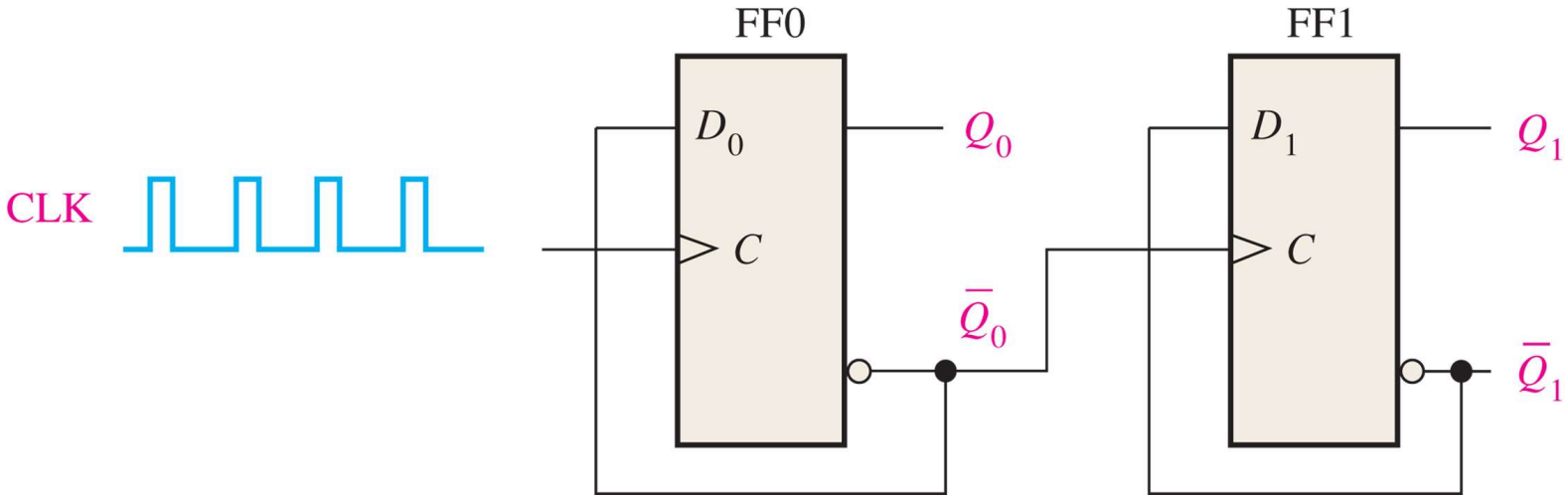
# Digital Fundamentals

Thomas L. Floyd

## Counters
## **Chapter 9**

# A 2-bit Asynchronous Counter

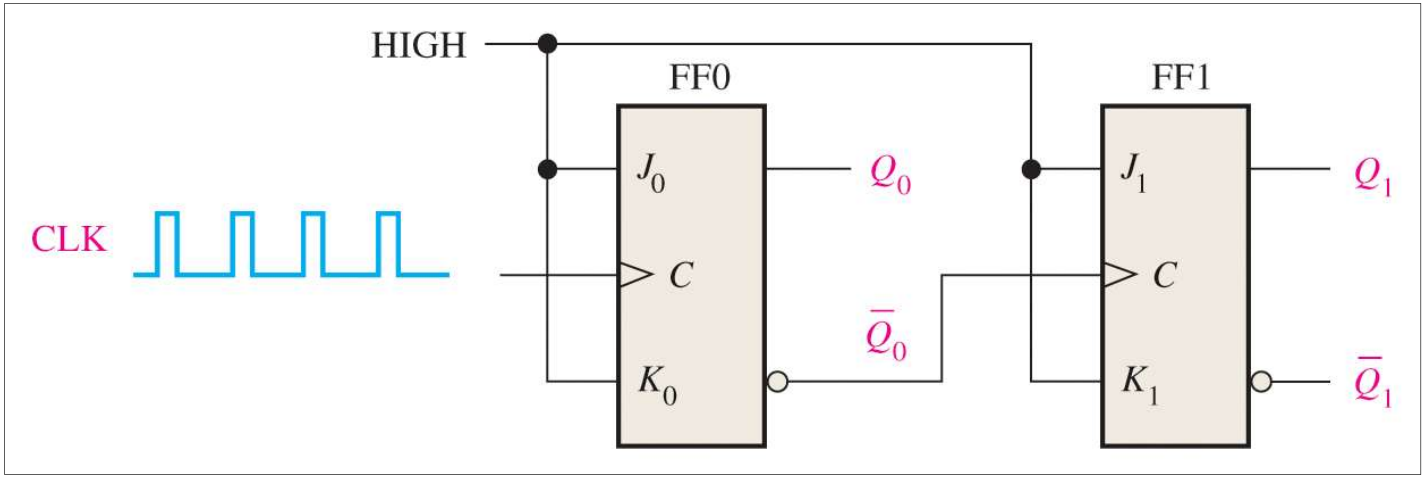
In an **asynchronous counter**, the clock is applied only to the first stage. Subsequent stages derive the clock from the previous stage.




The 2-bit asynchronous counter shown is typical.

# A 2-bit Asynchronous Counter

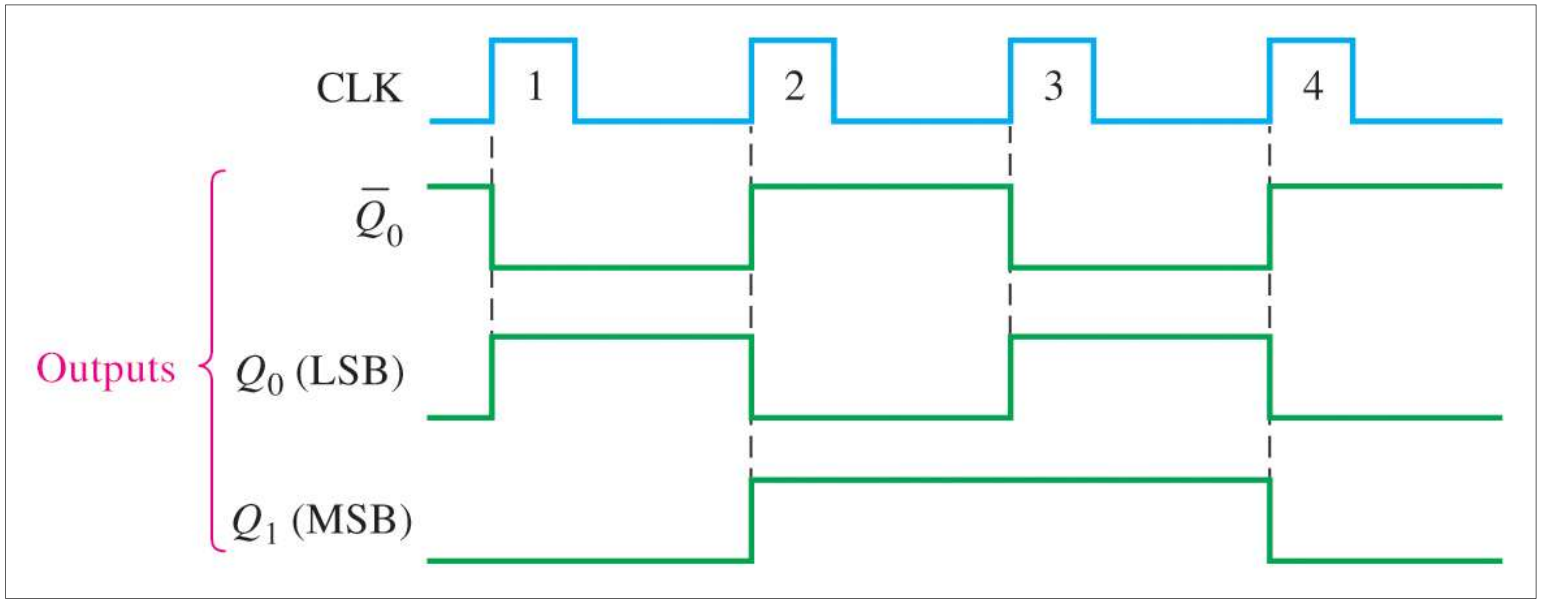
In an **asynchronous counter**, the clock is applied only to the first stage. Subsequent stages derive the clock from the previous stage.



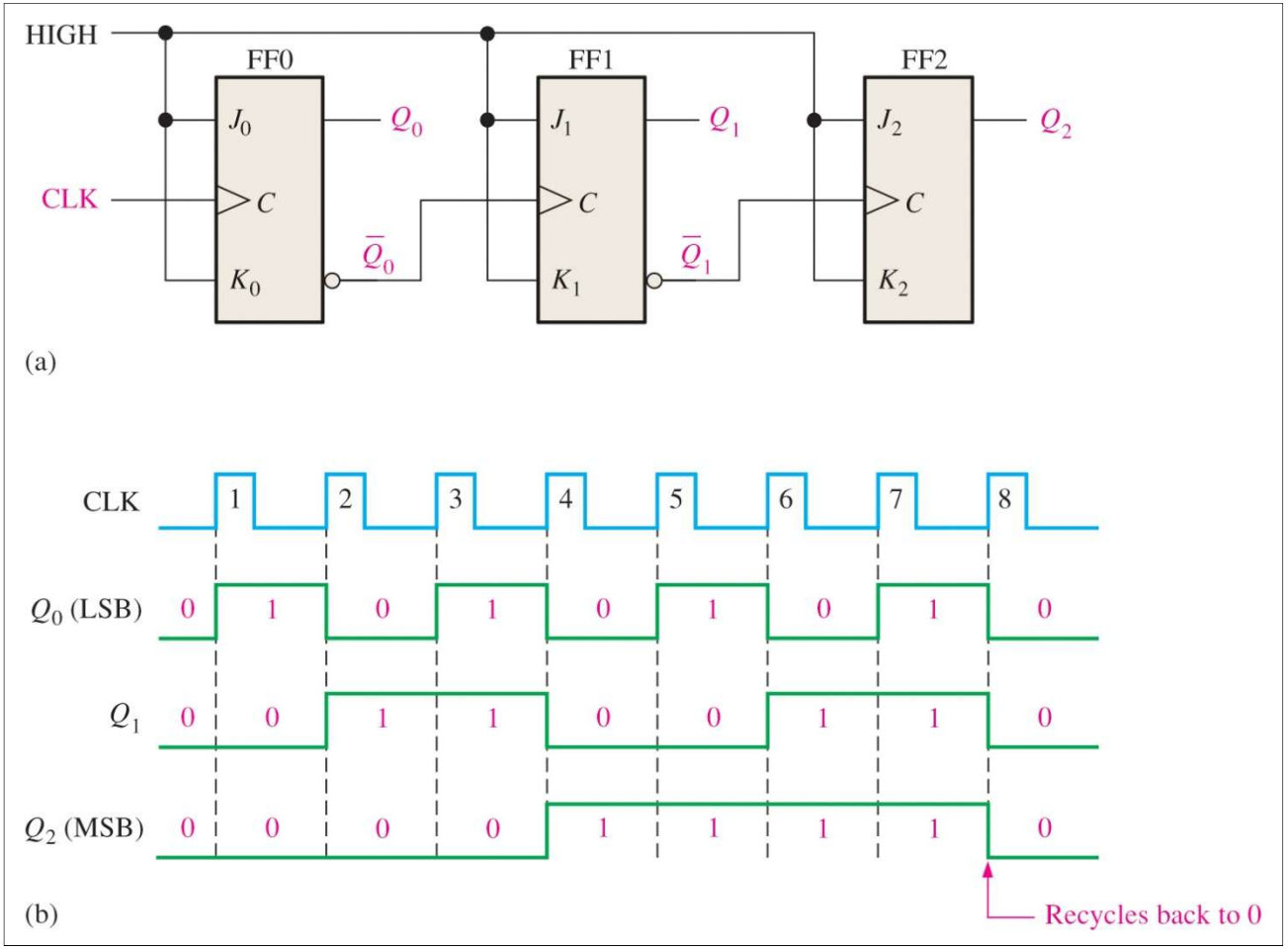

The 2-bit asynchronous counter shown is typical. It uses J-K flip-flops in the toggle mode.

# A 2-bit Asynchronous Counter Timing Diagram

Shown here is the timing diagram for the 2-bit asynchronous counter shown on the previous slide.



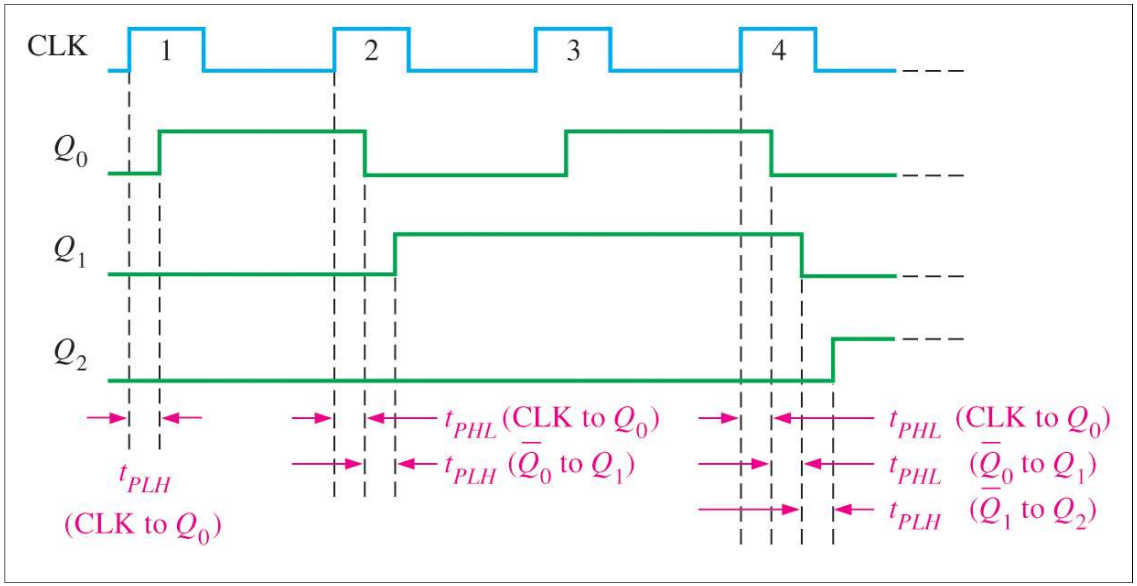
# A 3-bit Asynchronous Counter



# Propagation Delay

Asynchronous counters are sometimes called **ripple counters**, because the stages do not all change together. For certain applications requiring high clock rates, this is a major disadvantage.

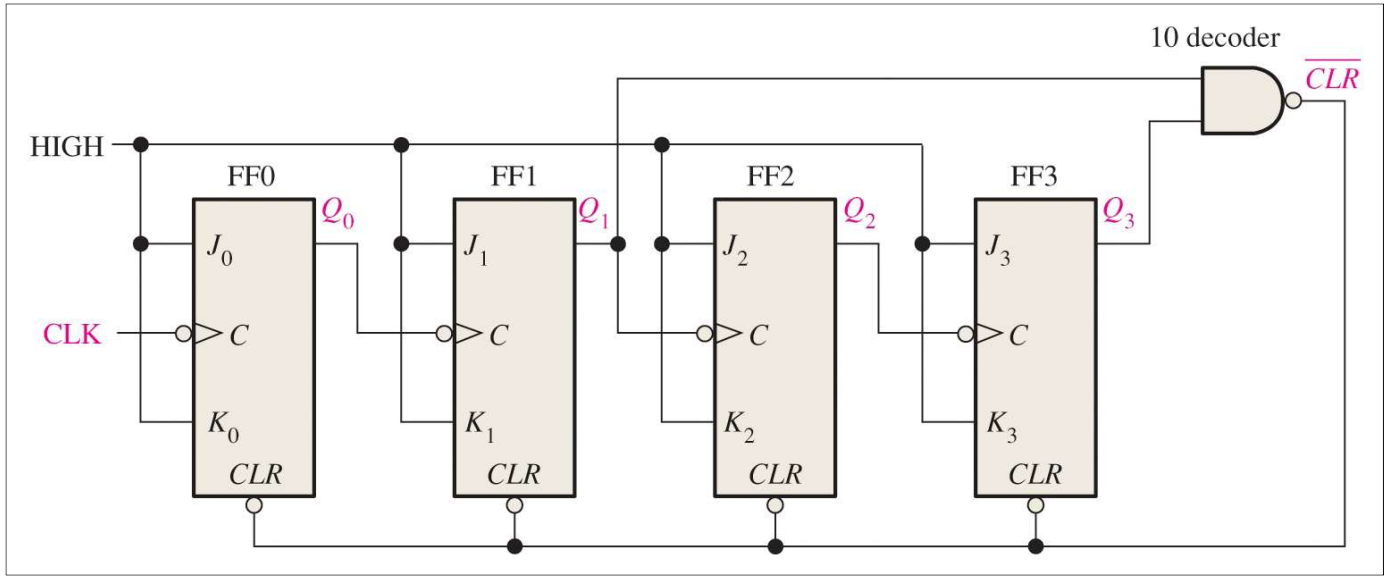
Note how the delays are cumulative as each stage in a counter is clocked later than the previous stage.




4-bit counter with tp=10ns, frequency <=25MHz

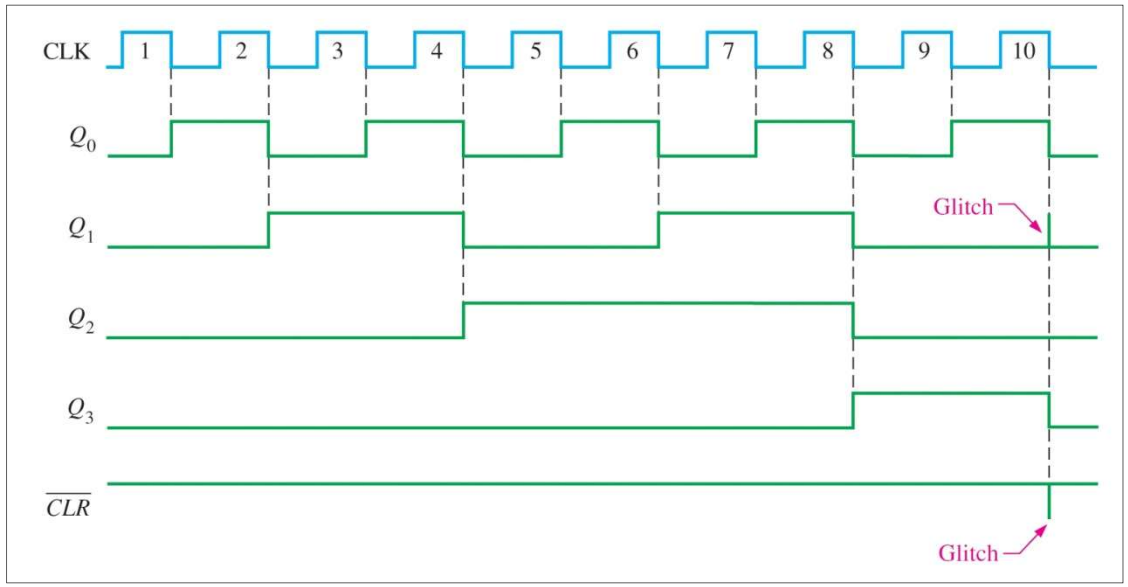
# Asynchronous Decade Counter

This counter uses partial decoding to recycle the count sequence to zero after the 1001 state. The flip-flops are trailing-edge triggered, so clocks are derived from the *Q* outputs.

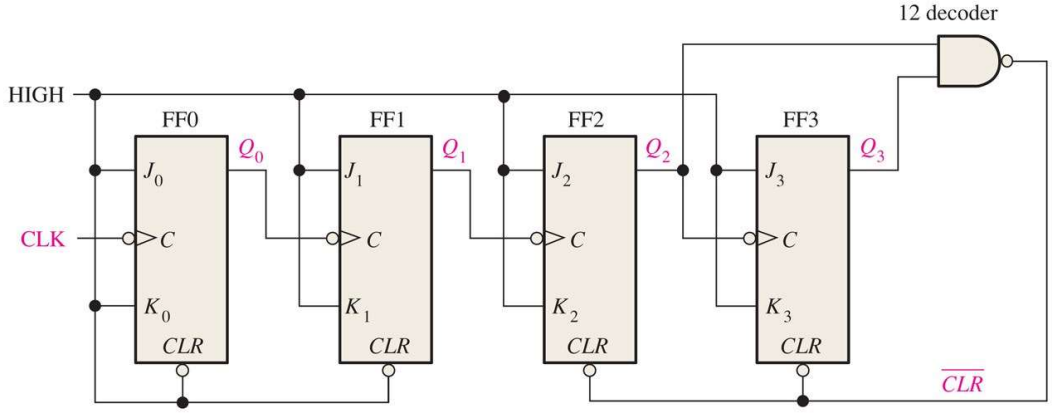


# Asynchronous Decade Counter Timing Diagram

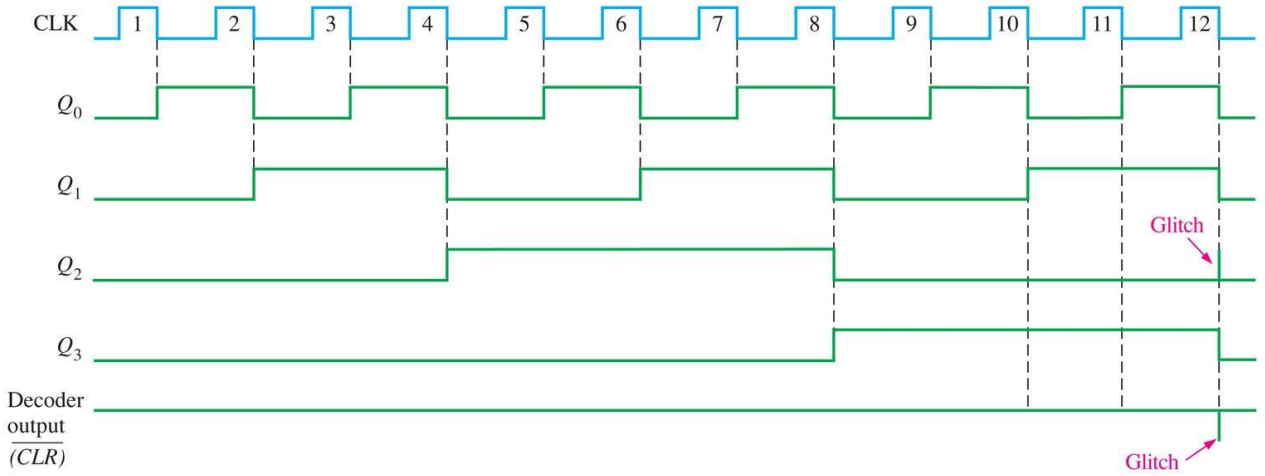
This is the timing diagram for the counter in the previous slide. When $Q_1$ and $Q_3$ are HIGH together, the counter is cleared by a "glitch" on the $\overline{CLR}$ line.



# 12-Decoder



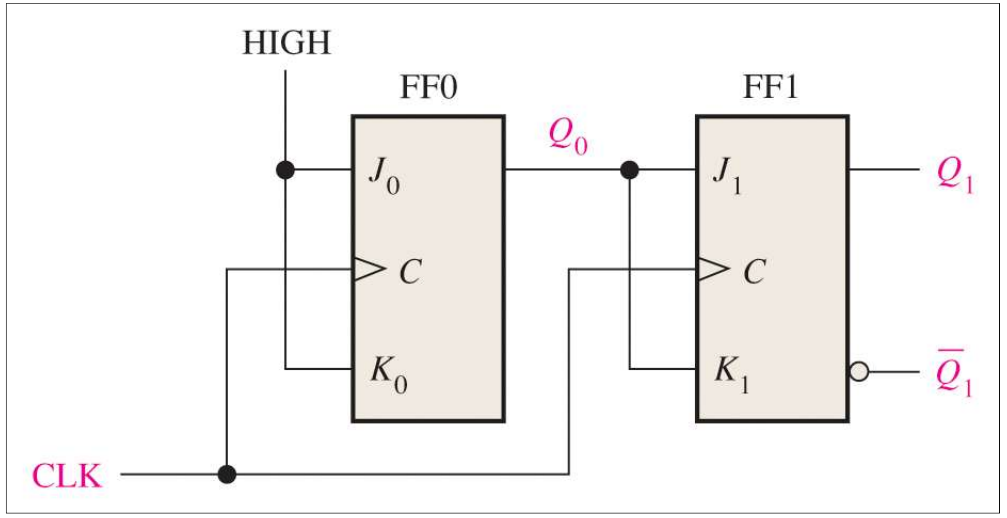

(a)




(b)

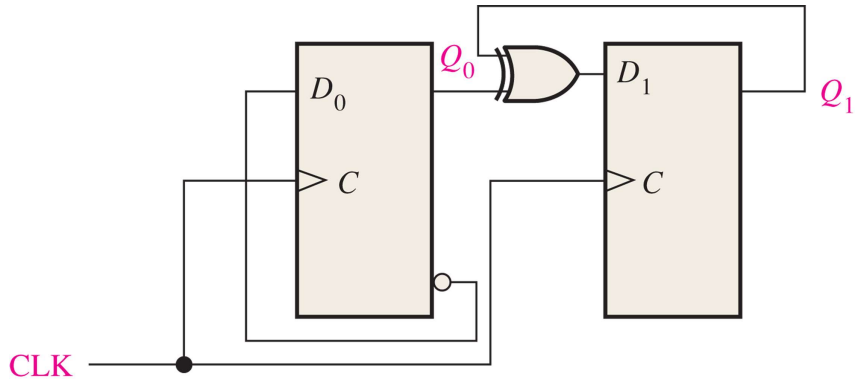
# A 2-bit Synchronous Counter

All of the flip-flops in a **synchronous counter** are clocked together with a common clock pulse. Synchronous counters overcome the disadvantage of accumulated propagation delays, but generally they require more circuitry to control states changes.



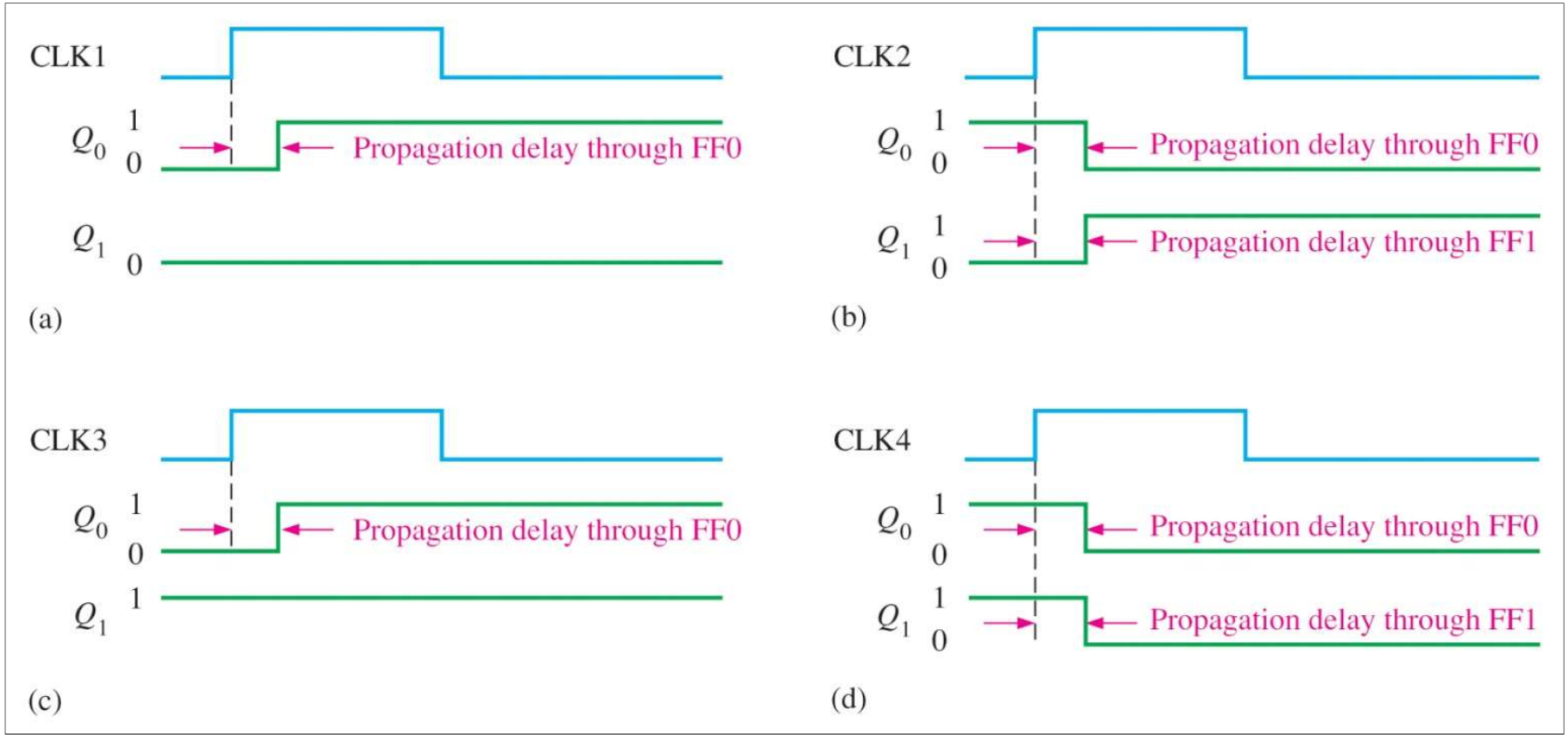
# A 2-bit Synchronous Counter

All of the flip-flops in a **synchronous counter** are clocked together with a common clock pulse. Synchronous counters overcome the disadvantage of accumulated propagation delays, but generally they require more circuitry to control states changes.

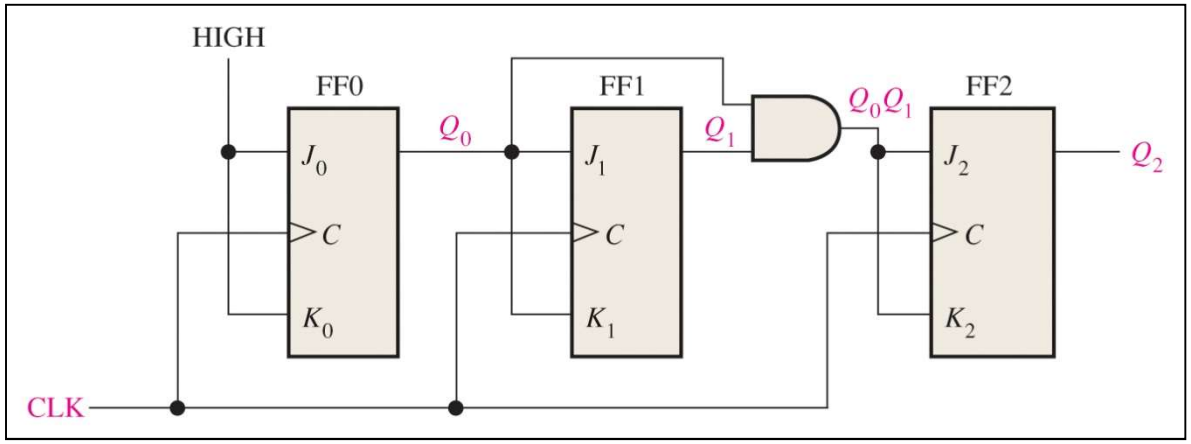



(b) D flip-flop

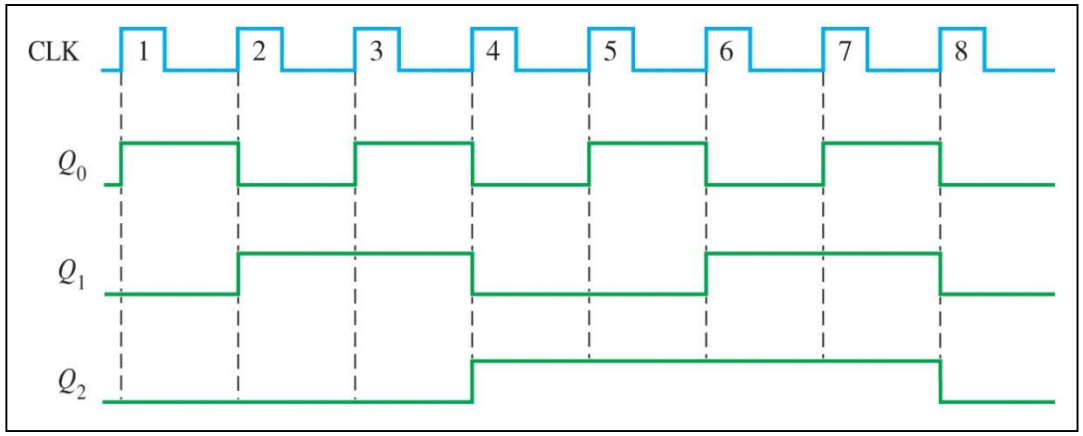
# 2-bit Synchronous Counter Timing Diagrams



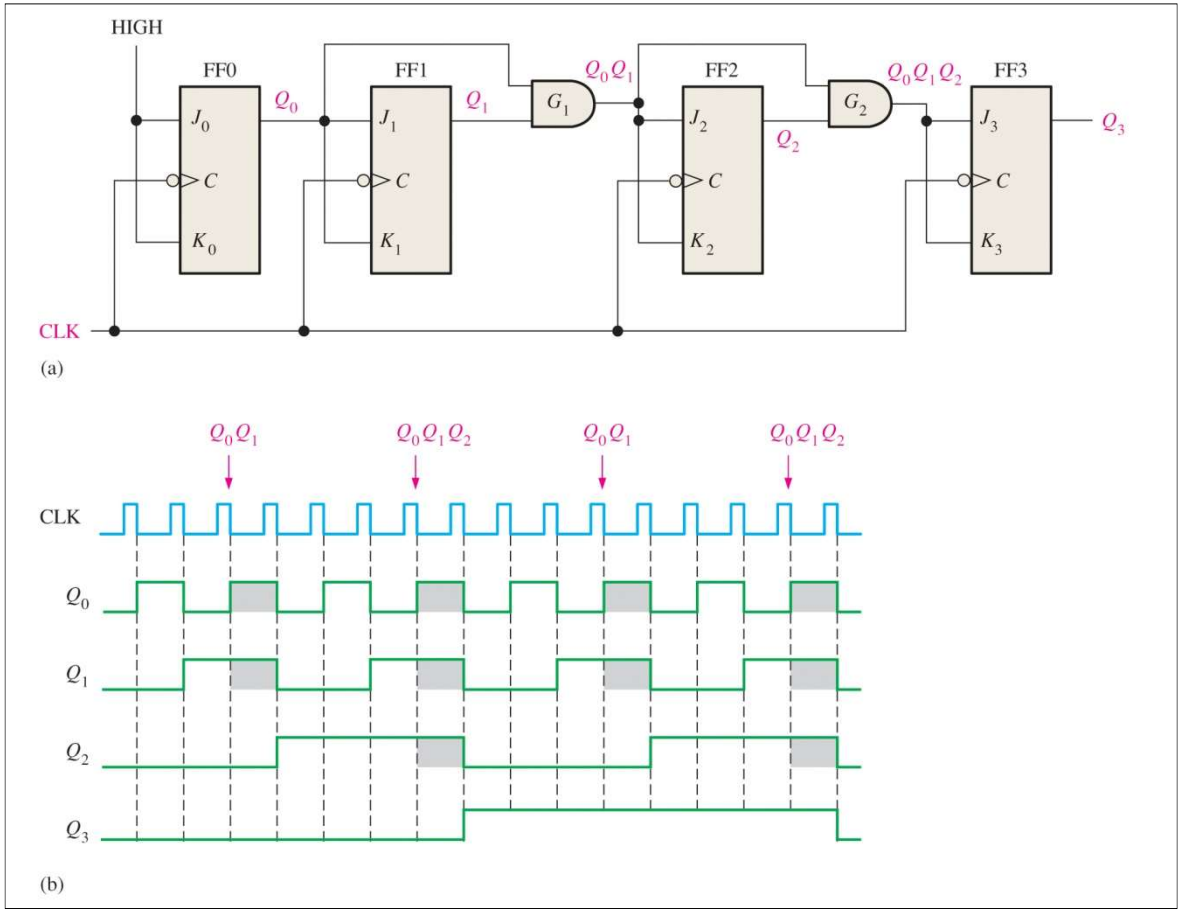
# A 3-bit Synchronous Binary Counter




The 3-bit binary counter has an AND gate, unlike the 2-bit counter just described. The output from the AND gate provides the J and K inputs to FF2.

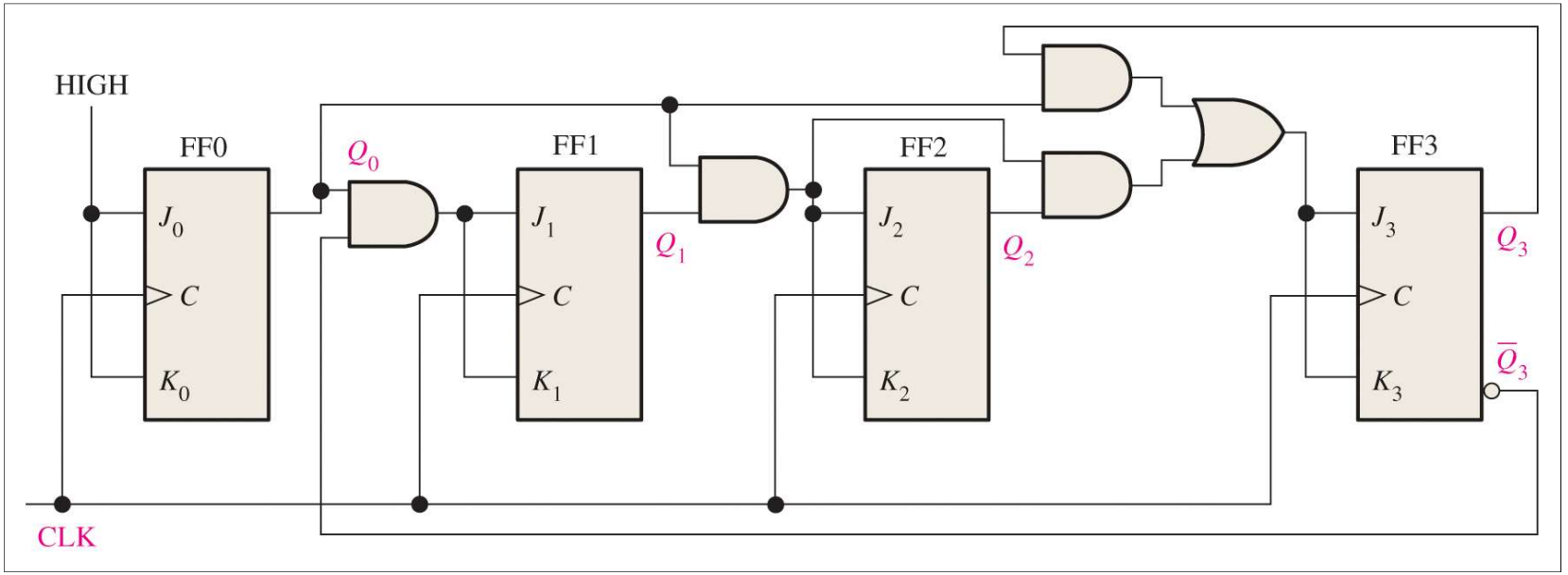


# A 4-bit Synchronous Binary Counter



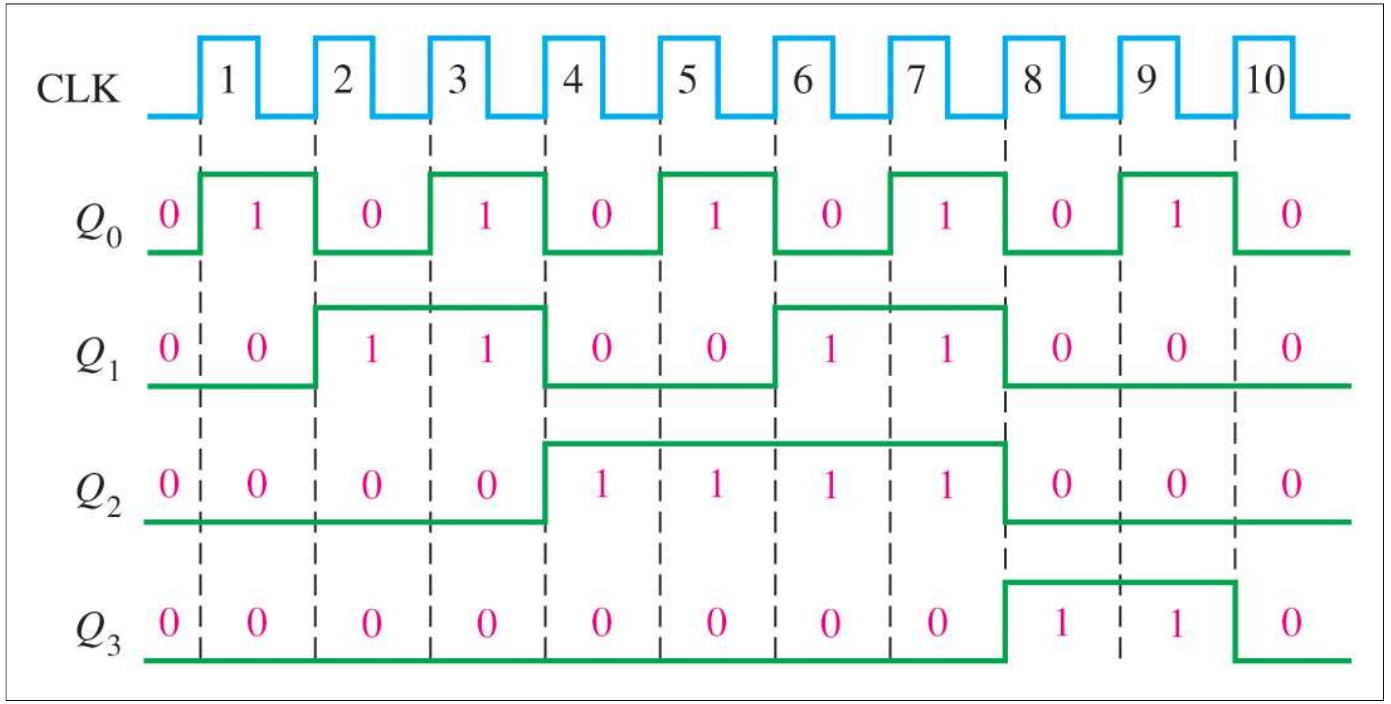
# A BCD Decade Counter

The circuit shown is a 4-bit BCD decade counter. After reaching the count 1001, the counter recycles to 0000.



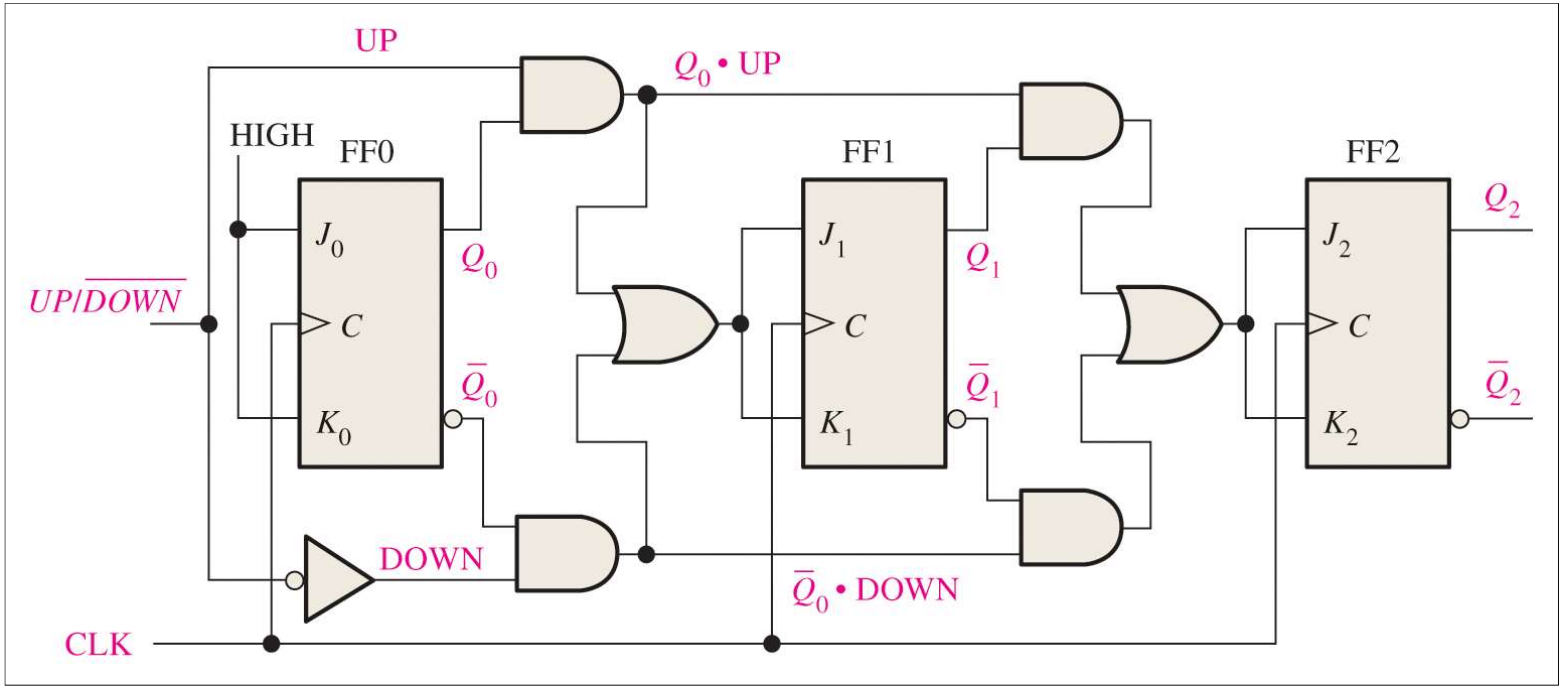
# BCD Decade Counter Waveforms

The timing diagram (below) is for the BCD decade counter shown on the previous slide.

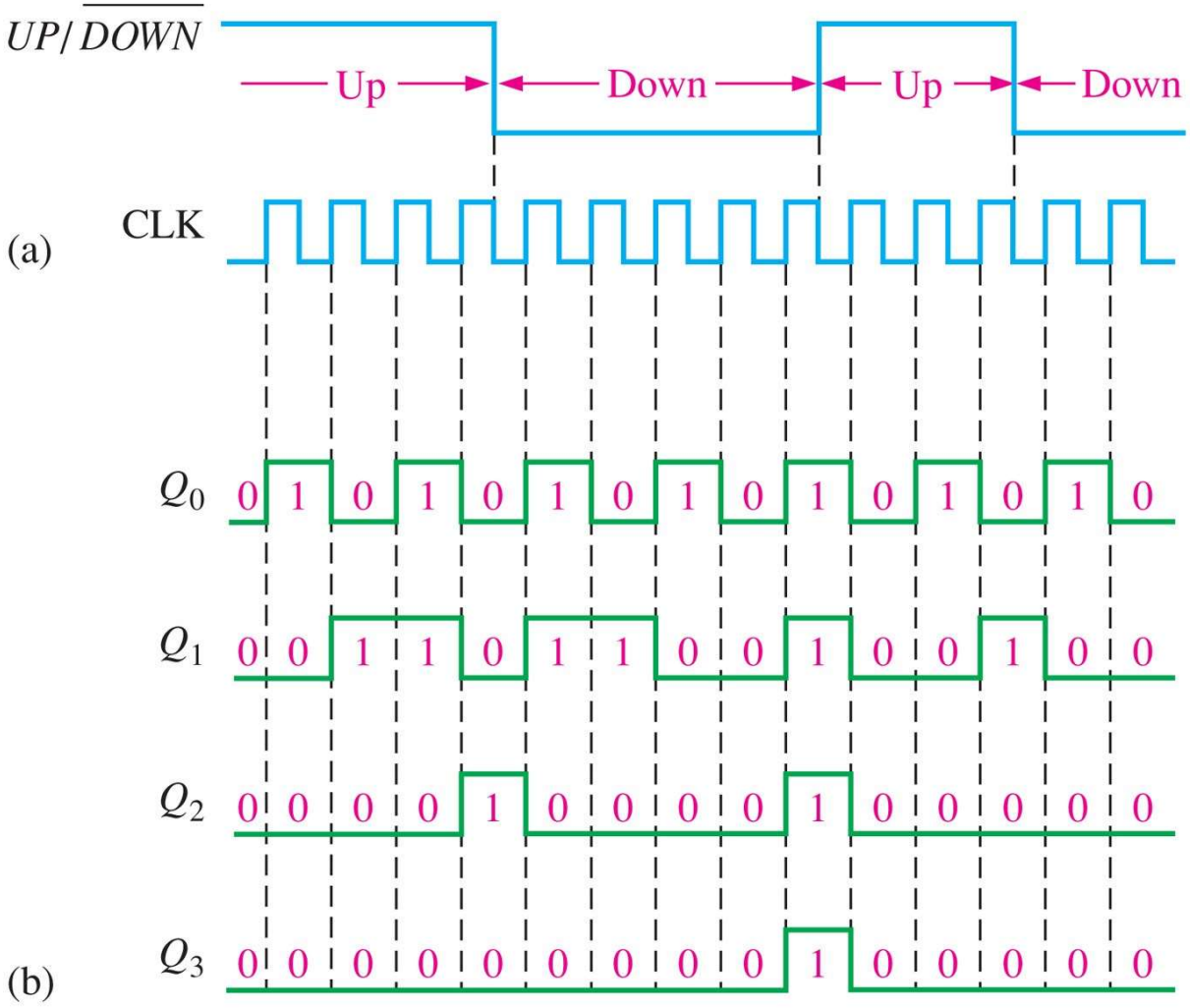


# Up/Down Synchronous Counters

An up/down counter is capable of progressing in either direction depending on a control input.



# Up/Down Synchronous Counters



# Design of Synchronous Counters

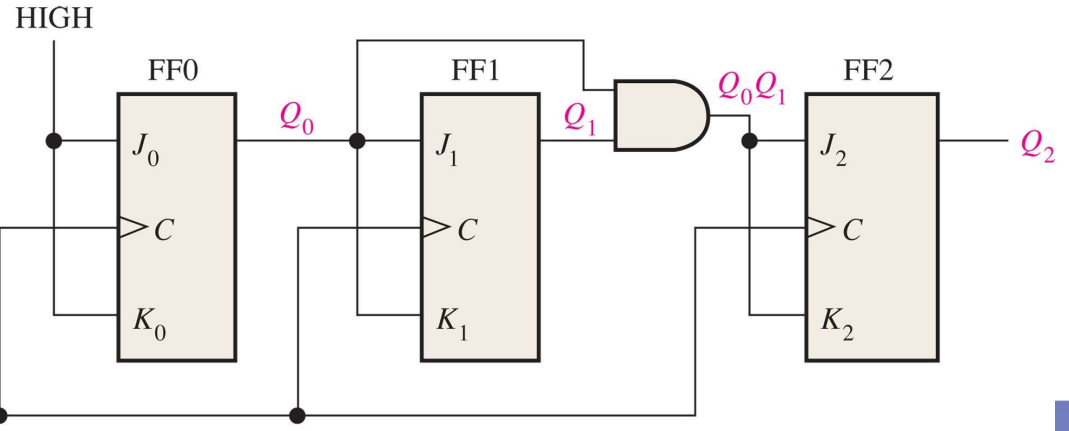


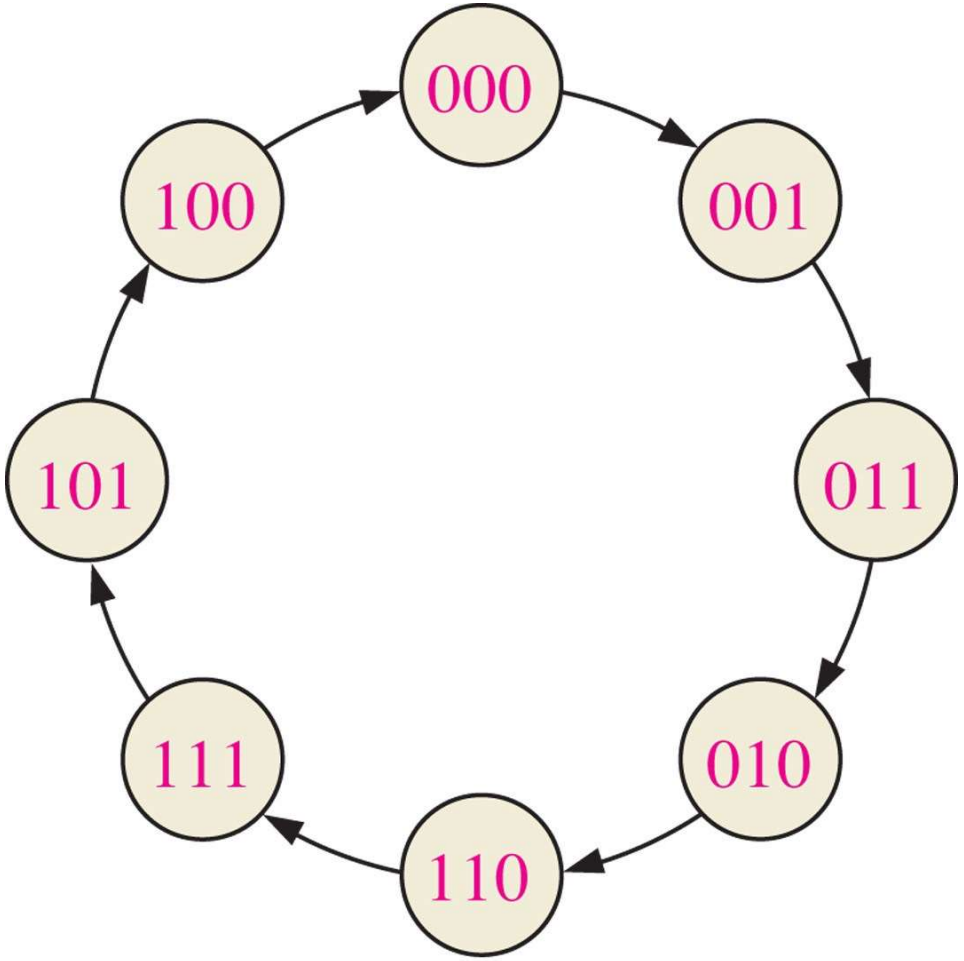

**TABLE 9–4**

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

| 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*.

# Step 1: State Diagram



# Step 2: Next-State Table

**TABLE 9–8**

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

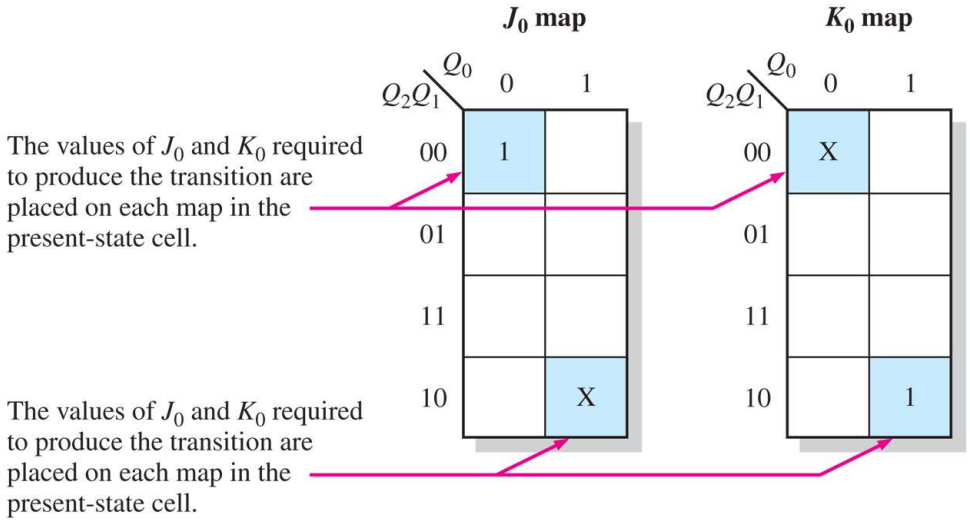
**TABLE 9–9**

Transition table for a J-K flip-flop.

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

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

# Step 4: Karnaugh Maps




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

Flip-flop transition table

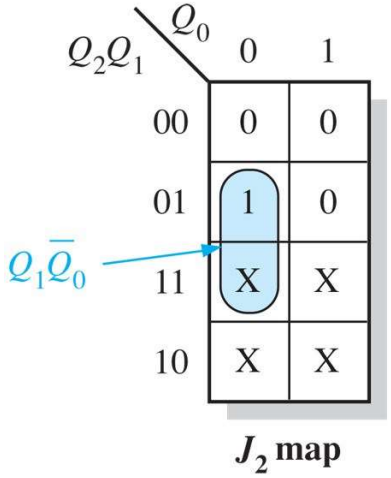
| Present State | | | Next State | | |
|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_2$ | $Q_1$ | $Q_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1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

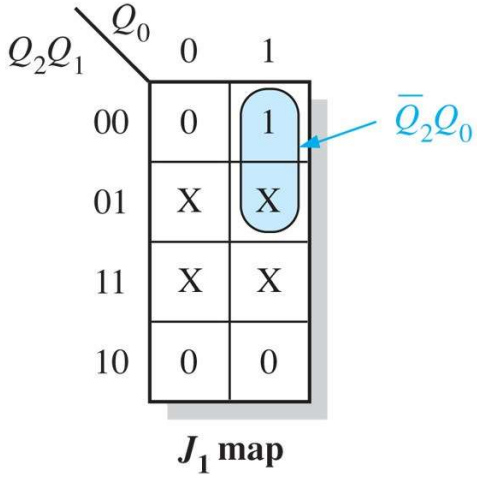
For the present state 000, $Q_0$ makes a transition from 0 to 1 to the next state.

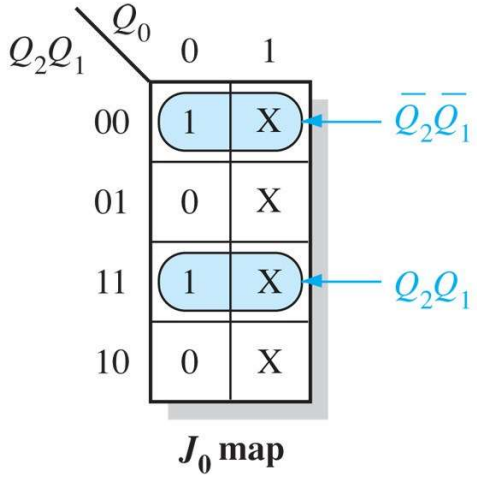
For the present state 101, $Q_0$ makes a transition from 1 to 0 to the next state.

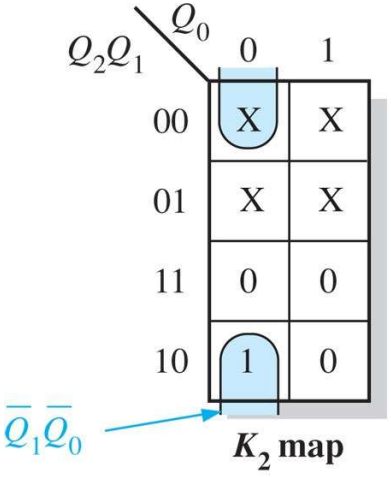
Next-state table

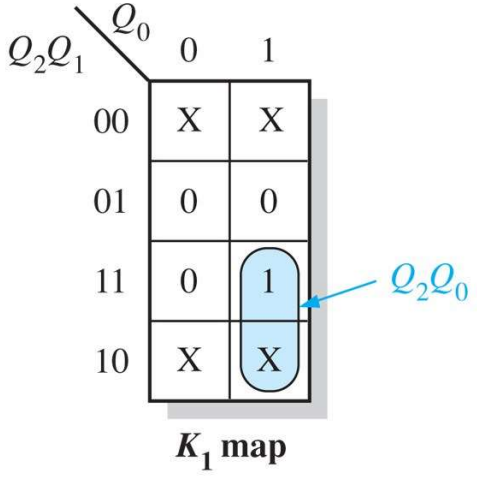
# Step 4: Karnaugh Maps

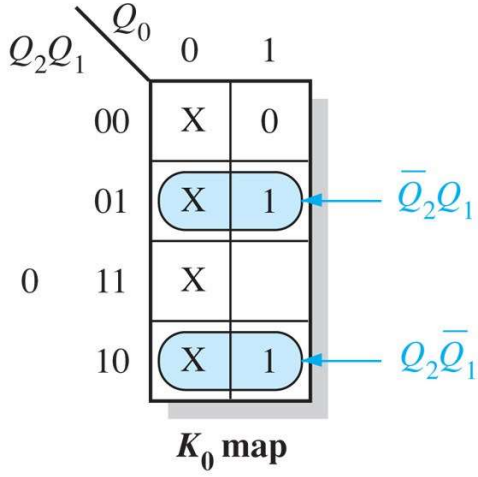


# Step 5: Logic Expressions

$$J_0 = Q_2\,Q_1 + \bar{Q}_2\bar{Q}_1$$
$$K_0 = Q_2\,\bar{Q}_1 + \bar{Q}_2 Q_1$$
$$J_1 = \bar{Q}_2 Q_0$$
$$K_1 = Q_2\,Q_0$$
$$J_2 = Q_1\bar{Q}_0$$
$$K_2 = \bar{Q}_1\bar{Q}_0$$

# Step 6: Counter Implementation

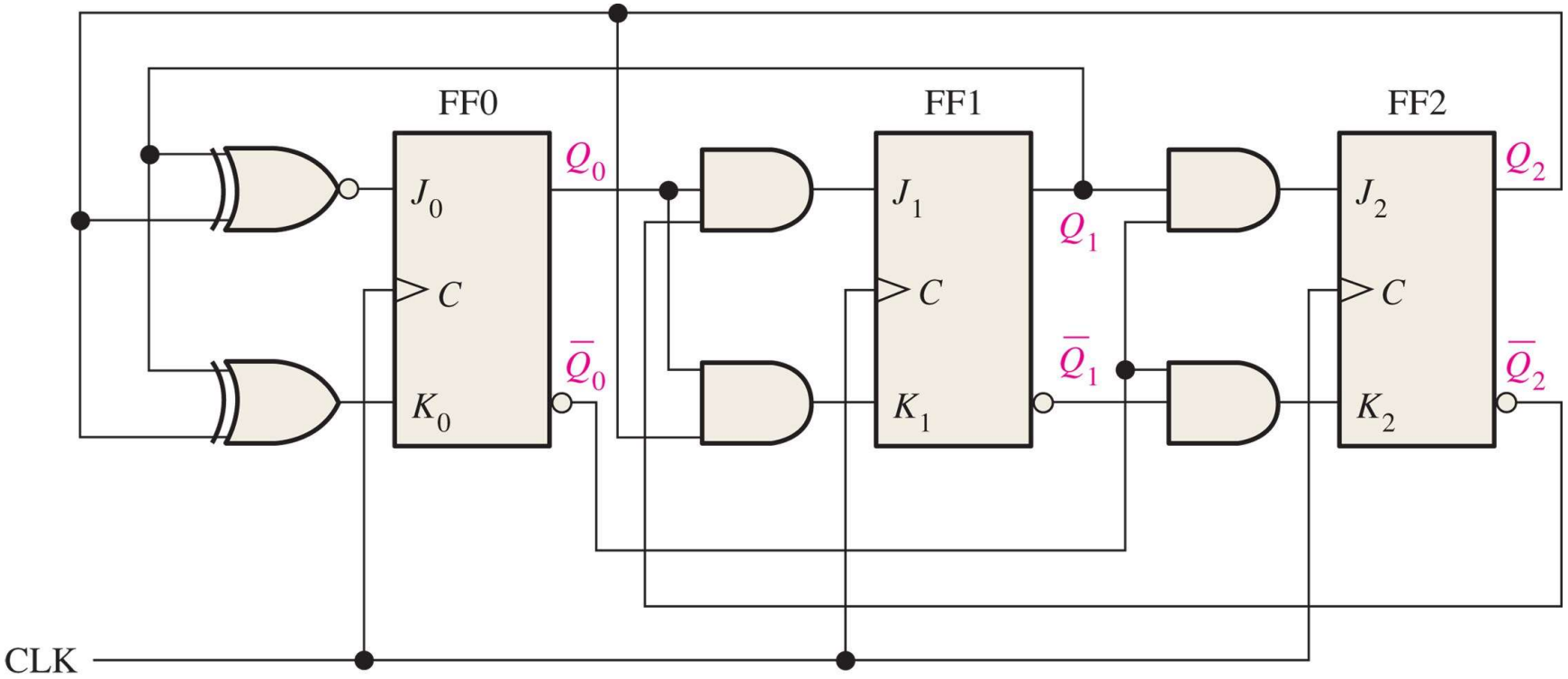
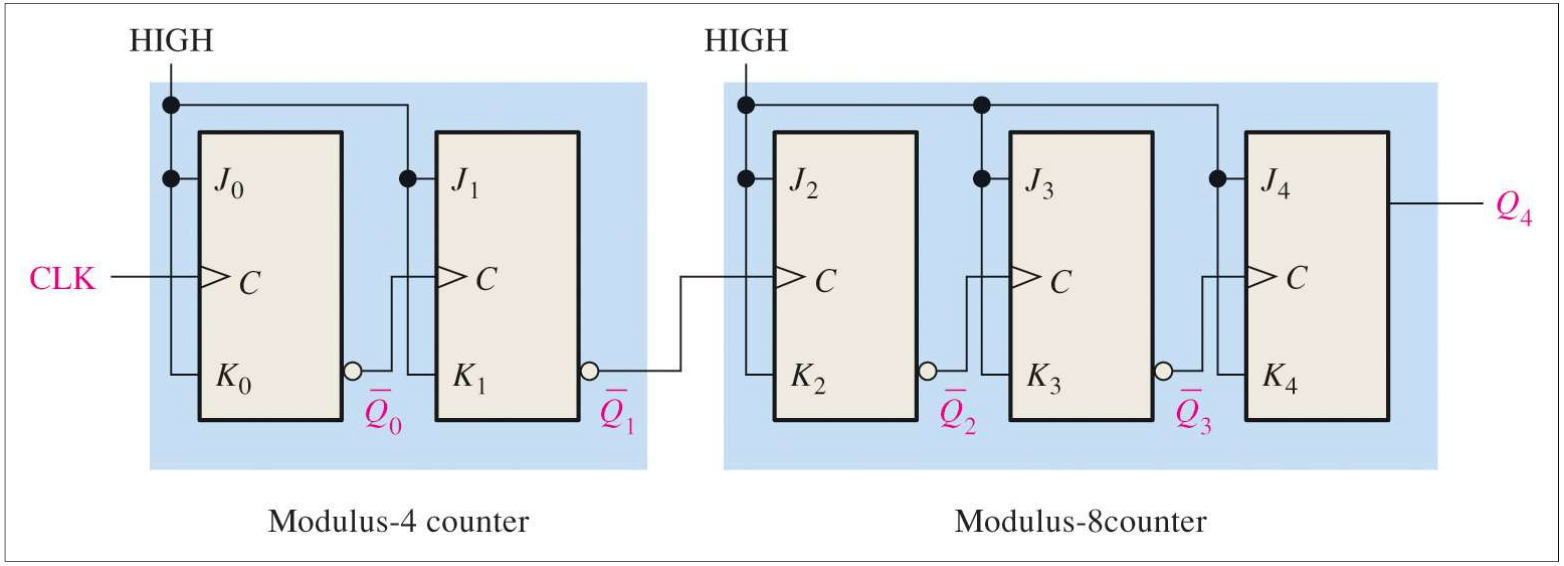



**FIGURE 9-29** Three-bit Gray code counter.

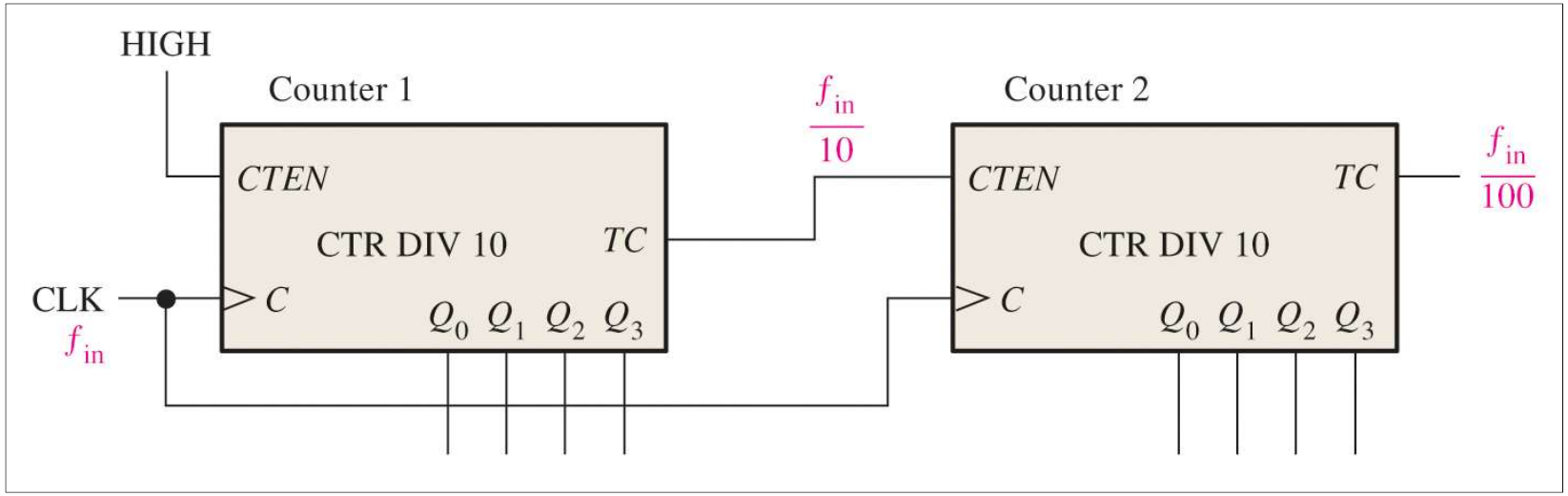
# Cascaded Counters

Counters can be cascaded to produce various divide-by (or *modulus*) values. In this case, a modulus-4 counter is cascaded with a modulus-8 counter. The resulting counter modulus equals the product of the two counters: Modulus 32.

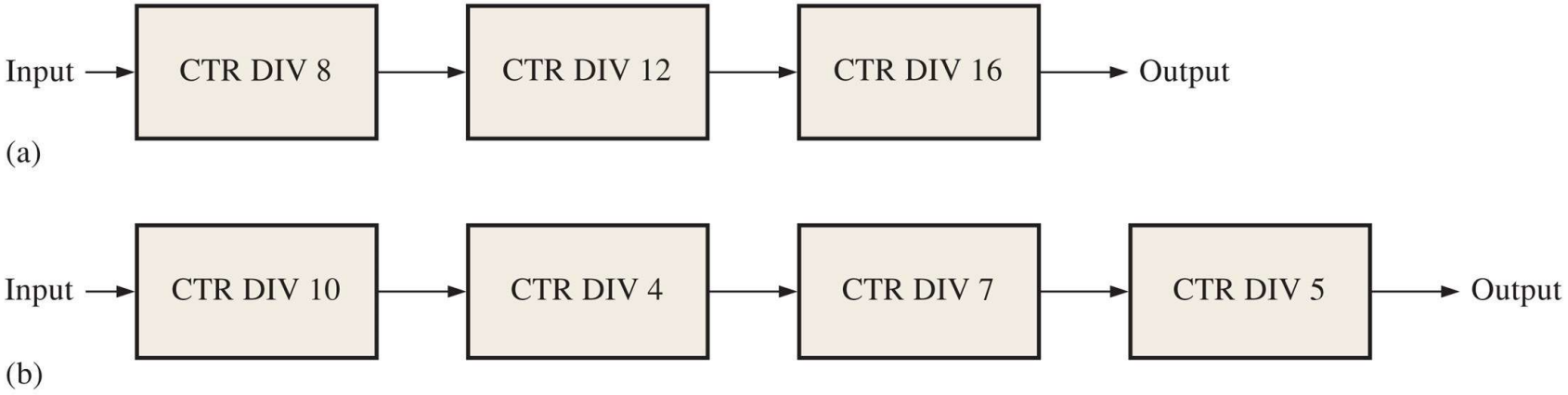


# Frequency Division Using A Modulus-100 Counter

The circuit below contains two cascaded Modulus-10 counters; forming a Modulus-100 counter. The output frequency from the circuit is 1 one-hundredth of the input frequency ( $f_{in}$ / 100 ).



# Cascade: Example



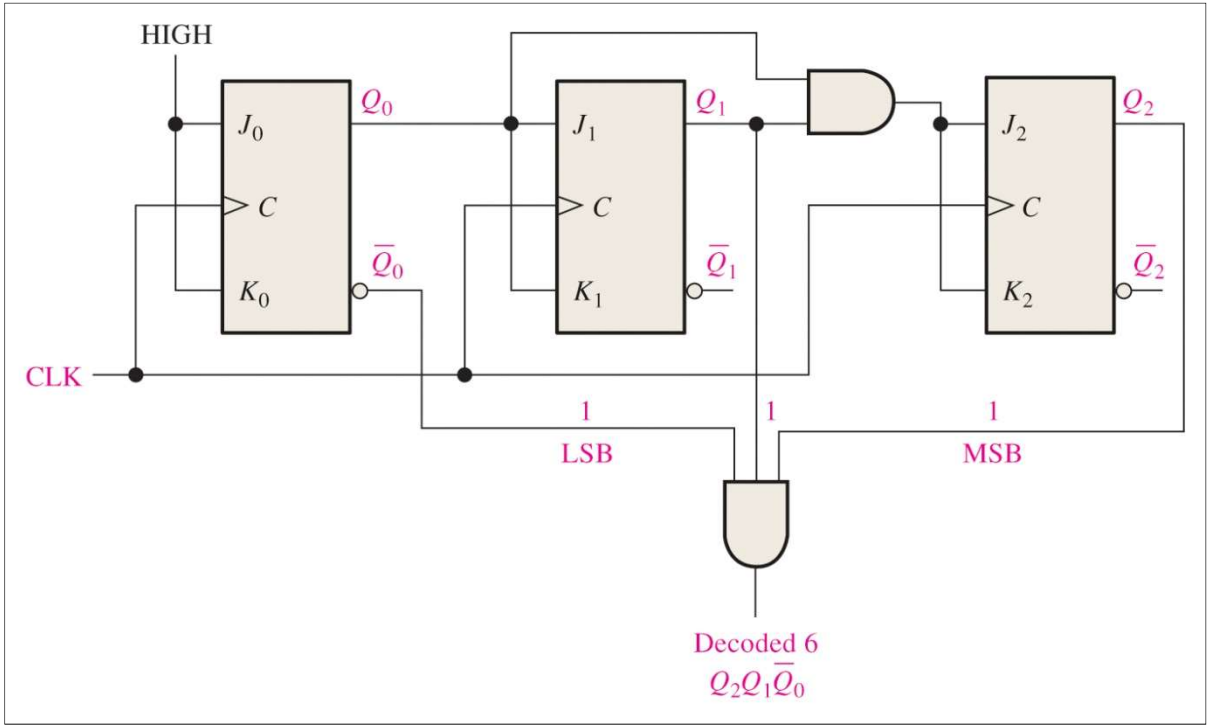

$$8 \times 12 \times 16 = 1536$$

$$10 \times 4 \times 7 \times 5 = 1400$$

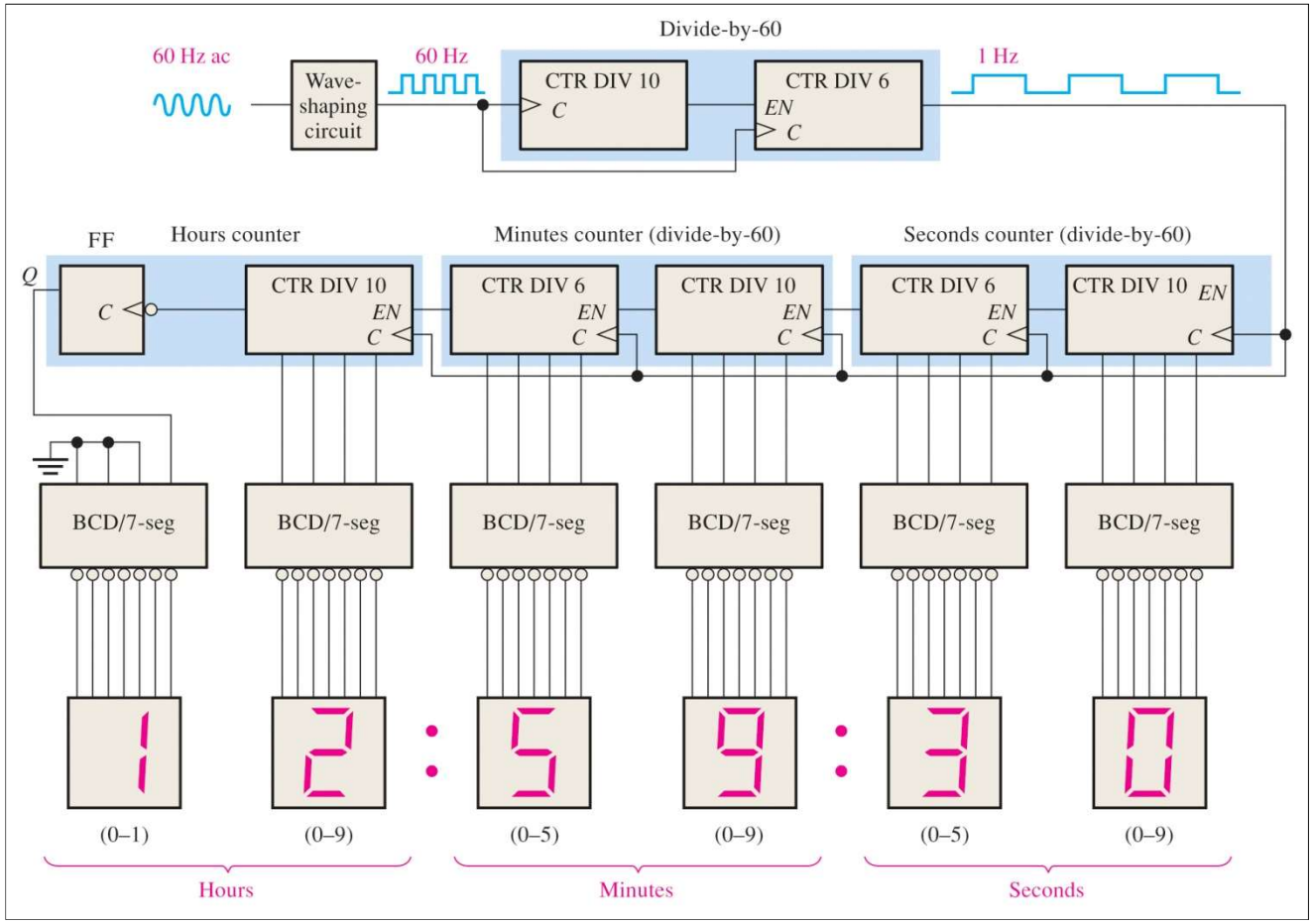
# Counter Decoding

Decoding is the detection of a binary number and can be done with an AND gate.

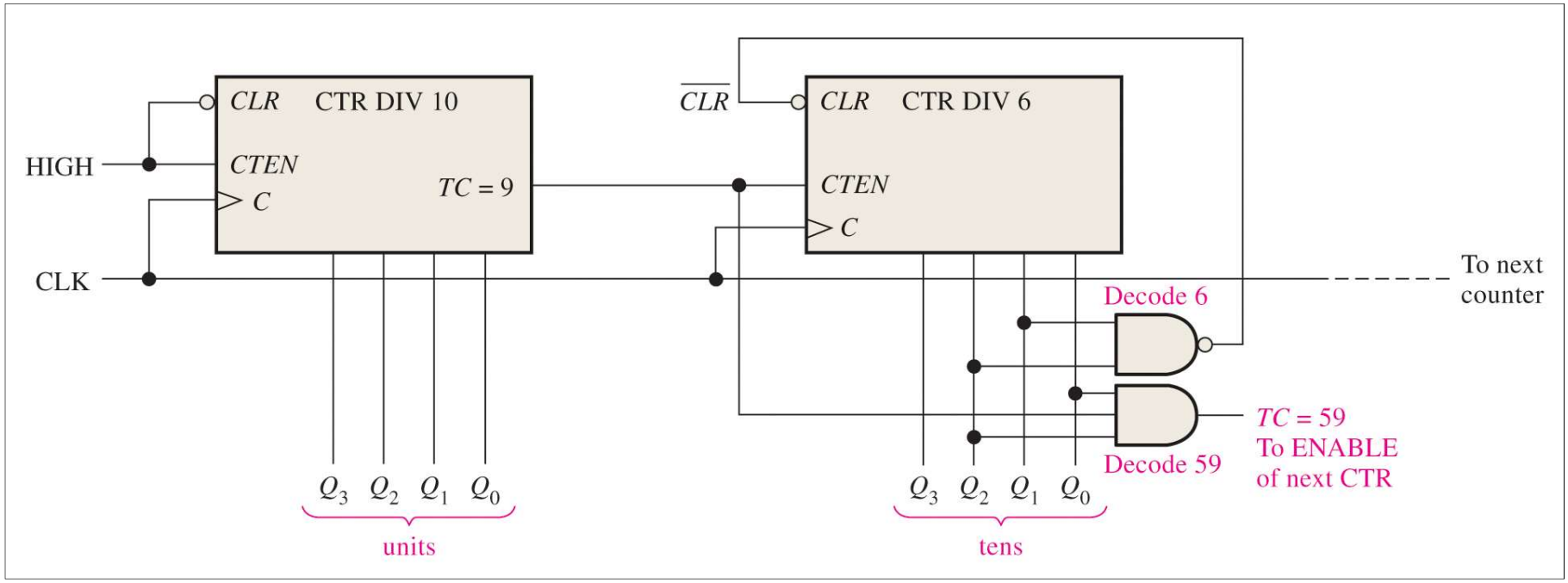
The output from the AND gate in the circuit shown goes high when the counter output equals six (110).



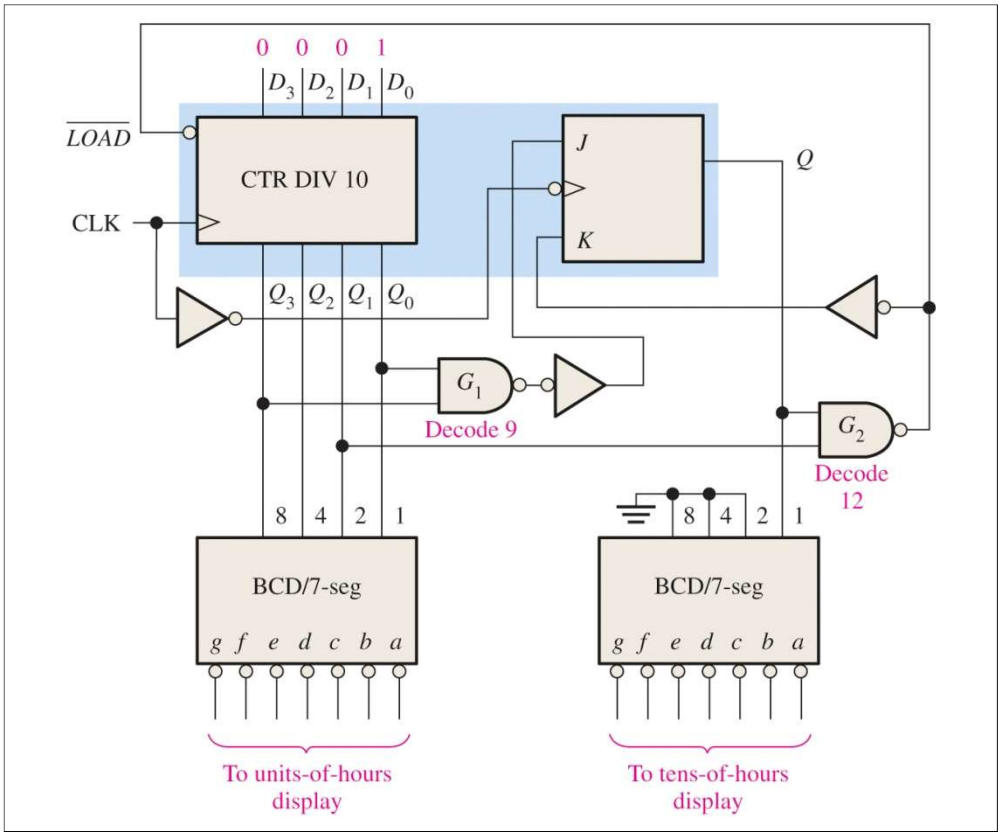
# A System: Digital Clock



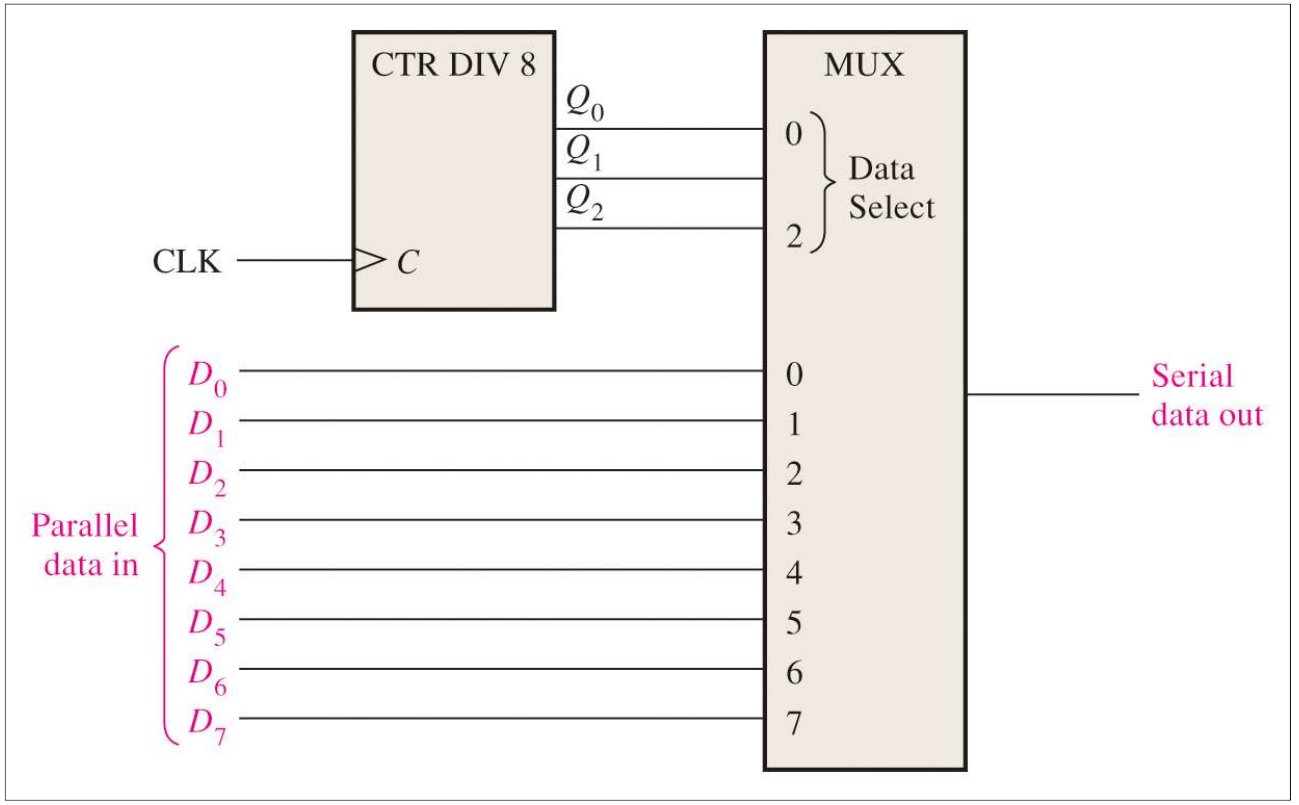
# Digital Clock: Counter Logic Diagram (Minutes and Seconds)



# Digital Clock: Hours Counter & Decoders



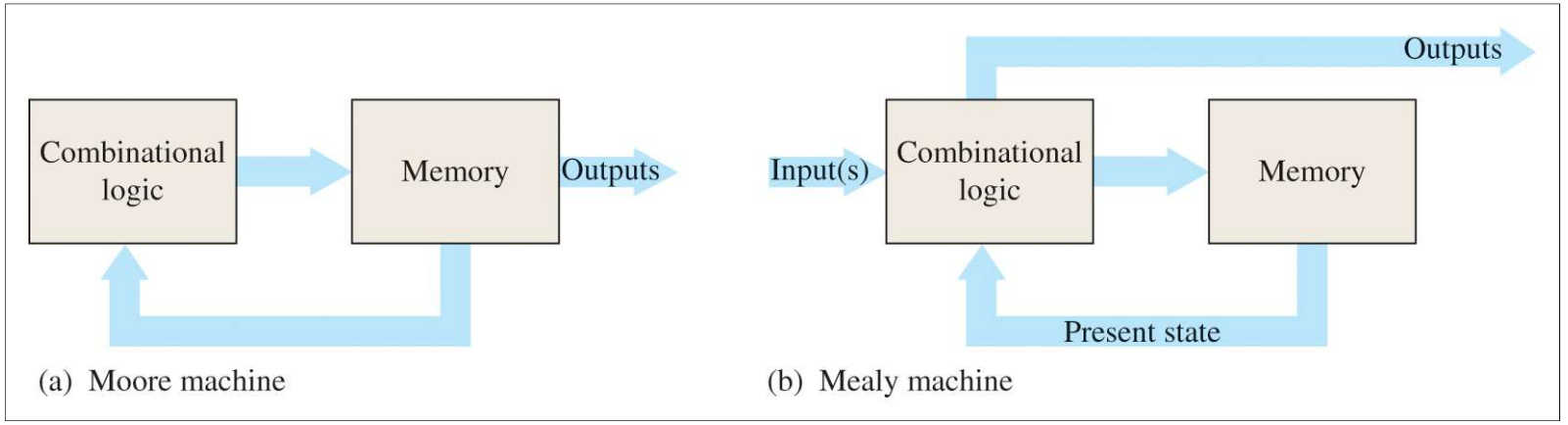
# Parallel-to-Serial Conversion (Multiplexing)



# Finite State Machines

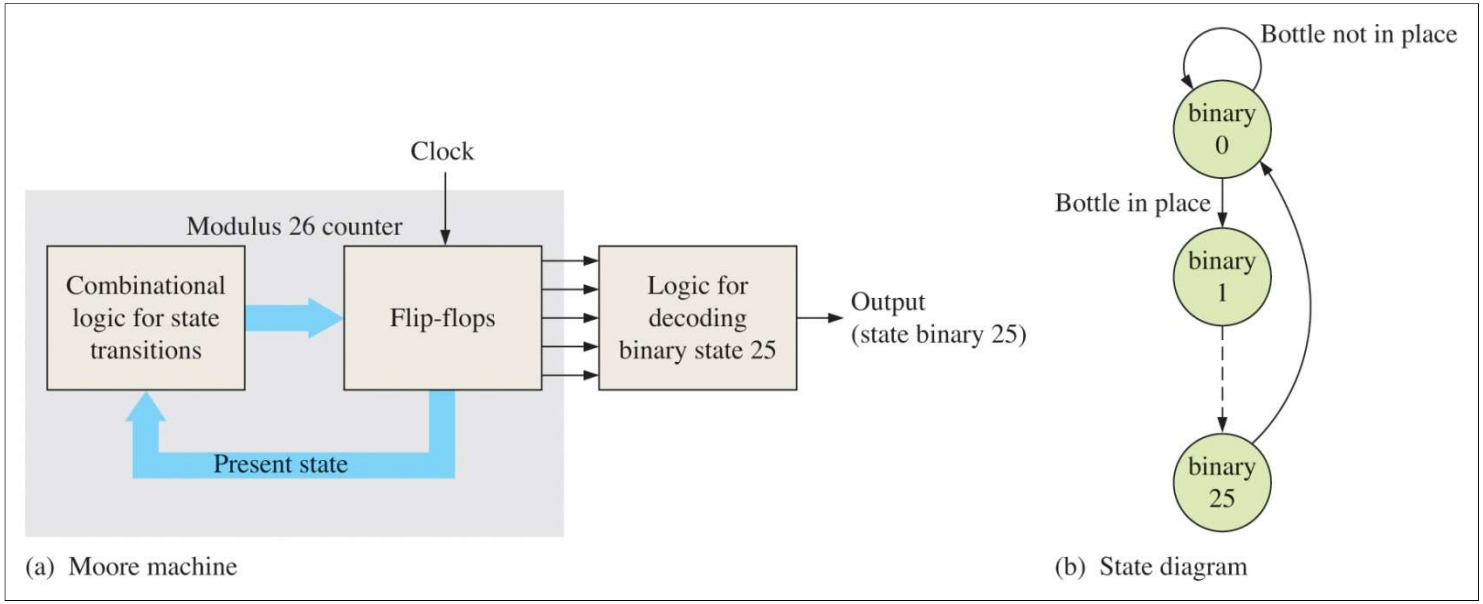
**Moore machine:** One whose outputs depends only on its internal present state.

**Mealy machine:** One whose outputs depends on both its internal present state and its inputs.



# A Moore Machine

The **Moore machine** (below) controls the number of tablets that go into each bottle. It counts out 25 tablets, then resets, stopping the clock until the next bottle is in place.



# A Mealy Machine

The **Mealy machine** (below) controls the number of tablets that go into bottles of various sizes. Its operation depends on the bottle size (external) and the number of tablets remaining in the count (internal).