9–2 Asynchronous Counters

The term **asynchronous** refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time. An **asynchronous counter** is one in which the flip-flops (FF) within the counter do not change states at exactly the same time because they do not have a common clock pulse.

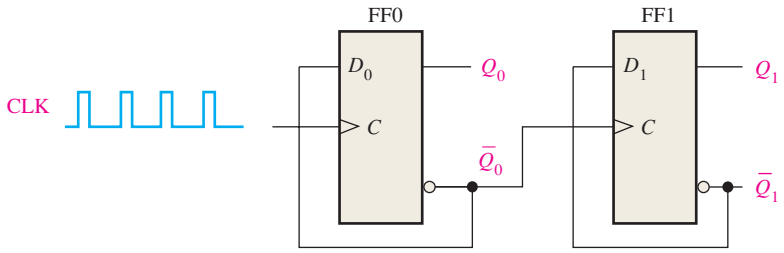
After completing this section, you should be able to

- Describe the operation of a 2-bit asynchronous binary counter
- Describe the operation of a 3-bit asynchronous binary counter
- Define *ripple* in relation to counters
- Describe the operation of an asynchronous decade counter
- Develop counter timing diagrams
- Discuss the implementation of a 4-bit asynchronous binary counter

A 2-Bit Asynchronous Binary Counter

The clock input of an asynchronous counter is always connected only to the LSB flip-flop.

Figure 9–4 shows a 2-bit counter connected for asynchronous operation. Notice that the clock (CLK) is applied to the clock input (C) of *only* the first flip-flop, FF0, which is always the least significant bit (LSB). The second flip-flop, FF1, is triggered by the $\overline{Q}_0$ output of FF0. FF0 changes state at the positive-going edge of each clock pulse, but FF1 changes only when triggered by a positive-going transition of the $\overline{Q}_0$ output of FF0. Because of the inherent propagation delay time through a flip-flop, a transition of the input clock pulse (CLK) and a transition of the $\overline{Q}_0$ output of FF0 can never occur at exactly the same time. Therefore, the two flip-flops are never simultaneously triggered, so the counter operation is asynchronous.



MultiSim

FIGURE 9–4 A 2-bit asynchronous binary counter. Open file F09-04 to verify operation. *A Multisim tutorial is available on the website.*

The Timing Diagram

Let's examine the basic operation of the asynchronous counter of Figure 9–4 by applying four clock pulses to FF0 and observing the Q output of each flip-flop. Figure 9–5 illustrates the changes in the state of the flip-flop outputs in response to the clock pulses. Both flip-flops are connected for toggle operation ($D = \overline{Q}$) and are assumed to be initially RESET (Q LOW).

Asynchronous counters are also known as ripple counters.

The positive-going edge of CLK1 (clock pulse 1) causes the Q_0 output of FF0 to go HIGH, as shown in Figure 9–5. At the same time the $\overline{Q}_0$ output goes LOW, but it has no effect on FF1 because a positive-going transition must occur to trigger the flip-flop. After the leading edge of CLK1, $Q_0 = 1$ and $Q_1 = 0$. The positive-going edge of CLK2 causes Q_0 to go LOW. Output $\overline{Q}_0$ goes HIGH and triggers FF1, causing Q_1 to go HIGH. After the leading edge of CLK2, $Q_0 = 0$ and $Q_1 = 1$. The positive-going edge of CLK3 causes Q_0 to go HIGH again. Output $\overline{Q}_0$ goes LOW and has no effect on FF1. Thus, after the leading edge of CLK3, $Q_0 = 1$ and $Q_1 = 1$. The positive-going edge of CLK4 causes Q_0 to go LOW, while $\overline{Q}_0$ goes HIGH and triggers FF1, causing Q_1 to go LOW. After the leading

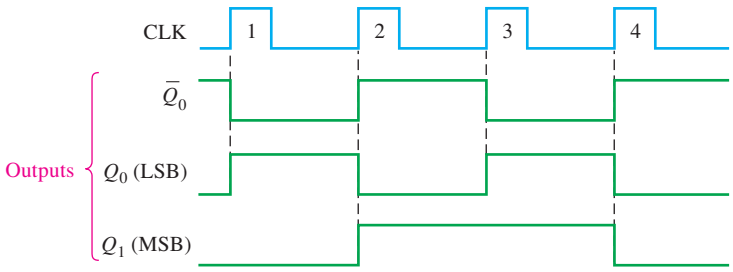


FIGURE 9–5 Timing diagram for the counter of Figure 9–4. As in previous chapters, output waveforms are shown in green.

edge of CLK4, $Q_0 = 0$ and $Q_1 = 0$. The counter has now recycled to its original state (both flip-flops are RESET).

In the timing diagram, the waveforms of the Q_0 and Q_1 outputs are shown relative to the clock pulses as illustrated in Figure 9–5. For simplicity, the transitions of Q_0, Q_1, and the clock pulses are shown as simultaneous even though this is an asynchronous counter. There is, of course, some small delay between the CLK and the Q_0 transition and between the $\overline{Q}_0$ transition and the Q_1 transition.

Note in Figure 9–5 that the 2-bit counter exhibits four different states, as you would expect with two flip-flops ($2^2 = 4$). Also, notice that if Q_0 represents the least significant bit (LSB) and Q_1 represents the most significant bit (MSB), the sequence of counter states represents a sequence of binary numbers as listed in Table 9–1.

In digital logic, Q_0 is always the LSB unless otherwise specified.

TABLE 9–1

Binary state sequence for the counter in Figure 9–4.

Clock Pulse	Q_1	Q_0
Initially	0	0
1	0	1
2	1	0
3	1	1
4 (recycles)	0	0

Since it goes through a binary sequence, the counter in Figure 9–4 is a binary counter. It actually counts the number of clock pulses up to three, and on the fourth pulse it recycles to its original state ($Q_0 = 0$, $Q_1 = 0$). The term **recycle** is commonly applied to counter operation; it refers to the transition of the counter from its final state back to its original state.

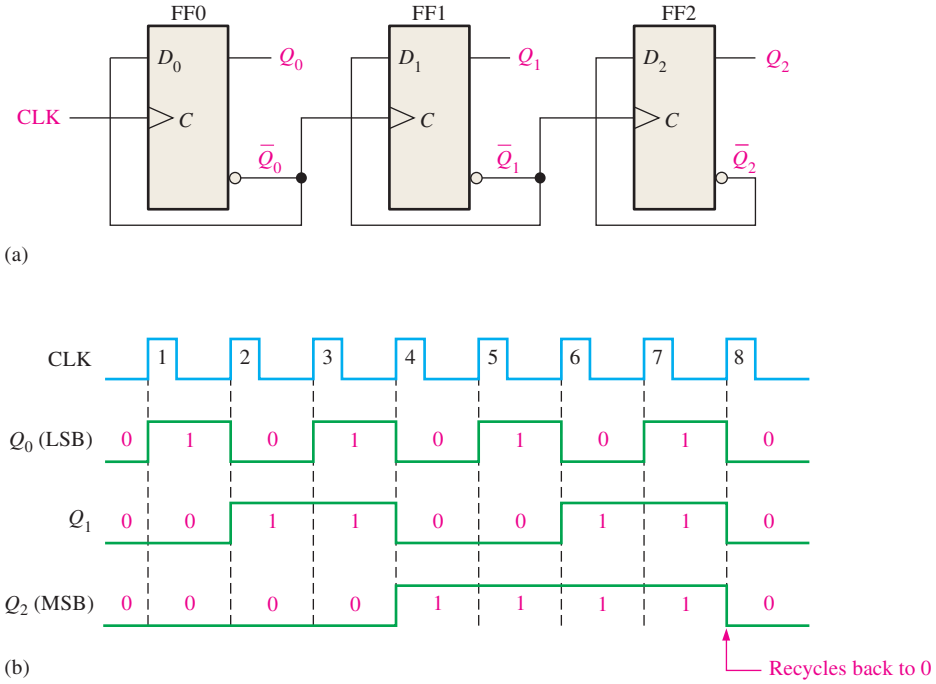
A 3-Bit Asynchronous Binary Counter

The state sequence for a 3-bit binary counter is listed in Table 9–2, and a 3-bit asynchronous binary counter is shown in Figure 9–6(a). The basic operation is the same as that of the 2-bit

TABLE 9–2

State sequence for a 3-bit binary counter.

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



MultiSim

FIGURE 9–6 Three-bit asynchronous binary counter and its timing diagram for one cycle. Open file F09-06 to verify operation.

counter except that the 3-bit counter has eight states, due to its three flip-flops. A timing diagram is shown in Figure 9–6(b) for eight clock pulses. Notice that the counter progresses through a binary count of zero through seven and then recycles to the zero state. This counter can be easily expanded for higher count, by connecting additional toggle flip-flops.

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 9–6 change state on the leading edge of CLK4. This ripple clocking effect is shown in Figure 9–7 for the first four clock pulses, with the propagation delays indicated. The LOW-to-HIGH transition of

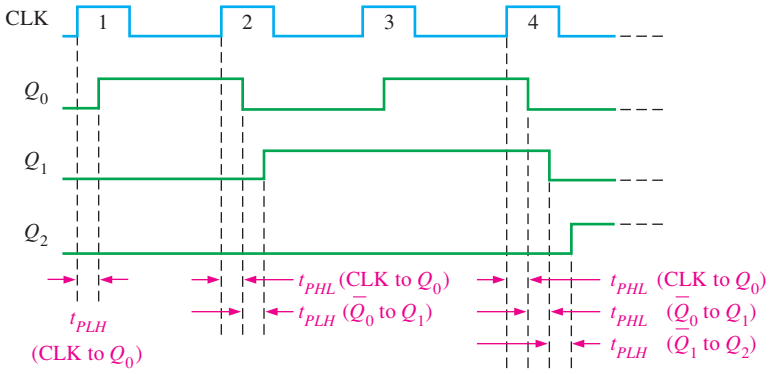


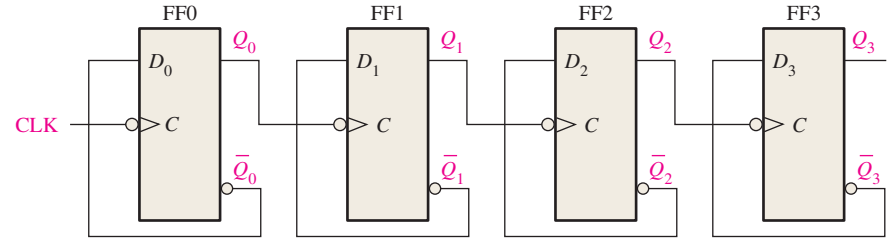
FIGURE 9–7 Propagation delays in a 3-bit asynchronous (ripple-clocked) binary counter.

Q_0 occurs one delay time (t_{PLH}) after the positive-going transition of the clock pulse. The LOW-to-HIGH transition of Q_1 occurs one delay time (t_{PLH}) after the positive-going transition of $\overline{Q}_0$. The LOW-to-HIGH transition of Q_2 occurs one delay time (t_{PLH}) 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.

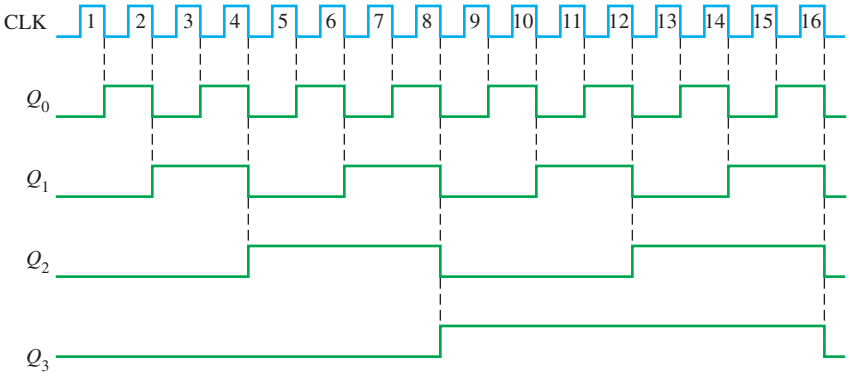
This cumulative delay of an asynchronous counter is a major disadvantage in many applications because it limits the rate at which the counter can be clocked and creates decoding problems. The maximum cumulative delay in a counter must be less than the period of the clock waveform.

EXAMPLE 9–1

A 4-bit asynchronous binary counter is shown in Figure 9–8(a). Each D 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.



(a)



(b)

FIGURE 9–8 Four-bit asynchronous binary counter and its timing diagram. Open file F09-08 and verify the operation.

MultiSim

Solution

The timing diagram with delays omitted is as shown in Figure 9–8(b). For the total delay time, the effect of CLK8 or CLK16 must propagate through four flip-flops before Q_3 changes, so

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

The maximum clock frequency is

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

The counter should be operated below this frequency to avoid problems due to the propagation delay.

Related Problem*

Show the timing diagram if all of the flip-flops in Figure 9–8(a) are positive edge-triggered.

*Answers are at the end of the chapter.

Asynchronous Decade Counters

A counter can have 2^n states, where n is the number of flip-flops.

The **modulus** of a counter is the number of unique states through which the counter will sequence. 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 be designed to have a number of states in their sequence that is less than the maximum of 2^n. This type of 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 $2^3 = 8$).

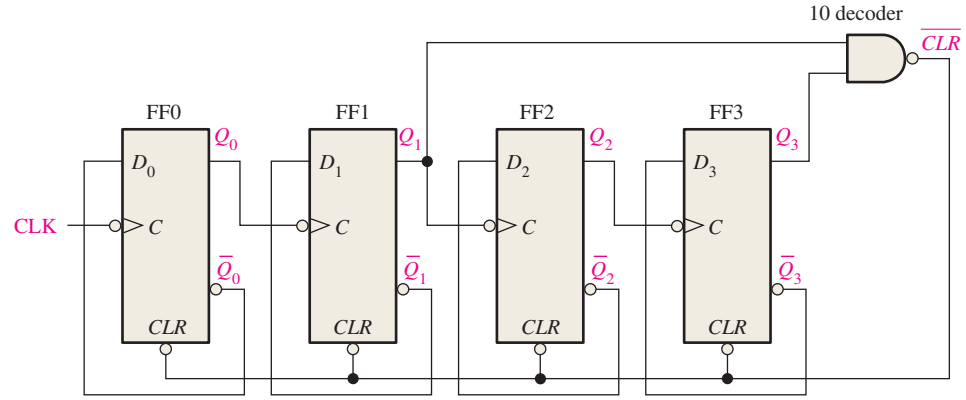
Let's use a 4-bit asynchronous counter such as the one in Example 9–1 and modify its sequence to illustrate the principle of truncated counters. One way to make the counter recycle after the count of nine (1001) is to decode count ten (1010) with a NAND gate and connect the output of the NAND gate to the clear ($\overline{CLR}$) inputs of the flip-flops, as shown in Figure 9–9(a).

Partial Decoding

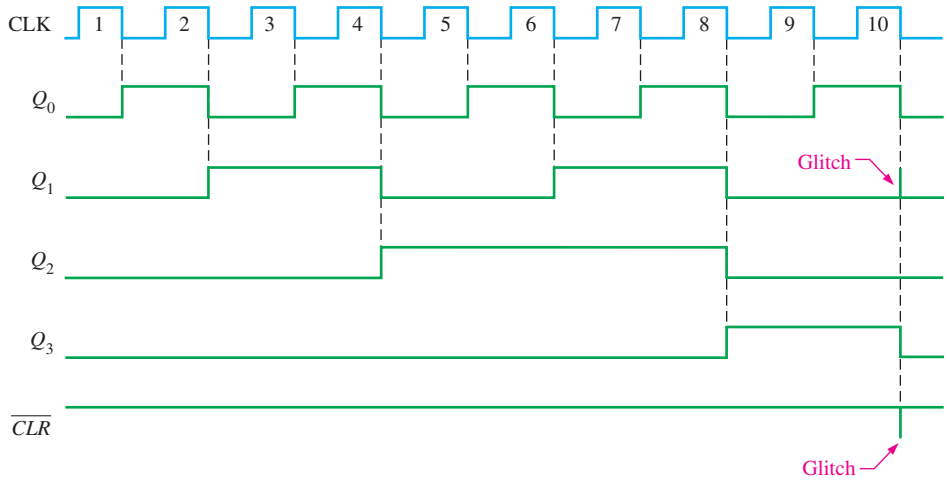
Notice in Figure 9–9(a) that only Q_1 and Q_3 are connected to the NAND gate inputs. This arrangement is an example of *partial decoding,* in which the two unique states ($Q_1 = 1$ and $Q_3 = 1$) are sufficient to decode the count of ten because none of the other states (zero through nine) have both Q_1 and Q_3 HIGH at the same time. When the counter goes 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 9–9(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 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 9–2 shows.



(a)



(b)

FIGURE 9–9 An asynchronously clocked decade counter with asynchronous recycling.

EXAMPLE 9–2

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

Solution

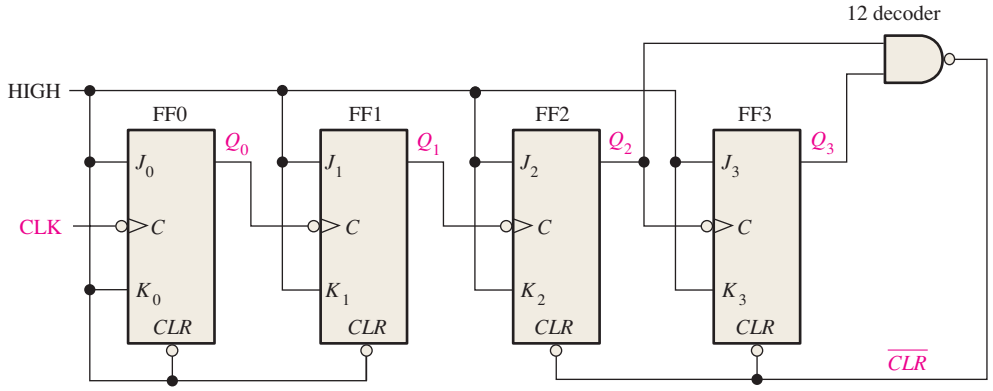
Since three flip-flops can produce a maximum of eight states, four flip-flops are required to produce any modulus greater than eight but less than or equal to sixteen.

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:

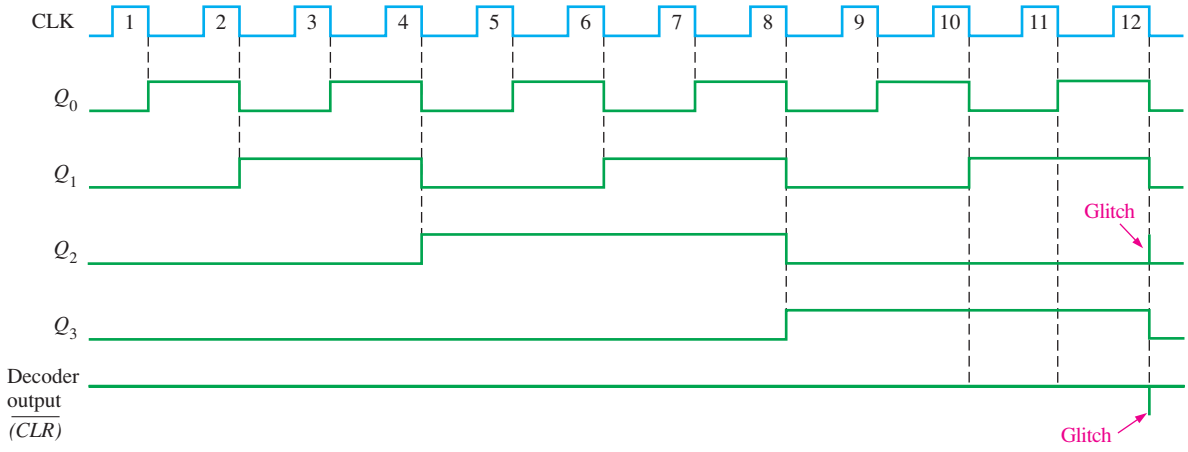
Q_3	Q_2	Q_1	Q_0	
0	0	0	0	←
.	.	.	.	
.	.	.	.	Recycles
.	.	.	.	
1	0	1	1	←
1	1	0	0	← Normal next state

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 9–10(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 9–10(b). (It is in count twelve for only a few nanoseconds before it is reset by the glitch on $\overline{CLR}$.)



(a)



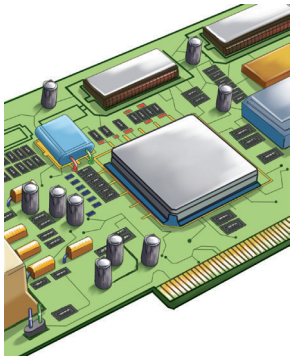
(b)

FIGURE 9–10 Asynchronously clocked modulus-12 counter with asynchronous recycling.

Related Problem

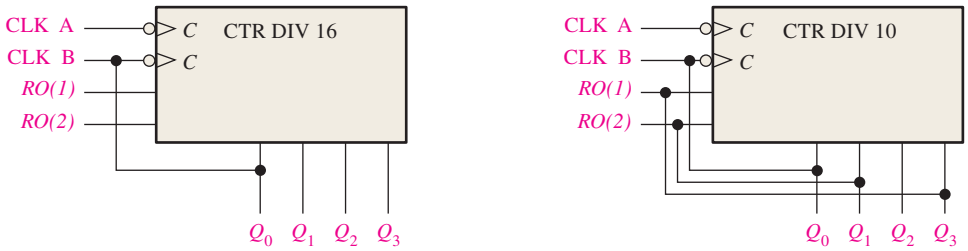
How can the counter in Figure 9–10(a) be modified to make it a modulus-13 counter?

IMPLEMENTATION: 4-BIT ASYNCHRONOUS BINARY COUNTER



Fixed-Function Device The 74HC93 is an example of a specific integrated circuit asynchronous counter. This device actually consists of a single flip-flop (CLK A) and a 3-bit asynchronous counter (CLK B). 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 $\overline{CLR}$.

Additionally, the 74HC93 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 by the logic symbol in Figure 9–11(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 by the logic symbol in Figure 9–11(b).



(a) 74HC93 connected as a modulus-16 counter

(b) 74HC93 connected as a decade counter

FIGURE 9–11 Two configurations of the 74HC93 asynchronous counter. (The qualifying label, CTR DIV *n*, indicates a counter with *n* states.)

Programmable Logic Device (PLD) The VHDL code for a generic 4-bit asynchronous binary counter using J-K flip flops with preset (PRN) and clear (CLRN) inputs is as follows:

```
library ieee;
use ieee.std_logic_1164.all;

entity AsyncFourBitBinCntr is
   port (Clock, Clr: in std_logic; Q0, Q1, Q2, Q3: inout std_logic);
end entity AsyncFourBitBinCntr;

architecture LogicOperation of AsyncFourBitBinCntr is
component jkff is
   port (J, K, Clk, PRN, CLRN: in std_logic; Q: out std_logic);
end component jkff;

begin
   FF0: jkff port map(J=>'1', K=>'1', Clk=>Clock,  CLRN=>Clr, PRN=>'1', Q=>Q0);
   FF1: jkff port map(J=>'1', K=>'1', Clk=>not Q0, CLRN=>Clr, PRN=>'1', Q=>Q1);
   FF2: jkff port map(J=>'1', K=>'1', Clk=>not Q1, CLRN=>Clr, PRN=>'1', Q=>Q2);
   FF3: jkff port map(J=>'1', K=>'1', Clk=>not Q2, CLRN=>Clr, PRN=>'1', Q=>Q3);
end architecture LogicOperation;
```

Inputs and outputs declared

J-K flip-flop component declaration

Instantiations define how each flip-flop is connected.

SECTION 9–2 CHECKUP

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?

9–3 Synchronous Counters

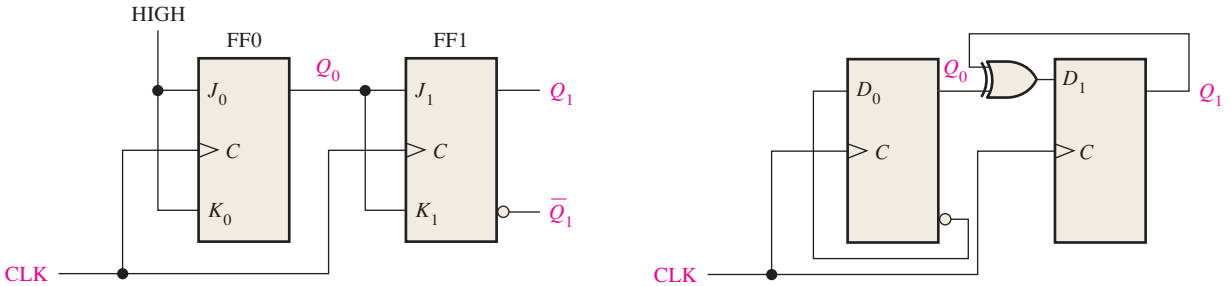
The term **synchronous** refers to events that have a fixed time relationship with each other. A **synchronous counter** is one in which all the flip-flops in the counter are clocked at the same time by a common clock pulse. J-K flip-flops are used to illustrate most synchronous counters. D flip-flops can also be used but generally require more logic because of having no direct toggle or no-change states.

After completing this section, you should be able to

- Describe the operation of a 2-bit synchronous binary counter
- Describe the operation of a 3-bit synchronous binary counter
- Describe the operation of a 4-bit synchronous binary counter
- Describe the operation of a synchronous decade counter
- Develop counter timing diagrams

A 2-Bit Synchronous Binary Counter

Figure 9–12 shows a 2-bit synchronous binary counter. Notice that an arrangement different from that for the asynchronous counter must be used for the J_1 and K_1 inputs of FF1 in order to achieve a binary sequence. A D flip-flop implementation is shown in part (b).



(a) J-K flip-flop (b) D flip-flop

FIGURE 9–12 2-bit synchronous binary counters.

The clock input goes to each flip-flop in a synchronous counter.

The operation of a J-K flip-flop synchronous counter is as follows: First, assume that the counter is initially in the binary 0 state; that is, both flip-flops are RESET. When the positive edge of the first clock pulse is applied, FF0 will toggle and Q_0 will therefore go HIGH. What happens to FF1 at the positive-going edge of CLK1? To find out, let's look at the input conditions of FF1. Inputs J_1 and K_1 are both LOW because Q_0, to which they are connected, has not yet gone HIGH. Remember, there is a propagation delay from the triggering edge of the clock pulse until the Q output actually makes a transition. So, $J = 0$ and $K = 0$ when the leading edge of the first clock pulse is applied. This is a no-change condition, and therefore FF1 does not change state. A timing detail of this portion of the counter operation is shown in Figure 9–13(a).

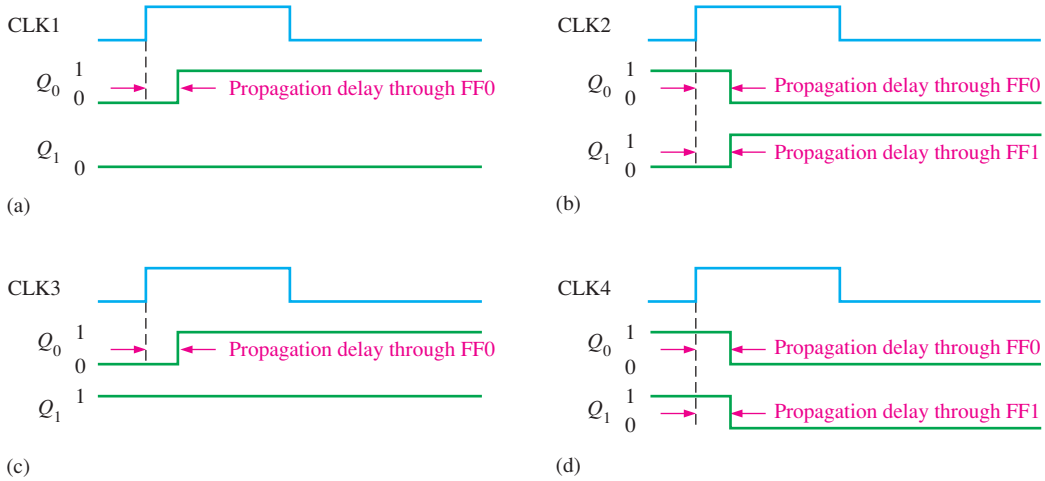


FIGURE 9–13 Timing details for the 2-bit synchronous counter operation (the propagation delays of both flip-flops are assumed to be equal).

After CLK1, $Q_0 = 1$ and $Q_1 = 0$ (which is the binary 1 state). When the leading edge of CLK2 occurs, FF0 will toggle and Q_0 will go LOW. Since FF1 has a HIGH ($Q_0 = 1$) on its J_1 and K_1 inputs at the triggering edge of this clock pulse, the flip-flop toggles and Q_1 goes HIGH. Thus, after CLK2, $Q_0 = 0$ and $Q_1 = 1$ (which is a binary 2 state). The timing detail for this condition is shown in Figure 9–13(b).

When the leading edge of CLK3 occurs, FF0 again toggles to the SET state ($Q_0 = 1$), and FF1 remains SET ($Q_1 = 1$) because its J_1 and K_1 inputs are both LOW ($Q_0 = 0$). After this triggering edge, $Q_0 = 1$ and $Q_1 = 1$ (which is a binary 3 state). The timing detail is shown in Figure 9–13(c).

Finally, at the leading edge of CLK4, Q_0 and Q_1 go LOW because they both have a toggle condition on their *J* and *K* inputs. The timing detail is shown in Figure 9–13(d). The counter has now recycled to its original state, binary 0. Examination of the D flip-flop counter in Figure 9–12(b) will show the timing diagram is the same as for the J-K flip-flop counter.

The complete timing diagram for the counters in Figure 9–12 is shown in Figure 9–14. Notice that all the waveform transitions appear coincident; that is, the propagation delays are not indicated. Although the delays are an important factor in the synchronous counter operation, in an overall timing diagram they are normally omitted for simplicity. Major waveform relationships resulting from the normal operation of a circuit can be conveyed completely without showing small delay and timing differences. However, in high-speed digital circuits, these small delays are an important consideration in design and troubleshooting.

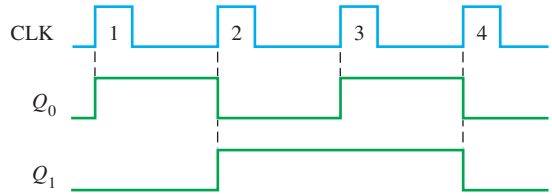
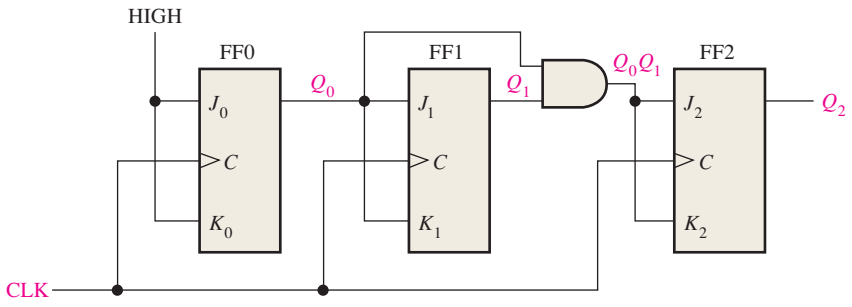


FIGURE 9–14 Timing diagram for the counters of Figure 9–12.

A 3-Bit Synchronous Binary Counter

A 3-bit synchronous binary counter is shown in Figure 9–15, and its timing diagram is shown in Figure 9–16. You can understand this counter operation by examining its sequence of states as shown in Table 9–3.



MultiSim

FIGURE 9–15 A 3-bit synchronous binary counter. Open file F09-15 to verify the operation.

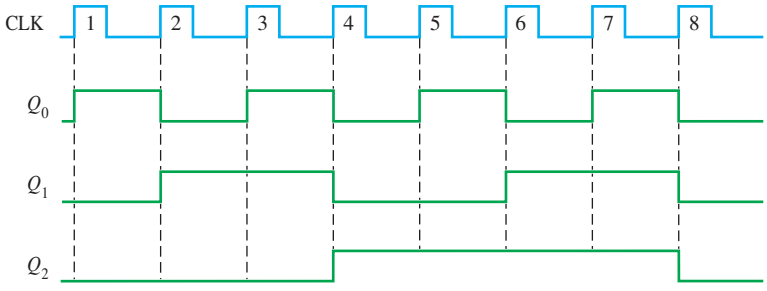


FIGURE 9–16 Timing diagram for the counter of Figure 9–15.

TABLE 9–3

State sequence for a 3-bit binary counter.

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

InfoNote

The TSC or *time stamp counter* in some microprocessors is used for performance monitoring, which enables a number of parameters important to the overall performance of a system to be determined exactly. By reading the TSC before and after the execution of a procedure, the precise time required for the procedure can be determined based on the processor cycle time. In this way, the TSC forms the basis for all time evaluations in connection with optimizing system operation. For example, it can be accurately determined which of two or more programming sequences is more efficient. This is a very useful tool for compiler developers and system programmers in producing the most effective code.

First, let's 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's 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.

The analysis of the counter in Figure 9–15 is summarized in Table 9–4.

TABLE 9–4

Summary of the analysis of the counter in Figure 9–15.

	Outputs			*J-K* Inputs						At the Next Clock Pulse		
Clock Pulse	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	FF2	FF1	FF0
Initially	0	0	0	0	0	0	0	1	1	NC*	NC	Toggle
1	0	0	1	0	0	1	1	1	1	NC	Toggle	Toggle
2	0	1	0	0	0	0	0	1	1	NC	NC	Toggle
3	0	1	1	1	1	1	1	1	1	Toggle	Toggle	Toggle
4	1	0	0	0	0	0	0	1	1	NC	NC	Toggle
5	1	0	1	0	0	1	1	1	1	NC	Toggle	Toggle
6	1	1	0	0	0	0	0	1	1	NC	NC	Toggle
7	1	1	1	1	1	1	1	1	1	Toggle	Toggle	Toggle
										Counter recycles back to 000.		

*NC indicates *No Change*.

A 4-Bit Synchronous Binary Counter

Figure 9–17(a) shows a 4-bit synchronous binary counter, and Figure 9–17(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

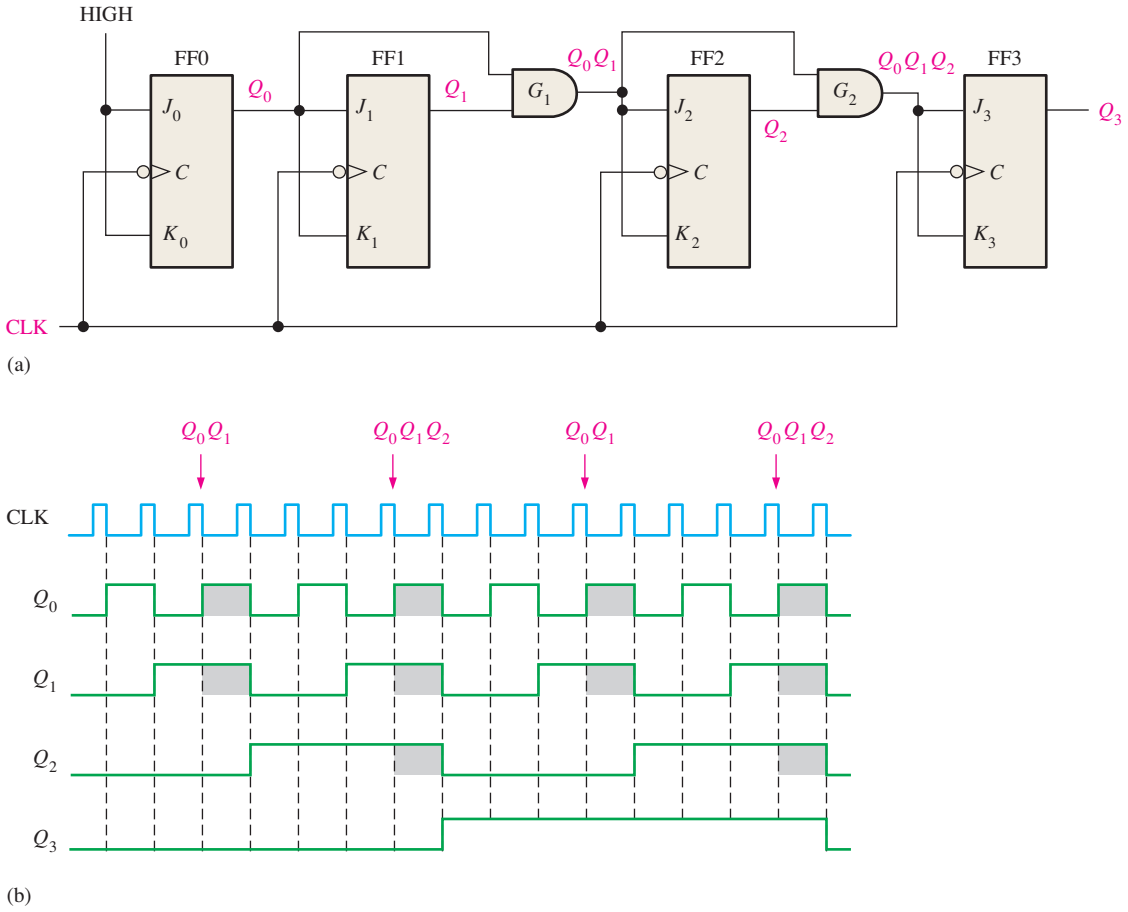


FIGURE 9–17 A 4-bit synchronous binary counter and timing diagram. Times where the AND gate outputs are HIGH are indicated by the shaded areas.

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.

A 4-Bit Synchronous Decade Counter

A decade counter has ten states.

As you know, a BCD decade counter exhibits a truncated binary sequence and goes from 0000 through the 1001 state. Rather than going from the 1001 state to the 1010 state, it recycles to the 0000 state. A synchronous BCD decade counter is shown in Figure 9–18. The timing diagram for the decade counter is shown in Figure 9–19.

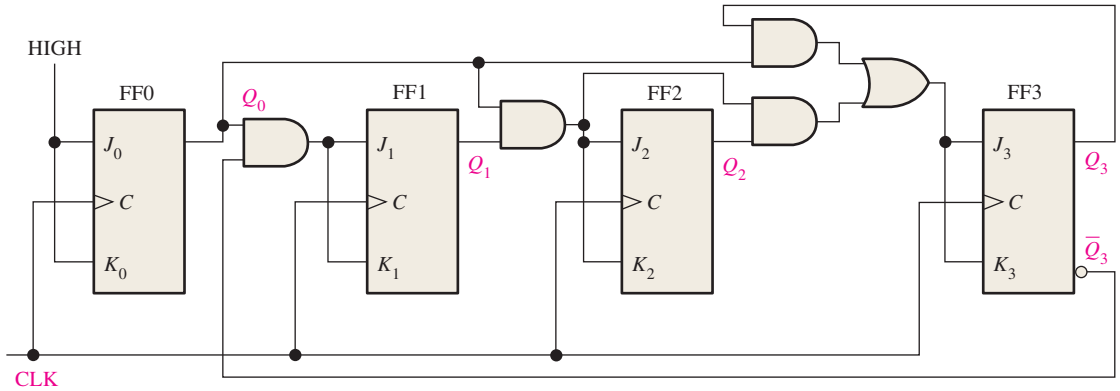


FIGURE 9–18 A synchronous BCD decade counter. Open file F09-18 to verify operation.

MultiSim

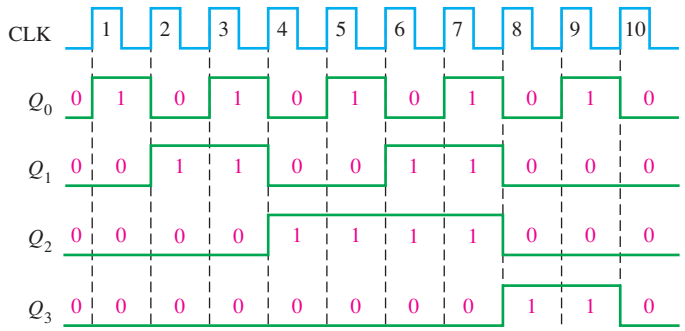


FIGURE 9–19 Timing diagram for the BCD decade counter (Q_0 is the LSB).

The counter operation is shown by the sequence of states in Table 9–5. First, notice that FF0 (Q_0) toggles on each clock pulse, so the logic equation for its J_0 and K_0 inputs is

$$J_0 = K_0 = 1$$

This equation is implemented by connecting J_0 and K_0 to a constant HIGH level.

TABLE 9–5

States of a BCD decade counter.

Clock Pulse	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (recycles)	0	0	0	0

Next, notice in Table 9–5 that FF1 (Q_1) changes on the next clock pulse each time $Q_0 = 1$ and $Q_3 = 0$, so the logic equation for the J_1 and K_1 inputs is

$$J_1 = K_1 = Q_0\overline{Q}_3$$

This equation is implemented by ANDing Q_0 and $\overline{Q}_3$ and connecting the gate output to the J_1 and K_1 inputs of FF1.

Flip-flop 2 (Q_2) changes on the next clock pulse each time both $Q_0 = 1$ and $Q_1 = 1$. This requires an input logic equation as follows:

$$J_2 = K_2 = Q_0Q_1$$

This equation is implemented by ANDing Q_0 and Q_1 and connecting the gate output to the J_2 and K_2 inputs of FF2.

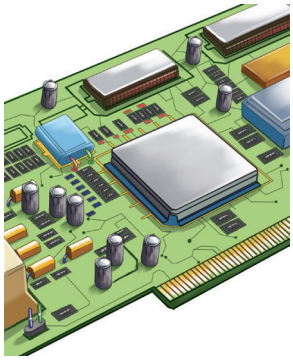
Finally, FF3 (Q_3) changes to the opposite state on the next clock pulse each time $Q_0 = 1$, $Q_1 = 1$, and $Q_2 = 1$ (state 7), or when $Q_0 = 1$ and $Q_3 = 1$ (state 9). The equation for this is as follows:

$$J_3 = K_3 = Q_0Q_1Q_2 + Q_0Q_3$$

This function is implemented with the AND/OR logic connected to the J_3 and K_3 inputs of FF3 as shown in the logic diagram in Figure 9–18. Notice that the differences between this

decade counter and the modulus-16 binary counter in Figure 9–17(a) are the $Q_0\overline{Q}_3$ AND gate, the Q_0Q_3 AND gate, and the OR gate; this arrangement detects the occurrence of the 1001 state and causes the counter to recycle properly on the next clock pulse.

IMPLEMENTATION: 4-BIT SYNCHRONOUS BINARY COUNTER



Fixed-Function Device The 74HC163 is an example of an integrated circuit 4-bit synchronous binary counter. A logic symbol is shown in Figure 9–20 with pin numbers in parentheses. This counter has several features in addition to the basic functions previously discussed for the general synchronous binary counter.

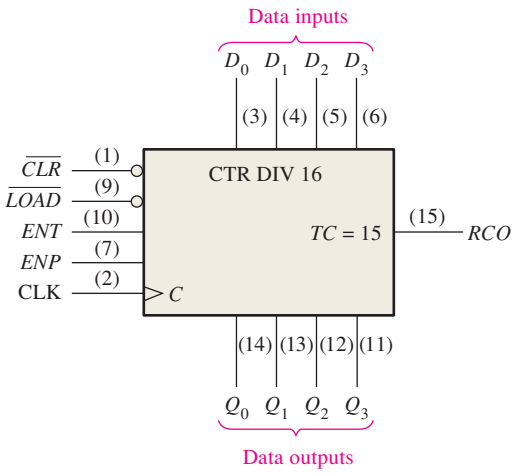


FIGURE 9–20 The 74HC163 4-bit synchronous binary counter. (The qualifying label CTR DIV 16 indicates a counter with sixteen states.)

First, the counter can be synchronously preset to any 4-bit binary number by applying the proper levels to the parallel data inputs. When a LOW is applied to the $\overline{LOAD}$ input, the counter will assume the state of the data inputs on the next clock pulse. Thus, the counter sequence can be started with any 4-bit binary number.

Also, there is an active-LOW clear input ($\overline{CLR}$), which synchronously resets all four flip-flops in the counter. There are two enable inputs, *ENP* and *ENT*. These inputs must both be HIGH for the counter to sequence through its binary states. When at least one input is LOW, the counter is disabled. The ripple clock output (*RCO*) goes HIGH when the counter reaches the last state in its sequence of fifteen, called the **terminal count** ($TC = 15$). This output, in conjunction with the enable inputs, allows these counters to be cascaded for higher count sequences.

Figure 9–21 shows a timing diagram of this counter being preset to twelve (1100) and then counting up to its terminal count, fifteen (1111). Input D_0 is the least significant input bit, and Q_0 is the least significant output bit.

Let's examine this timing diagram in detail. This will aid you in interpreting timing diagrams in this chapter or on manufacturers' data sheets. To begin, the LOW level pulse on the $\overline{CLR}$ input causes all the outputs (Q_0, Q_1, Q_2, and Q_3) to go LOW.

Next, the LOW level pulse on the $\overline{LOAD}$ input synchronously enters the data on the data inputs (D_0, D_1, D_2, and D_3) into the counter. These data appear on the Q outputs at the time of the first positive-going clock edge after $\overline{LOAD}$ goes LOW. This is the preset operation. In this particular example, Q_0 is LOW, Q_1 is LOW, Q_2 is HIGH, and Q_3 is HIGH. This, of course, is a binary 12 (Q_0 is the LSB).

The counter now advances through states 13, 14, and 15 on the next three positive-going clock edges. It then recycles to 0, 1, 2 on the following clock pulses. Notice that

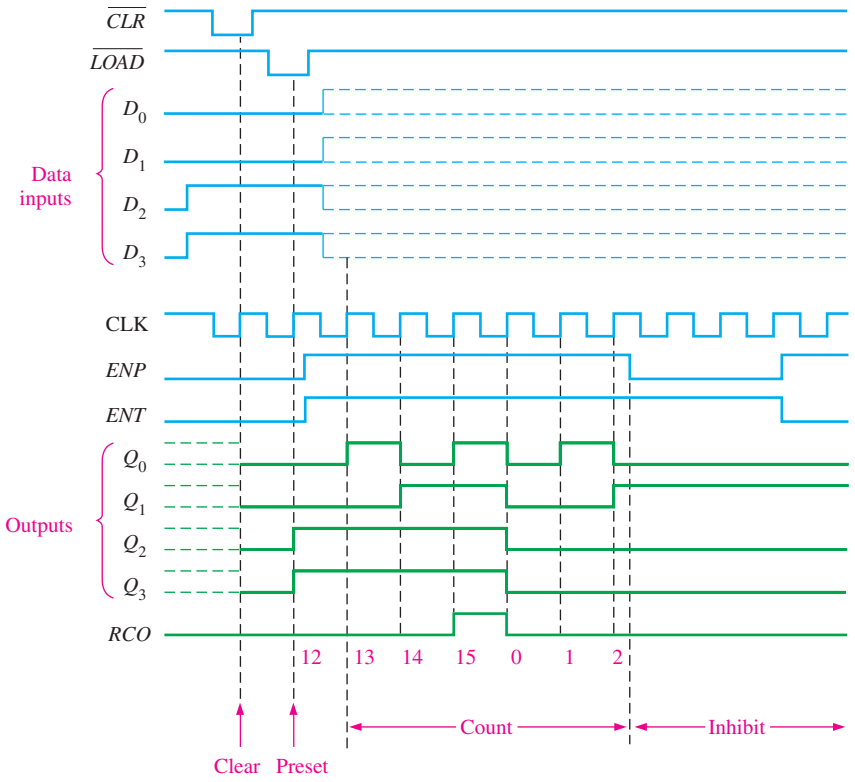


FIGURE 9–21 Timing example for a 74HC163.

both *ENP* and *ENT* inputs are HIGH during the state sequence. When *ENP* goes LOW, the counter is inhibited and remains in the binary 2 state.

Programmable Logic Device (PLD) The VHDL code for a 4-bit synchronous decade counter using J-K flip flops is as follows:

```
library ieee;
use ieee.std_logic_1164.all;

entity FourBitSynchDecadeCounter is
   port (Clk: in std_logic; Q0, Q1, Q2, Q3: inout std_logic);
end entity FourBitSynchDecadeCounter;

architecture LogicOperation of FourBitSynchDecadeCounter is

component jkff is
   port (J, K, Clk: in std_logic; Q: out std_logic);
end component jkff;

signal J1, J2, J3: std_logic;
begin
J1 <= Q0 and not Q3;
J2 <= Q1 and Q0;
J3 <= (Q2 and J2) or (Q0 and Q3);

FF0: jkff port map (J => '1', K => '1', Clk => Clk, Q => Q0);
FF1: jkff port map (J => J1, K => J1, Clk => Clk, Q => Q1);
FF2: jkff port map (J => J2, K => J2, Clk => Clk, Q => Q2);
FF3: jkff port map (J => J3, K => J3, Clk => Clk, Q => Q3);
end architecture LogicOperation;
```

Input and outputs declared

Component declaration for the J-K flip-flop

Boolean expressions for *J* input of each flip-flop (*J* = *K*)

Instantiations define connections for each flip-flop.

SECTION 9–3 CHECKUP

1. How does a synchronous counter differ from an asynchronous counter?
2. Explain the function of the preset feature of counters such as the 74HC163.
3. Describe the purpose of the *ENP* and *ENT* inputs and the *RCO* output for the 74HC163 counter.

9–4 Up/Down Synchronous Counters

An **up/down counter** is one that is capable of progressing in either direction through a certain sequence. An up/down counter, sometimes called a bidirectional counter, can have any specified sequence of states. A 3-bit binary counter that advances upward through its sequence (0, 1, 2, 3, 4, 5, 6, 7) and then can be reversed so that it goes through the sequence in the opposite direction (7, 6, 5, 4, 3, 2, 1, 0) is an illustration of up/down sequential operation.

After completing this section, you should be able to

- Explain the basic operation of an up/down counter
- Discuss the 74HC190 up/down decade counter

In general, most up/down counters can be reversed at any point in their sequence. For instance, the 3-bit binary counter can be made to go through the following sequence:

$$\overbrace{0, 1, 2, 3, 4, 5,}^{\text{UP}}\ \underbrace{4, 3, 2,}_{\text{DOWN}}\ \overbrace{3, 4, 5, 6, 7,}^{\text{UP}}\ \underbrace{6, 5,}_{\text{DOWN}}\ \text{etc.}$$

Table 9–6 shows the complete up/down sequence for a 3-bit binary counter. The arrows indicate the state-to-state movement of the counter for both its UP and its DOWN modes of operation. An examination of Q_0 for both the up and down sequences shows that FF0 toggles on each clock pulse. Thus, the J_0 and K_0 inputs of FF0 are

$$J_0 = K_0 = 1$$

TABLE 9–6

Up/Down sequence for a 3-bit binary counter.

Clock Pulse	Up	Q_2	Q_1	Q_0	Down
0	↺	0	0	0	↻
1	↺	0	0	1	↻
2	↺	0	1	0	↻
3	↺	0	1	1	↻
4	↺	1	0	0	↻
5	↺	1	0	1	↻
6	↺	1	1	0	↻
7	↺	1	1	1	↻

For the up sequence, Q_1 changes state on the next clock pulse when $Q_0 = 1$. For the down sequence, Q_1 changes on the next clock pulse when $Q_0 = 0$. Thus, the J_1 and K_1 inputs of FF1 must equal 1 under the conditions expressed by the following equation:

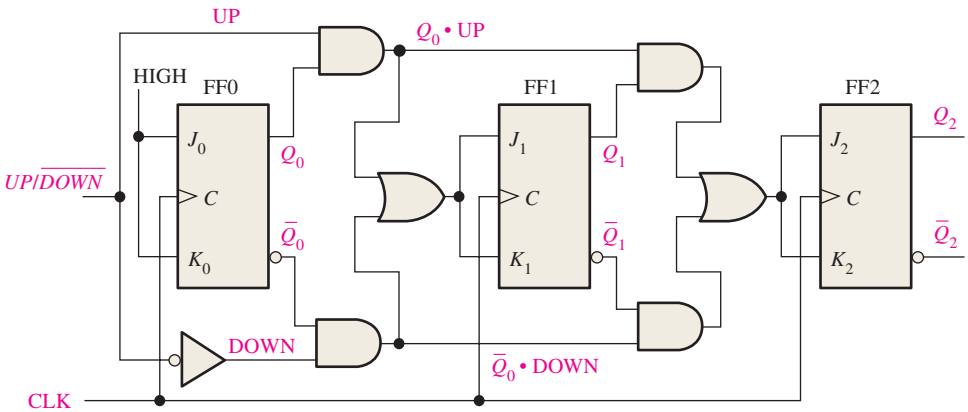
$$J_1 = K_1 = (Q_0 \cdot \text{UP}) + (\overline{Q}_0 \cdot \text{DOWN})$$

For the up sequence, Q_2 changes state on the next clock pulse when $Q_0 = Q_1 = 1$. For the down sequence, Q_2 changes on the next clock pulse when $Q_0 = Q_1 = 0$. Thus, the J_2 and K_2 inputs of FF2 must equal 1 under the conditions expressed by the following equation:

$$J_2 = K_2 = (Q_0 \cdot Q_1 \cdot \text{UP}) + (\overline{Q}_0 \cdot \overline{Q}_1 \cdot \text{DOWN})$$

Each of the conditions for the J and K inputs of each flip-flop produces a toggle at the appropriate point in the counter sequence.

Figure 9–22 shows a basic implementation of a 3-bit up/down binary counter using the logic equations just developed for the J and K inputs of each flip-flop. Notice that the $UP/\overline{DOWN}$ control input is HIGH for UP and LOW for DOWN.



MultiSim

FIGURE 9–22 A basic 3-bit up/down synchronous counter. Open file F09-22 to verify operation.

EXAMPLE 9–3

Show the timing diagram and determine the sequence of a 4-bit synchronous binary up/down counter if the clock and $UP/\overline{DOWN}$ control inputs have waveforms as shown in Figure 9–23(a). The counter starts in the all-0s state and is positive edge-triggered.

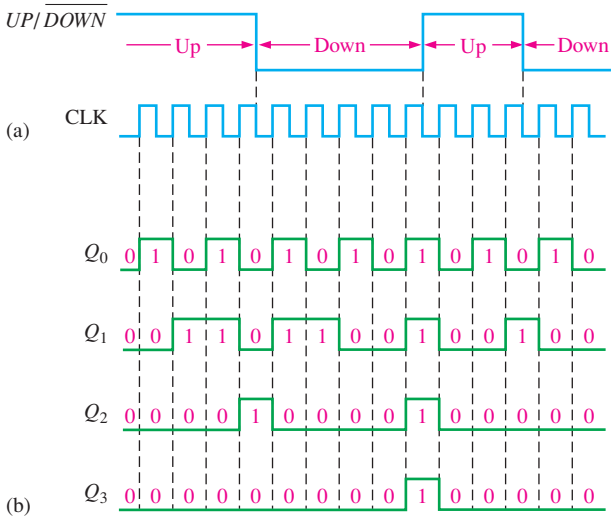


FIGURE 9–23

Solution

The timing diagram showing the Q outputs is shown in Figure 9–23(b). From these waveforms, the counter sequence is as shown in Table 9–7.

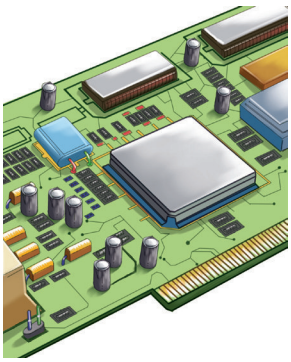
TABLE 9–7

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	
0	0	0	1	
0	0	1	0	UP
0	0	1	1	
0	1	0	0	
0	0	1	1	
0	0	1	0	
0	0	0	1	DOWN
0	0	0	0	
1	1	1	1	
0	0	0	0	
0	0	0	1	UP
0	0	1	0	
0	0	0	1	DOWN
0	0	0	0	

Related Problem

Show the timing diagram if the $UP/\overline{DOWN}$ control waveform in Figure 9–23(a) is inverted.

IMPLEMENTATION: UP/DOWN DECADE COUNTER



Fixed-Function Device Figure 9–24 shows a logic diagram for the 74HC190, an example of an integrated circuit up/down synchronous decade counter. The direction of the count is determined by the level of the up/down input ($D/\overline{U}$). When this input is HIGH, the counter counts down; when it is LOW, the counter counts up. Also, this device can be preset to any desired BCD digit as determined by the states of the data inputs when the $\overline{LOAD}$ input is LOW.

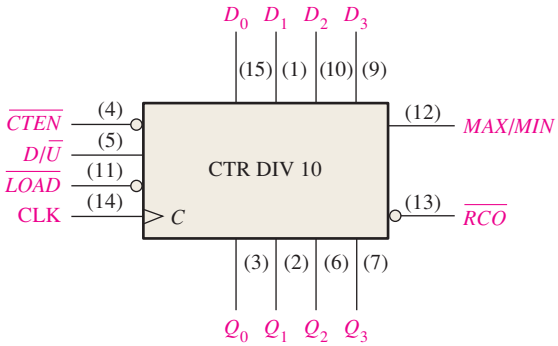


FIGURE 9–24 The 74HC190 up/down synchronous decade counter.

The *MAX/MIN* output produces a HIGH pulse when the terminal count nine (1001) is reached in the UP mode or when the terminal count zero (0000) is reached in the DOWN mode. The *MAX/MIN* output, the ripple clock output ($\overline{RCO}$), and the count enable input ($\overline{CTEN}$) are used when cascading counters. (Cascaded counters are discussed in Section 9–6.)

Figure 9–25 is a timing diagram that shows the 74HC190 counter preset to seven (0111) and then going through a count-up sequence followed by a count-down sequence. The *MAX/MIN* output is HIGH when the counter is in either the all-0s state (*MIN*) or the 1001 state (*MAX*).

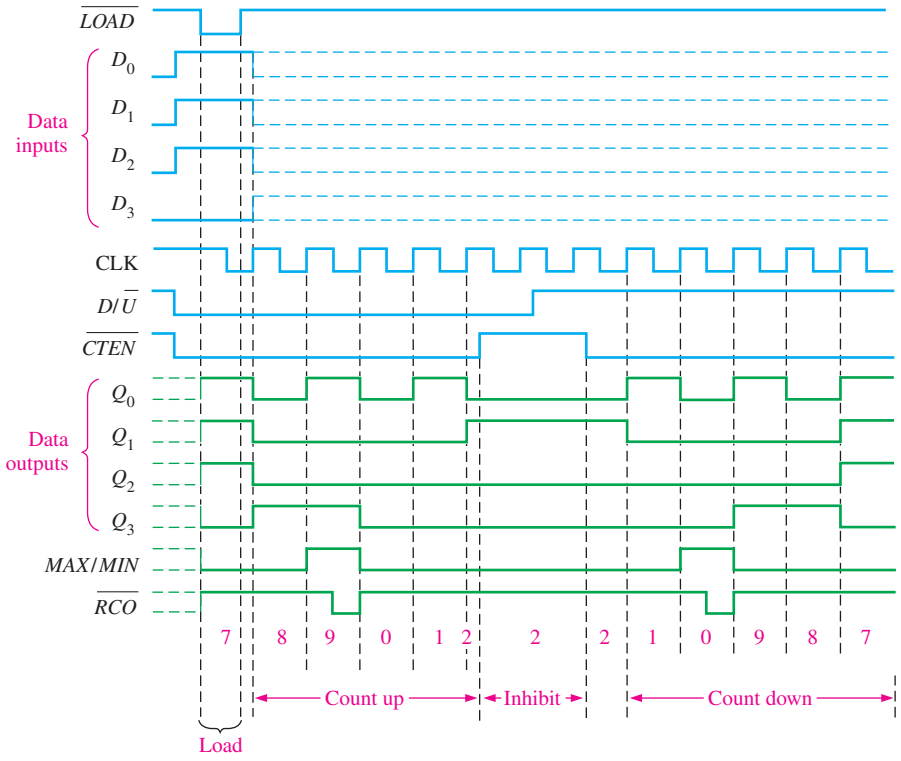


FIGURE 9–25 Timing example for a 74HC190.

Programmable Logic Device (PLD) A VHDL code for an up/down decade counter using J-K flip-flops is as follows:

```
library ieee;
use ieee.std_logic_1164.all;

entity UpDnDecadeCntr is
   port (UPDN, Clk: in std_logic; Q0, Q1, Q2, Q3: buffer std_logic);
end entity UpDnDecadeCntr;

architecture LogicOperation of UpDnDecadeCntr is

component jkff is
   port (J, K, Clk: in std_logic; Q: buffer std_logic);
end component jkff;
```

UPDN: Counter direction
Clk: System clock
Q0-Q3: Counter output

J-K flip flop component

```
function UpDown(A, B, C, D: in std_logic)
   return std_logic is
begin
   return((A and B) or (C and D));
end function UpDown;

signal J1Up, J1Dn, J1, J2, J3: std_logic;
begin
   J1Up <= UPDN and Q0; J1Dn <= not UPDN and not Q0;
   UpDn1: J1 <= UpDown(UPDN, Q0, not UPDN, not Q0);
   UpDn2: J2 <= UpDown(J1Up, Q1, J1Dn, not Q1);
   UpDn3: J3 <= UpDown(J1Up and Q1, Q2, J1Dn and not Q1, not Q2);

   FF0: jkff port map (J =>'1', K =>'1', Clk => Clk, Q => Q0);
   FF1: jkff port map (J => J1, K => J1, Clk => Clk, Q => Q1);
   FF2: jkff port map (J => J2, K => J2, Clk => Clk, Q => Q2);
   FF3: jkff port map (J => J3, K => J3, Clk => Clk, Q => Q3);
end architecture LogicOperation;
```

Function UpDown is a helper function performing the common logic between stages performed by the two AND gates applied to the OR gate supplying the J K inputs of the next stage. See Figure 9–22.

J1Up: Initial Up logic for FF1.
J1Dn: Initial Down logic for FF1.
J1-J3: Variable for combined UpDown applied to FF1-FF3.

Identifiers J1, J2, and J3 complete the up/down logic applied to the J and K inputs of flip-flop stages FF0-FF1. Using a function to perform operations common to multiple tasks simplifies the overall code design and implementation.

Flip-flop stages FF0-FF3 complete the Up/Down counter.

SECTION 9–4 CHECKUP

1. A 4-bit up/down binary counter is in the DOWN mode and in the 1010 state. On the next clock pulse, to what state does the counter go?
2. What is the terminal count of a 4-bit binary counter in the UP mode? In the DOWN mode? What is the next state after the terminal count in the DOWN mode?

9–5 Design of Synchronous Counters

In this section, you will learn the six steps to design a counter (state machine). As you learned in Section 9–1, sequential circuits can be classified into two types: (1) those in which the output or outputs depend only on the present internal state (Moore state machines) and (2) those in which the output or outputs depend on both the present state and the input or inputs (Mealy state machines). This section is recommended for those who want an introduction to counter design or to state machine design in general. It is not a prerequisite for any other material.

After completing this section, you should be able to

- Develop a state diagram for a given sequence
- Develop a next-state table for a specified counter sequence
- Create a flip-flop transition table
- Use the Karnaugh map method to derive the logic requirements for a synchronous counter
- Implement a counter to produce a specified sequence of states

Step 1: State Diagram

The first step in the design of a state machine (counter) is to create a state diagram. A **state diagram** shows the progression of states through which the counter advances when it is