:

   (a) into which a complete binary number can be loaded in one operation and then shifted out one bit at a time;
   (b) into which data can be entered only one bit at a time but have all data bits available as outputs;
   (c) in which we have access to only the leftmost or rightmost flip-flop.

Table 11.16 Commonly used IC counters and registers belonging to the TTL, CMOS and ECL logic families.

Type number	Function	Logic family
7490	Decade counter	TTL
7491	Eight-bit shift register (serial-in/serial-out)	TTL
7493	Four-bit binary counter	TTL
74160	BCD decade counter with asynchronous CLEAR	TTL
74161	Four-bit binary counter with asynchronous CLEAR	TTL
74162	BCD decade counter with synchronous CLEAR	TTL
74163	Four-bit binary counter with synchronous CLEAR	TTL
74164	Eight-bit shift register (serial-in/parallel-out)	TTL
74165	Eight-bit shift register (parallel-in/serial-out)	
74166	Eight-bit shift register (parallel-in/serial-out)	TTL
74178	Four-bit parallel access shift register	TTL
74190	Presettable BCD decade UP/DOWN counter	TTL
74191	Presettable four-bit binary UP/DOWN counter	TTL
74192	Presettable BCD decade UP/DOWN counter	TTL
74193	Presettable four-bit binary UP/DOWN counter	TTL
74194	Four-bit right/left universal shift register	TTL
74198	Eight-bit universal shift register (parallel-in/parallel-out bidirectional)	TTL
74199	Eight-bit universal shift register (parallel-in/parallel-out bidirectional)	TTL
74290	Decade counter	TTL
74293	Four-bit binary counter	TTL
74390	Dual decade counter	TTL
74393	Dual four-bit binary counter	TTL
4014 B	Eight-bit static shift register (synchronous parallel or serial-in/serial-out)	CMOS
4015 B	Dual four-bit static shift register (serial-in/parallel-out)	CMOS
4017 B	Five-stage Johnson counter	CMOS
4021 B	Eght-bit static shift register (asynchronous parallel-in or synchronous serial-in/serial-out)	CMOS
4029 B	Synchronous presettable four-bit UP/DOWN counter	CMOS
4035 B	Four-bit universal shift register	CMOS
40160 B	Decade counter with asynchronous CLEAR	CMOS
40161 B	Binary counter with asynchronous CLEAR	CMOS
40162 B	Decade counter	CMOS
40163 B	Binary Counter	CMOS
40192 B	Presettable BCD UP/DOWN counter	CMOS
40193 B	Presettable Binary UP/DOWN counter	CMOS
4510 B	Presettable UP/DOWN BCD counter	CMOS

Table 11.16 (*continued*).

Type number	Function	Logic family
4518 B	Dual four-bit decade counter	CMOS
4520B	Dual four-bit binary counter	CMOS
4522 B	Four-bit BCD programmable divide-by-N counter	CMOS
4722 B	Programmable counter/timer	CMOS
4731 B	Quad 64-bit static shift register	CMOS
MC 10136	Universal hexadecimal counter	ECL
MC 10137	Universal decade counter	ECL
MC 10141	Four-bit universal shift register	ECL
MC 10154	Binary counter (four-bit)	ECL
MC 10178	Four-bit binary counter	ECL

8. What do you understand when the PRESET, CLEAR, UP/DOWN, master reset and parallel load functions of a counter are designated as $\overline{PR}$, CLR, $U/\overline{D}$, $\overline{MR}$ and PL respectively?
9. What are counters with arbitrary count sequences? Briefly describe the procedure for designing a counter with a given arbitrary count sequence.
10. Give at least one IC type number for:

    (a) a four-bit binary ripple counter;
    (b) a four-bit synchronous counter;
    (c) an eight-bit serial-in serial-out shift register;
    (d) a bidirectional universal shift register.

Problems

1. For the multistage counter arrangement of Fig.11.54, determine the frequency of the output signal.

125 Hz

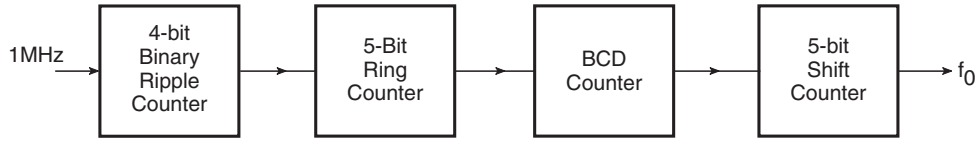


Figure 11.54 Problem 1.

2. A four-bit binary UP counter is initially in the 0000 state. Then the clock pulses are applied. Some time later the clock pulses are removed, and at that the counter is observed to be in the 0011 state. What is the minimum number of clock pulses that could possibly have occurred?

3

3. An eight-bit binary ripple UP counter with a modulus of 256 is holding the count 01111111. What will be the count after 135 clock pulses be?

00000110

4. Three four-bit BCD decade counters are connected in cascade. The MSB output of the first counter is fed to the clock input of the second counter, and the MSB output of the second counter is fed to the clock input of the third counter. If the counters are negatively edge triggered and the input clock frequency is 256 kHz, what is the frequency of the waveform available at the MSB of the third counter?

256 Hz

5. The flip-flops used in a four-bit binary ripple counter have a HIGH-to-LOW and LOW-to-HIGH propagation delay of 25 and 10 ns respectively. Determine the maximum usable clock frequency of this counter.

10 MHz

6. Refer to the counter schematic shown in Fig. 11.55. Determine the count sequence of this counter.

000, 001, 010, 011, 100, 101, 110, 000, . . .

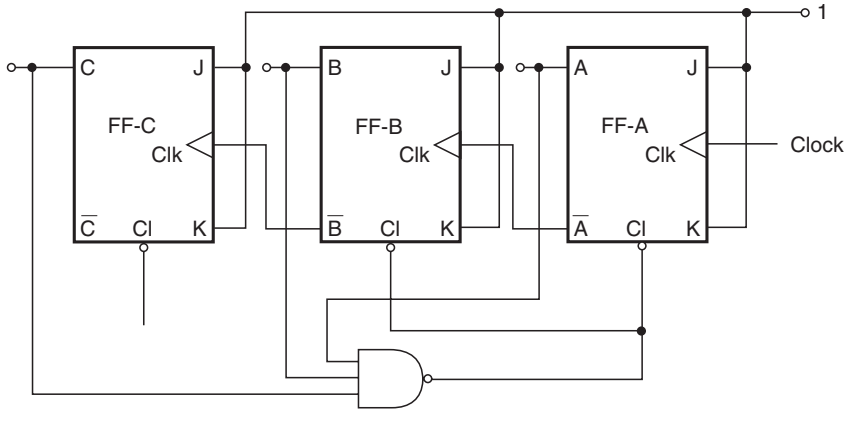


Figure 11.55 Problem 6.

7. Refer to the counter arrangement of Fig. 11.56. Determine the modulus of the counter and also the frequency of the *B* output and the duty cycle of the *C* output if the clock frequency is 600 kHz.

3; 200 kHz; 0 %

8. A four-bit ring counter and a four-bit Johnson counter are in turn clocked by a 10 MHz clock signal. Determine the frequency and duty cycle of the output of the output flip-flop in the two cases.

Ring counter: 2.5 MHz, 25 %; Johnson counter: 1.25 MHz, 50 %

9. A 100-stage serial-in/serial-out shift register is clocked at 100 kHz. How long will the data be delayed in passing through this register?

1 ms

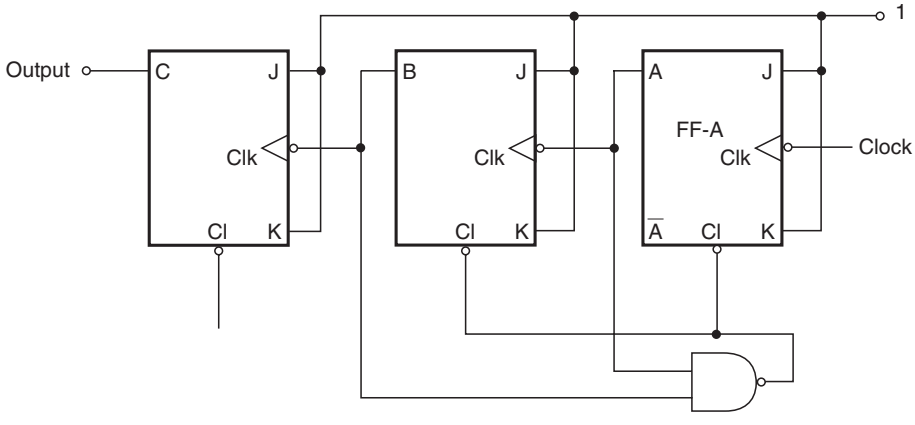


Figure 11.56 Problem 7.

10. Refer to the three-bit counter arrangement of Fig. 11.57. Determine its count sequence and also determine whether the counter is self-starting. (A counter is self-starting if it automatically goes to one of the desired states with subsequent clock pulse in case it lands itself accidentally into any of the undesired states.)

000, 001, 010, 011, 100, 000, . . . ; not self starting

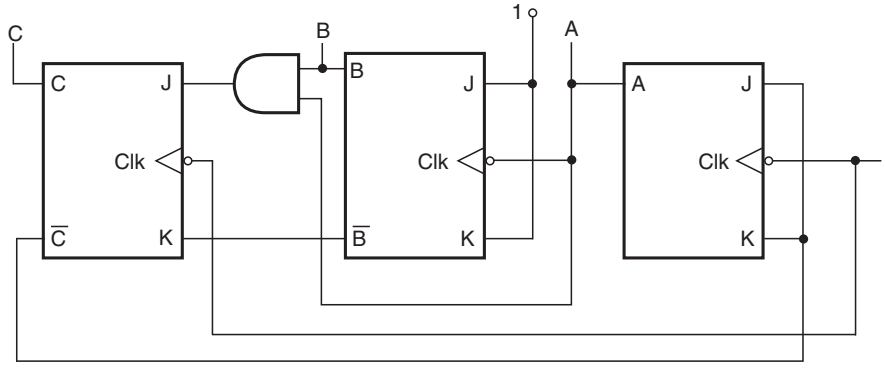


Figure 11.57 Problem 10.

Further Reading

1. Langholz, G., Mott, J. L. and Kandel, A (1998) *Foundations of Digital Logic Design*, World Scientific Publ. Co. Inc., NJ, USA.
2. Cook, N. P. (2003) *Practical Digital Electronics*, Prentice-Hall, NJ, USA.

3. Floyd, T. L. (2005) *Digital Fundamentals*, Prentice-Hall Inc., USA.
4. Tokheim, R. L. (1994) *Schaum's Outline Series of Digital Principles*, McGraw-Hill Companies Inc., USA.
5. Tocci, R. J. (2006) *Digital Systems – Principles and Applications*, Prentice-Hall Inc., NJ, USA.
6. Malvino, A. P. and Leach, D. P. (1994) *Digital Principles and Applications*, McGraw-Hill Book Company, USA.