INTRODUCTION

As you learned in Chapter 7, flip-flops can be connected together to perform counting operations. Such a group of flip-flops is a counter. The number of flip-flops used and the way in which they are connected determine the number of states (called the modulus) and also the specific sequence of states that the counter goes through during each complete cycle.

Counters are classified into two broad categories according to the way they are clocked: asynchronous and synchronous. In asynchronous counters, commonly called *ripple counters*, the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the output of the preceding flip-flop. In synchronous counters, the clock input is connected to all of the flip-flops so that they are clocked simultaneously. Within each of these two categories, counters are classified primarily by the type of sequence, the number of states, or the number of flip-flops in the counter.

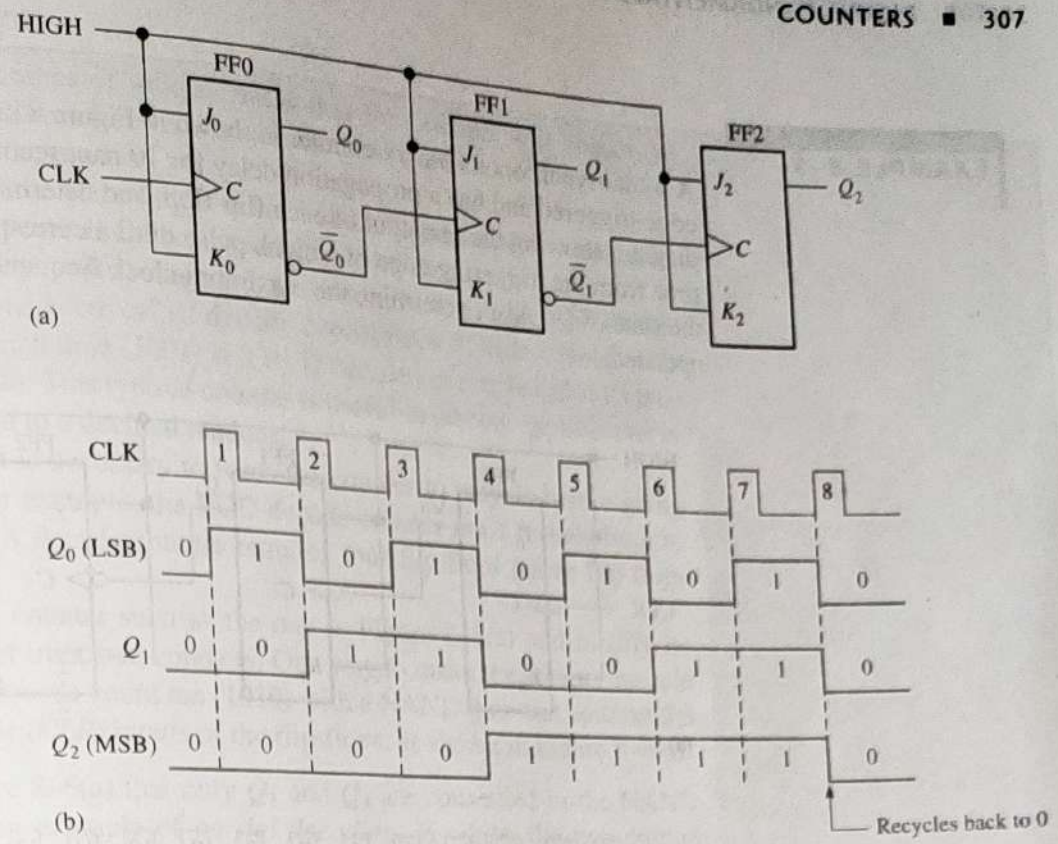
A 3-Bit Asynchronous Binary Co...

A 3-bit asynchronous binary counter is shown in Figure 8–3(a). The basic operation is the same as that of the 2-bit counter just discussed, except that the 3-bit counter has eight states, due to its three flip-flops. A timing diagram is shown in Figure 8–3(b) for eight clock pulses. Notice that the counter in Figure 8–3 progresses through a binary count of zero through seven and then recycles to the zero state. This counter sequence is listed in Table 8–2. This counter can be easily expanded for higher count, by connecting additional toggle flip-flops.

▶ TABLE 8–2

CLOCK PULSE	Q_2	Q_1	Q_0
Initially	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recycles)	0	0	0

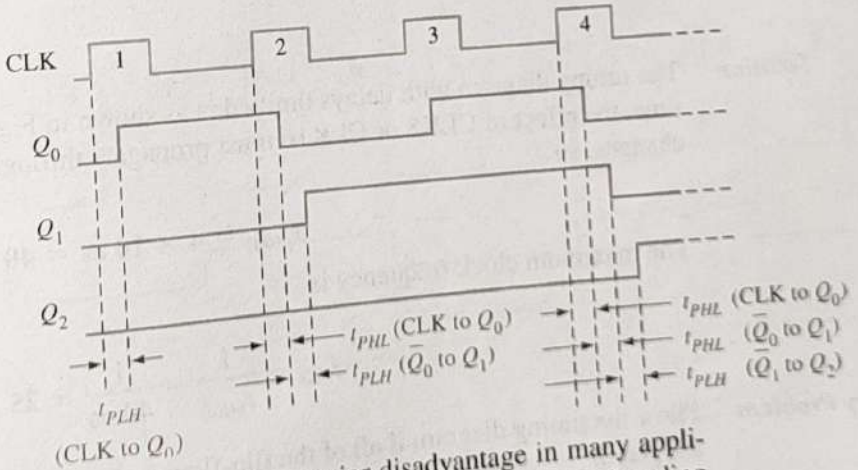
▶ FIGURE 8–3



Propagation Delay Asynchronous counters are commonly referred to as **ripple counters** for the following reason: The effect of the input clock pulse is first "felt" by FF0. This effect cannot get to FF1 immediately because of the propagation delay through FF0. Then, there is the propagation delay through FF1 before FF2 can be triggered. Thus, the effect of an input clock pulse "ripples" through the counter, taking some time, due to propagation delays, to reach the last flip-flop.

To illustrate, notice that all three flip-flops in the counter of Figure 8–3 change state on the leading edge of CLK4. This ripple clocking effect is shown in Figure 8–4 for the first four clock pulses, with the propagation delays indicated. The HIGH-to-LOW transition of Q_0 occurs one delay time (t_{PHL}) after the positive-going transition of the clock pulse. The HIGH-to-LOW transition of Q_1 occurs one delay time (t_{PHL}) after the positive-going transition of $\overline{Q}_0$. The LOW-to-HIGH transition of Q_2 occurs one delay time (t_{PHL}) after the positive-going transition of $\overline{Q}_1$. As you can see, FF2 is not triggered until two delay times after the positive-going edge of the clock pulse, CLK4. Thus, it takes three propagation delay times for the effect of the clock pulse, CLK4, to ripple through the counter and change Q_2 from LOW to HIGH.

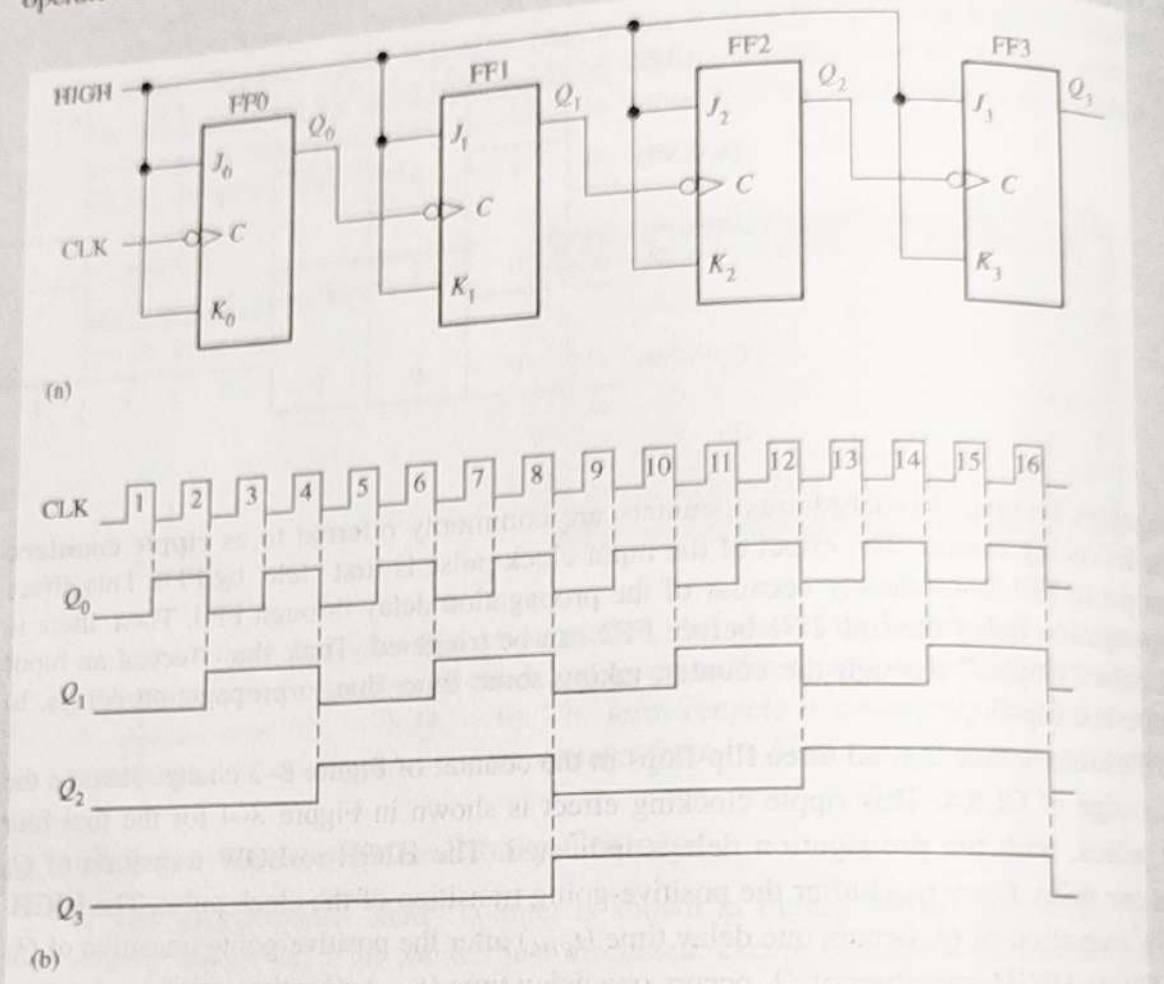
▶ FIGURE 8–4



This cumulative delay of an asynchronous counter is a major disadvantage in many applications because it limits the rate at which the counter can be clocked and creates decoding problems. The maximum cumulative delay in a counter must be less than the period of the clock waveform.

EXAMPLE 8–1

A 4-bit asynchronous binary counter is shown in Figure 8–5(a). Each flip-flop is negative edge-triggered and has a propagation delay for 10 nanoseconds (ns). Develop a timing diagram showing the Q output of each flip-flop, and determine the total propagation delay time from the triggering edge of a clock pulse until a corresponding change can occur in the state of Q_3. Also, determine the maximum clock frequency at which the counter can be operated.



▲ **FIGURE 8–5**

4-bit asynchronous binary counter and its timing diagram

Solution The timing diagram with delays omitted is as shown in Figure 8–5(b). For the total delay time, the effect of CLK8 or CLK16 must propagate through four flip-flops before Q_3 changes, so

$$t_{p(tot)} = 4 \times 10 \text{ ns} = \mathbf{40 \text{ ns}}$$

The maximum clock frequency is

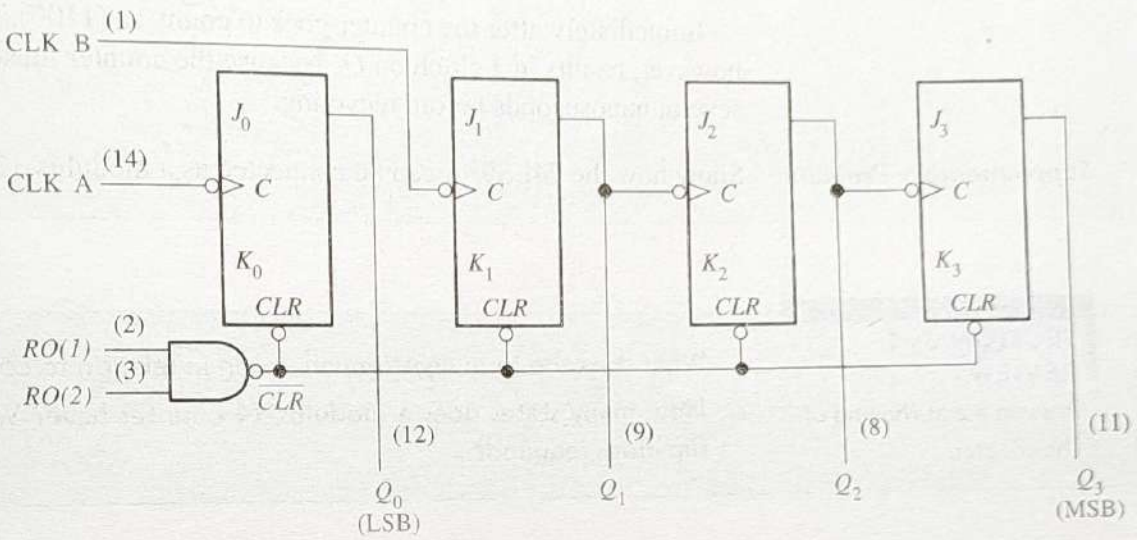
$$f_{max} = \frac{1}{t_{p(tot)}} = \frac{1}{40 \text{ ns}} = \mathbf{25 \text{ MHz}}$$

plementary Problem Show the timing diagram if all of the flip-flops in Figure 8–5(a) are positive edge-triggered.

A 4-BIT ASYNCHRONOUS BINARY COUNTER

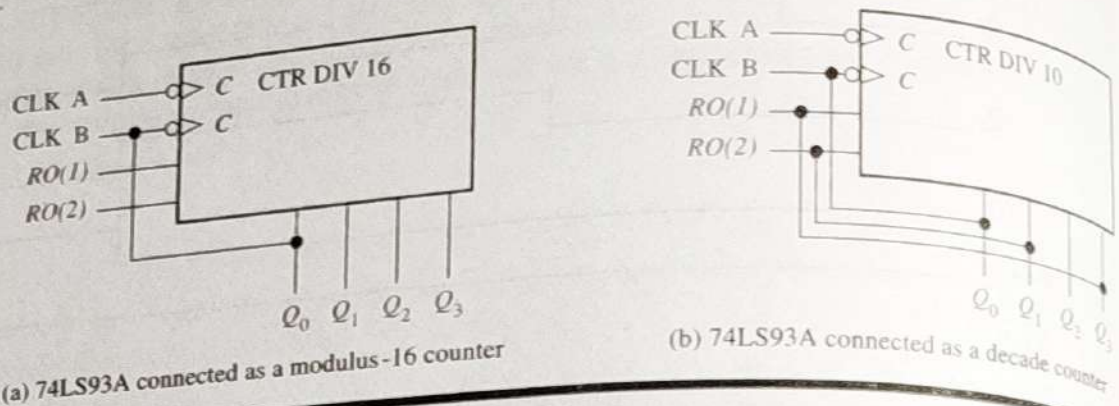
The 74LS93A is an example of a specific integrated circuit asynchronous counter. As the logic diagram in Figure 8–8 shows, this device actually consists of a single flip-flop and a 3-bit asynchronous counter. This arrangement is for flexibility. It can be used as a divide-by-2 device if only the single flip-flop is used, or it can be used as a modulus-8 counter if only the 3-bit counter portion is used. This device also provides gated reset inputs, *RO(1)* and *RO(2)*. When both of these inputs are HIGH, the counter is reset to the 0000 state $\overline{CLR}$.

▶ **FIGURE 8–8**



Additionally, the 74LS93A can be used as a 4-bit modulus-16 counter (counts 0 through 15) by connecting the Q_0 output to the CLK B input as shown in Figure 8–9(a). It can also be configured as a decade counter (counts 0 through 9) with asynchronous recycling by using the gated reset inputs for partial decoding of count ten, as shown in Figure 8–9(b).

▶ FIGURE 8–9



(a) 74LS93A connected as a modulus-16 counter

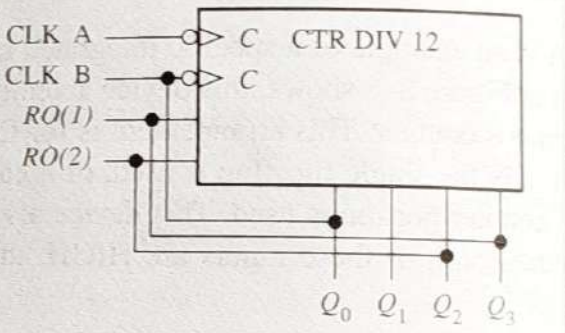
(b) 74LS93A connected as a decade counter

EXAMPLE 8–3

Show how the 74LS93A can be used as a modulus-12 counter.

Solution Use the gated reset inputs, *RO(1)* and *RO(2)*, to partially decode count 12 (remember, there is an internal NAND gate associated with these inputs). The count-12 decoding is accomplished by connecting Q_3 to *RO(1)* and Q_2 to *RO(2)*, as shown in Figure 8–10. Output Q_0 is connected to CLK B to create a 4-bit counter.

▶ FIGURE 8–10



Immediately after the counter goes to count 12 (1100), it is reset to 0000. The recycling, however, results in a glitch on Q_2 because the counter must go into the 1100 state for several nanoseconds before recycling.

Supplementary Problem Show how the 74LS93A can be connected as a modulus-13 counter.

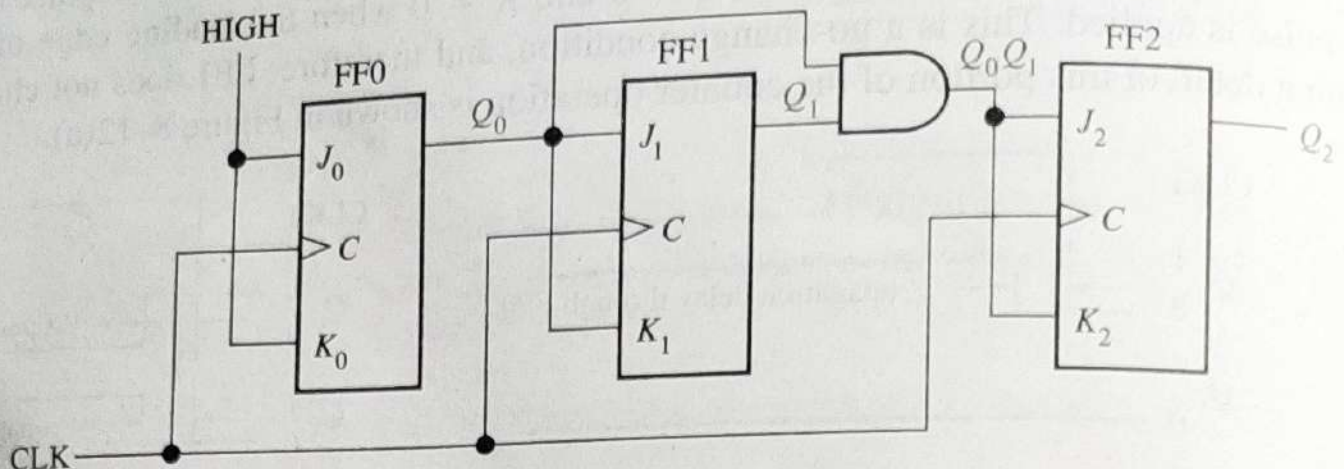
SECTION 8–1 REVIEW

Answers are at the end of the chapter.

1. What does the term *asynchronous* mean in relation to counters?
2. How many states does a modulus-14 counter have? What is the minimum number of flip-flops required?

A 3-Bit Synchronous Binary Counter

A 3-bit synchronous binary counter is shown in Figure 8–14, and its timing diagram is shown in Figure 8–15. You can understand this counter operation by examining its sequence of states as shown in Table 8–3.



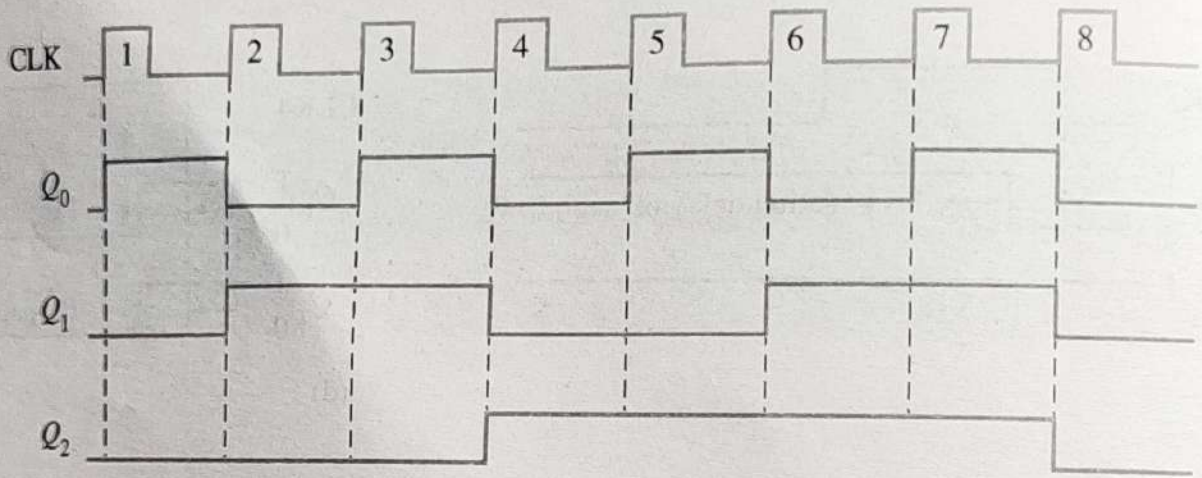


TABLE 8–3

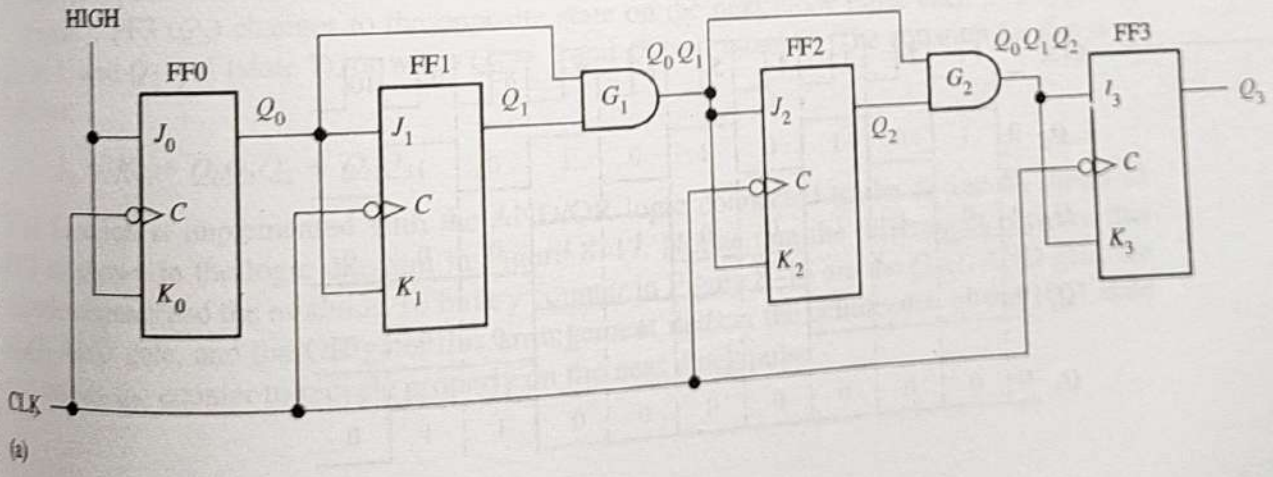
CLOCK PULSE	Q_2	Q_1	Q_0
Initially	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recycles)	0	0	0

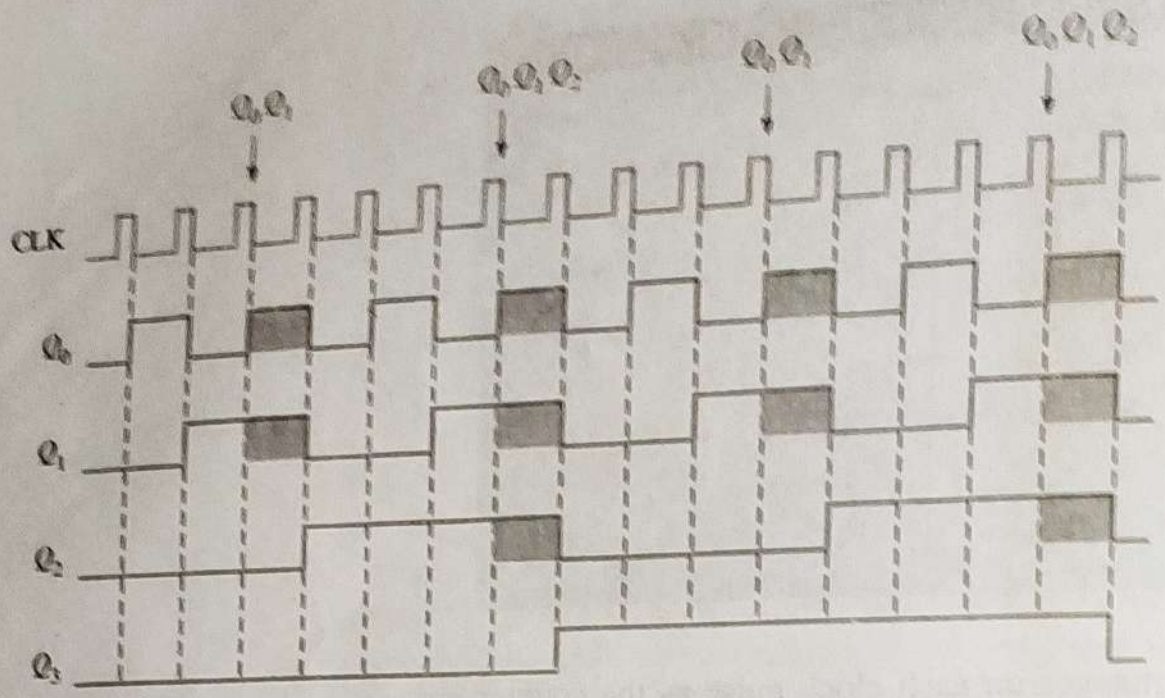
First, let us look at Q_0. Notice that Q_0 changes on each clock pulse as the counter progresses from its original state to its final state and then back to its original state. To produce this operation, FF0 must be held in the toggle mode by constant HIGHs on its J_0 and K_0 inputs. Notice that Q_1 goes to the opposite state following each time Q_0 is a 1. This change occurs at CLK2, CLK4, CLK6, and CLK8. The CLK8 pulse causes the counter to recycle. To produce this operation, Q_0 is connected to the J_1 and K_1 inputs of FF1. When Q_0 is a 1 and a clock pulse occurs, FF1 is in the toggle mode and therefore changes state. The other times, when Q_0 is a 0, FF1 is in the no-change mode and remains in its present state.

Next, let us see how FF2 is made to change at the proper times according to the binary sequence. Notice that both times Q_2 changes state, it is preceded by the unique condition in which both Q_0 and Q_1 are HIGH. This condition is detected by the AND gate and applied to the J_2 and K_2 inputs of FF2. Whenever both Q_0 and Q_1 are HIGH, the output of the AND gate makes the J_2 and K_2 inputs of FF2 HIGH, and FF2 toggles on the following clock pulse. At all other times, the J_2 and K_2 inputs of FF2 are held LOW by the AND gate output, and FF2 does not change state.

A 4-Bit Synchronous Binary Counter

Figure 8–16(a) shows a 4-bit synchronous binary counter, and Figure 8–16(b) shows its timing diagram. This particular counter is implemented with negative edge-triggered flip-flops. The reasoning behind the J and K input control for the first three flip-flops is the same as previously discussed for the 3-bit counter. The fourth stage, FF3, changes only twice in the sequence. Notice that both of these transitions occur following the times that Q_0, Q_1, and Q_2 are all HIGH. This condition is decoded by AND gate G_2 so that when a clock pulse occurs, FF3 will change state. For all other times the J_3 and K_3 inputs of FF3 are LOW, and it is in a no-change condition.





(b)

▲ FIGURE 8–16

Asynchronous Decade Counters

The **modulus** of a counter is the number of unique states that the counter will sequence through. The maximum possible number of states (maximum modulus) of a counter is 2^n, where n is the number of flip-flops in the counter.

Counters can also be designed to have a number of states in their sequence that is less than the maximum of 2^n. The resulting sequence is called a **truncated sequence**.

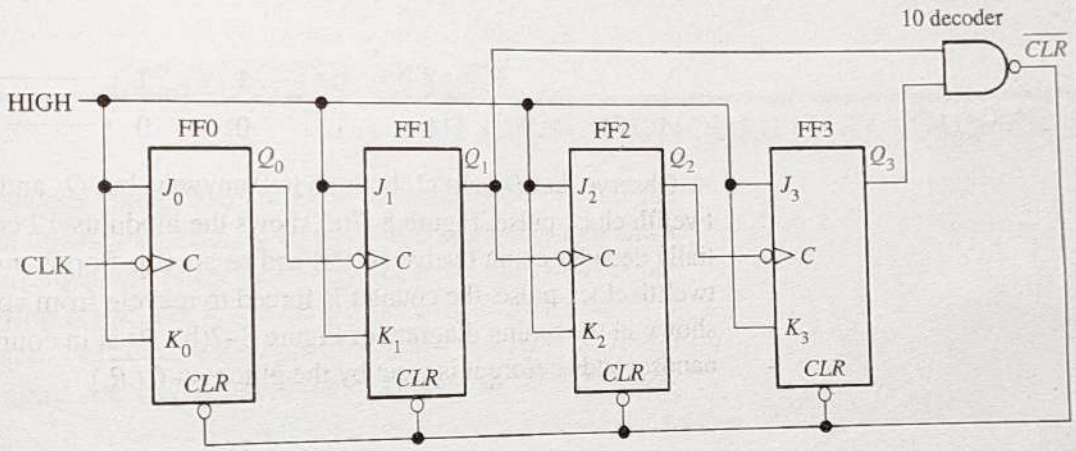
One common modulus for counters with **truncated** sequences is ten (called MOD10). Counters with ten states in their sequence are called **decade counters**. A **decade** counter with a count sequence of zero (0000) through nine (1001) is a BCD decade counter because its ten-state sequence produces the BCD code. This type of counter is useful in display applications in which BCD is required for conversion to a decimal readout.

To obtain a truncated sequence, it is necessary to force the counter to recycle before going through all of its possible states. For example, the BCD decade counter must recycle back to the 0000 state after the 1001 state. A decade counter requires four flip-flops (three flip-flops are insufficient because $2^3 = 8$).

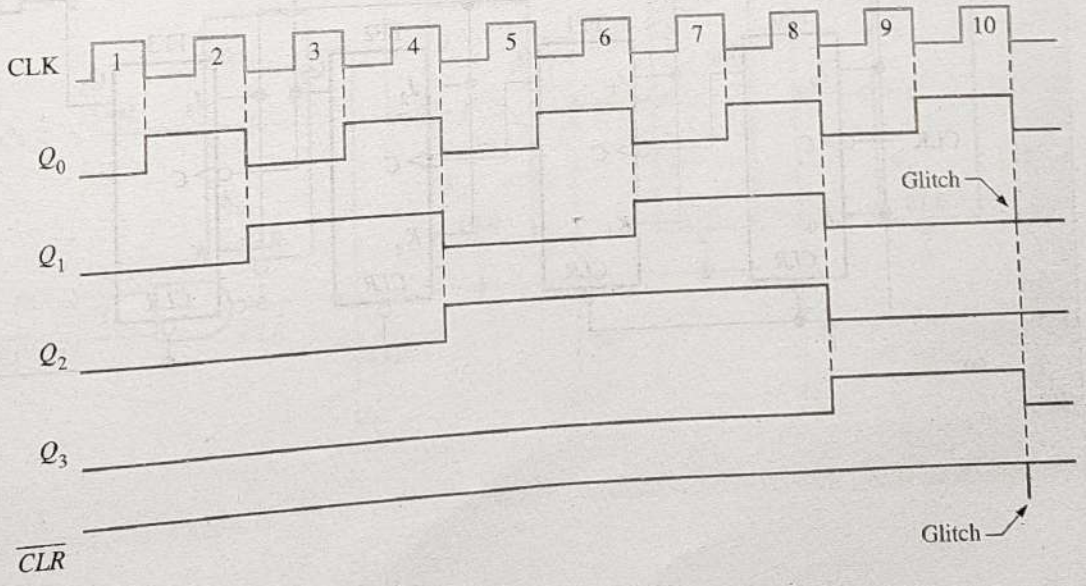
Let us use a 4-bit asynchronous counter such as the one in Figure 8–5(a) and modify its sequence to illustrate the principle of truncated counters. One way to make the counter recycle after the count of nine (1001) is to decode count ten (1010) with a NAND gate and connect the output of the NAND gate to the clear ($\overline{CLR}$) inputs of the flip-flops, as shown in Figure 8–6(a).

Partial Decoding Notice in Figure 8–6(a) that only Q_1 and Q_3 are connected to the NAND gate inputs. This arrangement is an example of *partial decoding*, in which the two unique states ($Q_1 = 1$ and $Q_3 = 1$) are sufficient to decode the count of ten because none of the other states (zero through nine) have both Q_1 and Q_3 HIGH at the same time. When the counter goes

FIGURE 8–6



(a)



(b)

into count ten (1010), the decoding gate output goes LOW and asynchronously resets all the flip-flops.

The resulting timing diagram is shown in Figure 8–6(b). Notice that there is a glitch on the Q_1 waveform. The reason for this glitch is that Q_1 must first go HIGH before the count of ten does can be decoded. Not until several nanoseconds after the counter goes to the count of ten does the output of the decoding gate go LOW (both inputs are HIGH). Thus, the counter is in the 1010 state for a short time before it is reset to 0000, thus producing the glitch on Q_1 and the resulting glitch on the $\overline{CLR}$ line that resets the counter.

Other truncated sequences can be implemented in a similar way, as Example 8–2 shows.

EXAMPLE 8–2

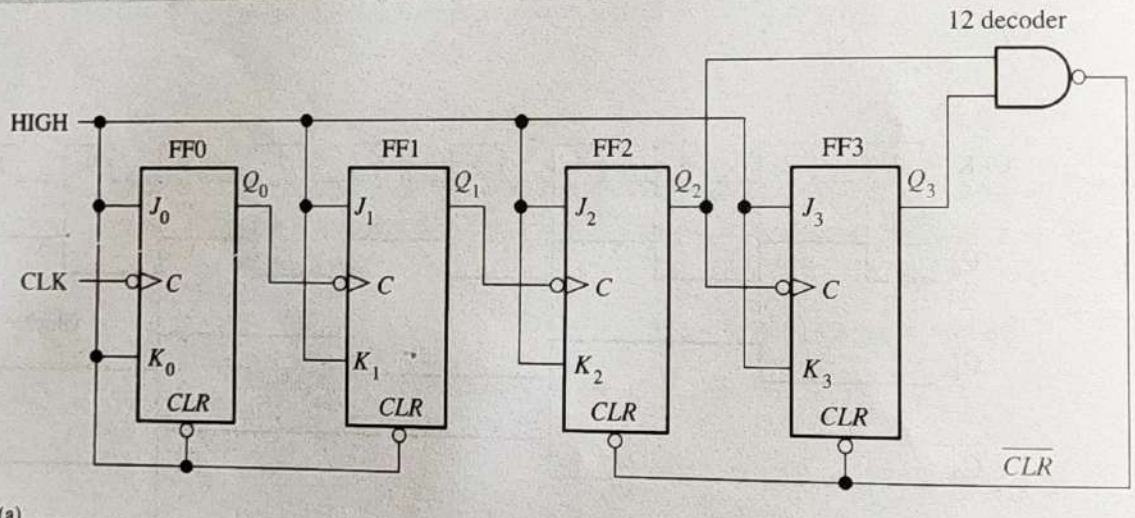
Show how an asynchronous counter can be implemented having a modulus of twelve with a straight binary sequence from 0000 through 1011.

Solution Since three flip-flops can produce a maximum of eight states, four flip-flops are required to produce any modulus greater than eight but less than or equal to sixteen.

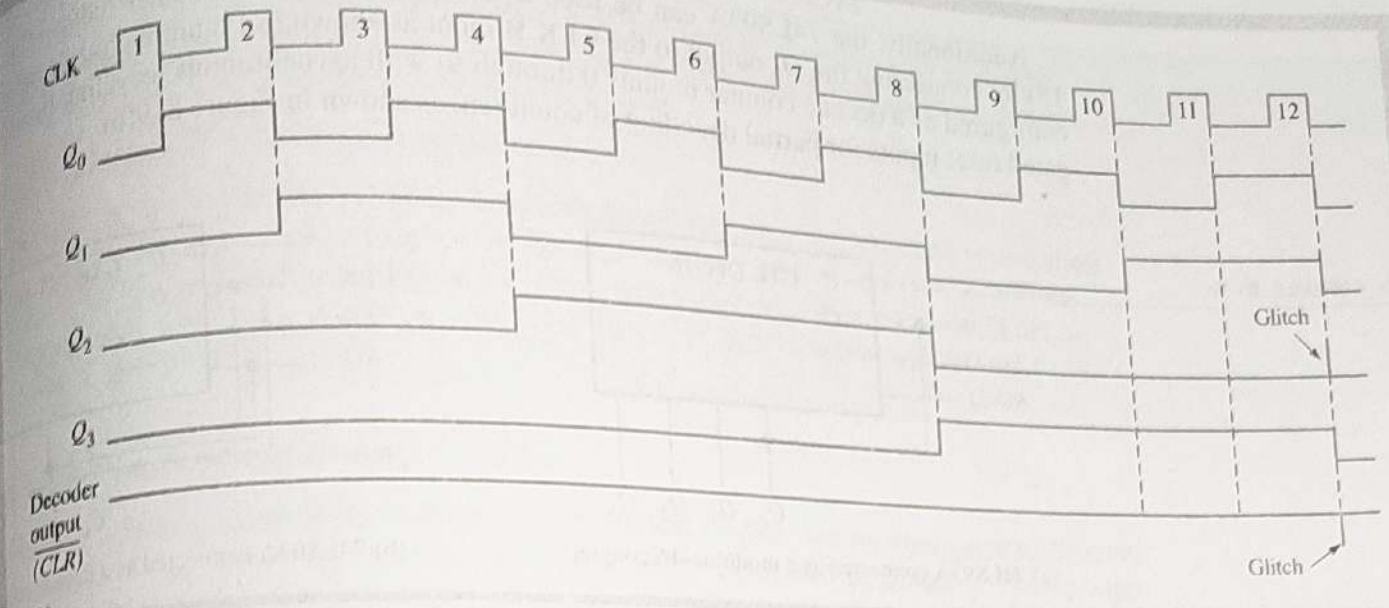
When the counter gets to its last state, 1011, it must recycle back to 0000 rather than going to its normal next state of 1100, as illustrated in the following sequence chart:

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	←
.	.	.	.	
.	.	.	.	Recycles
.	.	.	.	
1	0	1	1	
1	1	0	0	← Normal next state

Observe that Q_0 and Q_1 both go to 0 anyway, but Q_2 and Q_3 must be forced to 0 on the twelfth clock pulse. Figure 8–7(a) shows the modulus-12 counter. The NAND gate partially decodes count twelve (1100) and resets flip-flop 2 and flip-flop 3. Thus, on the twelfth clock pulse, the counter is forced to recycle from count eleven to count zero, as shown in the timing diagram of Figure 8–7(b). (It is in count twelve for only a few nanoseconds before it is reset by the glitch on $\overline{CLR}$.)



(a)


CLK
1
2
3
4
5
6
7
8
9
10
11
12
Q_0
Q_1
Q_2
Q_3
Glitch
Decoder output
$\overline{(CLR)}$
Glitch
(b)

▲ FIGURE 8–7