# CSE231 – Digital Logic Design

# Lecture – 8

v

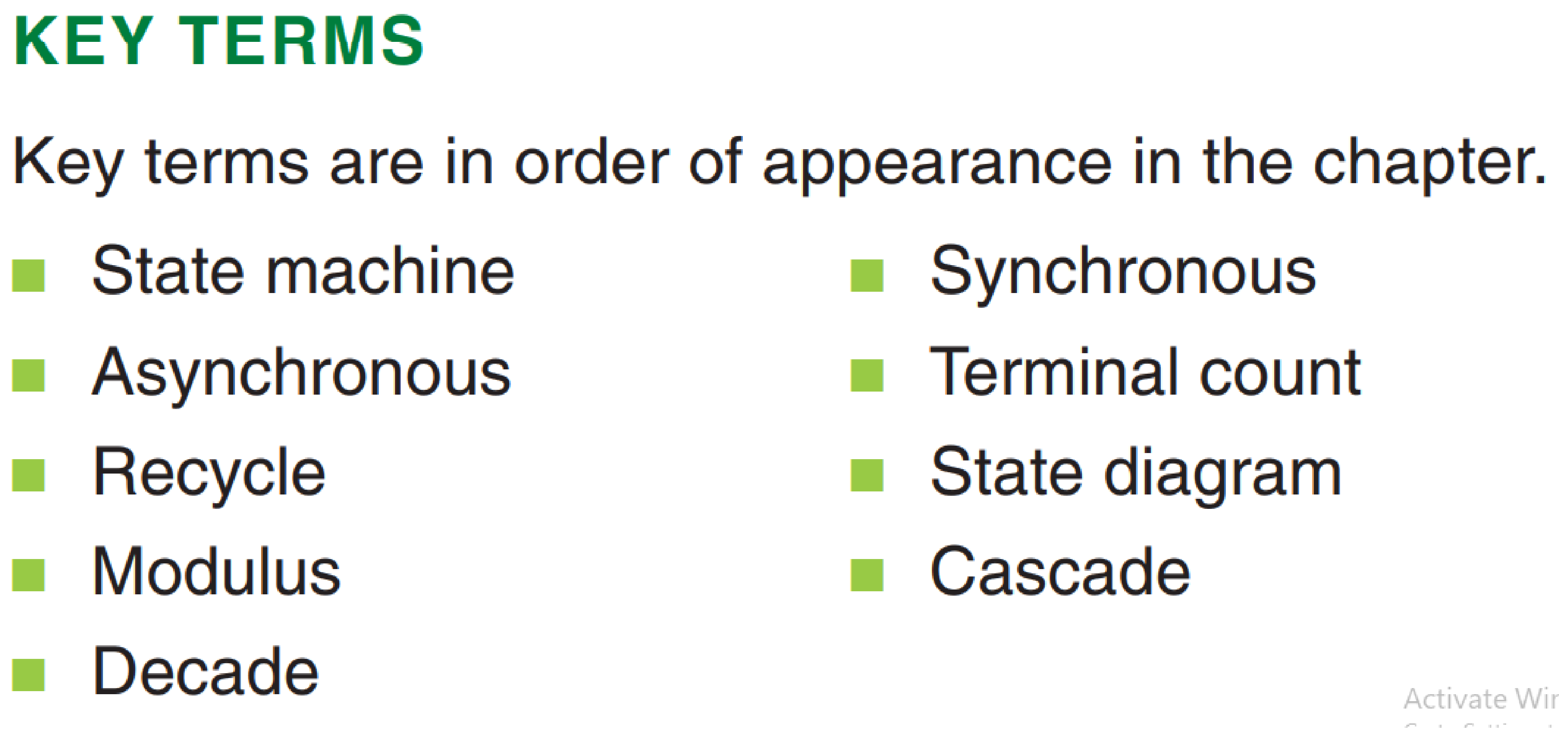
# Counters

## *Lesson Outcomes*

After completing this lecture, students will be able to

- *Describe the difference between an asynchronous and a synchronous counter*
- *Analyze counter timing diagrams & counter circuits*
- *Explain how propagation delays affect the operation of a counter*
- *Determine & modify the modulus of a counter*
- *Use an up/down counter to generate forward and reverse binary sequences*
- *Use cascaded counters to achieve a higher modulus*
- *Use logic gates to decode any given state of a counter*
- *Apply counters for various applications : digital clock, parking controller, elevator controller etc.*

## Key Terms



### KEY TERMS

Key terms are in order of appearance in the chapter.

- State machine
- Asynchronous
- Recycle
- Modulus
- Decade
- Synchronous
- Terminal count
- State diagram
- Cascade

# 2-bit asynchronous binary counter

- 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.

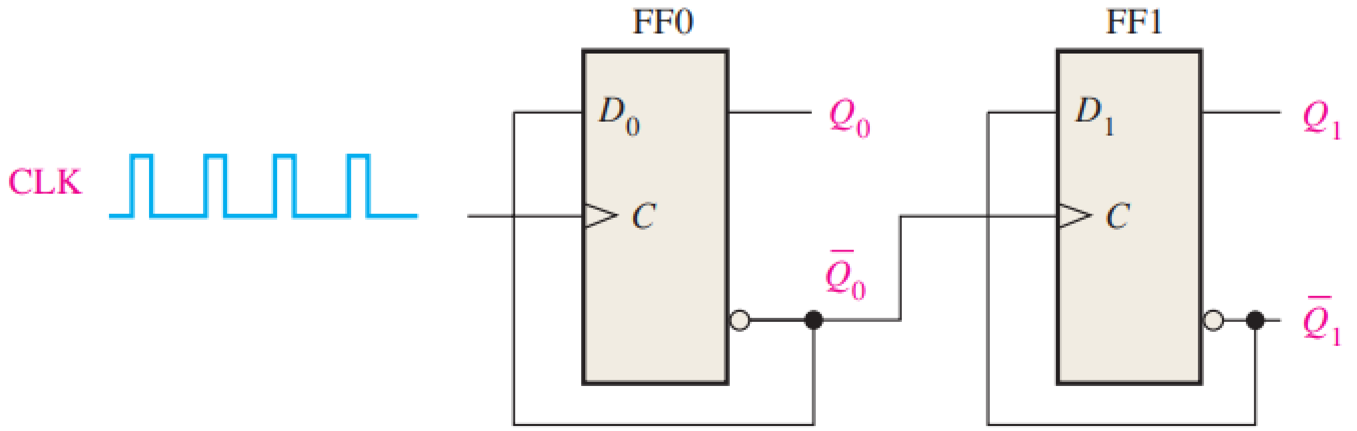
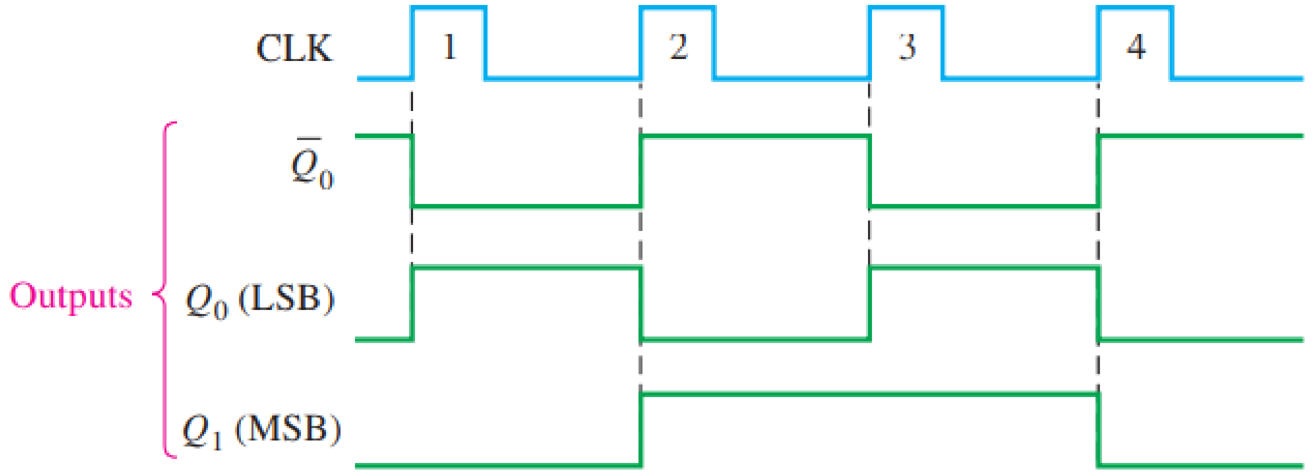


**TABLE 9–1**

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 |



# 3-bit / mod 8 asynchronous / ripple counter

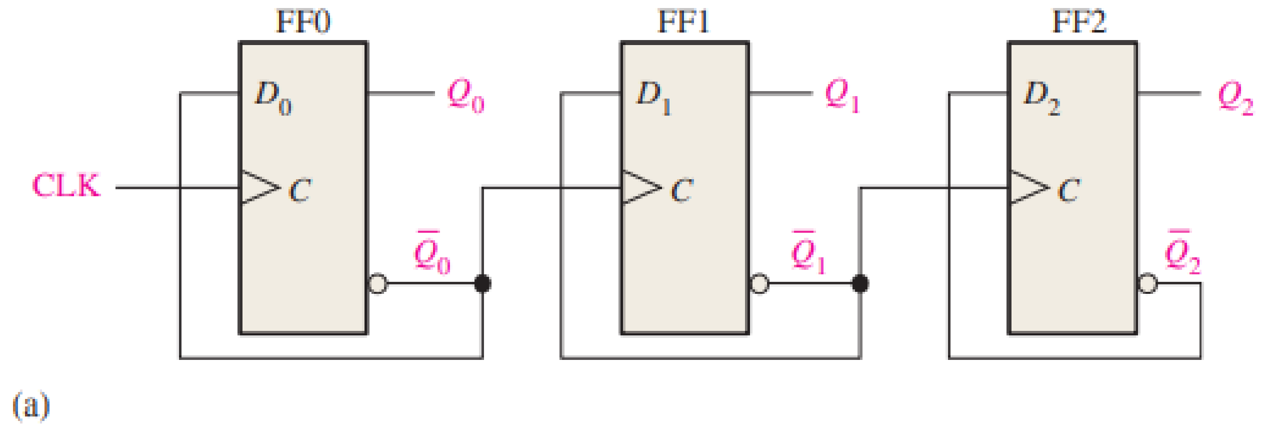
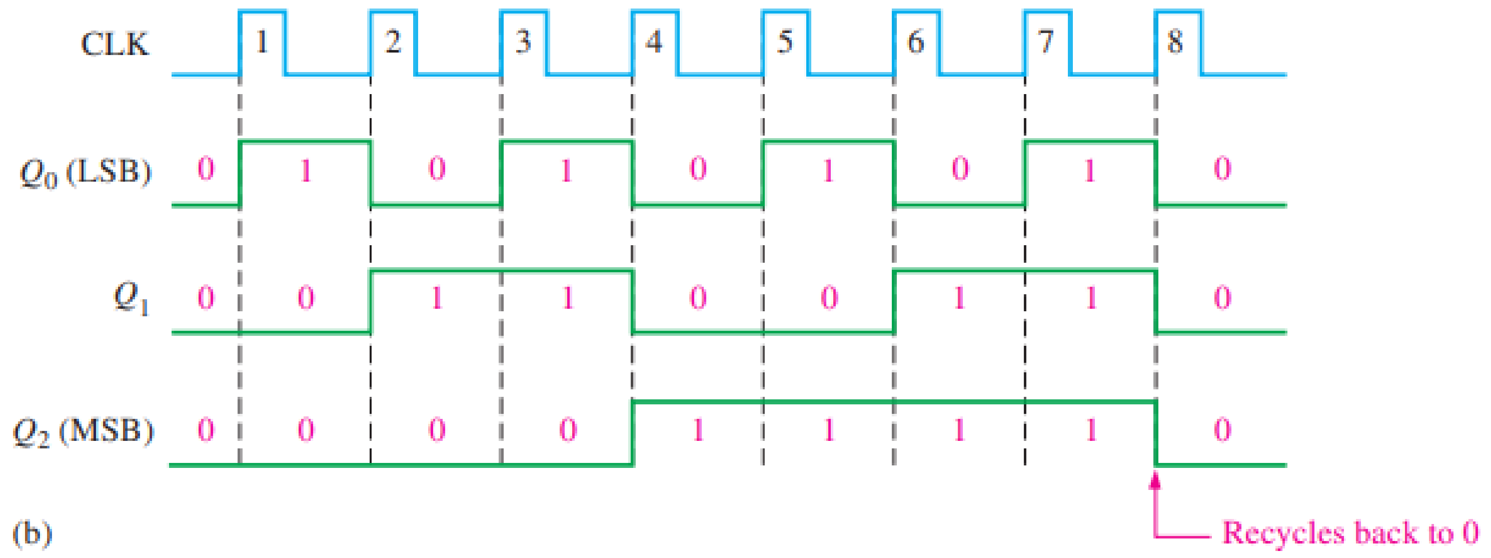


**TABLE 9–2**

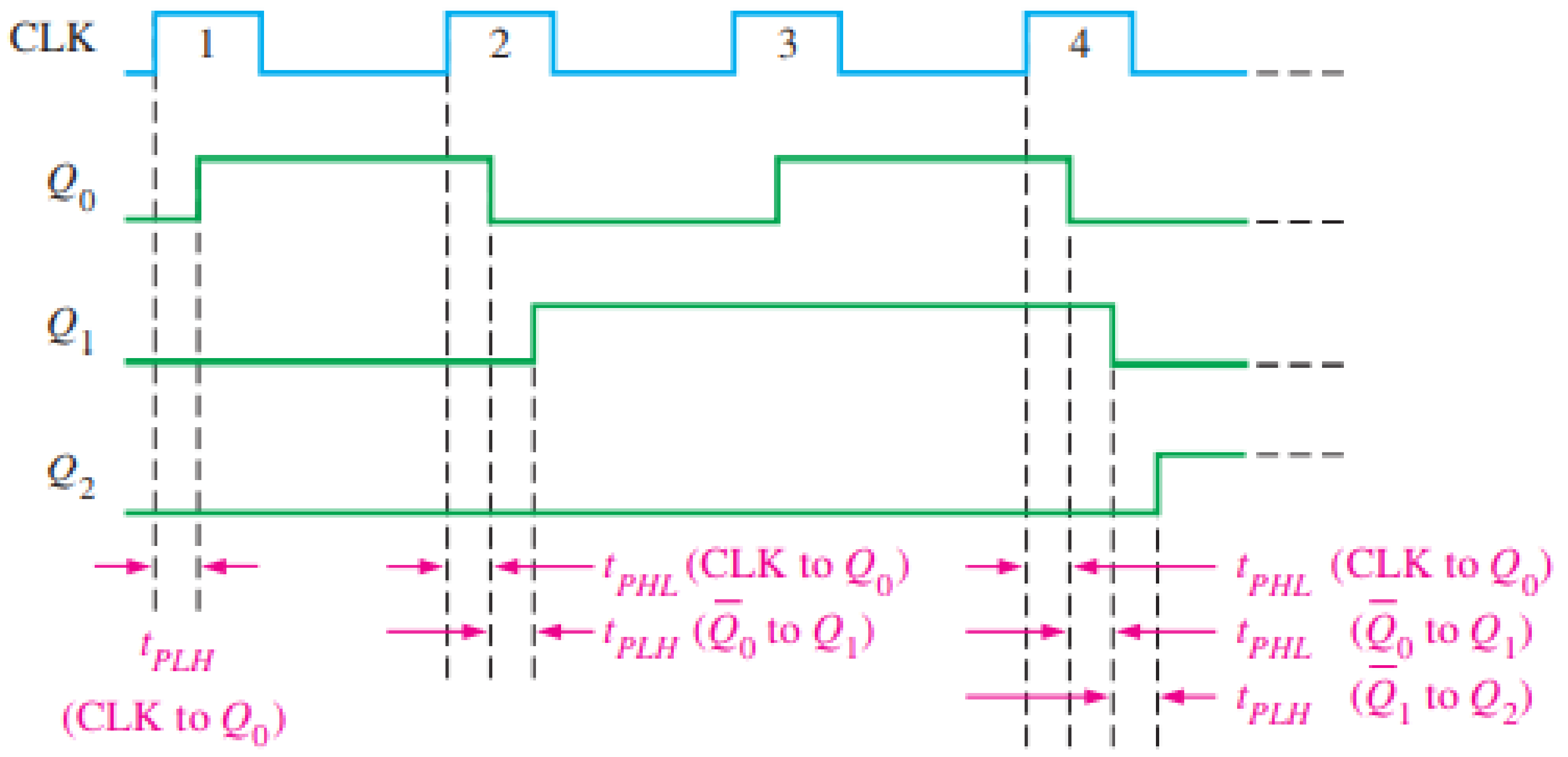
State sequence for a 3-bit binary counter.

| 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 |

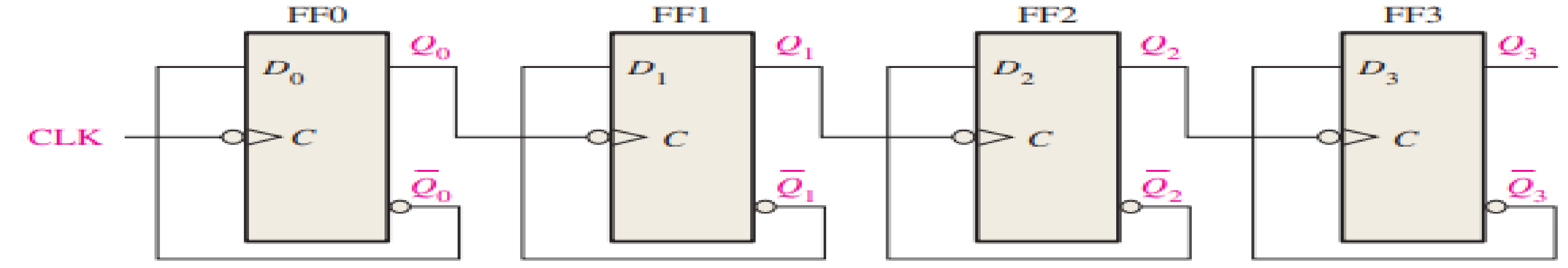


# Propagation delay: ripple counter

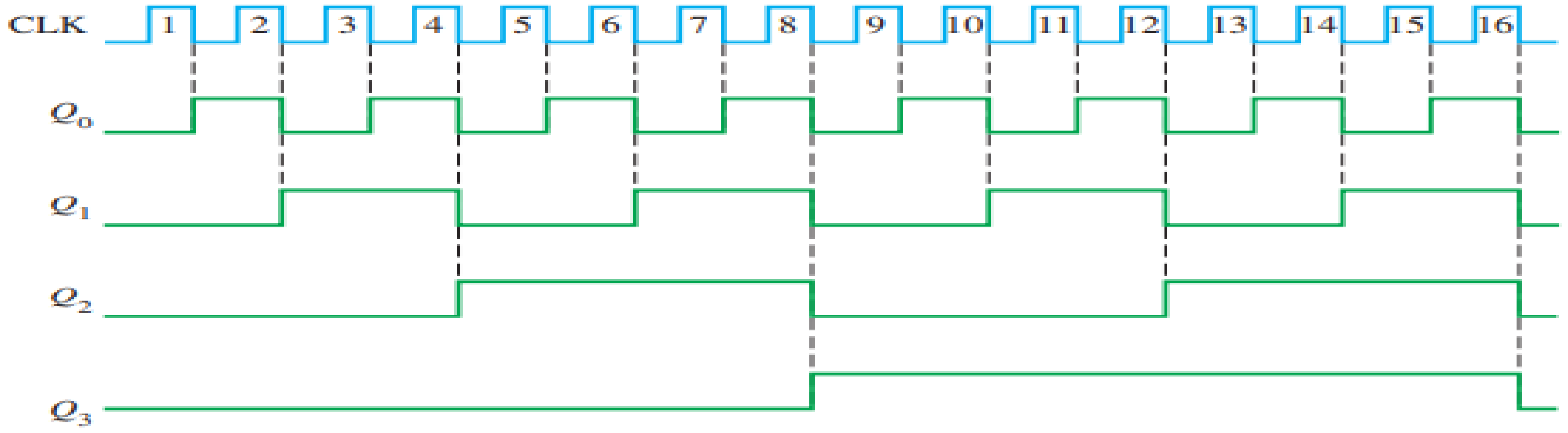
- *Asynchronous counters* are commonly referred to as *ripple counters* for the following reason: The effect of the input clock pulse is first "felt" by FF0. This effect cannot get to FF1 immediately because of the propagation delay through FF0. Then there is the propagation delay through FF1 before 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.


CLK
1
2
3
4
$Q_0$
$Q_1$
$Q_2$
$t_{PLH}$ (CLK to $Q_0$)
$t_{PHL}$ (CLK to $Q_0$)
$t_{PLH}$ ($\bar{Q}_0$ to $Q_1$)
$t_{PHL}$ (CLK to $Q_0$)
$t_{PHL}$ ($\bar{Q}_0$ to $Q_1$)
$t_{PLH}$ ($\bar{Q}_1$ to $Q_2$)

# 4-bit / mod16 asynchronous / ripple counter



(a)



(b)

## 3-bit asynchronous / ripple counter

**PROBE:** A 4-bit asynchronous binary counter is shown in Figure (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.

**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

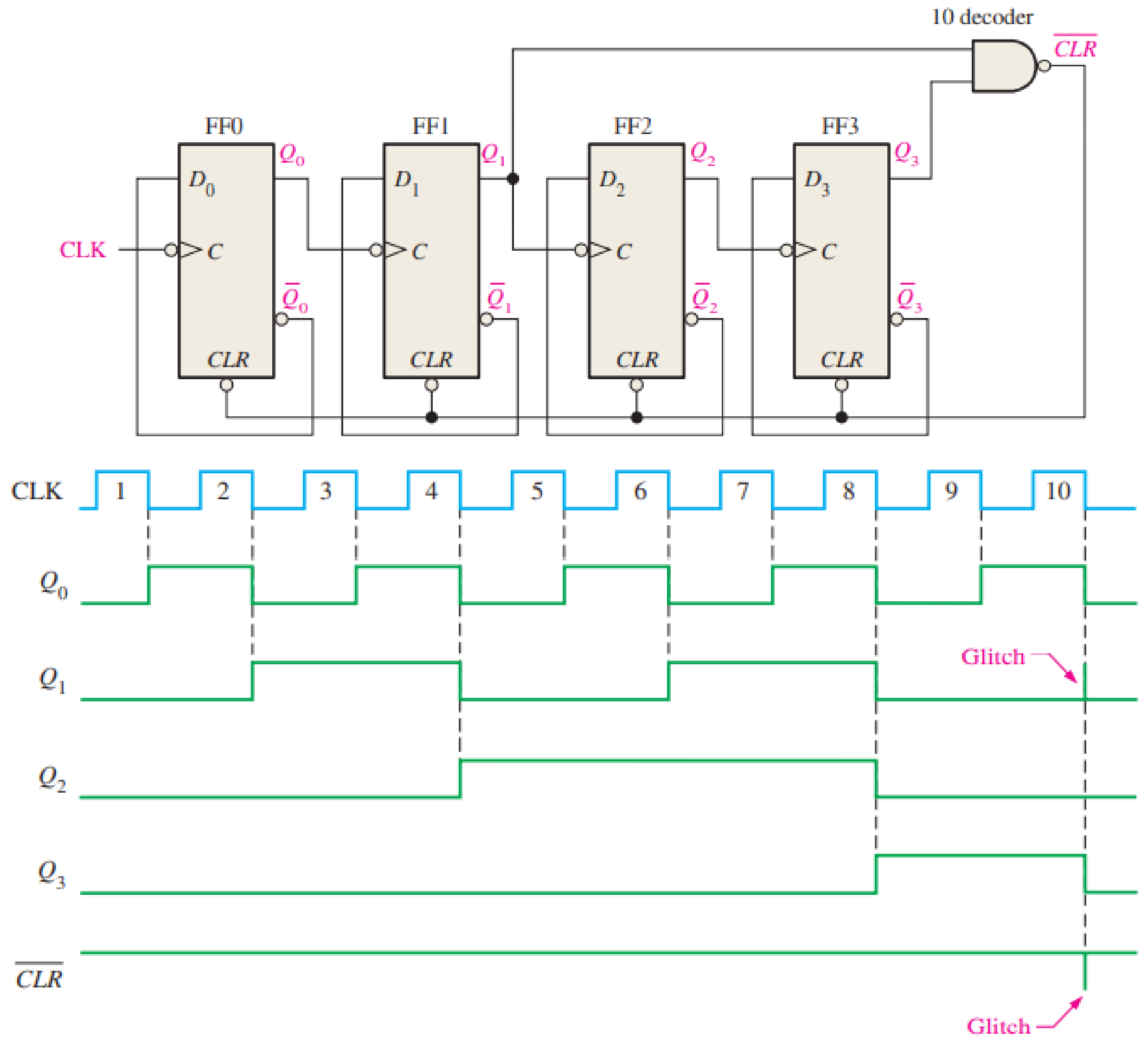
$$t_{p(tot)} = 4 \times 10\text{ ns} = \mathbf{40\text{ ns}}$$

The maximum clock frequency is

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

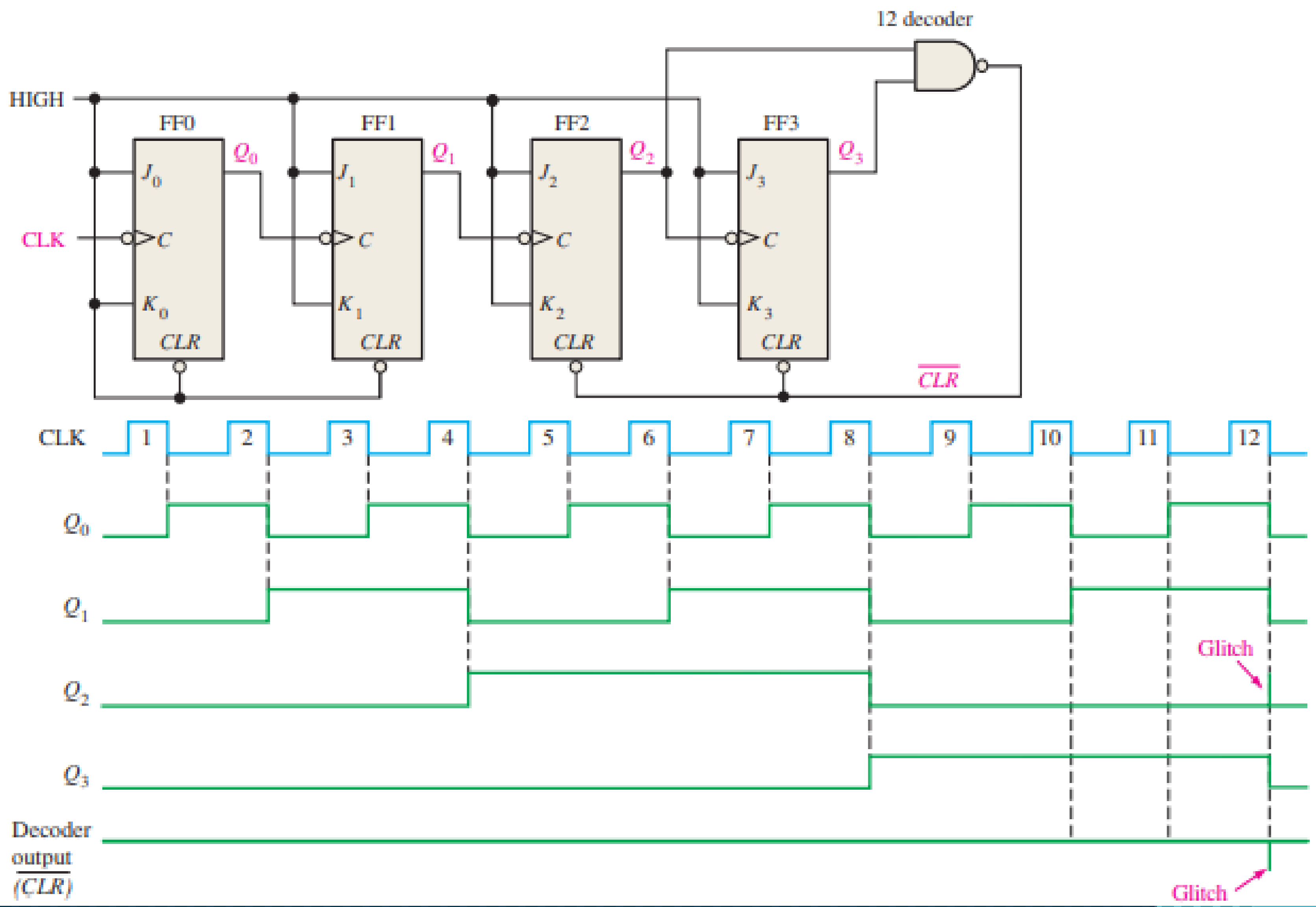
The counter should be operated below this frequency to avoid problems due to the propagation delay.

# Asynchronous decade / MOD10 / BCD counter



| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | Decimal | Comments |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 1 | |
| 0 | 0 | 1 | 0 | 2 | |
| 0 | 0 | 1 | 1 | 3 | |
| 0 | 1 | 0 | 0 | 4 | |
| 0 | 1 | 0 | 1 | 5 | |
| 0 | 1 | 1 | 0 | 6 | |
| 0 | 1 | 1 | 1 | 7 | |
| 1 | 0 | 0 | 0 | 8 | |
| 1 | 0 | 0 | 1 | 9 | |
| 1 | 0 | 1 | 0 | 0 | Reset |

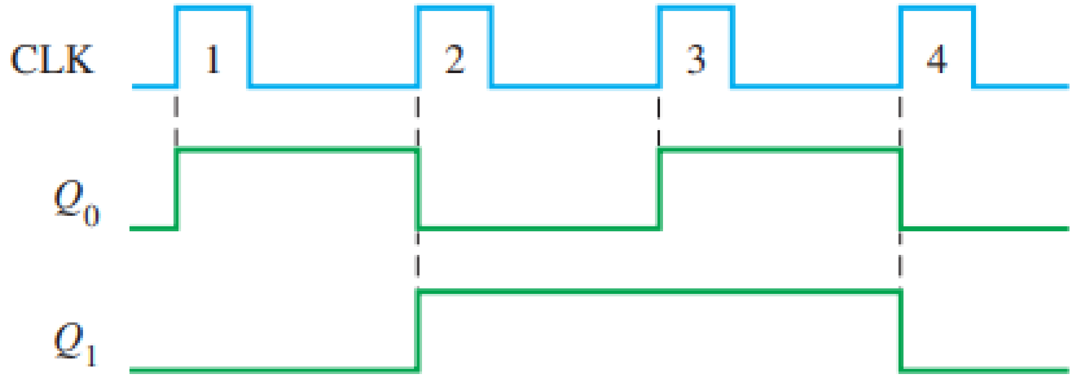
# Asynchronous MOD12 counter

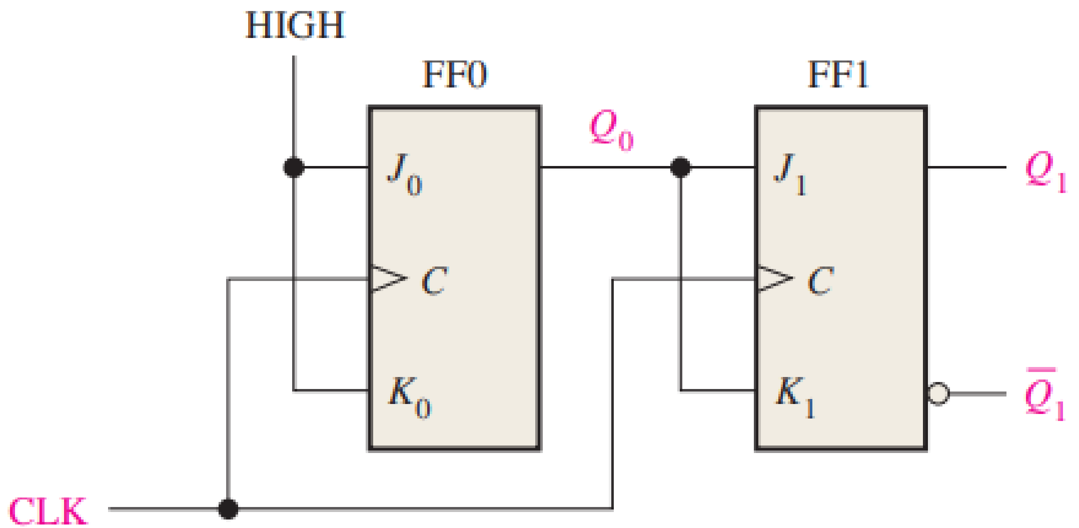


| $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ | Decimal | Comments |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 1 | |
| 0 | 0 | 1 | 0 | 2 | |
| 0 | 0 | 1 | 1 | 3 | |
| 0 | 1 | 0 | 0 | 4 | |
| 0 | 1 | 0 | 1 | 5 | |
| 0 | 1 | 1 | 0 | 6 | |
| 0 | 1 | 1 | 1 | 7 | |
| 1 | 0 | 0 | 0 | 8 | |
| 1 | 0 | 0 | 1 | 9 | |
| 1 | 0 | 1 | 0 | 10 | |
| 1 | 0 | 1 | 1 | 11 | |
| 1 | 1 | 0 | 0 | 0 | Reset |

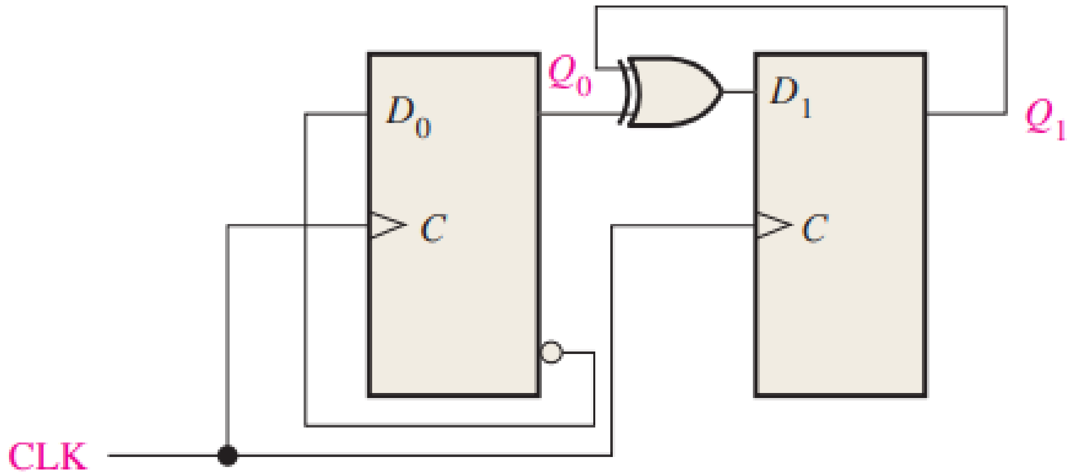
# 2-bit synchronous counter

- 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.



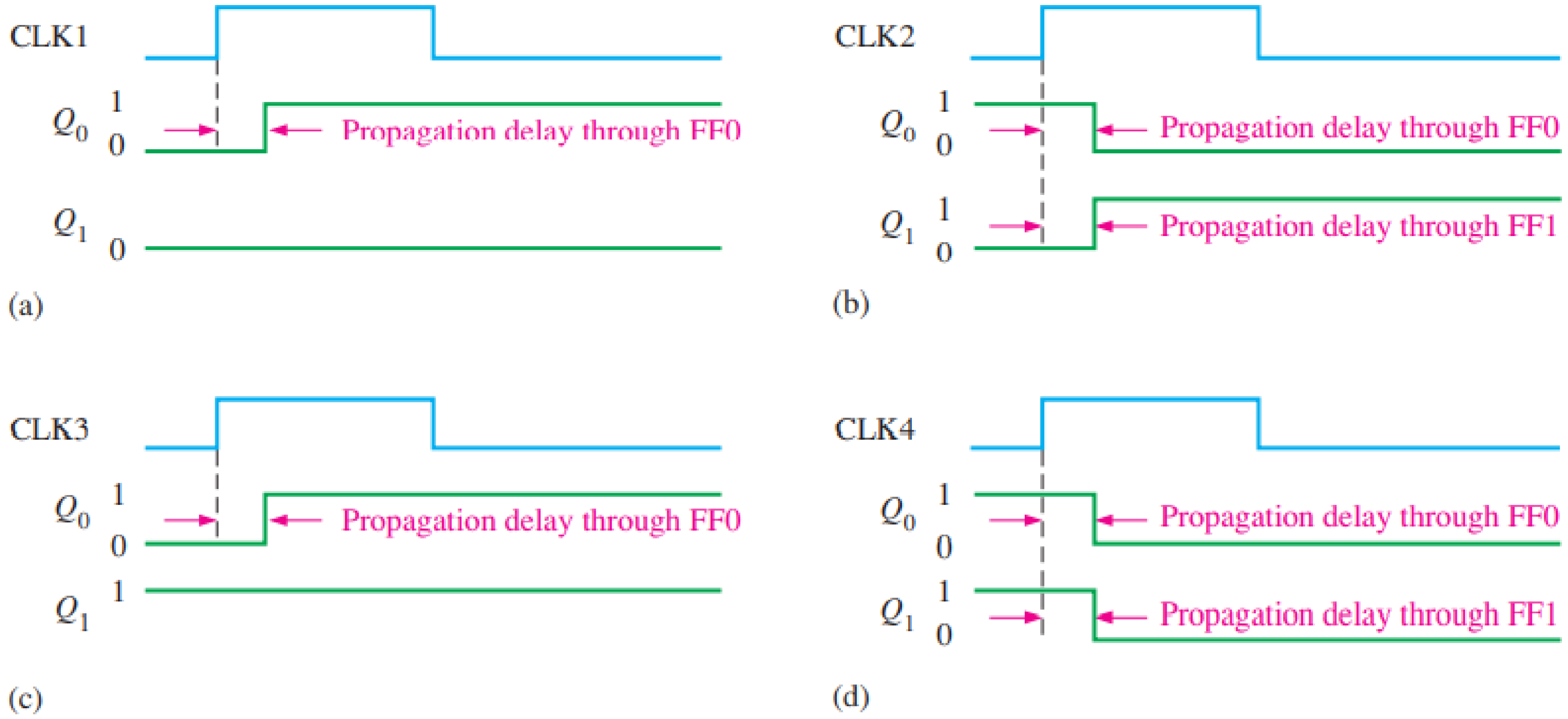


(a) J-K flip-flop

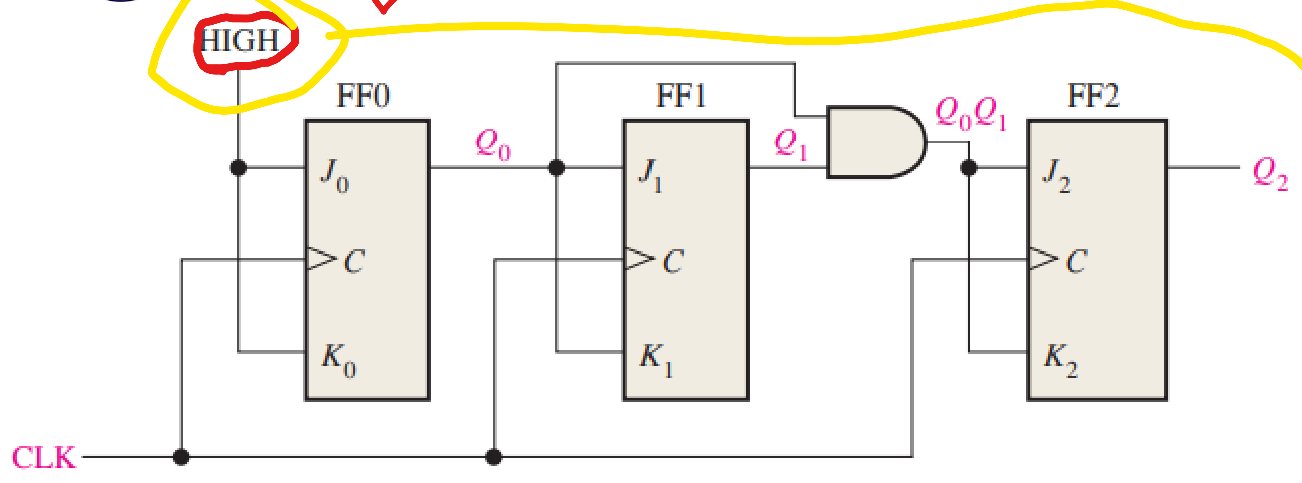


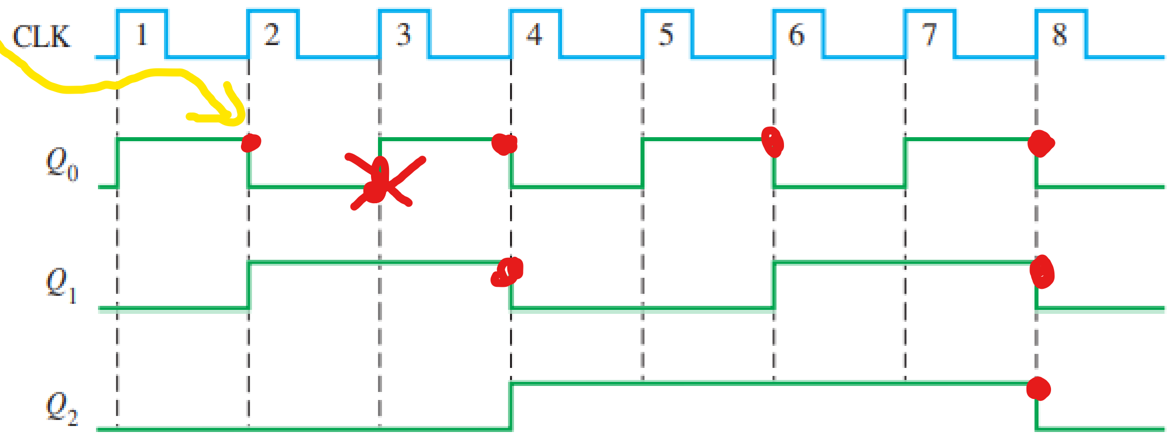
(b) D flip-flop

## 2-bit synchronous counter: propagation delay


CLK1
$Q_0$ 1 0
Propagation delay through FF0
$Q_1$ 0
(a)
CLK2
$Q_0$ 1 0
Propagation delay through FF0
$Q_1$ 1 0
Propagation delay through FF1
(b)
CLK3
$Q_0$ 1 0
Propagation delay through FF0
$Q_1$ 1
(c)
CLK4
$Q_0$ 1 0
Propagation delay through FF0
$Q_1$ 1 0
Propagation delay through FF1
(d)

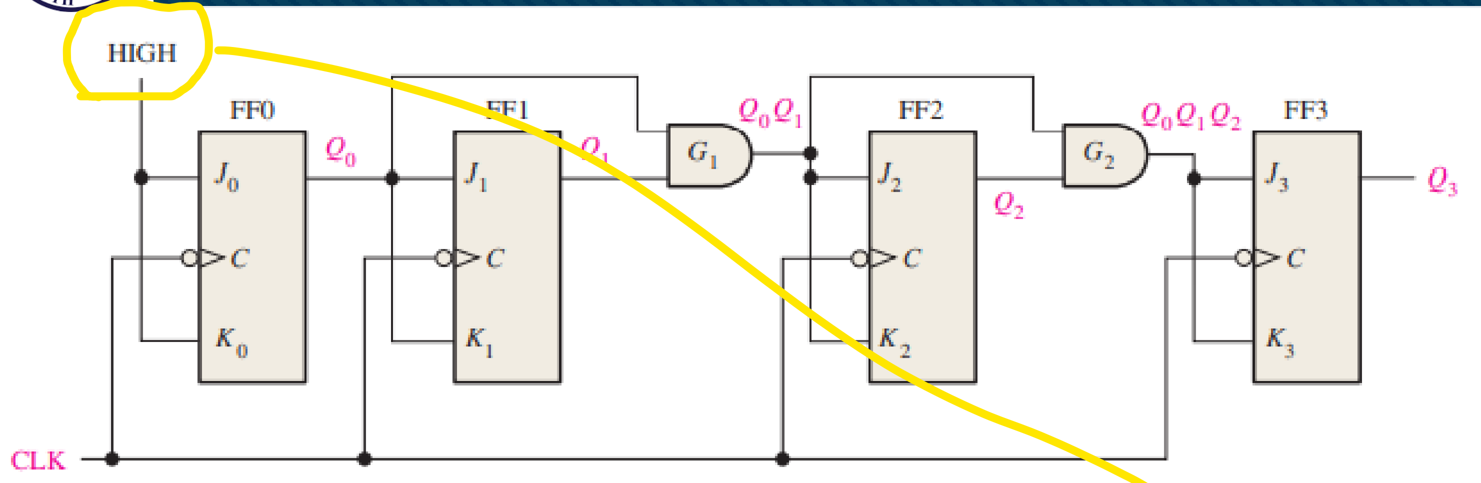
# 3-bit synchronous counter: divide by 8 / mod 8 counter





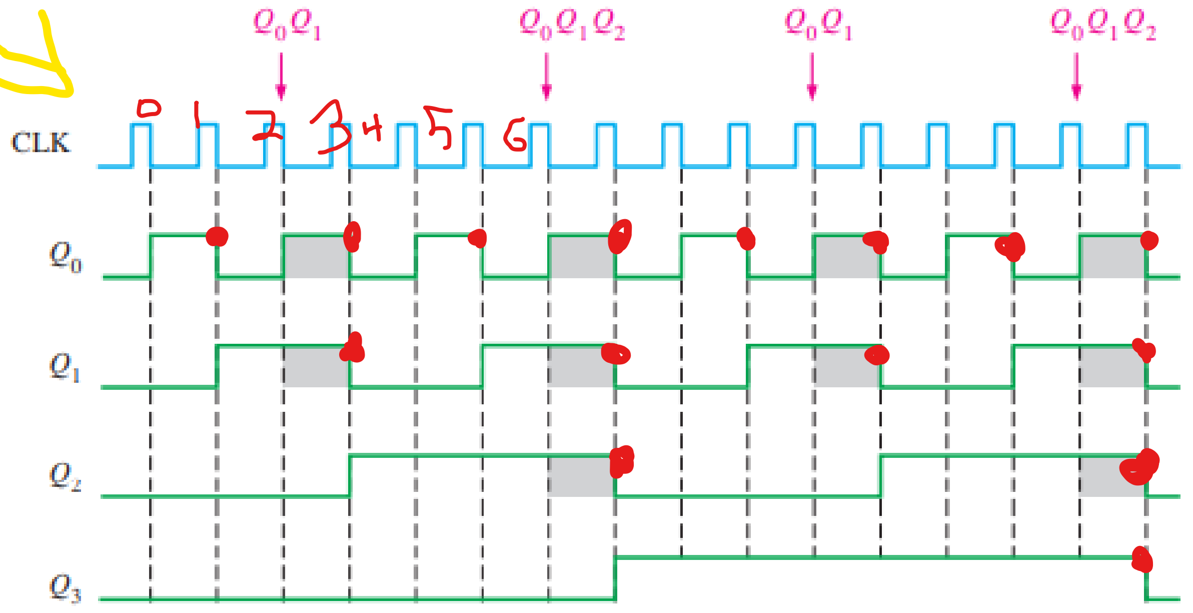
| Clock Pulse | Outputs $Q_2$ | $Q_1$ | $Q_0$ | J-K Inputs $J_2$ | $K_2$ | $J_1$ | $K_1$ | $J_0$ | $K_0$ | At the Next Clock Pulse FF2 | FF1 | FF0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initially | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | NC* | NC | Toggle |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 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. | | |

# 4-bit synchronous counter: divide by / mod 16 counter

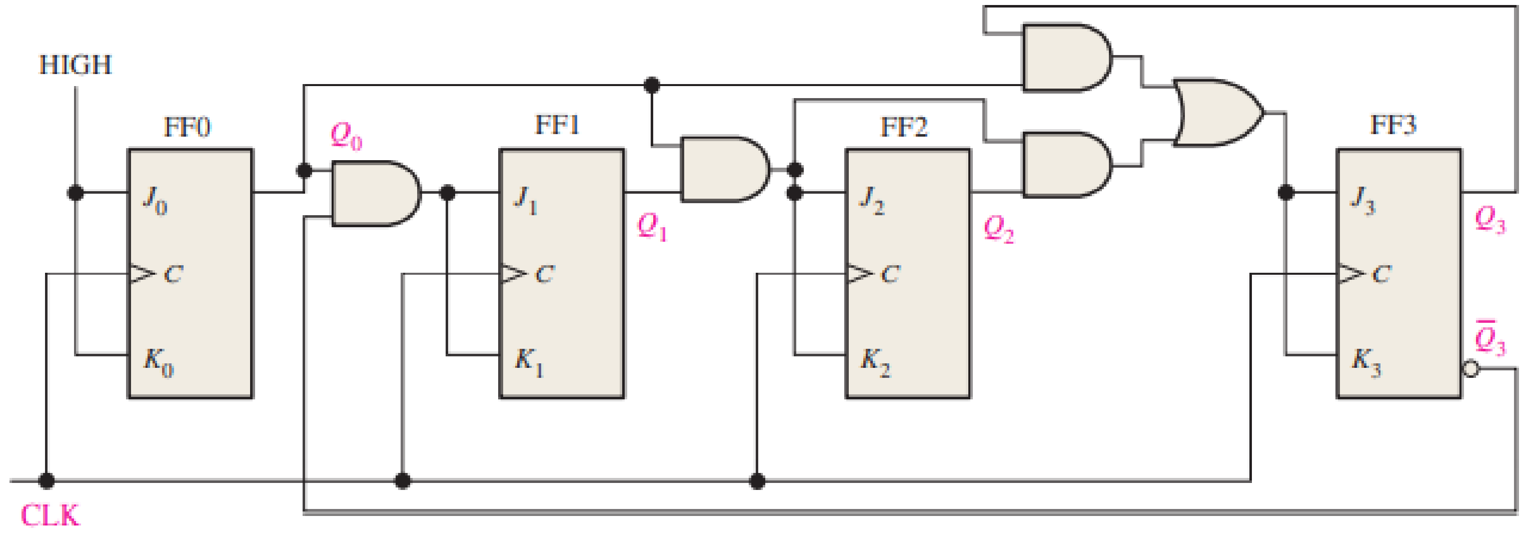


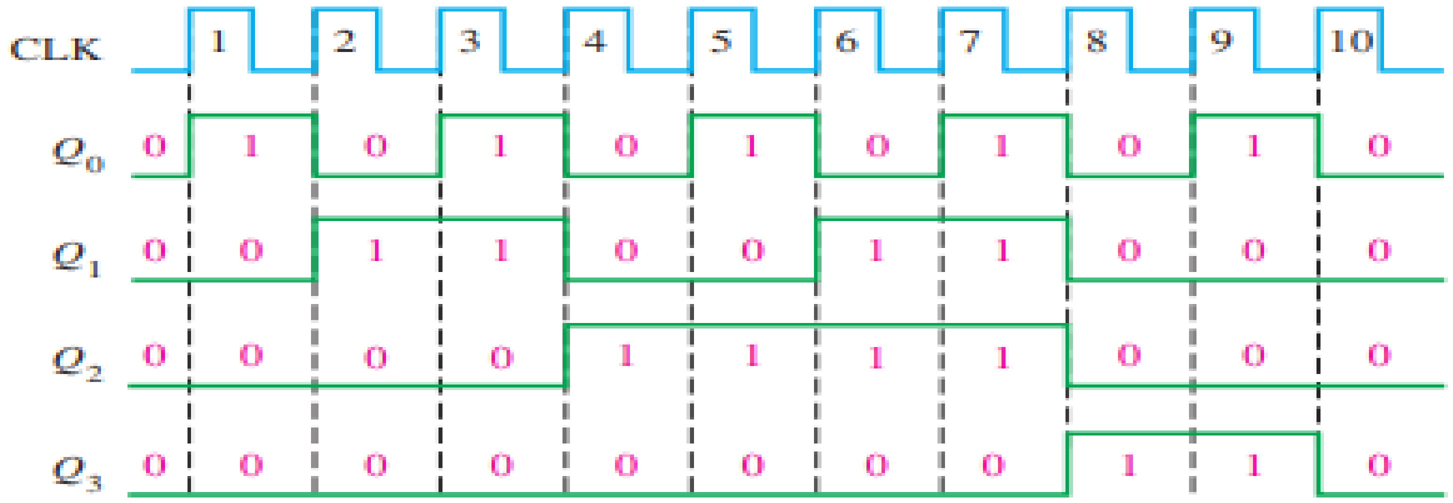
- J and K input control for the first three flip-flops is the same as previously discussed for the 3-bit counter.

- 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.



# 4-bit synchronous BCD counter



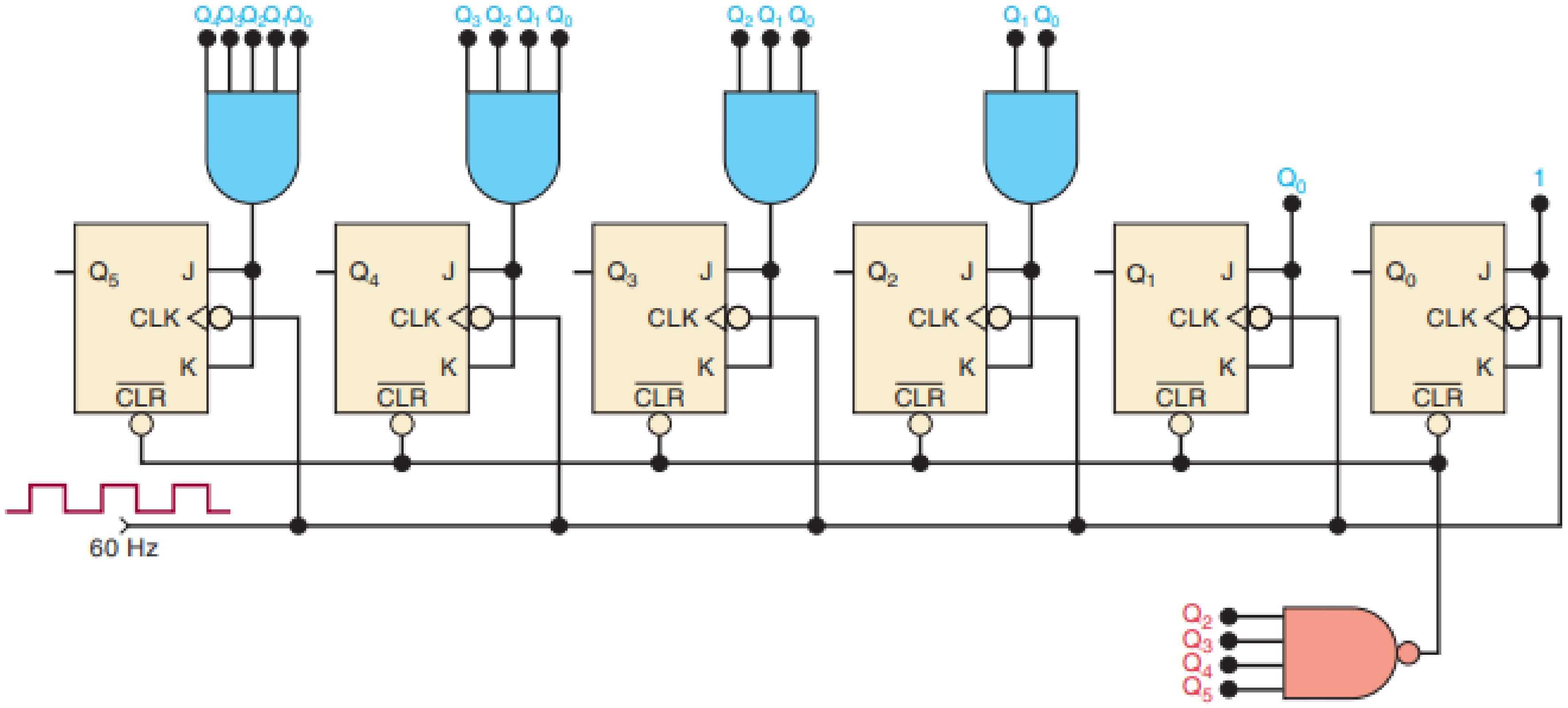


| Clock Pulse | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| Initially | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10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

# Mod 60 counter

**PROB**: Construct an appropriate MOD-60 counter to divide the 60-Hz line frequency down to 1 Hz.

**Solution** $2^5 = 32$ and $2^6 = 64$, and so we need six FFs, as shown in Figure 7-9. The counter is to be cleared when it reaches the count of 60 (111100). Thus, the outputs of flip-flops Q5, Q4, Q3, and Q2 must be connected to the NAND gate. The output of flip-flop Q5 will have a frequency of 1 Hz.



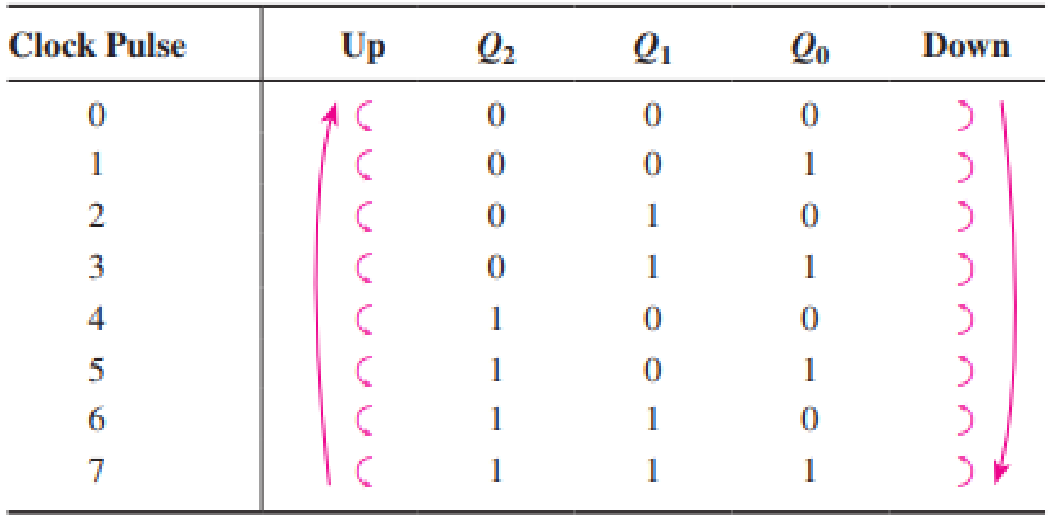
# Up/down synchronous binary counter

- 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.
- $Q_0$ 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 $\mathbf{J_0 = K_0 = 1}$
- 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: $J_1 = K_1 = (Q_0 \bullet UP) + (\underline{Q_0} \bullet DOWN)$
- 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:

$$J_2 = K_2 = (Q_0 \bullet Q_1 \bullet UP) + (\underline{Q_0} \bullet \underline{Q_1} \bullet DOWN)$$

- 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.

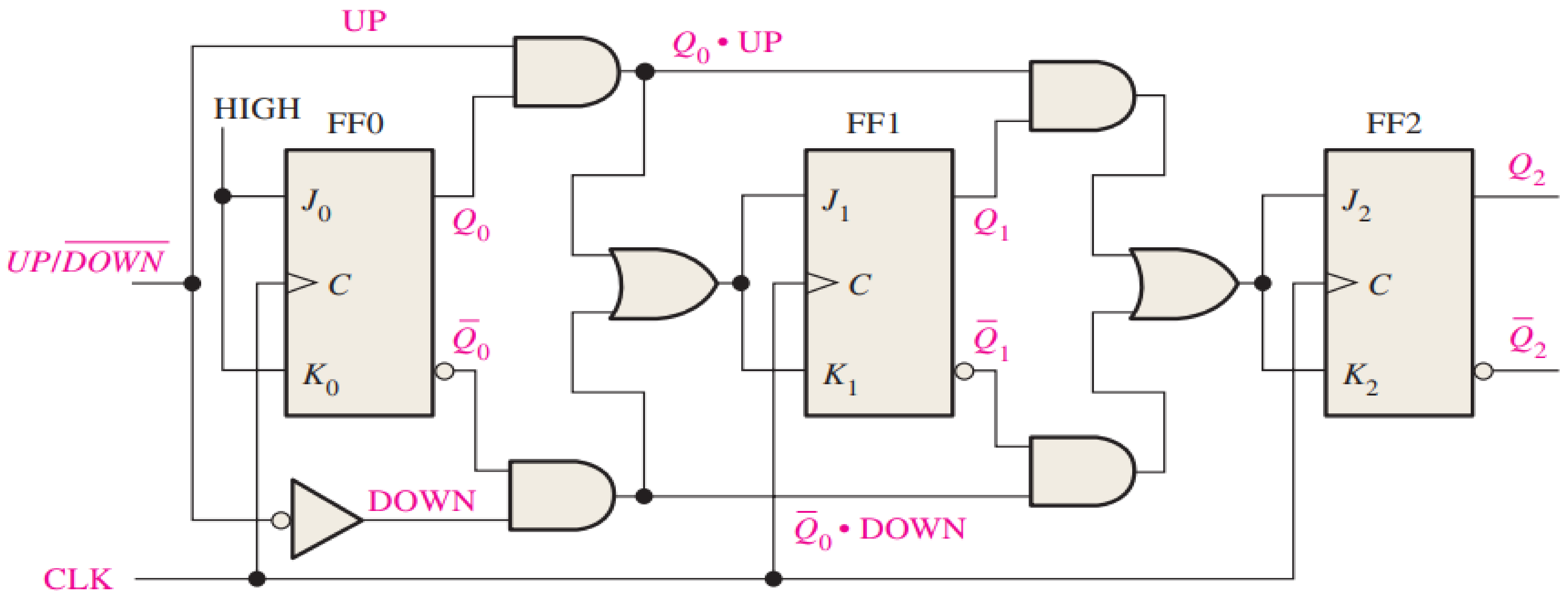
Up/Down sequence for a 3-bit binary counter.



| 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 | |

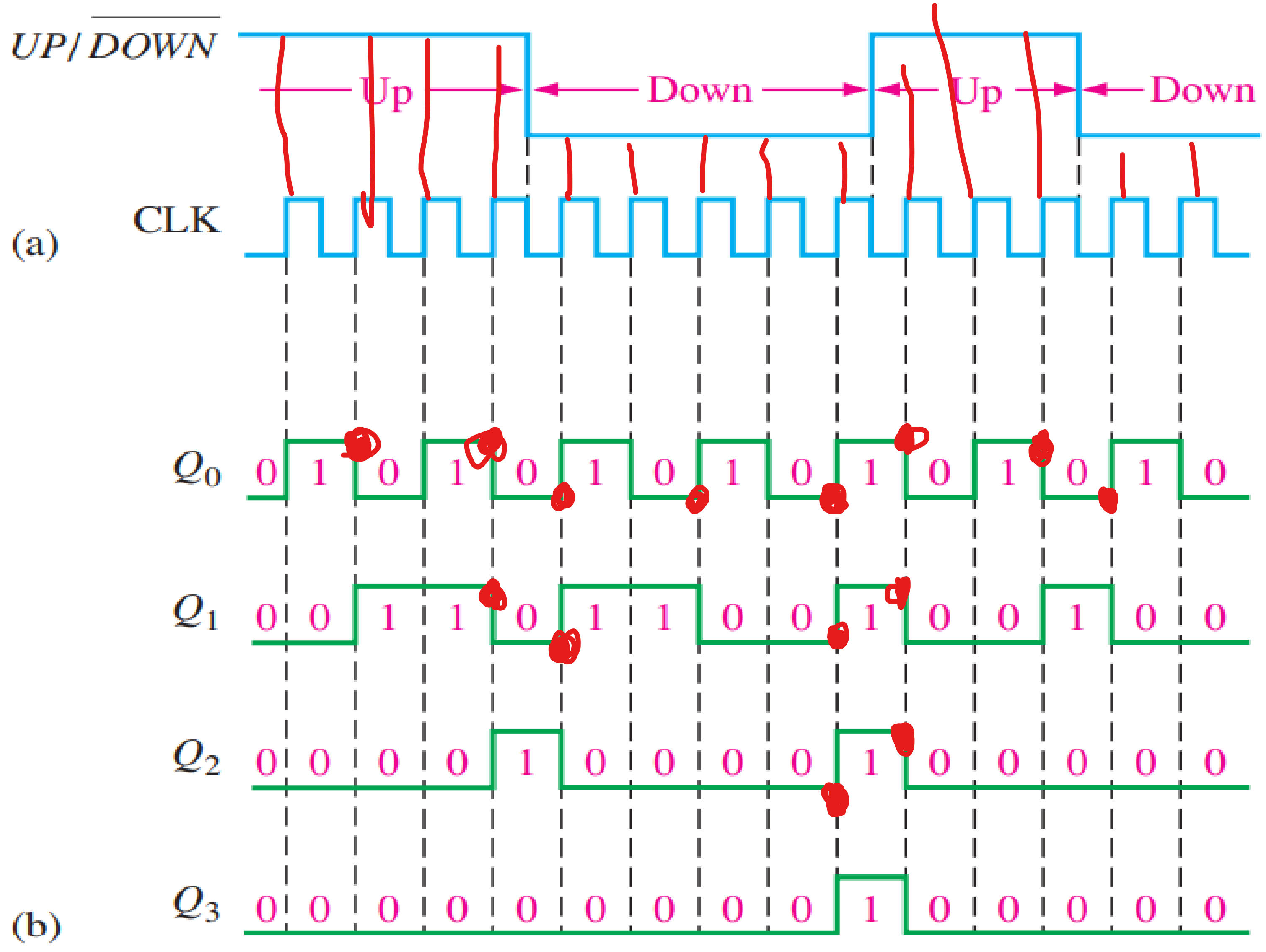
## 3-bit Up/down synchronous binary counter

- Figure 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/DOWN control input is HIGH for UP and LOW for DOWN.

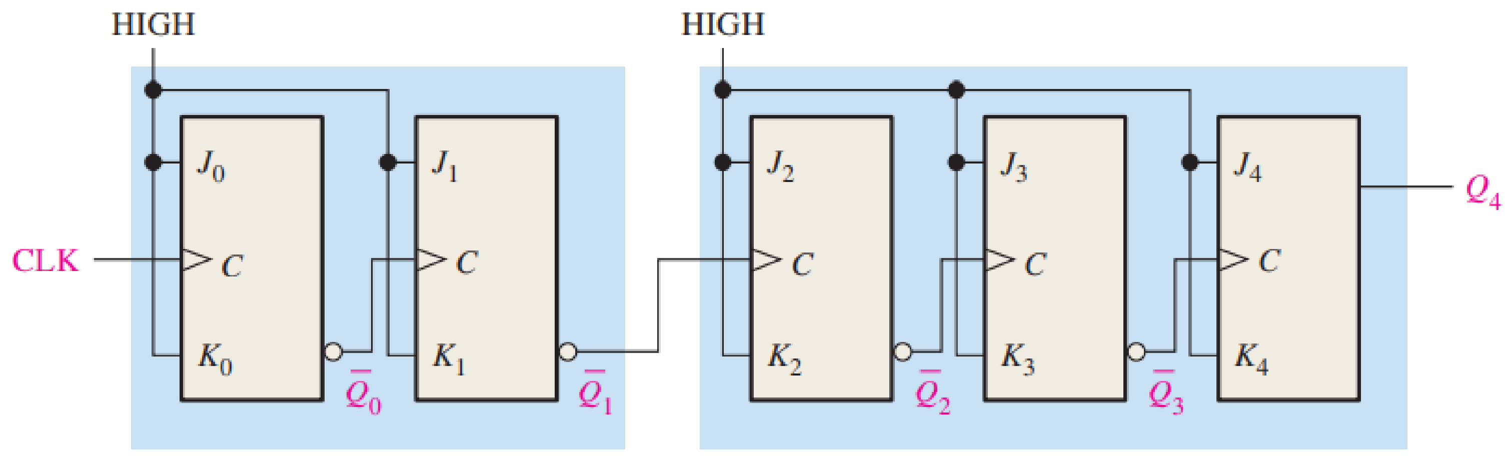


# 4-bit synchronous up/down counter

**PROB:** Show the timing diagram and determine the sequence of a 4-bit synchronous binary up/down counter if the clock and UP/DOWN control inputs have waveforms as shown in Figure (a). The counter starts in the all-0s state and is positive edge-triggered.

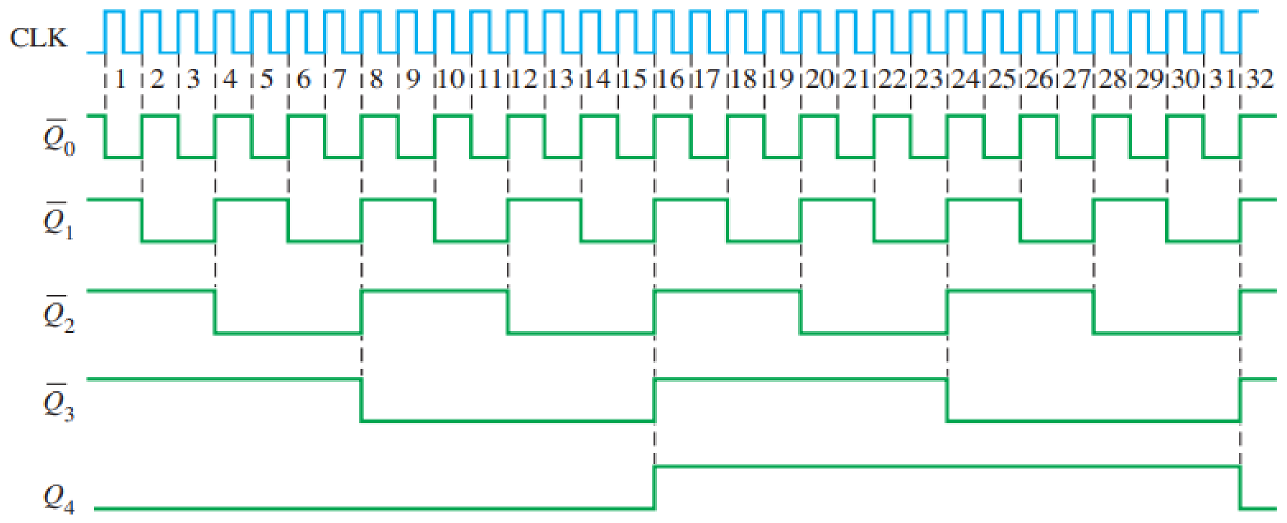


## 2-bit & 3-bit cascaded asynchronous counters: MOD 4x8=32

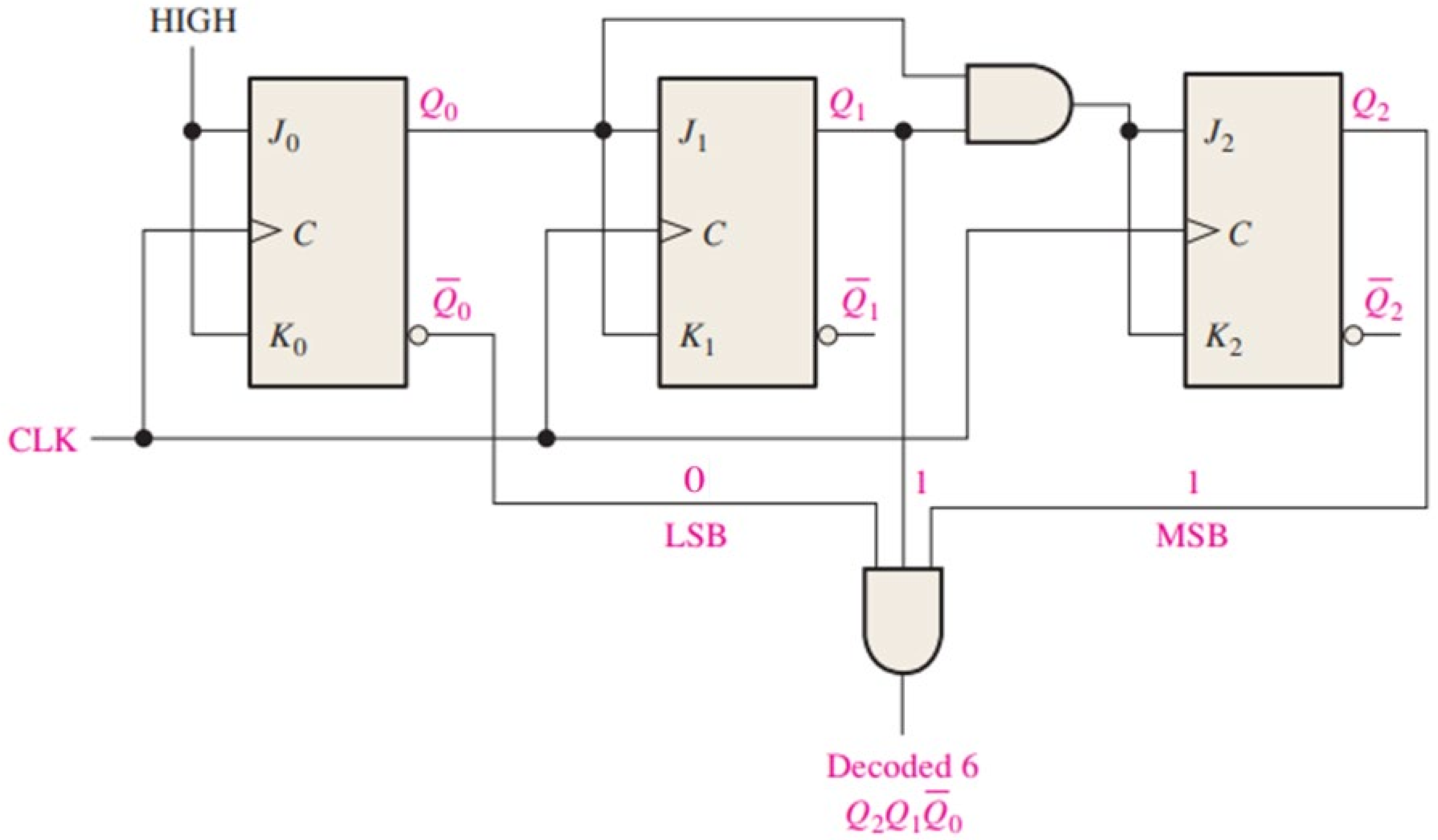


Modulus-4 counter

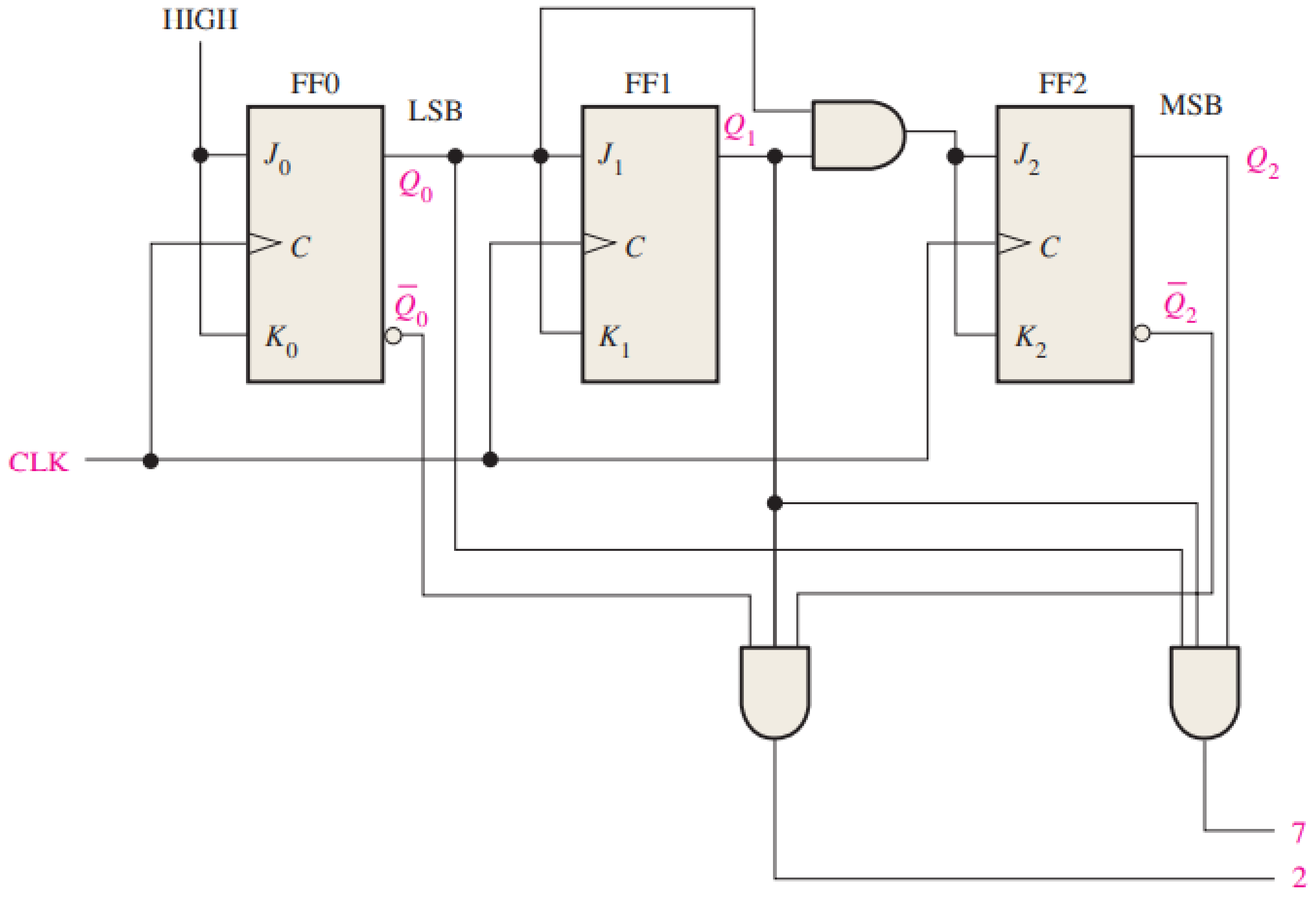
Modulus-8 counter

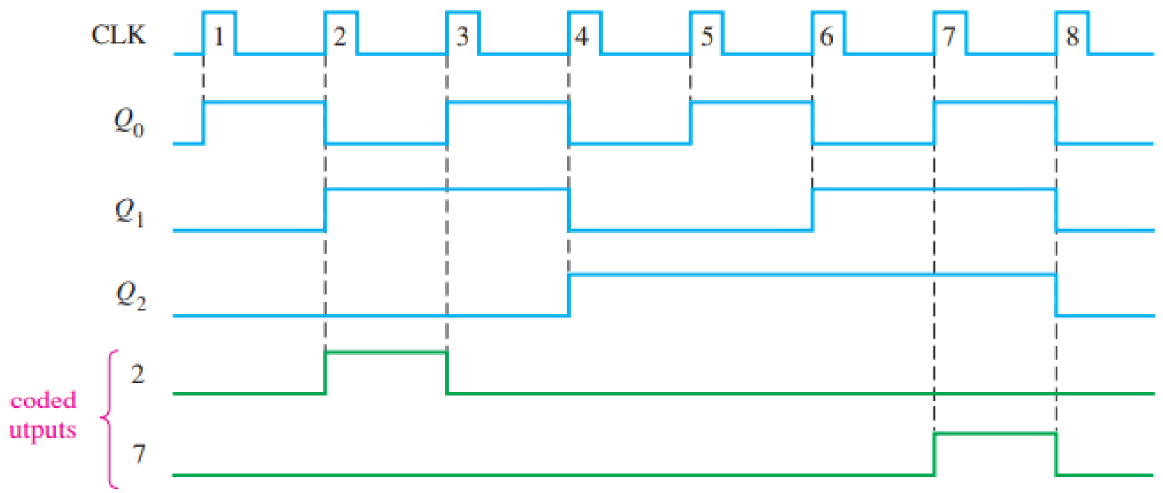


## Counter decoding: decoding of state 6

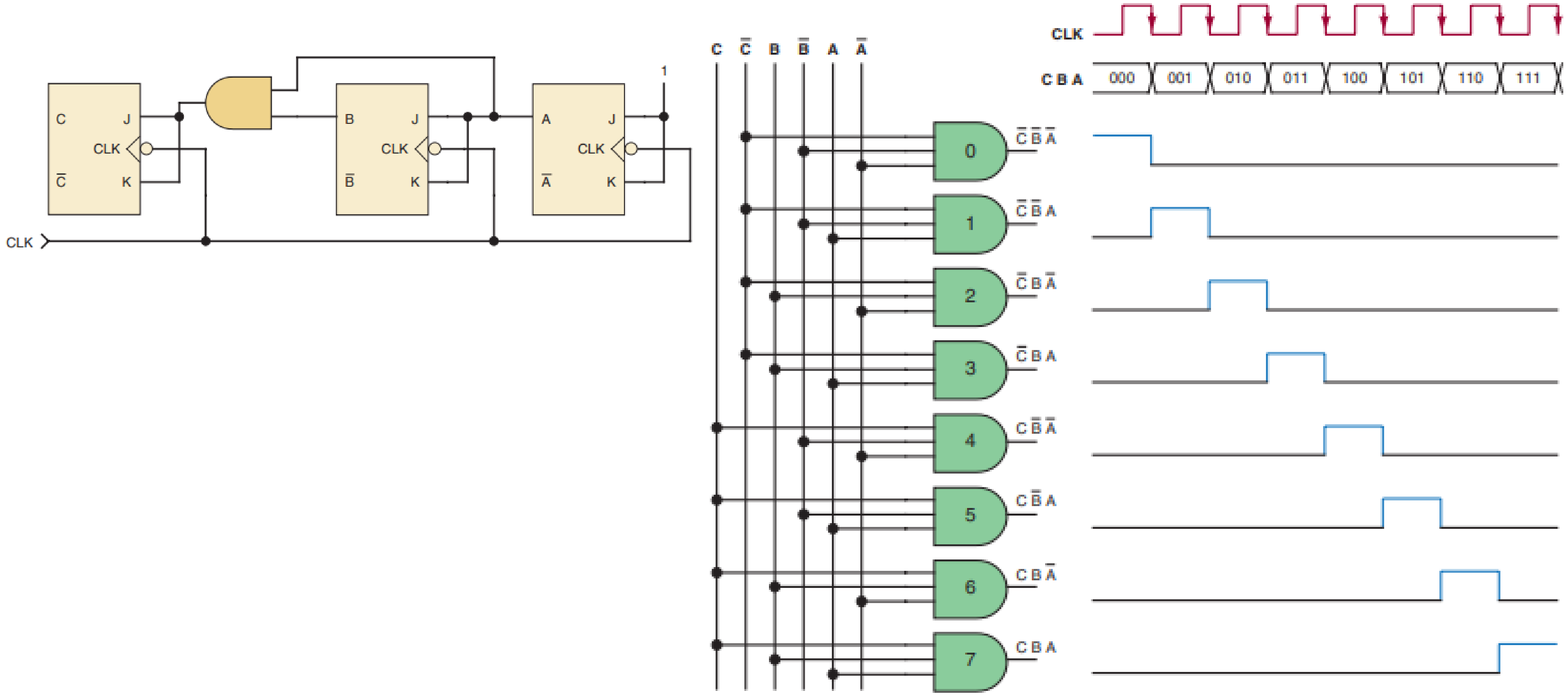


# Counter decoding: decoding of state 2 and 7



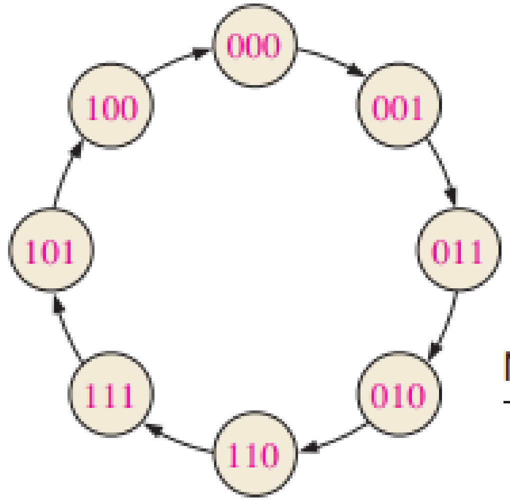


# Mod 8 counter decoding



## Design of synchronous counter (3-bit gray counter)

### Step 1: State Diagram



### Step 2: Next-State Table

Next-state table for 3-bit Gray code counter.

| Present State | | | Next State | | |
|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |

### Step 3: Flip-Flop Transition Table

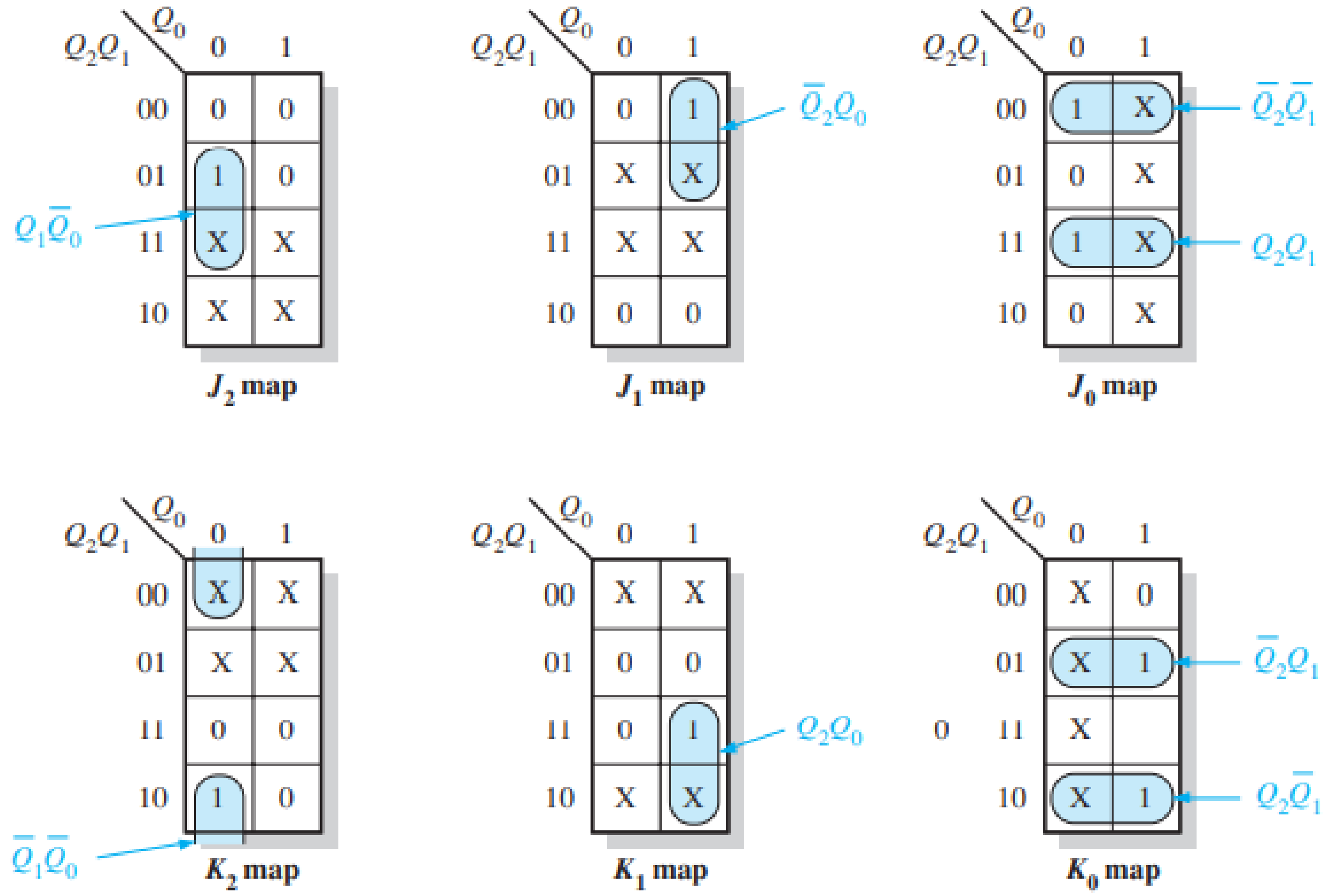
Transition table for a J-K flip-flop.

| Output Transitions | | | Flip-Flop Inputs | |
|---|---|---|---|---|
| $Q_N$ | | $Q_{N+1}$ | $J$ | $K$ |
| 0 | $\longrightarrow$ | 0 | 0 | X |
| 0 | $\longrightarrow$ | 1 | 1 | X |
| 1 | $\longrightarrow$ | 0 | X | 1 |
| 1 | $\longrightarrow$ | 1 | X | 0 |

$Q_N$: present state
$Q_{N+1}$: next state
X: "don't care"

# Design of synchronous counter (3-bit gray counter)

## Step 4: Karnaugh Maps



**$J_2$ map**

| $Q_2Q_1 \backslash Q_0$ | 0 | 1 |
|---|---|---|
| 00 | 0 | 0 |
| 01 | 1 | 0 |
| 11 | X | X |
| 10 | X | X |

**$J_1$ map**

| $Q_2Q_1 \backslash Q_0$ | 0 | 1 |
|---|---|---|
| 00 | 0 | 1 |
| 01 | X | X |
| 11 | X | X |
| 10 | 0 | 0 |

**$J_0$ map**

| $Q_2Q_1 \backslash Q_0$ | 0 | 1 |
|---|---|---|
| 00 | 1 | X |
| 01 | 0 | X |
| 11 | 1 | X |
| 10 | 0 | X |

**$K_2$ map**

| $Q_2Q_1 \backslash Q_0$ | 0 | 1 |
|---|---|---|
| 00 | X | X |
| 01 | X | X |
| 11 | 0 | 0 |
| 10 | 1 | 0 |

**$K_1$ map**

| $Q_2Q_1 \backslash Q_0$ | 0 | 1 |
|---|---|---|
| 00 | X | X |
| 01 | 0 | 0 |
| 11 | 0 | 1 |
| 10 | X | X |

**$K_0$ map**

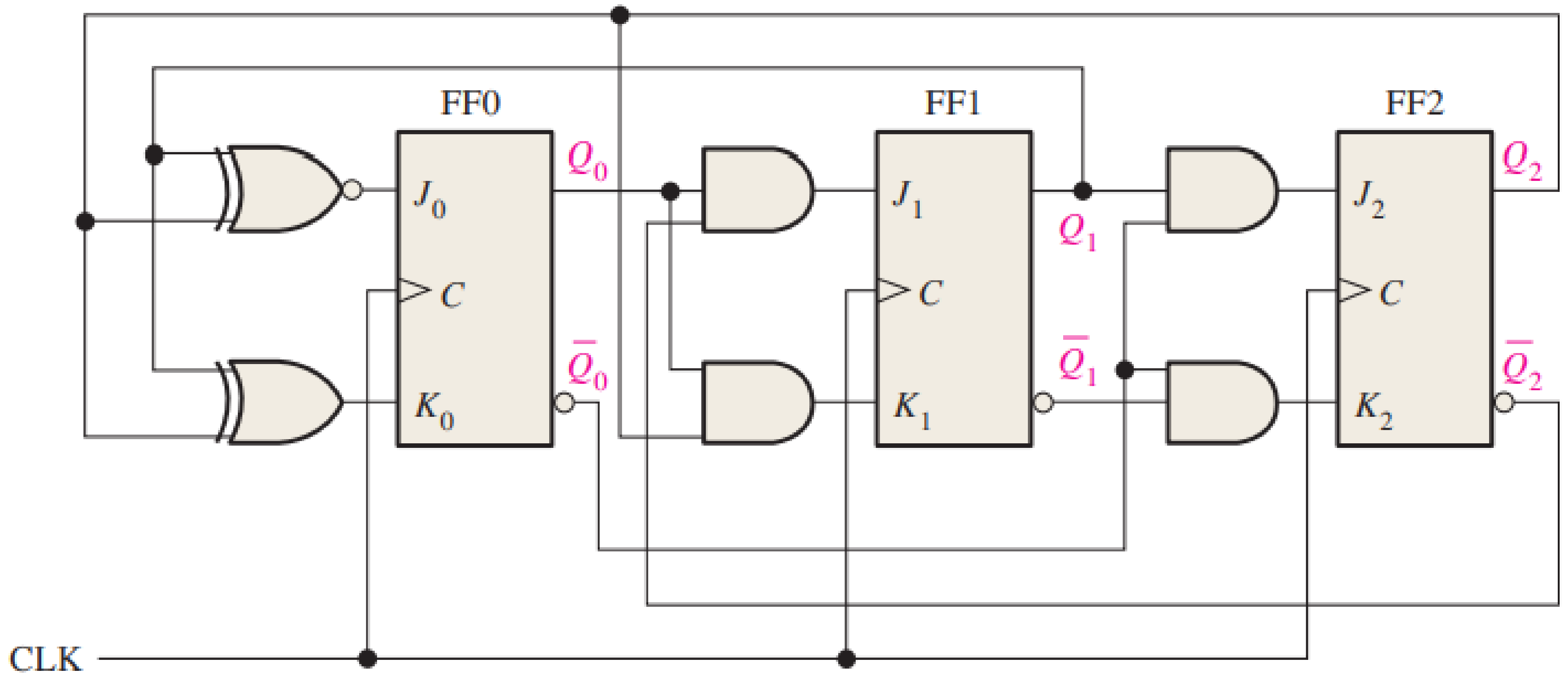
| $Q_2Q_1 \backslash Q_0$ | 0 | 1 |
|---|---|---|
| 00 | X | 0 |
| 01 | X | 1 |
| 11 | X |  |
| 10 | X | 1 |

## Step 5: Logic Expressions for Flip-Flop Inputs

$$J_0 = Q_2Q_1 + \overline{Q}_2\overline{Q}_1 = \overline{Q_2 \oplus Q_1}$$
$$K_0 = Q_2\overline{Q}_1 + \overline{Q}_2Q_1 = Q_2 \oplus Q_1$$
$$J_1 = \overline{Q}_2Q_0$$
$$K_1 = Q_2Q_0$$
$$J_2 = Q_1\overline{Q}_0$$
$$K_2 = \overline{Q}_1\overline{Q}_0$$

## Design of synchronous counter (3-bit gray counter)

### Step 6: Counter Implementation



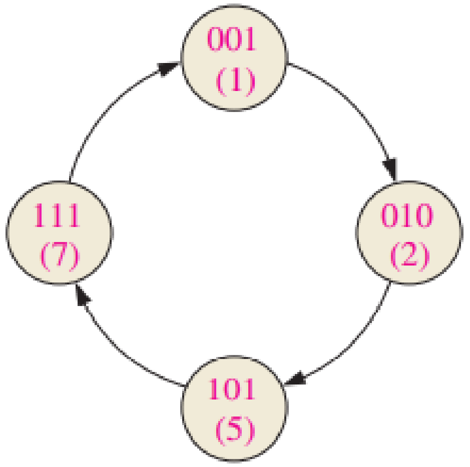
# Design of synchronous counter

**PROB.** Design a counter with the irregular binary count sequence 001-010-101-111. Use D flip-flops.

Step 1: State Diagram

Step 2: Next-State Table

Step 3: Flip-Flop Transition Table



Next-state table.

| Present State | | | Next State | | |
|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

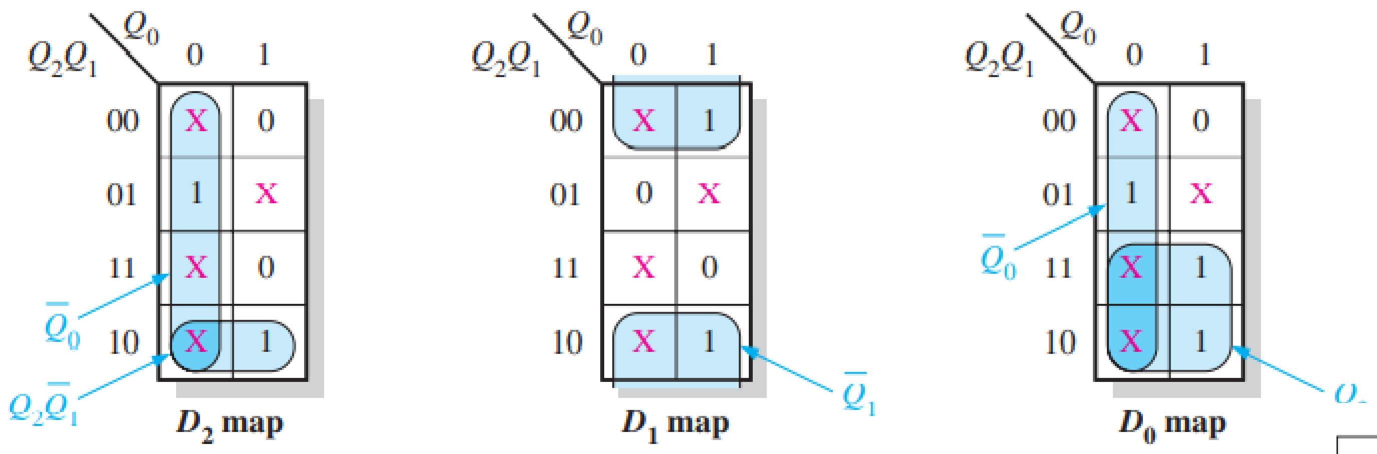
Transition table for a D flip-flop.

| Output Transitions | | | Flip-Flop Input |
|---|---|---|---|
| $Q_N$ | | $Q_{N+1}$ | $D$ |
| 0 | $\longrightarrow$ | 0 | 0 |
| 0 | $\longrightarrow$ | 1 | 1 |
| 1 | $\longrightarrow$ | 0 | 0 |
| 1 | $\longrightarrow$ | 1 | 1 |

The invalid states (0, 3, 4, and 6) can be treated as "don't cares" in the design.

# Design of synchronous counter

## Step 4: Karnaugh Maps



| $Q_2Q_1$ \ $Q_0$ | 0 | 1 |
|---|---|---|
| 00 | X | 0 |
| 01 | 1 | X |
| 11 | X | 0 |
| 10 | X | 1 |

$D_2$ map

| $Q_2Q_1$ \ $Q_0$ | 0 | 1 |
|---|---|---|
| 00 | X | 1 |
| 01 | 0 | X |
| 11 | X | 0 |
| 10 | X | 1 |

$D_1$ map

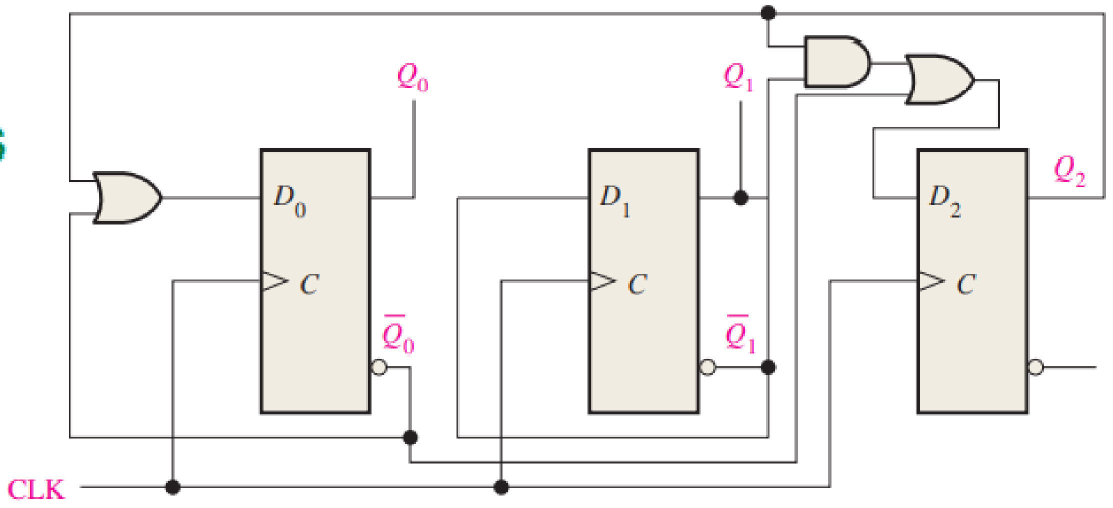
| $Q_2Q_1$ \ $Q_0$ | 0 | 1 |
|---|---|---|
| 00 | X | 0 |
| 01 | 1 | X |
| 11 | X | 1 |
| 10 | X | 1 |

$D_0$ map

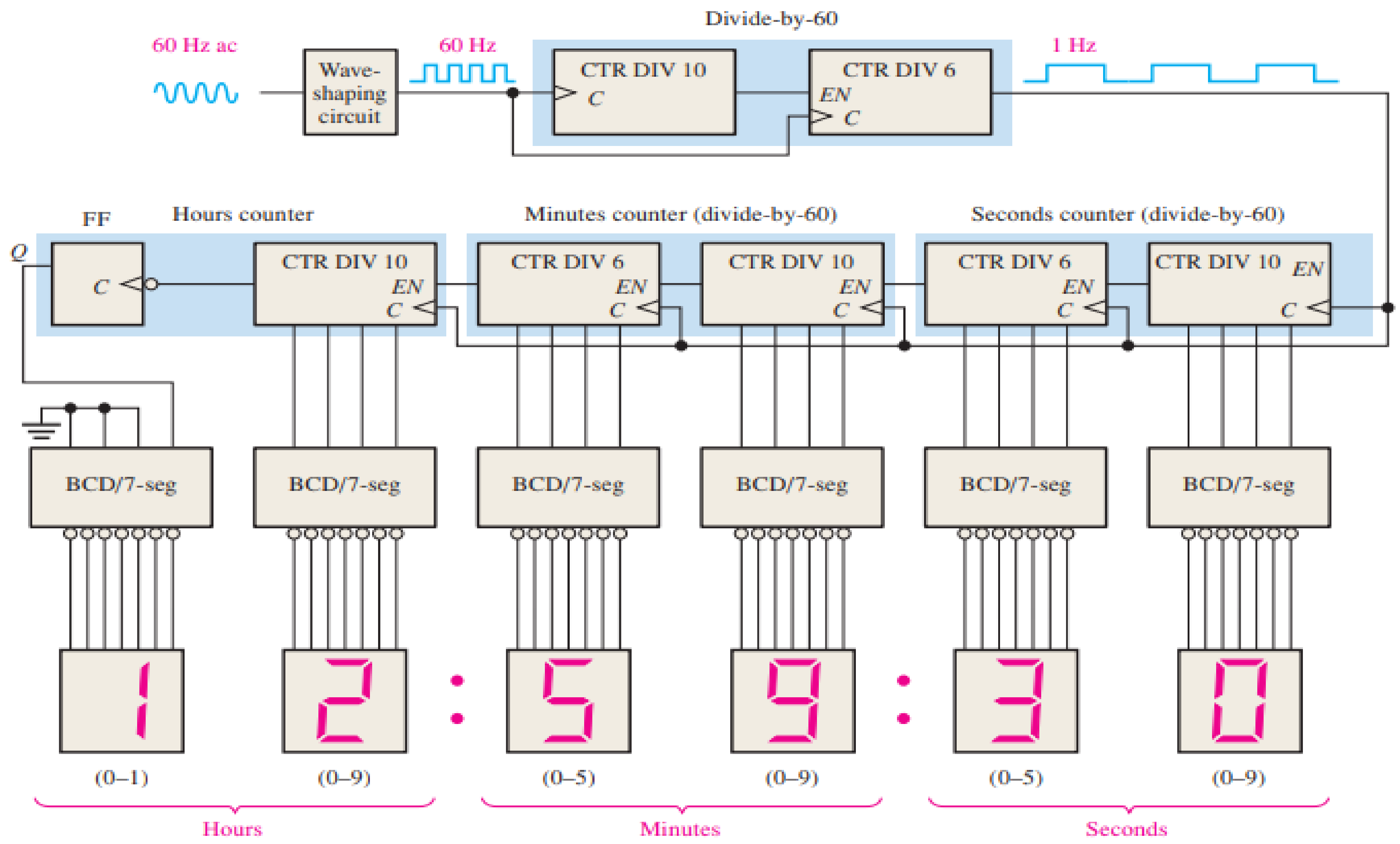
## Step 5: Logic Expressions for Flip-Flop Inputs

$$D_0 = \overline{Q}_0 + Q_2$$
$$D_1 = \overline{Q}_1$$
$$D_2 = \overline{Q}_0 + Q_2\overline{Q}_1$$

## Step 6: Counter Implementation



# Counter applications: digital clock



## Counter applications: digital clock

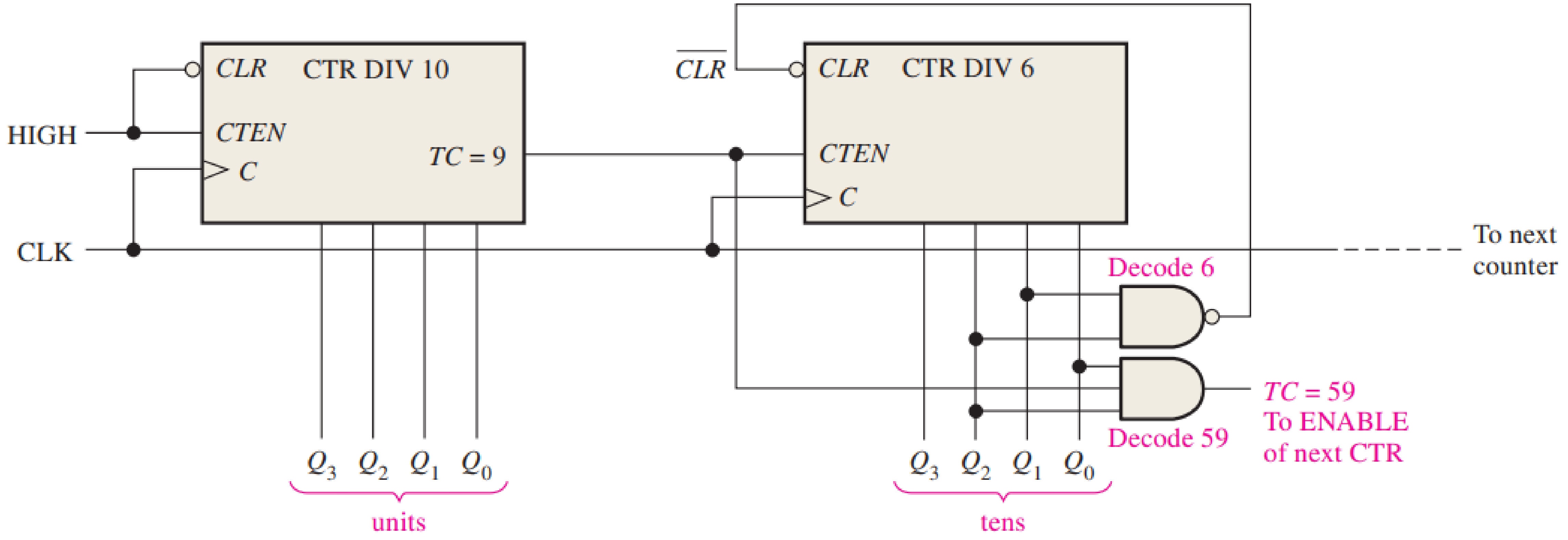


**FIGURE** Logic diagram of typical divide-by-60 counter using synchronous decade counters. Note that the outputs are in binary order (the right-most bit is the LSB)

# Counter applications: digital clock

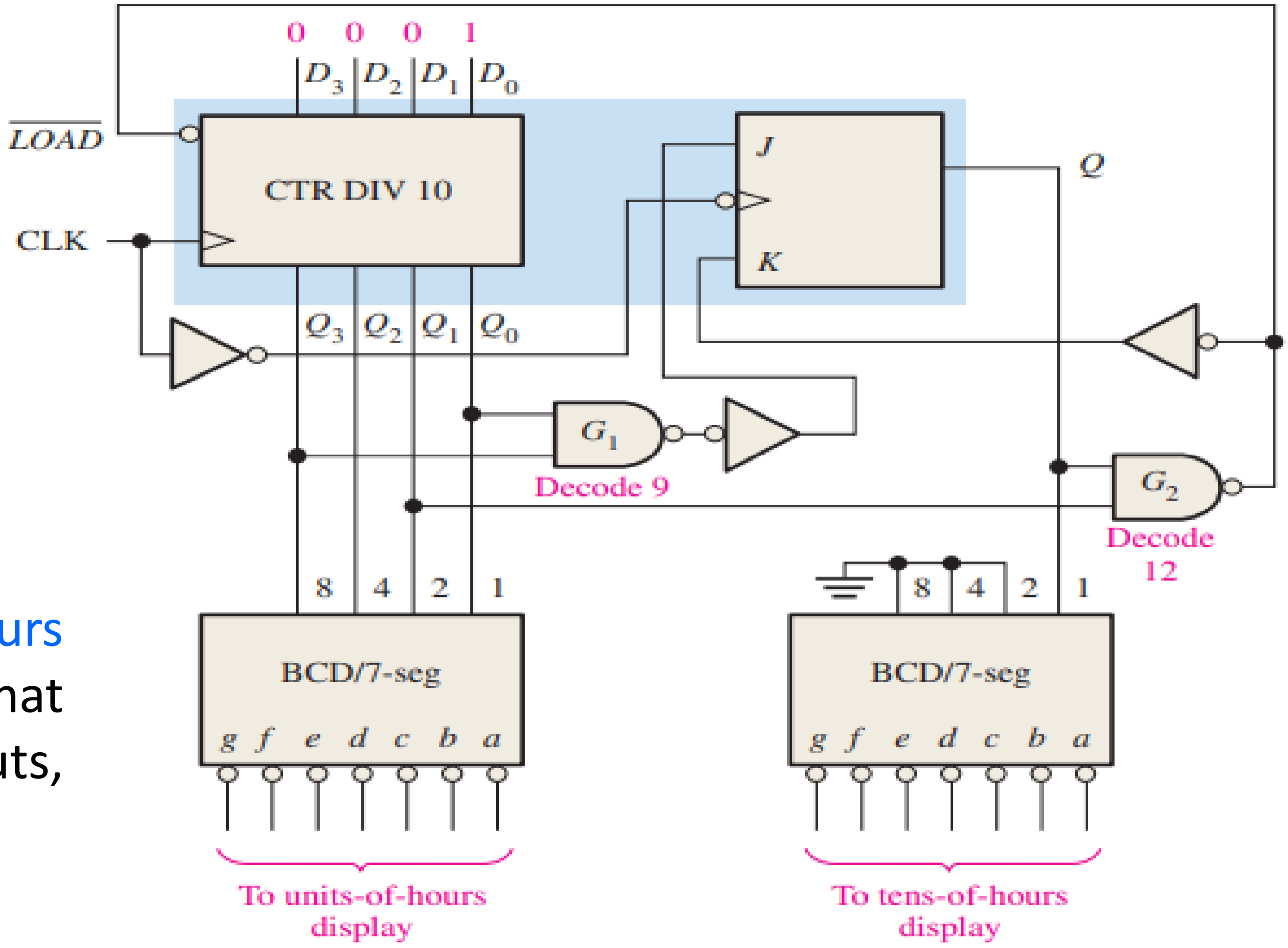
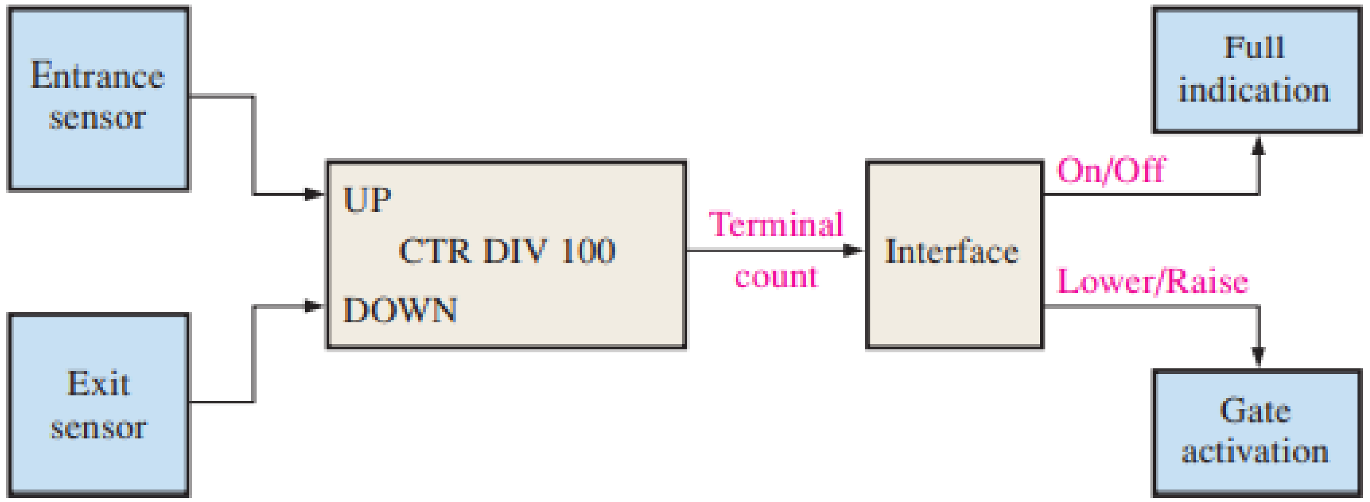


**FIGURE** Logic diagram for hours counter and decoders. Note that on the counter inputs and outputs, the right-most bit is the LSB.

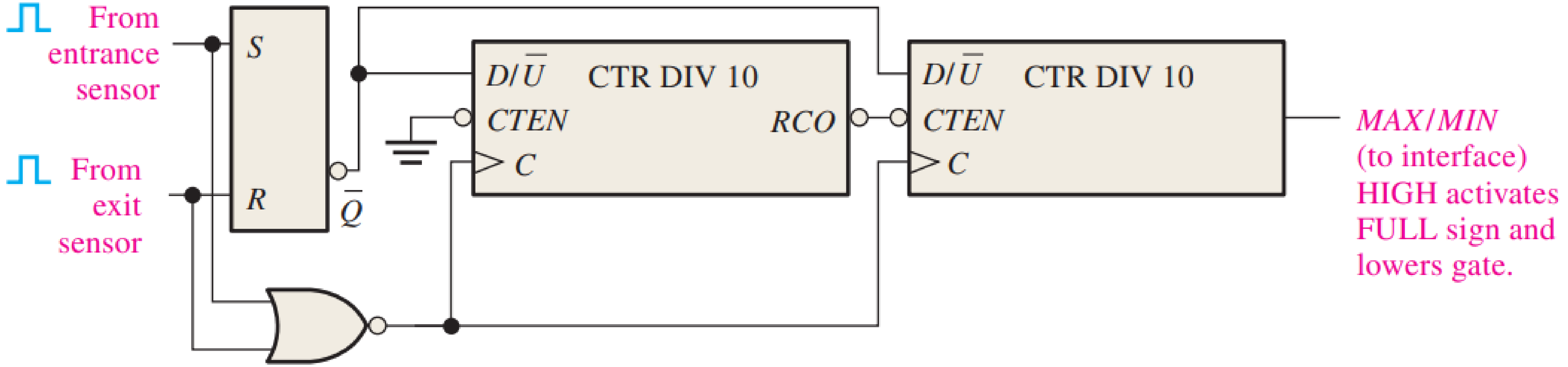
# Counter applications: automobile car parking control

**PROBLEM**. Design a parking monitoring available spaces in a one-hundred space parking garage and provide for an indication of a full condition by illuminating a display sign and lowering a gate bar at the entrance.

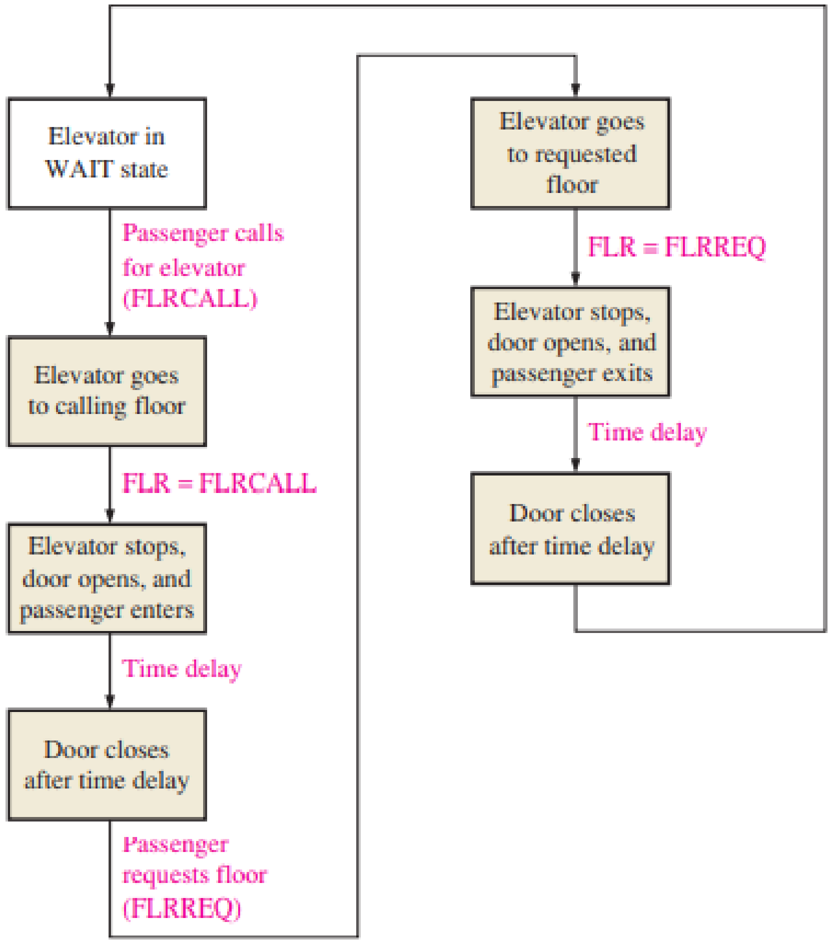
**SOLUTION:** A system that solves this problem consists of optoelectronic sensors at the entrance and exit of the garage, an up/down counter and associated circuitry, and an interface circuit that uses the counter output to turn the FULL sign on or off as required and lower or raise the gate bar at the entrance. A general block diagram of this system is shown in Figure



# Counter applications: automobile car parking control

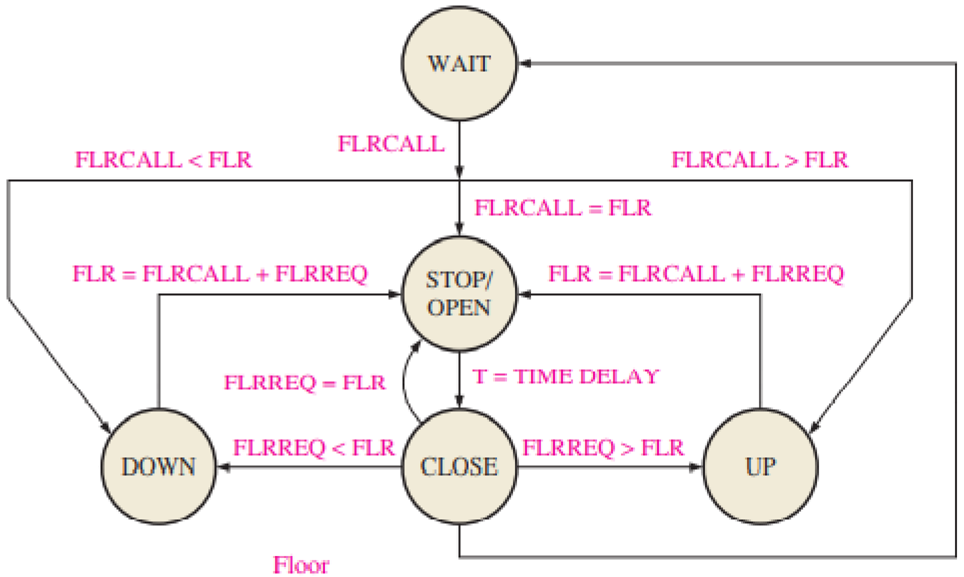


# Counter applications: elevator controller

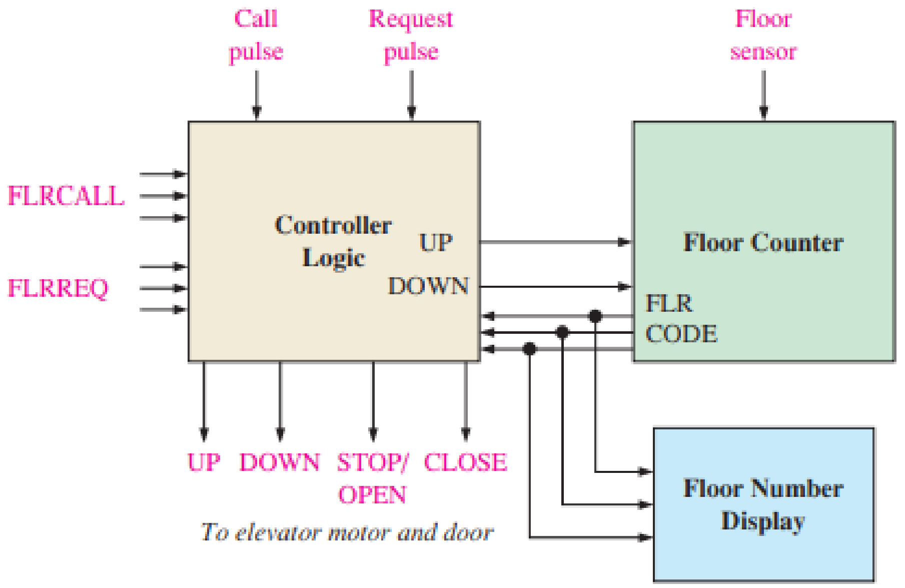


Elevator in WAIT state
Passenger calls for elevator (FLRCALL)
Elevator goes to calling floor
FLR = FLRCALL
Elevator stops, door opens, and passenger enters
Time delay
Door closes after time delay
Passenger requests floor (FLRREQ)
Elevator goes to requested floor
FLR = FLRREQ
Elevator stops, door opens, and passenger exits
Time delay
Door closes after time delay

One cycle of the elevator operation

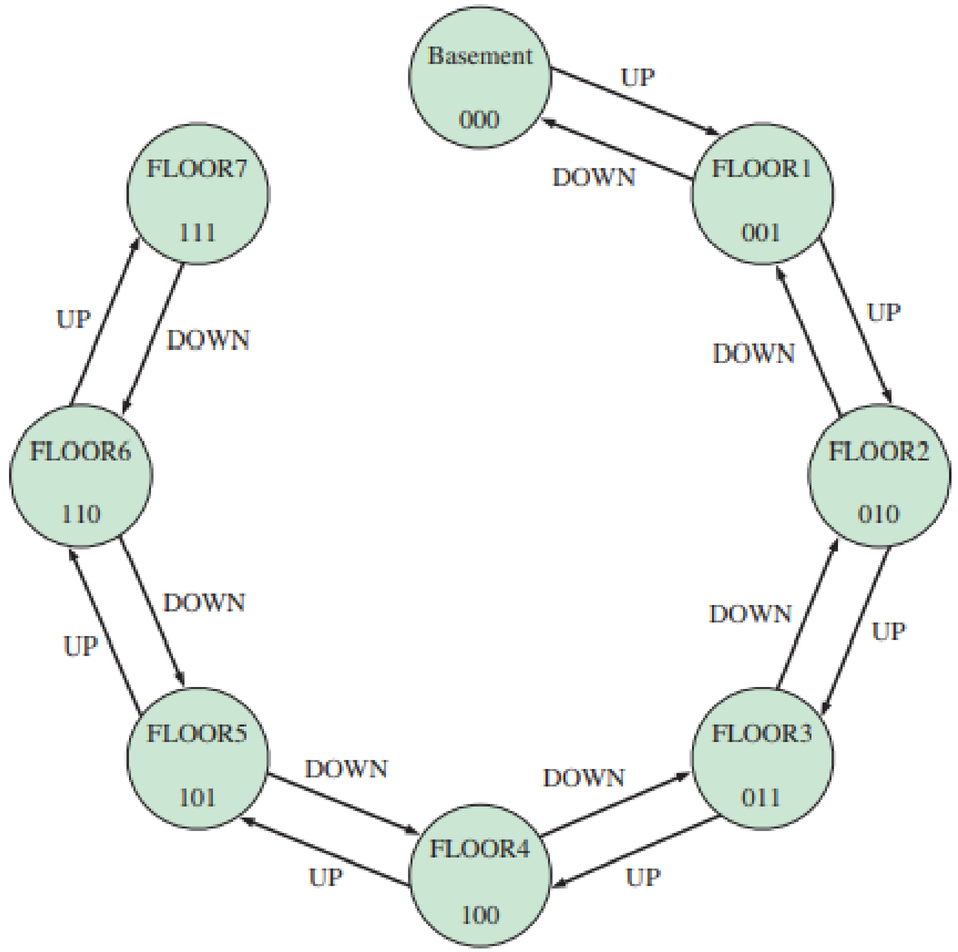


Elevator controller state diagram

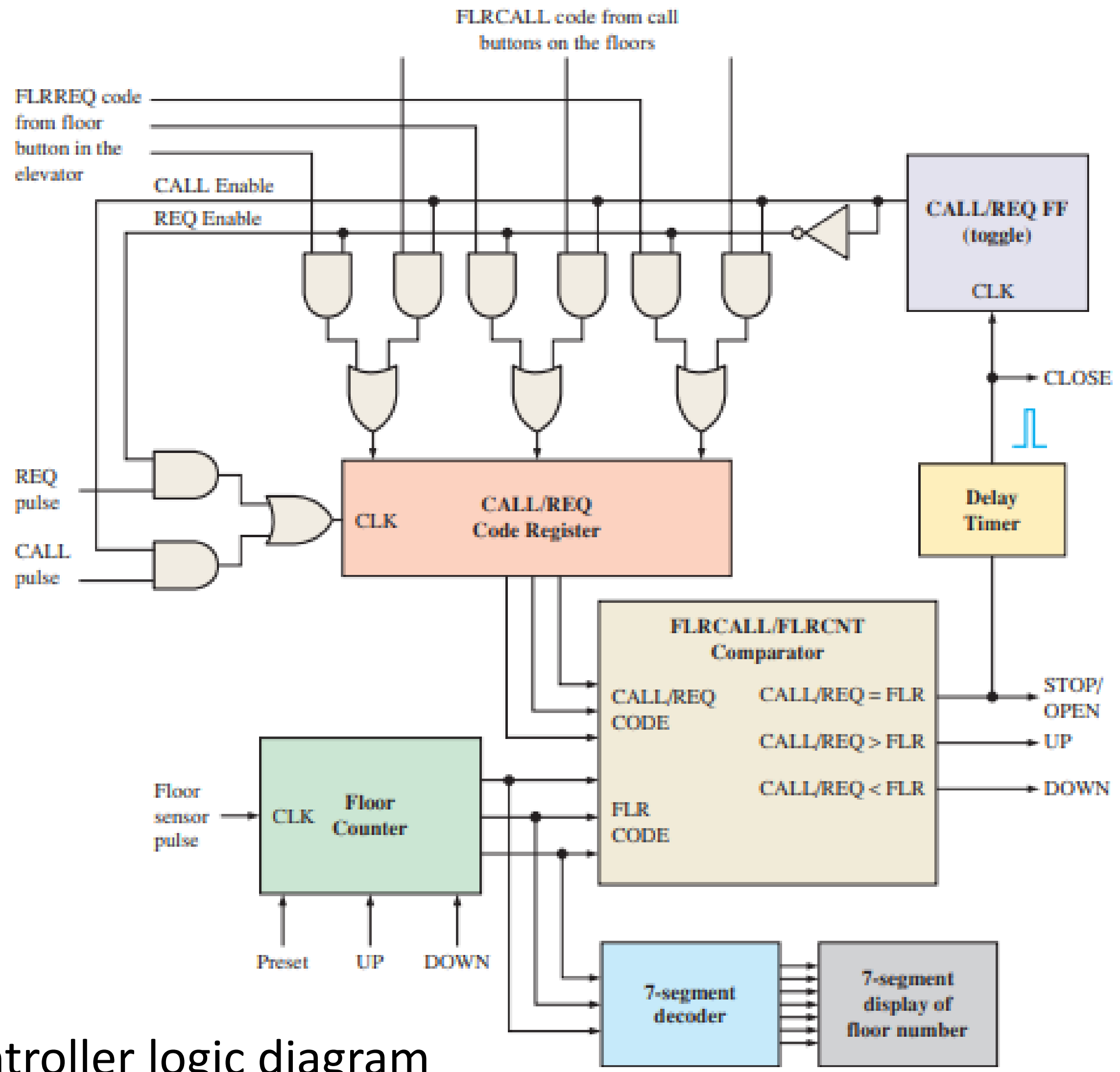


Elevator controller block diagram

# Counter applications: elevator controller



Floor counter state diagram



Elevator controller logic diagram

## References

1. ***Digital Fundamentals*** by Thomas Floyd, Pearson International Edition, 11th Edition, Chapter 9, Page 497-560.
2. ***Digital Systems: Principles and Applications*** by Ronald Tocci, Neal Widmer and Greg Moss, Pearson International Edition, 12th Edition, Chapter 7, Page 408-498.

## Next class

**Counters**