Digital System Design

Electrical Department-Fourth Stage

Lecture Four

By

Assistant Lecturer: Adnan Ali Abdullah

2024-2023

Finite State Machines and Algorithm State machine (ASM) with applications.

Lecture Outlines

- Finite State Machines
- Asynchronous Counters
- Synchronous Counters
- Up/Down Synchronous Counters

INTRODUCTION

- We learned that the flip-flops can be connected together to perform counting operations. Such a group of flip-flops is a counter, which is a type of finite state machine. 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.

Finite State Machines

- A **state machine** is a sequential circuit having a limited (finite) number of states occurring in a prescribed order.

- A counter is an example of a state machine; the number of states is called the modulus. Two basic types of state machines are **the Moore** and **the Mealy**.

- The **Moore state machine** is one where the outputs depend only on the internal present state.

- The **Mealy state machine** is one where the outputs depend on both the internal present state and on the inputs.

- Both types have a timing input (clock) that is not considered a controlling input.

General Models of Finite State Machines

- A Moore state machine consists of combinational logic that determines the sequence and memory (flip-flops), as shown in Figure 1(a). A Mealy state machine is shown in part (b).

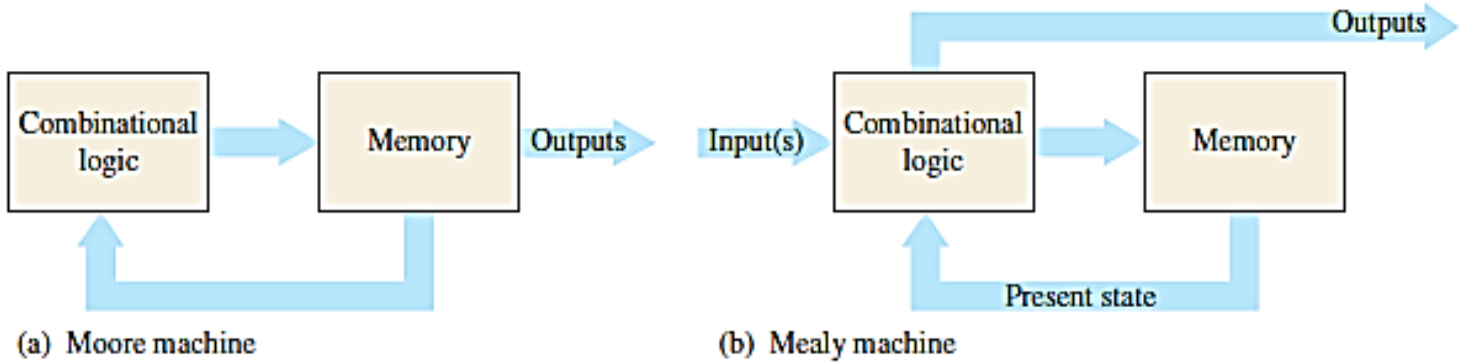


FIGURE 1 Two types of sequential logic.

- In the Moore machine, the combinational logic is a gate array with outputs that determine the next state of the flip-flops in the memory. There may or may not be inputs to the combinational logic. There may also be output combinational logic, such as a decoder. If there is an input(s), it does not affect the outputs because they always correspond to and are dependent only on the present state of the memory. For the Mealy machine, the present state affects the outputs, just as in the Moore machine; but in addition, the inputs also affect the outputs. The outputs come directly from the combinational logic and not the memory.

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.

❑ A 2-Bit Asynchronous Binary Counter

- Figure 2 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.

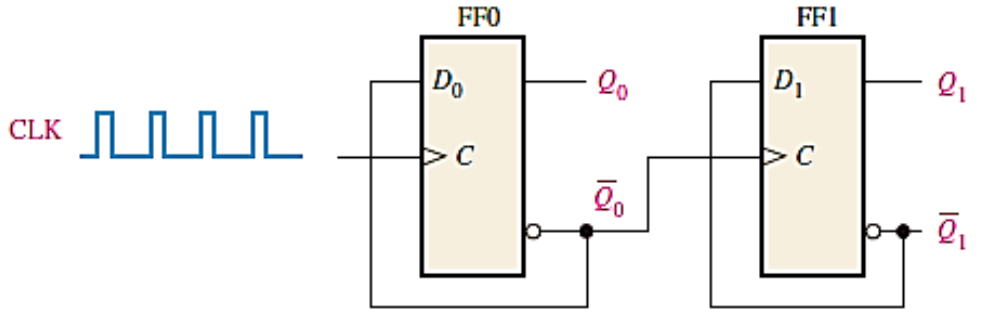


FIGURE 2 A 2-bit asynchronous binary counter.

- 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.

The Timing Diagram

- Let's examine the basic operation of the asynchronous counter of Figure 2 by applying four clock pulses to FF0 and observing the Q output of each flip-flop.
- Figure 3 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).
- The positive-going edge of CLK1 (clock pulse 1) causes the Q_0 output of FF0 to go HIGH, as shown in Figure 3.
- 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.

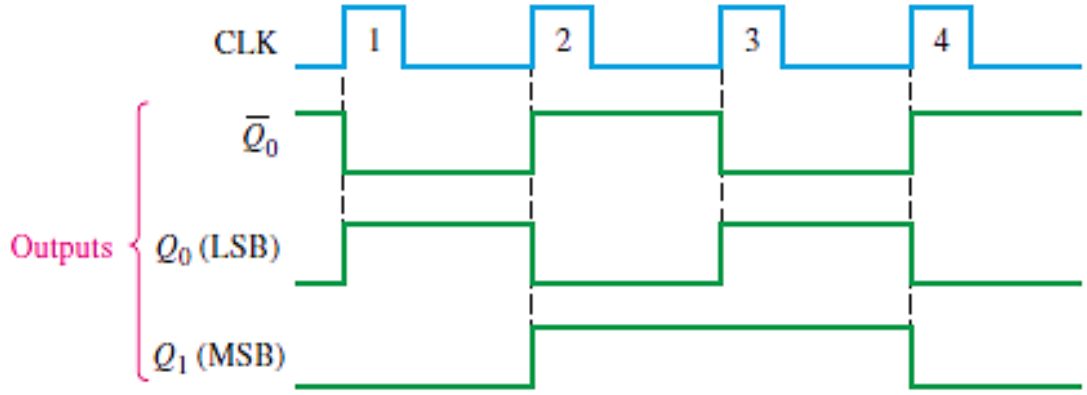


FIGURE 3 Timing diagram for the counter of Figure 2.

- 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 leadingedge of CLK4, $Q0 = 0$ and $Q1 = 0$. 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 3. 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 3 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 1.

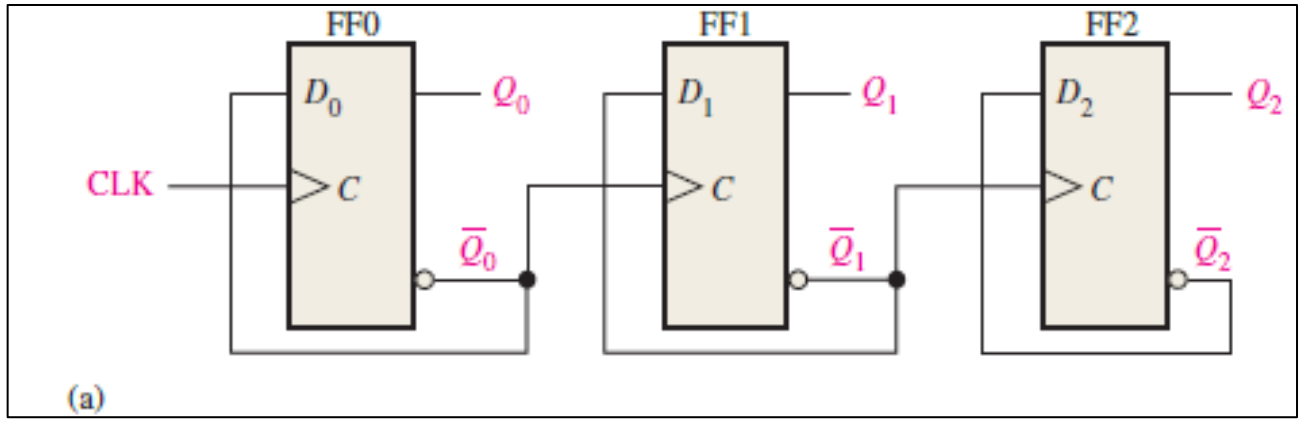
- Since it goes through a binary sequence, the counter in Figure 2 is a binary counter. It actually counts the number of clock pulses up to three, and on the fourth pulse it recycles to its original state ($Q_0 = 0$, $Q_1 = 0$). The term **recycle** is commonly applied to counter operation; it refers to the transition of the counter from its final state back to its original state.

Table 1 Binary state sequence for the counter in Figure 2.

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

A 3-Bit Asynchronous Binary Counter

- The state sequence for a 3-bit binary counter is listed in Table 2, and a 3-bit asynchronous binary counter is shown in Figure 4 (a). The basic operation is the same as that of the 2-bit counter except that the 3-bit counter has eight states, due to its three flip-flops.



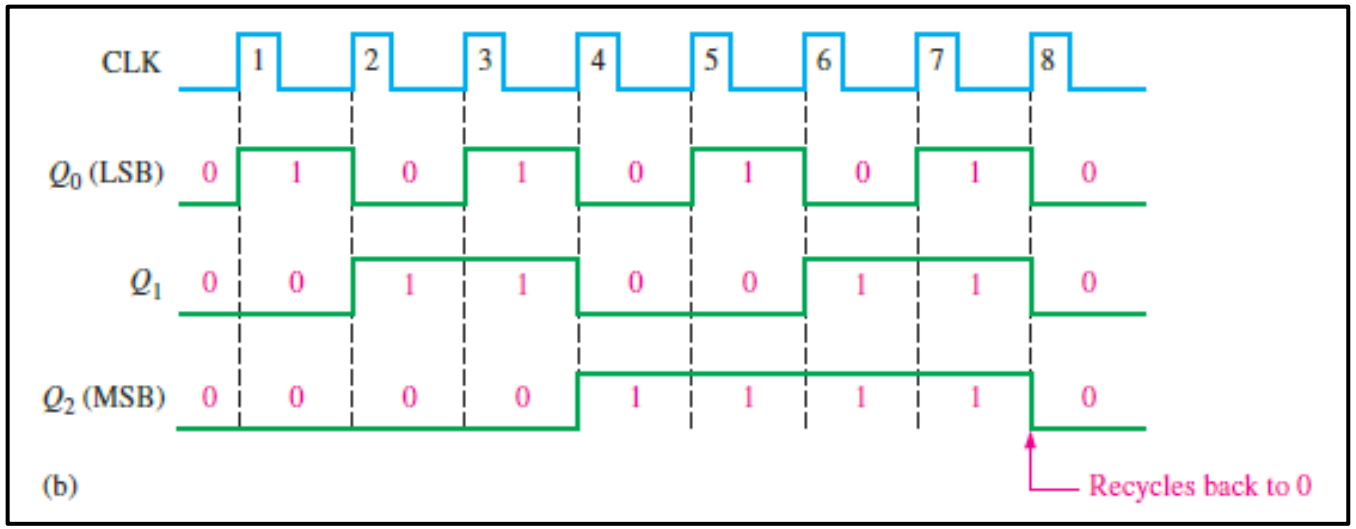


Table 2

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

FIGURE 4 Three-bit asynchronous binary counter and its timing diagram for one cycle.

- A timing diagram is shown in Figure 4 (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 4 change state on the leading edge of CLK4. This ripple clocking effect is shown in Figure 5 for the first four clock pulses, with the propagation delays indicated.

- The LOW-to-HIGH transition of $Q0$ 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.

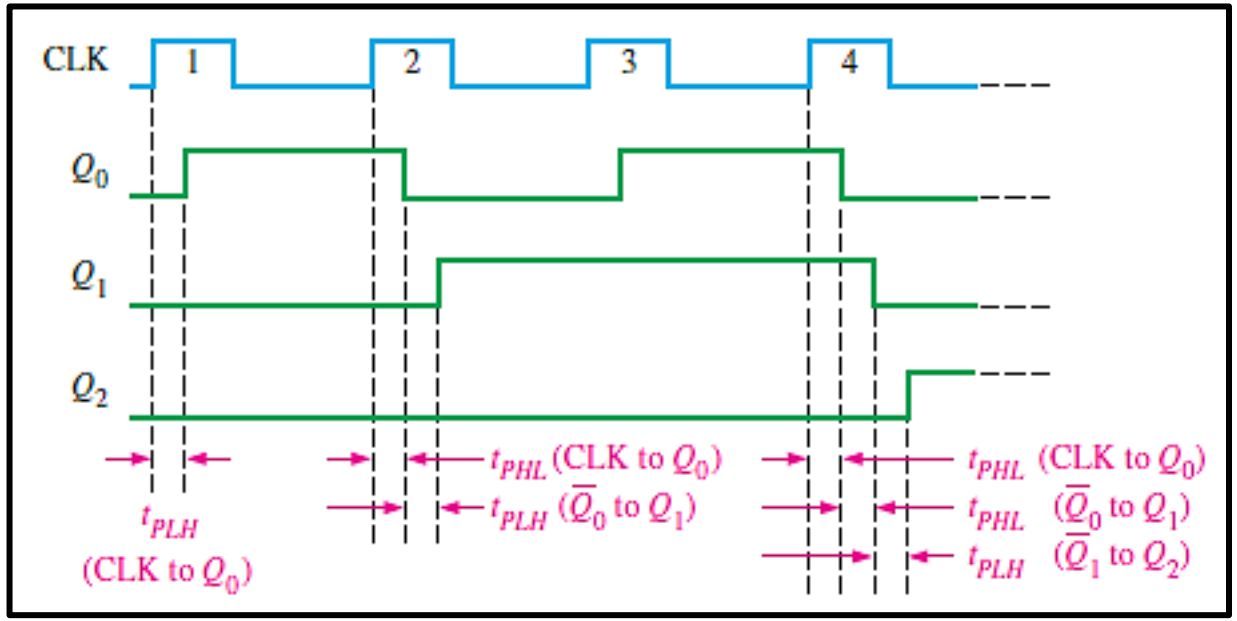


FIGURE 5 Propagation delays in a 3-bit asynchronous (ripple-clocked) binary counter.

EXAMPLE 1

A 4-bit asynchronous binary counter is shown in Figure 6(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.

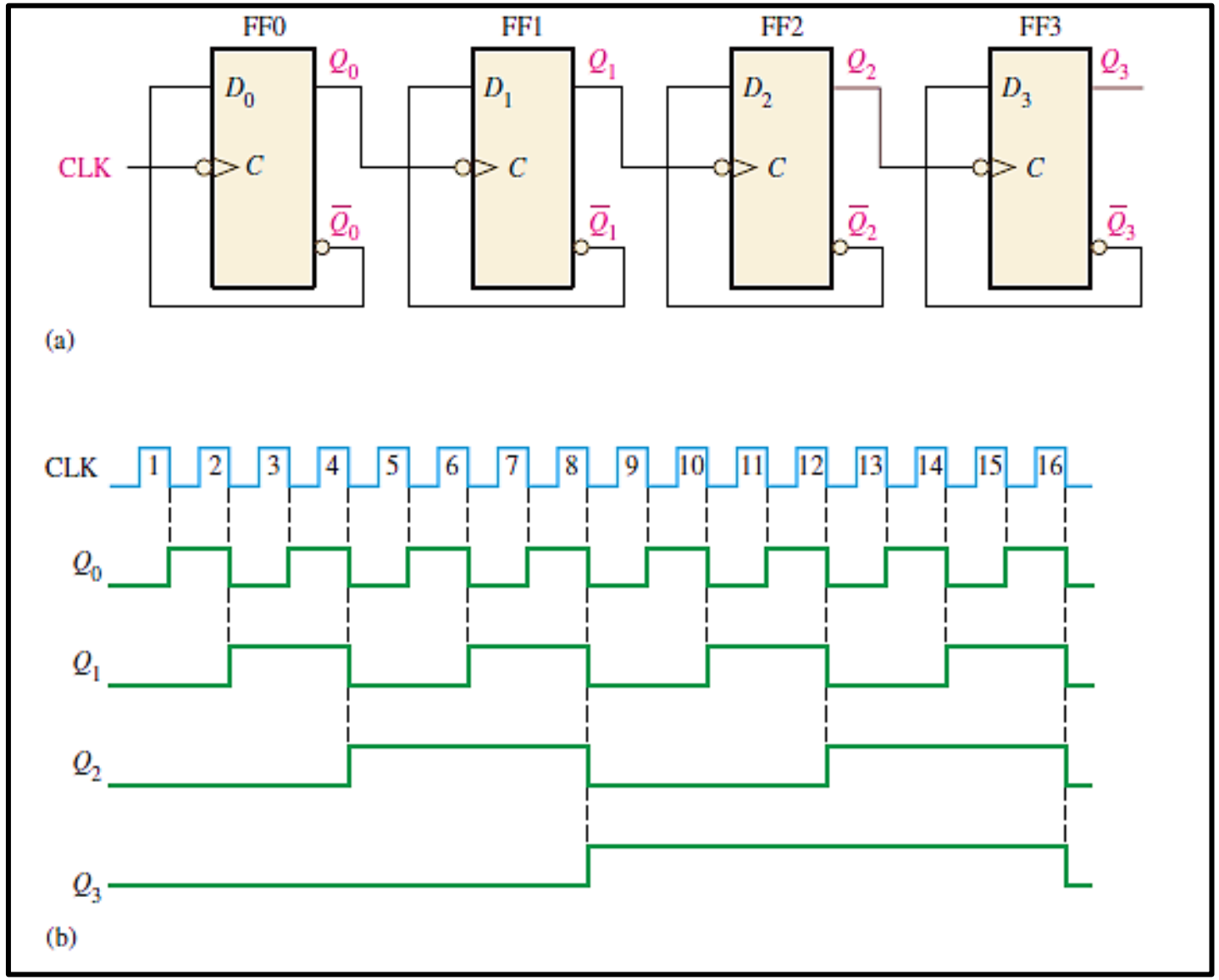


FIGURE 6 Four-bit asynchronous binary counter and its timing diagram.

Solution

The timing diagram with delays omitted is as shown in Figure 6 (b). For the total delay time, the effect of CLK8 or CLK16 must propagate through four flip-flops before
Q_3 changes, so

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

The maximum clock frequency is

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

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 6(a) are positive edge-triggered.

Asynchronous Decade Counters

- 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 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 7(a).

Partial Decoding

- Notice in Figure 7(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 7(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 2 shows.

EXAMPLE 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.

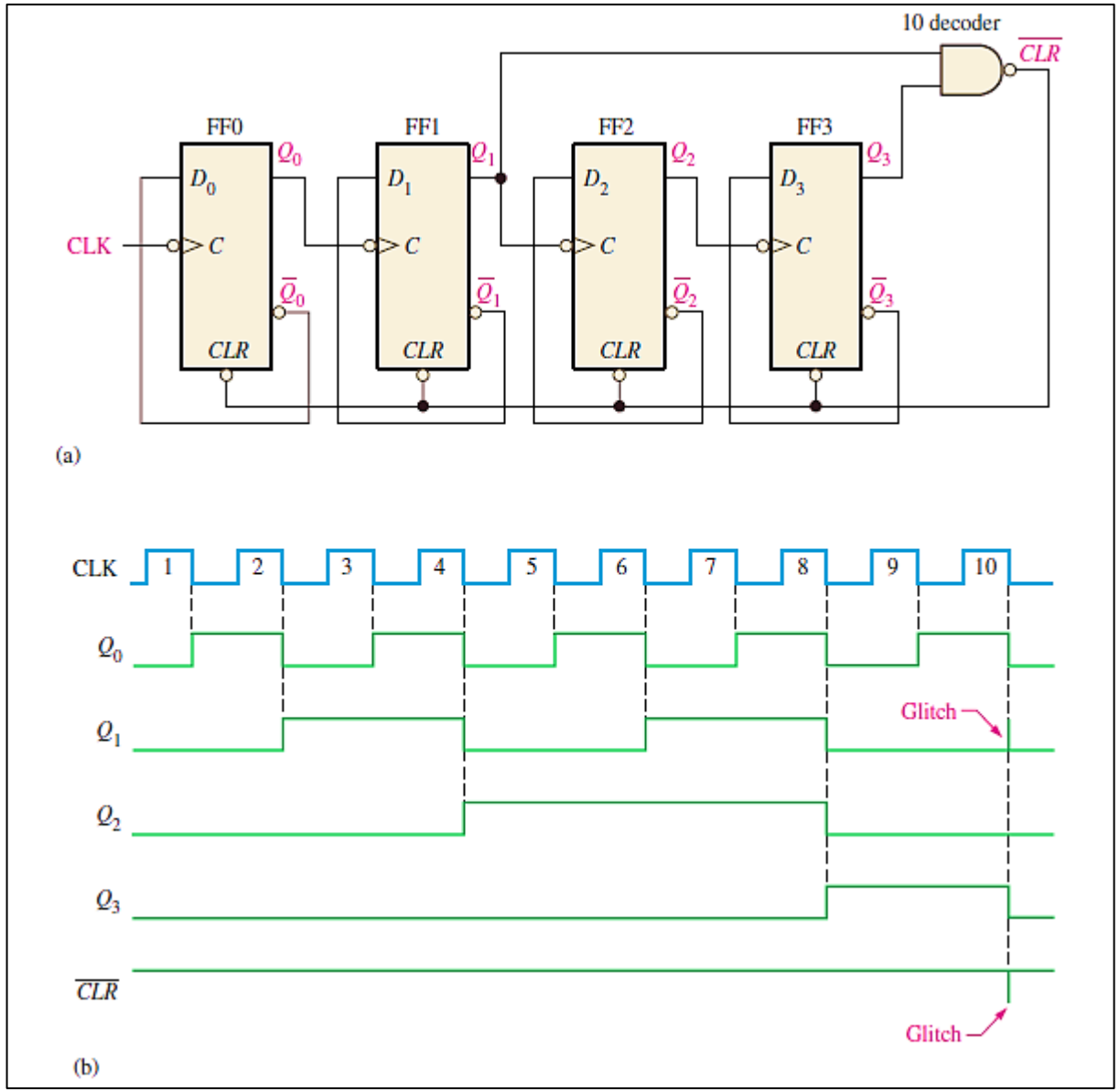
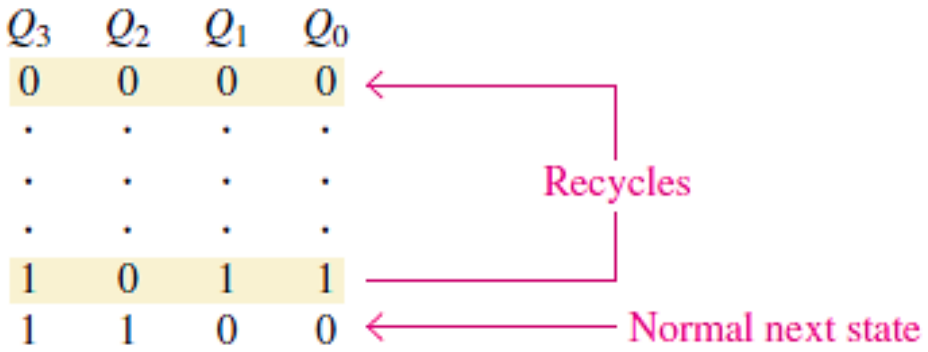


FIGURE 7 An asynchronously clocked decade counter with asynchronous recycling.

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:



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

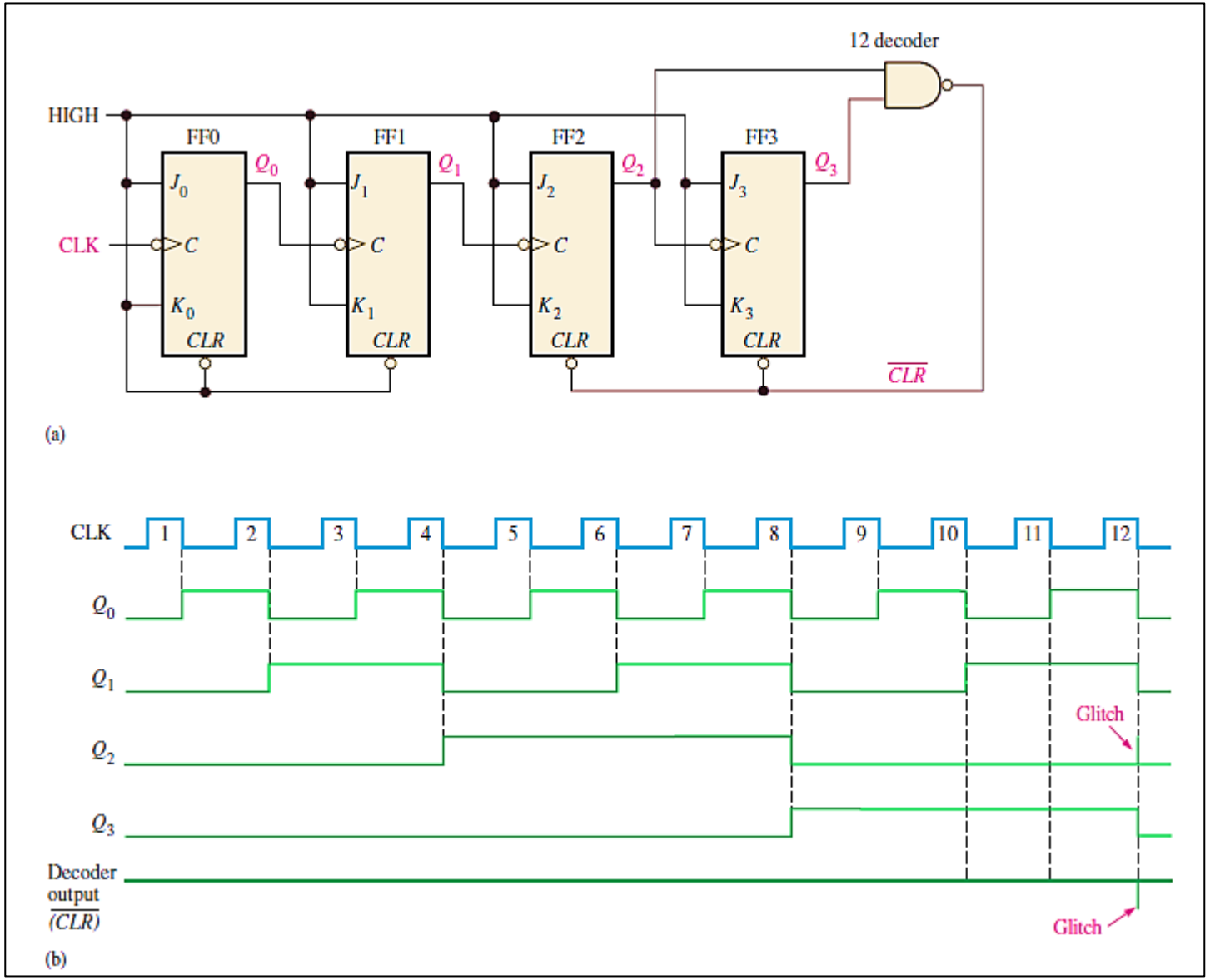


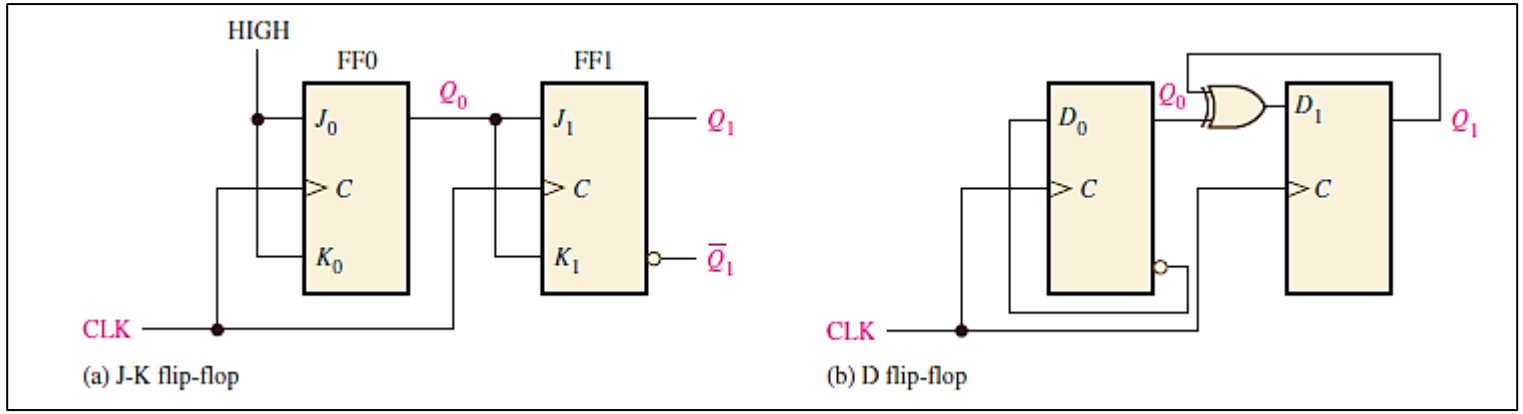
FIGURE 8 Asynchronously clocked modulus-12 counter with asynchronous recycling.

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.

A 2-Bit Synchronous Binary Counter

- Figure 9 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).



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

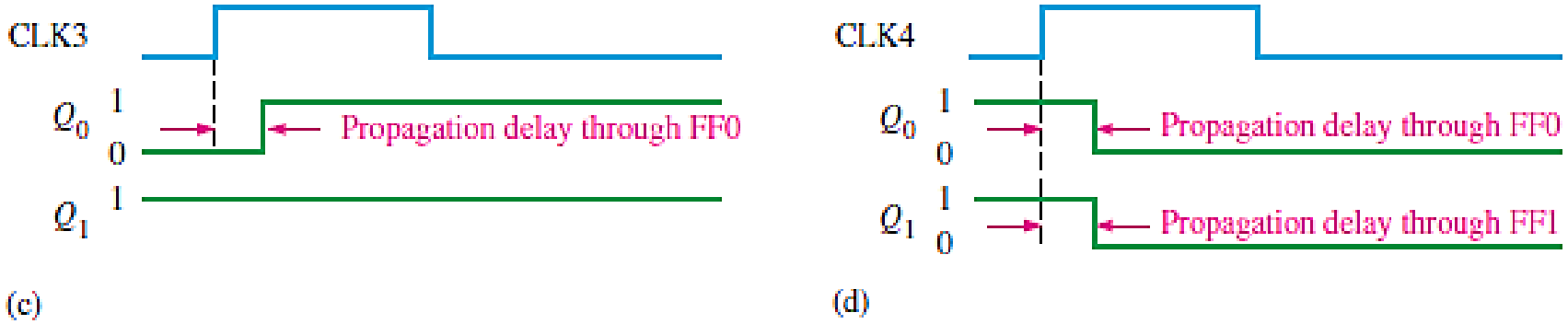
FIGURE 9 2-bit synchronous binary counters.

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 10 (a).

FIGURE 10 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 10 (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 10 (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 10 (d). The counter has now recycled to its original state, binary 0.

Examination of the D flip-flop counter in Figure 9(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 is shown in Figure 11. 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.

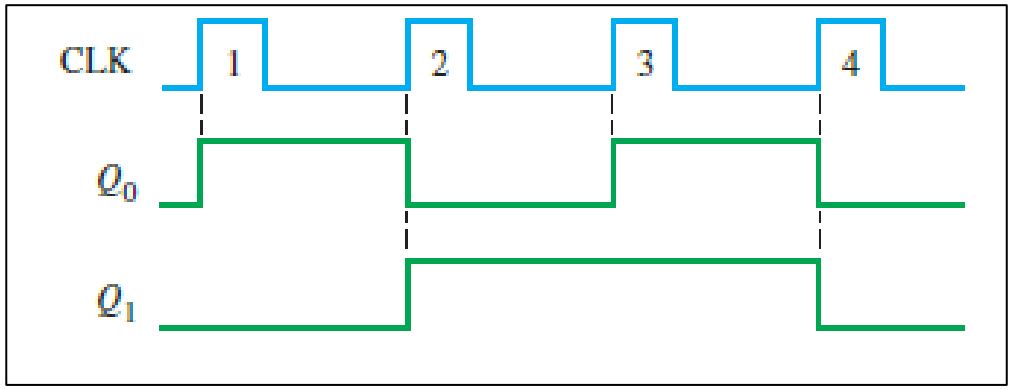


FIGURE 11 Timing diagram for the counters of Figure 9.

A 3-Bit Synchronous Binary Counter

A 3-bit synchronous binary counter is shown in Figure 12, and its timing diagram is shown in Figure 13. You can understand this counter operation by examining its sequence of states as shown in Table 3.

Table 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

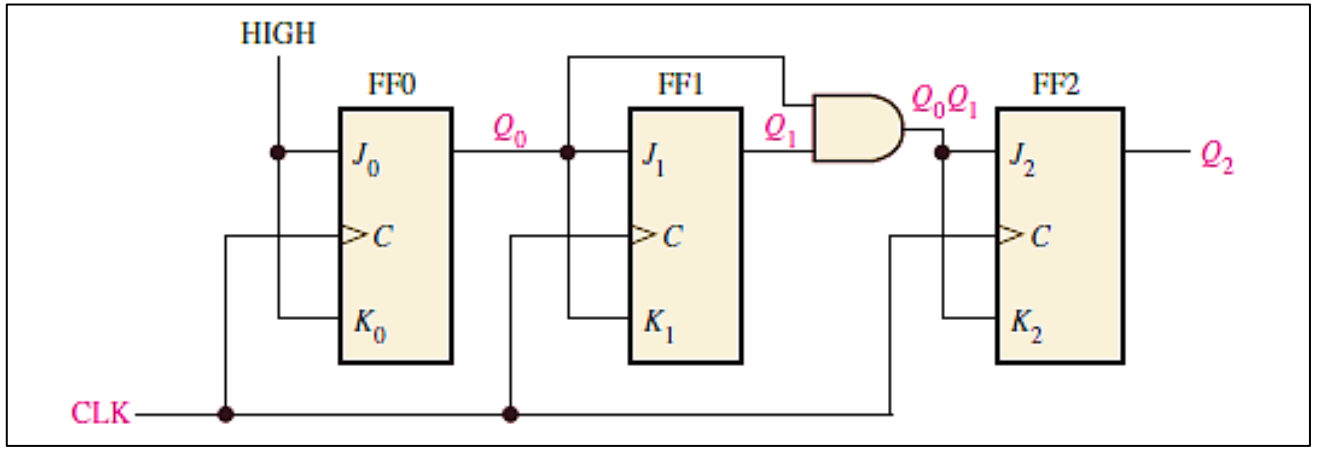


FIGURE 12 A 3-bit synchronous binary counter.

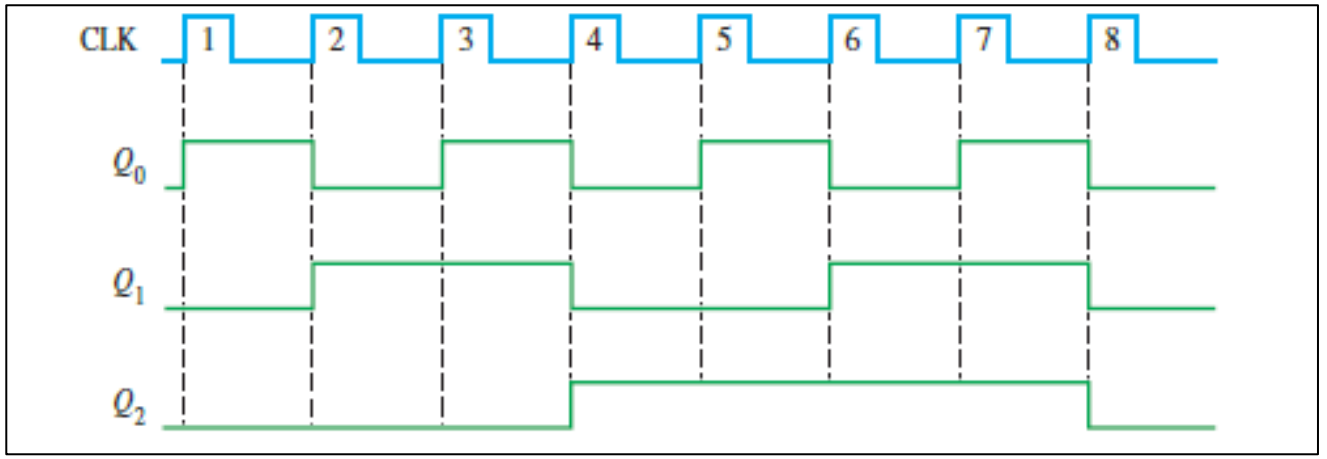


FIGURE 13 Timing diagram for the counter of Figure 12.

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 12 is summarized in Table 4.

Table 4

Summary of the analysis of the counter in Figure 12.

Clock Pulse	Outputs			*J-K* Inputs						At the Next 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 14 (a) shows a 4-bit synchronous binary counter, and Figure 14 (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.

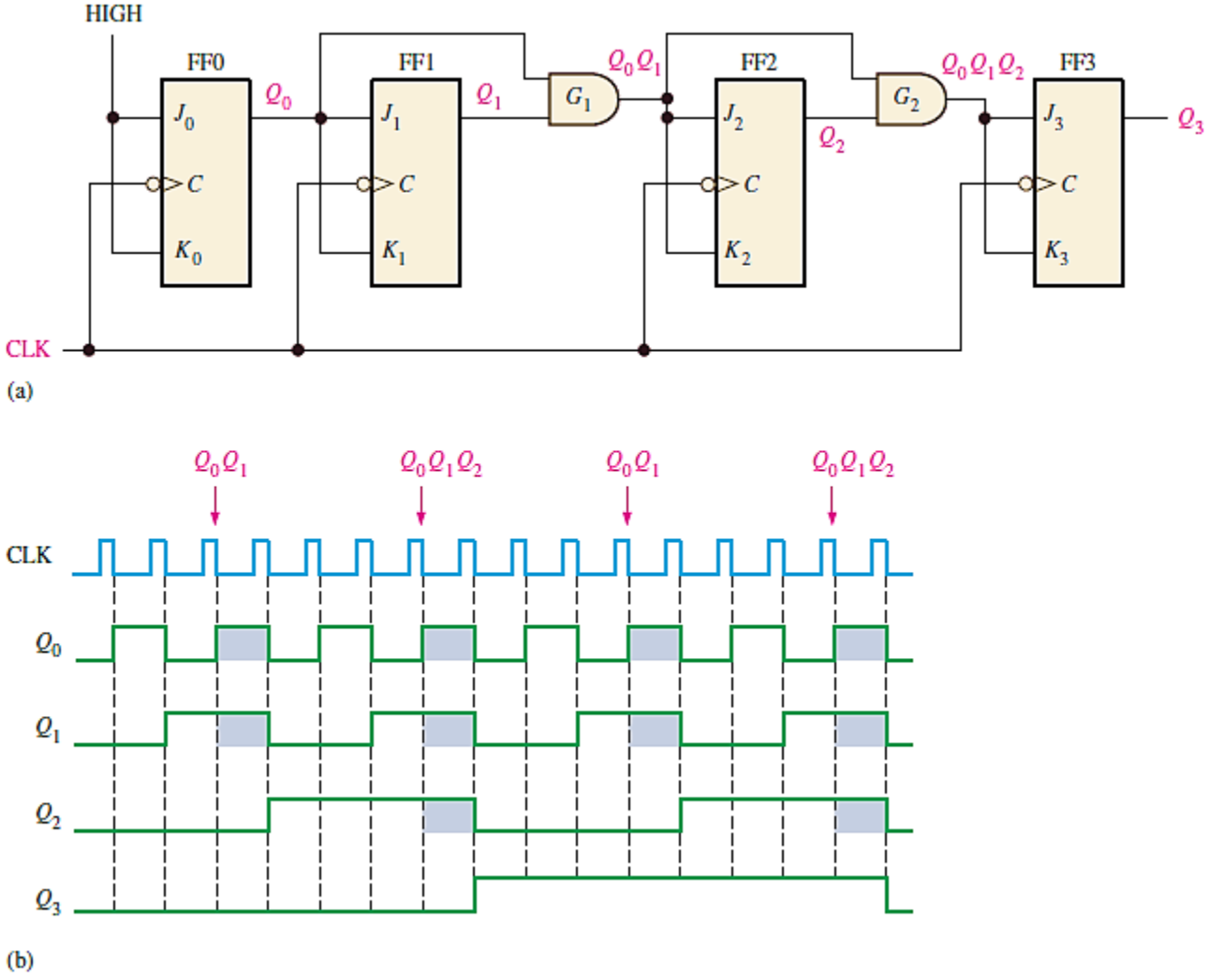


FIGURE 14 A 4-bit synchronous binary counter and timing diagram. Times where the AND gate outputs are HIGH are indicated by the shaded areas.

A 4-Bit Synchronous Decade Counter

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 15. The timing diagram for the decade counter is shown in Figure 16.

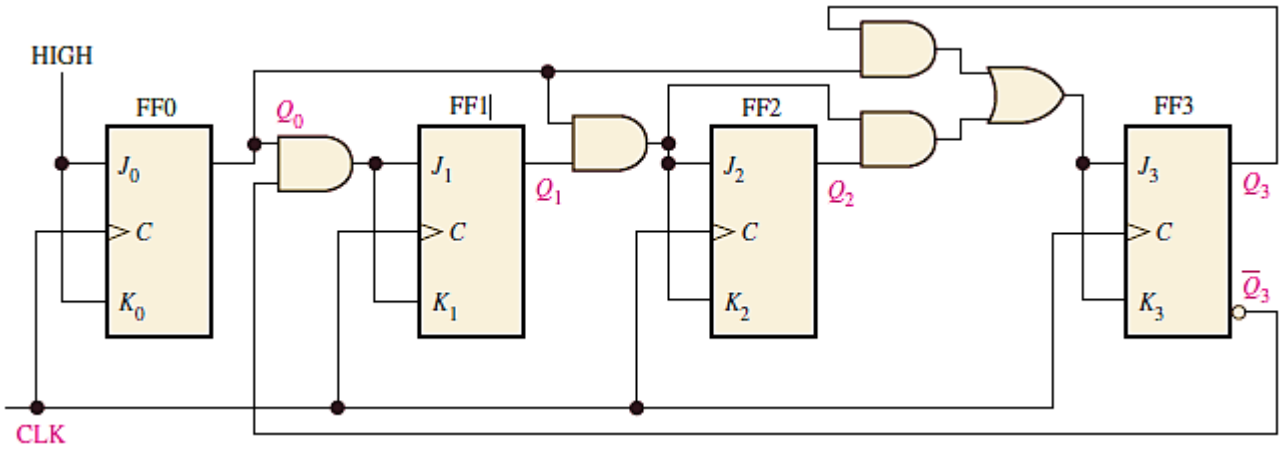


FIGURE 15 A synchronous BCD decade counter.

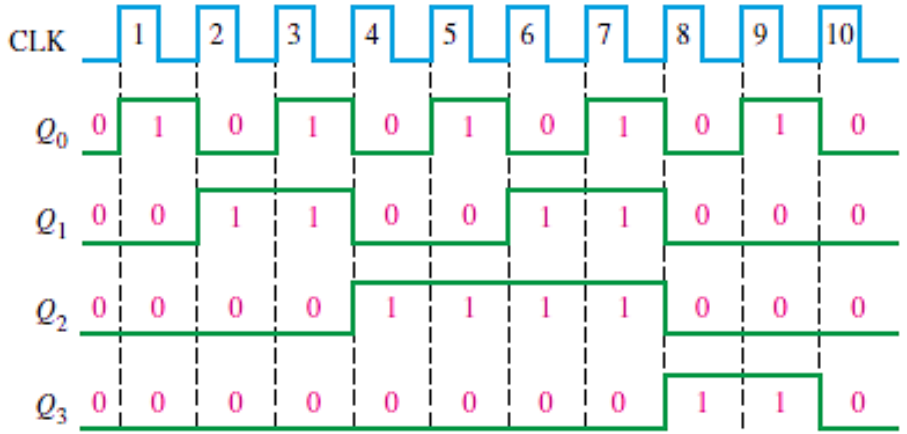


FIGURE Timing diagram for the BCD decade counter (Q_0 is the LSB).

The counter operation is shown by the sequence of states in Table 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.

Next, notice in Table 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 Q_3 and connecting the gate output to the J_1 and K_1 inputs of FF1

Table 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

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:

$$J_2 = K_2 = Q_0 Q_1$$

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.

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.

$$J_3 = K_3 = Q_0 Q_1 Q_2 + Q_0 Q_3$$

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 15. Notice that the differences between this decade counter and the modulus-16 binary counter in Figure14(a) are the $Q_0 \overline{Q_3}$ AND gate, the $Q_0 Q_3$ 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