Lecture 13

CHAPTER 6

RIPPLE COUNTERS

- A register that goes through a prescribed sequence of states upon the application of input pulses is called a *counter* .
- The input pulses may be clock pulses, or they may originate from some external source and may occur at a fixed interval of time or at random.
- The sequence of states may follow the binary number sequence or any other sequence of states.
- A counter that follows the binary number sequence is called a *binary counter* .
- An n -bit binary counter consists of n flip-flops and can count in binary from 0 through $2^n - 1$.

RIPPLE COUNTERS

- Counters are available in two categories: ripple counters and synchronous counters.
- In a ripple counter, a flip-flop output transition serves as a source for triggering other flip-flops.
- In other words, the *C* input of some or all flip-flops are triggered, not by the common clock pulses, but rather by the transition that occurs in other flip-flop outputs.
- In a synchronous counter, the *C* inputs of all flip-flops receive the common clock.

Binary Ripple Counter

- A binary ripple counter consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the *C* input of the next higher order flip-flop.
- The flip-flop holding the least significant bit receives the incoming count pulses.
- A complementing flip-flop can be obtained from a *JK* flip-flop with the *J* and *K* inputs tied together or from a *T* flip-flop.
- A third possibility is to use a *D* flip-flop with the complement output connected to the *D* input.
- The logic diagram of two 4-bit binary ripple counters is shown in Fig. 6.8 .

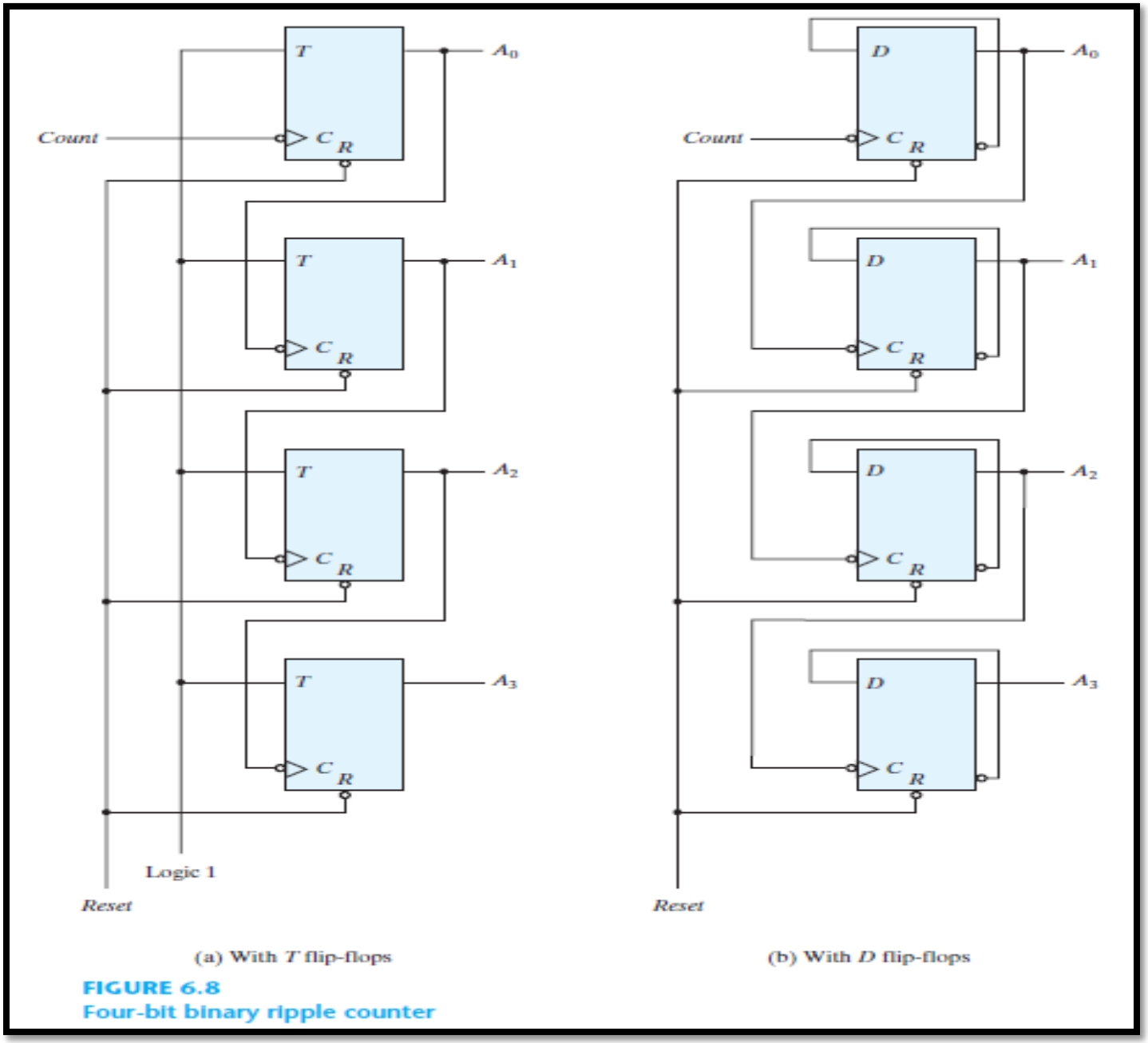


FIGURE 6.8
Four-bit binary ripple counter

Binary Ripple Counter

- The counter is constructed with complementing flip-flops of the *T* type in part (a) and *D* type in part (b).
- The output of each flip-flop is connected to the *C* input of the next flip-flop in sequence.
- The flip-flop holding the least significant bit receives the incoming count pulses.
- The *T* inputs of all the flip-flops in (a) are connected to a permanent logic 1, making each flipflop complement if the signal in its *C* input goes through a negative transition.
- The bubble in front of the dynamic indicator symbol next to *C* indicates that the flip-flops respond to the negative-edge transition of the input.
- The negative transition occurs from 1 to 0.

Binary Ripple Counter

- To understand the operation of the four-bit binary ripple counter, refer to the first nine binary numbers listed in Table.
- The count starts with binary 0 and increments by 1 with each count pulse input.
- After the count of 15, the counter goes back to 0 to repeat the count.
- The least significant bit, *A*0, is complemented with each count pulse input.
- Every time that *A*0 goes from 1 to 0, it complements *A*1.
- Every time that *A*1 goes from 1 to 0, it complements *A*2. Every time that *A*2 goes from 1 to 0, it complements *A*3, and so on for any other higher order bits of a ripple counter.

Binary Ripple Counter

Table 6.4
Binary Count Sequence

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

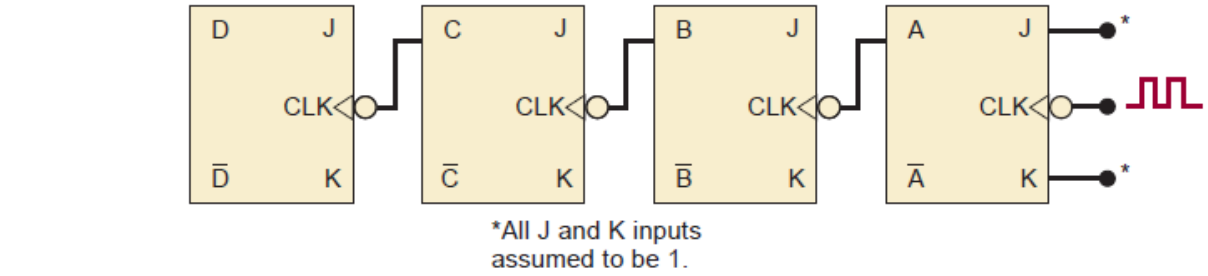
Binary Ripple Counter

- 1. The clock pulses are applied only to the *CLK* input of flip-flop *A*.Thus, flipflop *A* will toggle (change to its opposite state) each time the clock pulses make a negative (HIGH-to-LOW) transition. Note that *JK*1 for all FFs.
- 2. The normal output of flip-flop *A* acts as the *CLK* input for flip-flop *B*, and so flip-flop *B* will toggle each time the *A* output goes from 1 to 0. Similarly, flip-flop *C* will toggle when *B* goes from 1 to 0, and flip-flop *D* will toggle when *C* goes from 1 to 0.

Binary Ripple Counter

- 3. FF outputs *D*, *C*, *B*, and *A* represent a four-bit binary number, with *D* as the MSB. Let's assume that all FFs have been cleared to the 0 state (CLEAR inputs are not shown). The waveforms in Figure 7-1 show that a binary counting sequence from 0000 to 1111 is followed as clock pulses are continuously applied.
- 4. After the NGT of the fifteenth clock pulse has occurred, the counter FFs are in the 1111 condition. On the sixteenth NGT, flip-flop *A* goes from 1 to 0, which causes flip-flop *B* to go from 1 to 0, and so on, until the counter is in the 0000 state. In other words, the counter has gone through one complete cycle (0000 through 1111) and has *recycled* back to 0000.
- From this point, it will begin a new counting cycle as subsequent clock pulses are applied.

Binary Ripple Counter



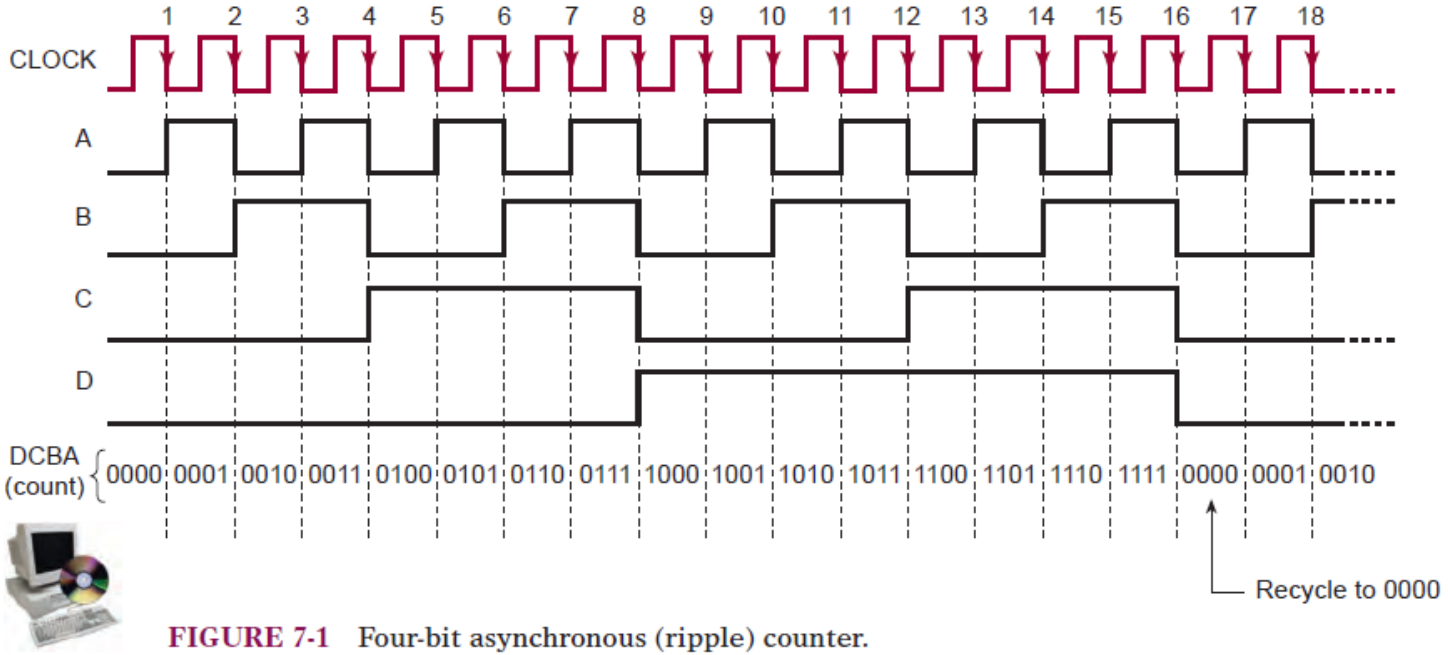
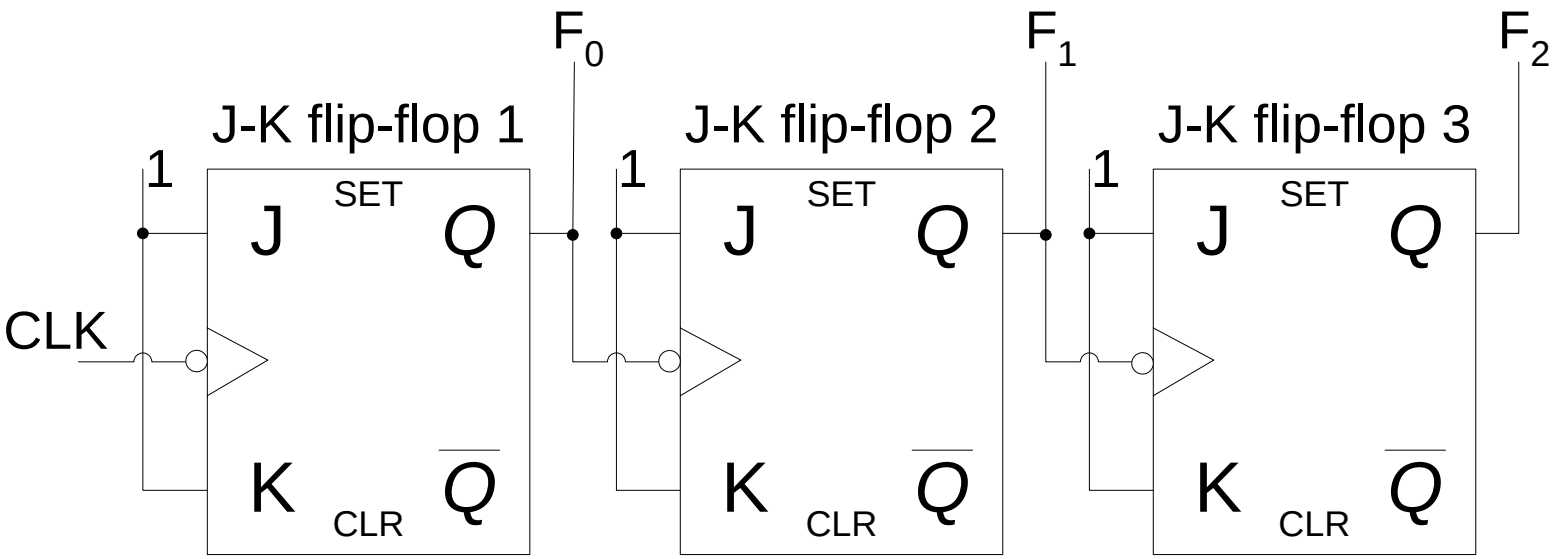
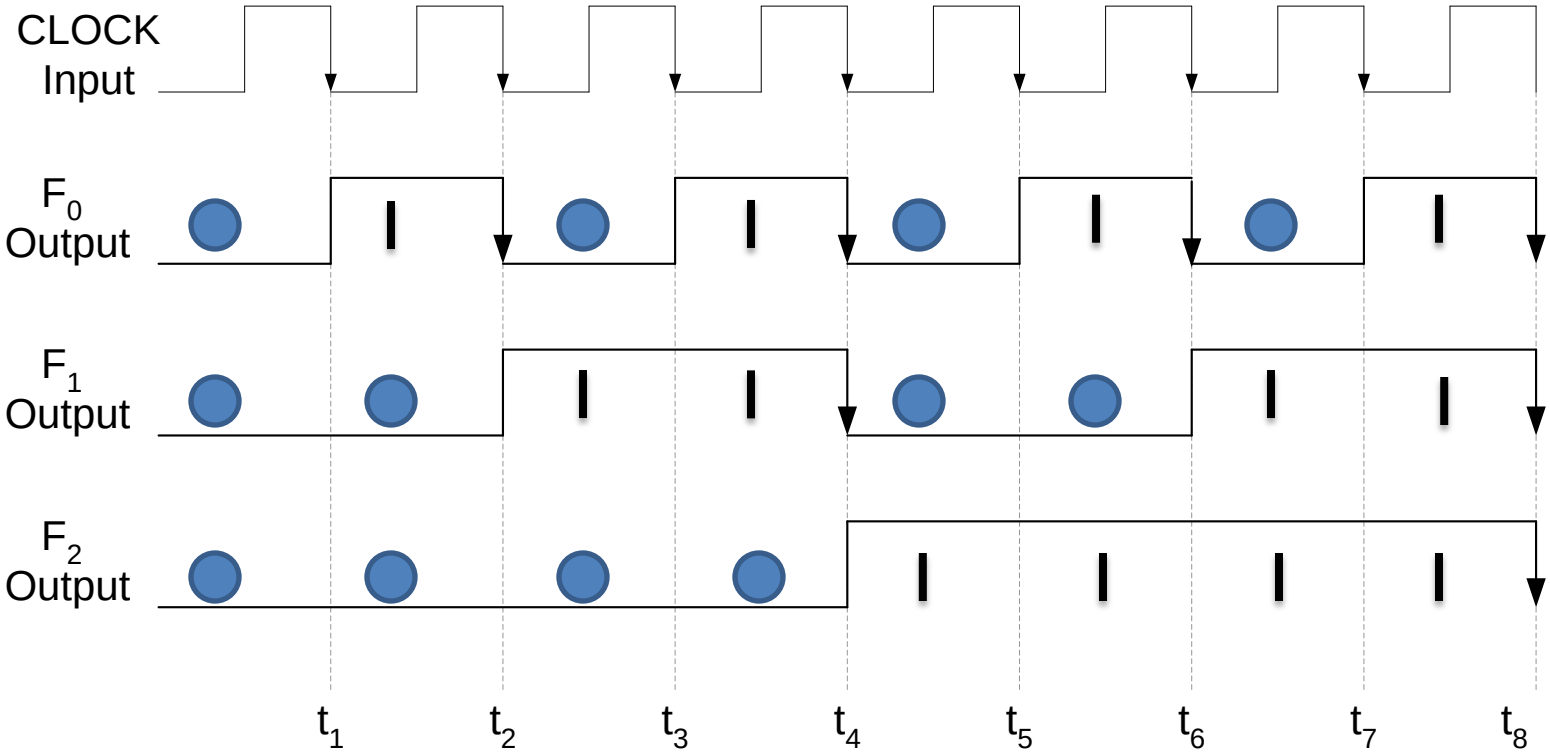


FIGURE 7-1 Four-bit asynchronous (ripple) counter.

3-bit Ripple Up-Counter



Timing Diagram of a 3-bit Ripple Up-Counter



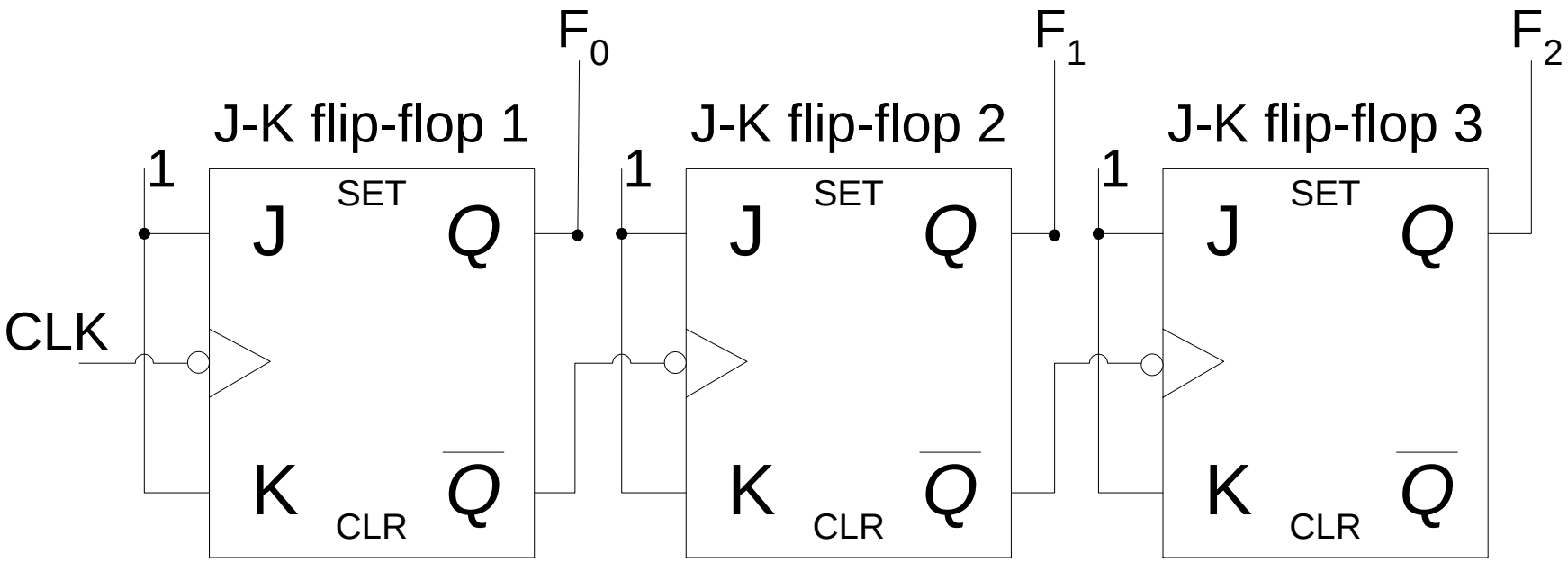
Output State of a 3-bit Ripple Up-Counter

Input	Output		
Clock Pulses	F_2	F_1	F_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Binary Ripple down Counter

- A binary counter with a reverse count is called a *binary countdown counter* .
- In a countdown counter, the binary count is decremented by 1 with every input count pulse.
- The count of a four-bit countdown counter starts from binary 15 and continues to binary counts 14, 13, 12, . . . , 0 and then back to 15.
- A list of the count sequence of a binary countdown counter shows that the least significant bit is complemented with every count pulse.
- Any other bit in the sequence is complemented if its previous least significant bit goes from 0 to 1.

3-bit Ripple Down-Counter



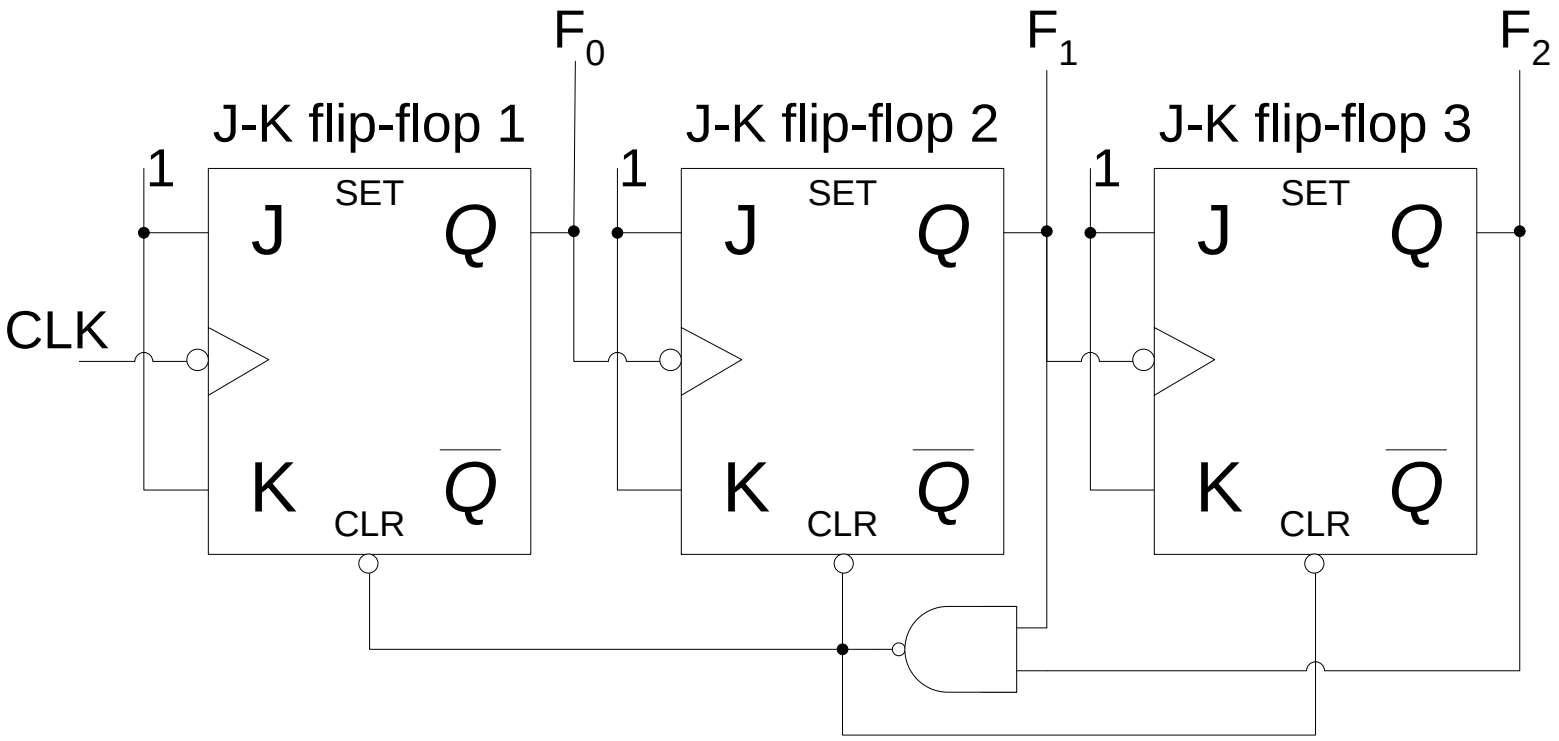
Mod-n Counters

- The term Mod represents the Modulus of the counter which is the total number of unique states through which the counter will sequence through.
- A 3-bit Asynchronous counter can count up from 0 to 7 or count down from 7 to 0.
- The 3-bit counter has 8 different states represented by the 8 outputs 0 to 7.
- The counter states or the range of numbers of a counter is determined by the formula 2^m. where m represents the number of flip-flops.
- Therefore, a Mod- 8 counter implemented using three flip-flops 2^3 has 8 output states.

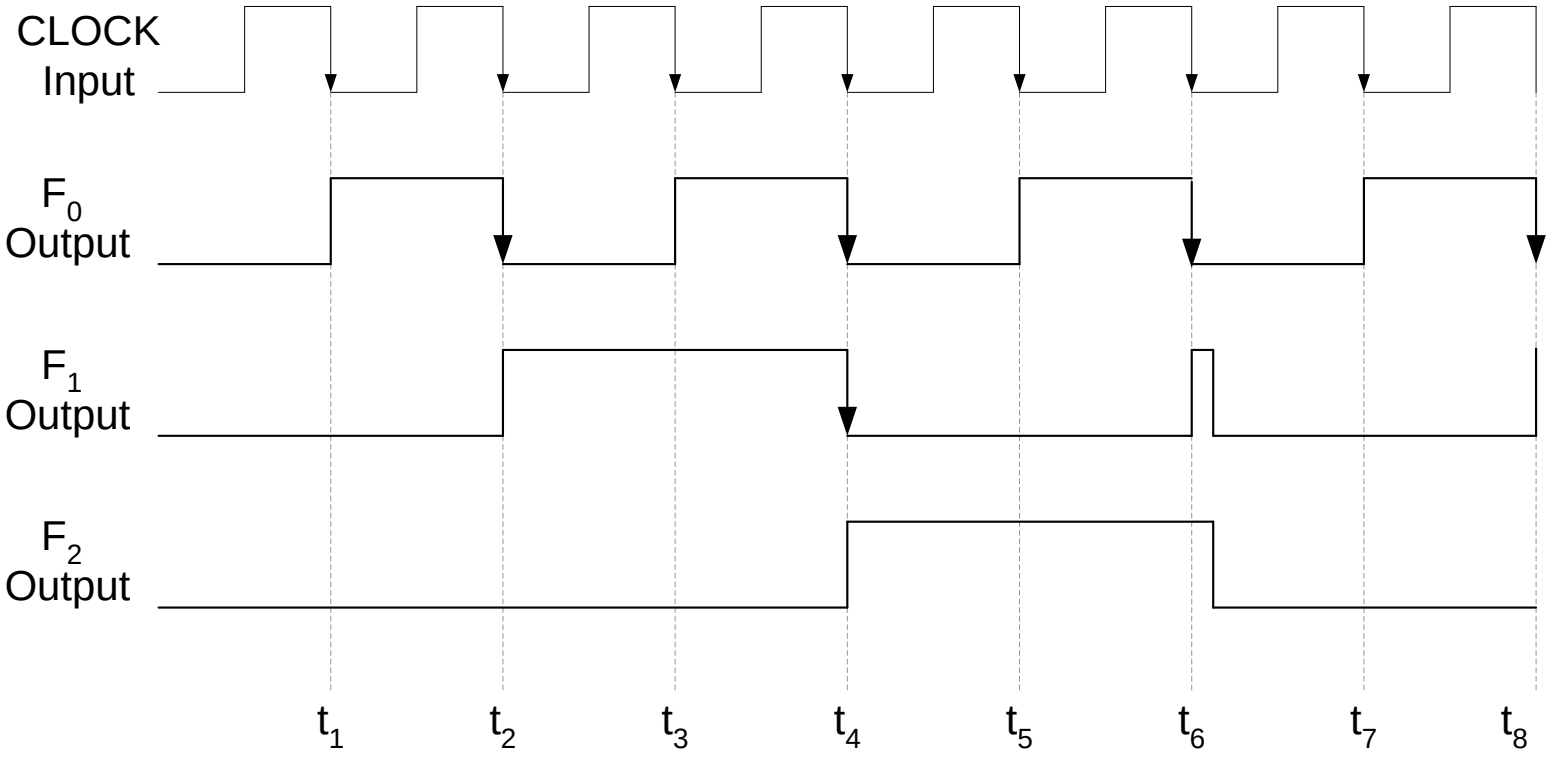
Mod-n Counters

- Counter can also be designed to have less number of states than 2^m.
- The resulting sequence is called a truncated sequence.
- The counter therefore counts up to the truncated sequence.
- Designing a truncated sequence counter is very simple.
- When the counter counts up to the intended sequence it is reset to the initial count value 0.
- The counter is reset to the initial count value by activating the Clear asynchronous inputs.
- The clears input is activated by the counter through a combinational circuit that activates its output when the appropriate count sequence is reached.

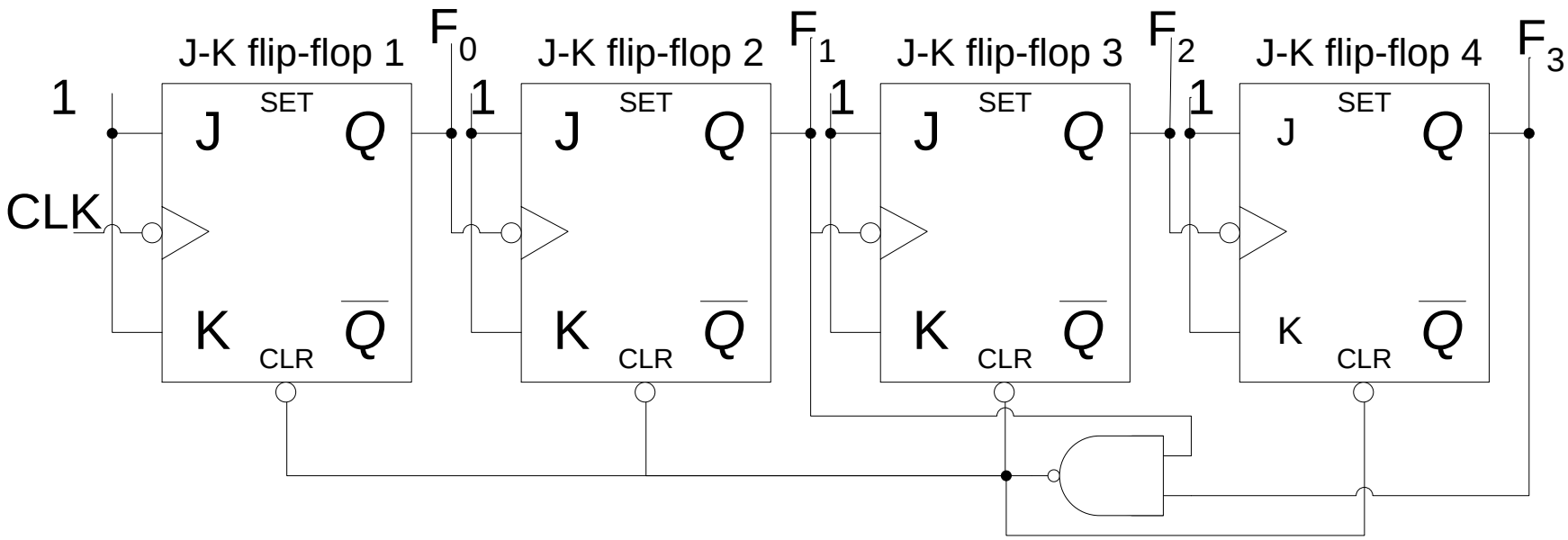
Mod-6 Counter



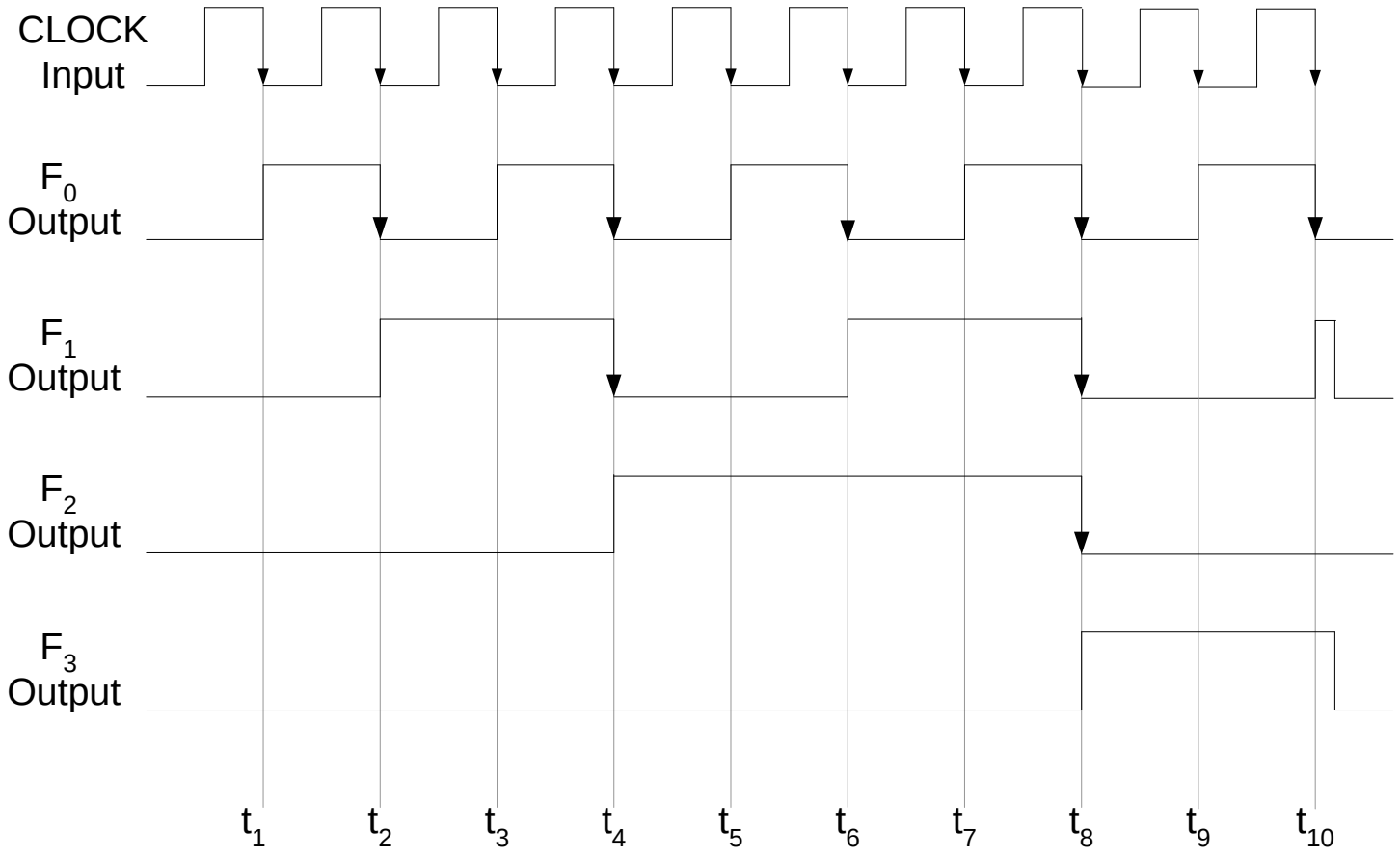
Timing diagram of a Mod-6 Counter



BCD Decade Counter/ MOD 10 Counter



Timing diagram of a Decode Counter



BCD Ripple Counter

- A decimal counter follows a sequence of 10 states and returns to 0 after the count of 9.
- Such a counter must have at least four flip-flops to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits.
- The sequence of states in a decimal counter is dictated by the binary code used to represent a decimal digit.
- If BCD is used, the sequence of states is as shown in the state diagram of Fig. 6.9 .
- A decimal counter is similar to a binary counter, except that the state after 1001 (the code for decimal digit 9) is 0000 (the code for decimal digit 0).

BCD Ripple Counter

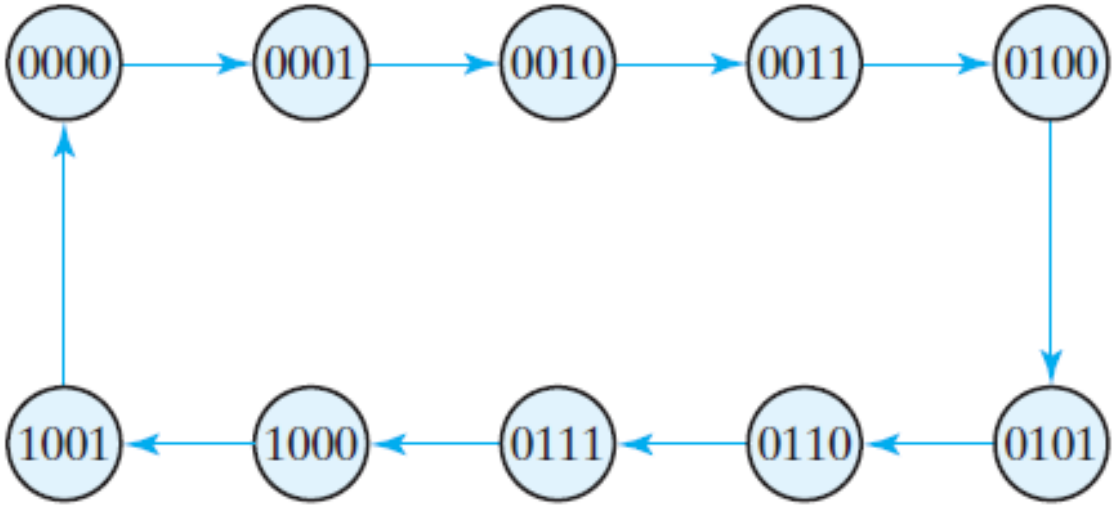


FIGURE 6.9
State diagram of a decimal BCD counter

BCD Ripple Counter

- The logic diagram of a BCD ripple counter using *JK* flip-flops is shown in Fig. 6.10 .
- The four outputs are designated by the letter symbol *Q*, with a numeric subscript equal to the binary weight of the corresponding bit in the BCD code.
- Note that the output of *Q*1 is applied to the *C* inputs of both *Q*2 and *Q*8 and the output of *Q*2 is applied to the *C* input of *Q*4.
- The *J* and *K* inputs are connected either to a permanent 1 signal or to outputs of other flip-flops.
- A ripple counter is an asynchronous sequential circuit.
- Signals that affect the flip-flop transition depend on the way they change from 1 to 0.

BCD Ripple Counter

- The operation of the counter can be explained by a list of conditions for flip-flop transitions.
- These conditions are derived from the logic diagram and from knowledge of how a *JK* flip-flop operates.
- Remember that when the *C* input goes from 1 to 0, the flip-flop is set if $J = 1$, is cleared if $K = 1$, is complemented if $J = K = 1$, and is left unchanged if $J = K = 0$.
- The BCD counter of Fig. 6.10 is a *decade* counter, since it counts from 0 to 9.
- To count in decimal from 0 to 99, we need a two-decade counter. To count from 0 to 999, we need a three-decade counter.

BCD Ripple Counter

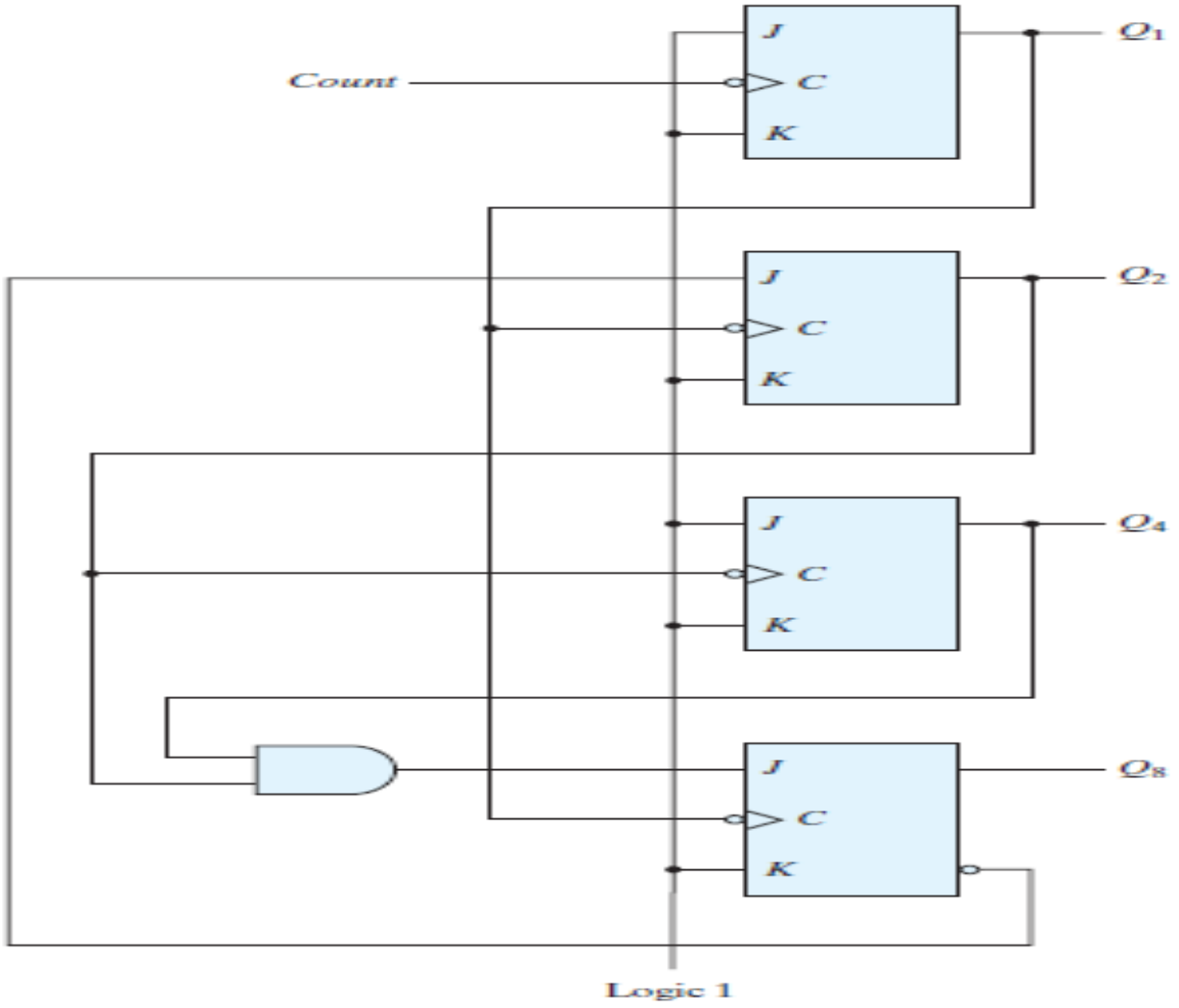


FIGURE 6.10
BCD ripple counter

BCD Ripple Counter

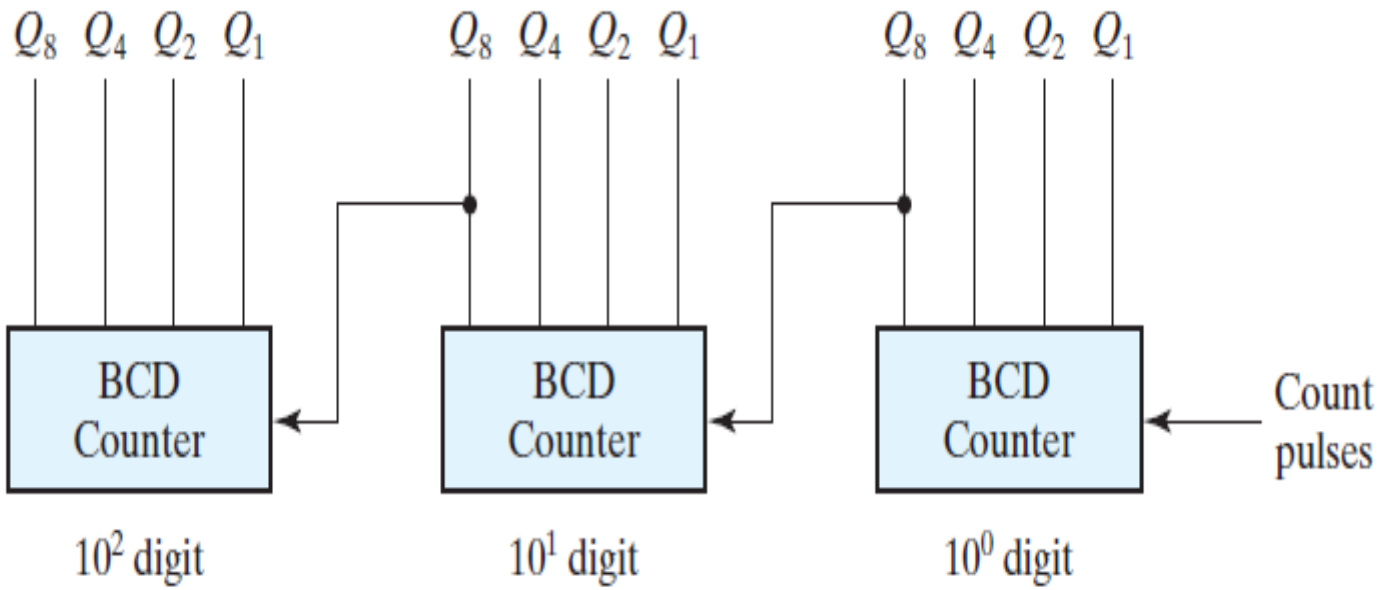


FIGURE 6.11
Block diagram of a three-decade decimal BCD counter

SYNCHRONOUS COUNTERS

- Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops.
- A common clock triggers all flip-flops simultaneously, rather than one at a time in succession as in a ripple counter.
- The decision whether a flip-flop is to be complemented is determined from the values of the data inputs, such as *T* or *J* and *K* at the time of the clock edge.
- If $T = 0$ or $J = K = 0$, the flip-flop does not change state.
- If $T = 1$ or $J = K = 1$, the flip-flop complements.

Binary Counter

- The design of a synchronous binary counter is so simple that there is no need to go through a sequential logic design process.
- In a synchronous binary counter, the flip-flop in the least significant position is complemented with every pulse.
- *A flip-flop in any other position is complemented when all the bits in the lower significant positions are equal to 1* .
- For example, if the present state of a four-bit counter is *A*3*A*2*A*1*A*0 = 0011, the next count is 0100.
- *A*0 is always complemented.
- *A*1 is complemented because the present state of *A*0 = 1.
- *A*2 is complemented because the present state of *A*1*A*0 = 11.
- However, *A*3 is not complemented, because the present state of *A*2*A*1*A*0 = 011, which does not give an all-1's condition.

Binary Counter

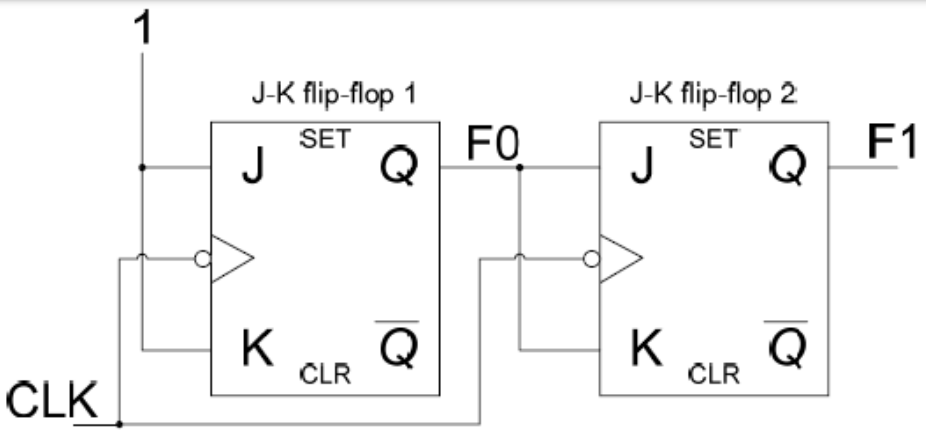


Figure 27.3a 2-bit Synchronous Counter

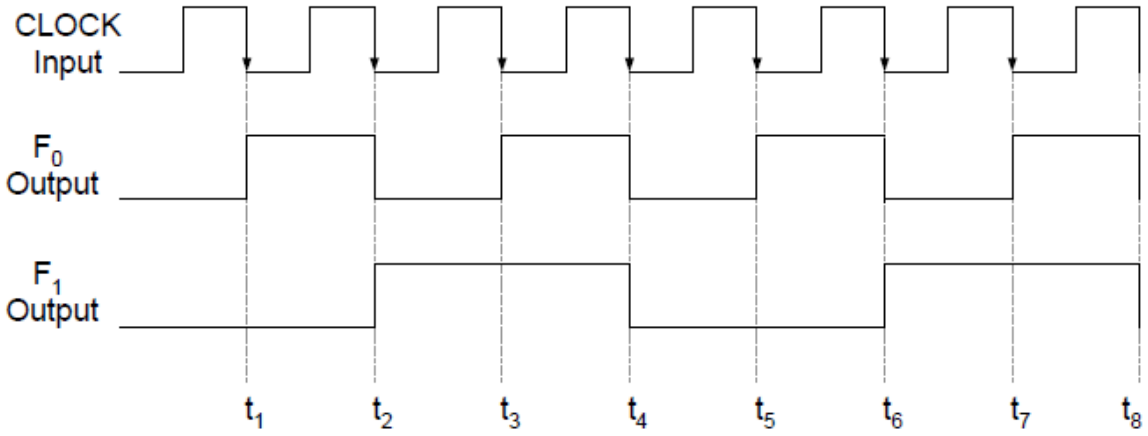


Figure 27.3b Timing diagram of a 2-bit Synchronous Counter

Binary Counter

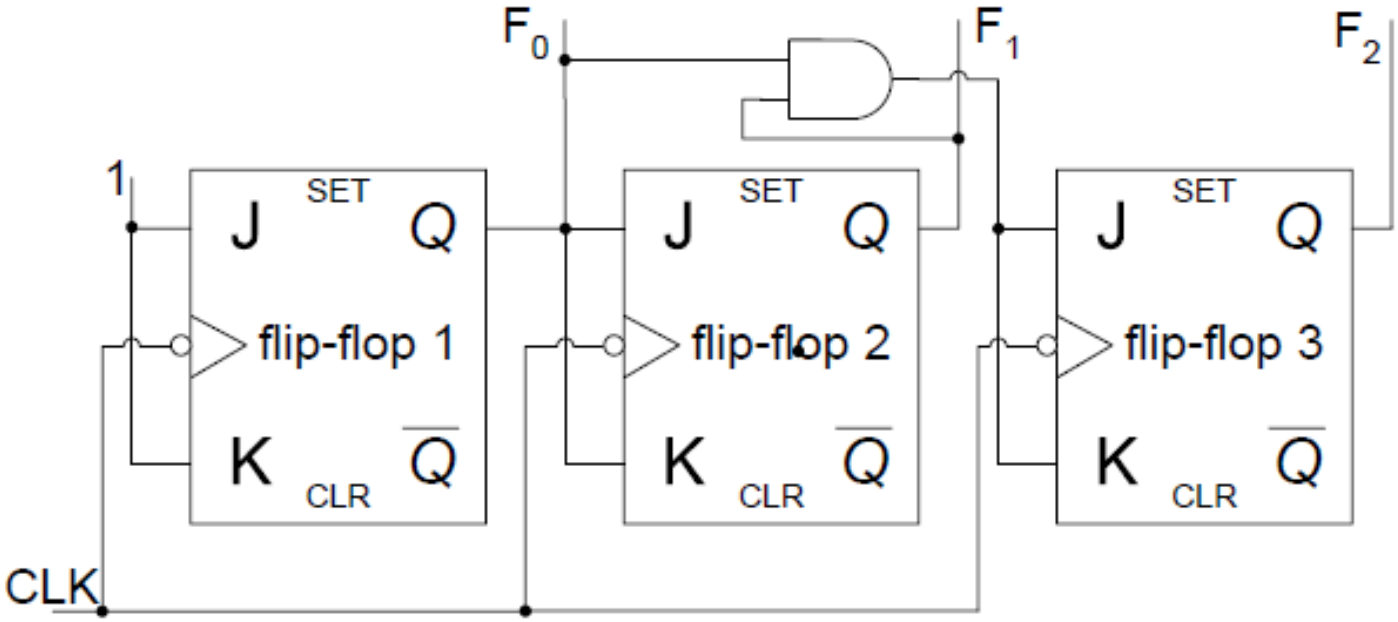


Figure 27.4a A 3-bit Synchronous Counter

Binary Counter

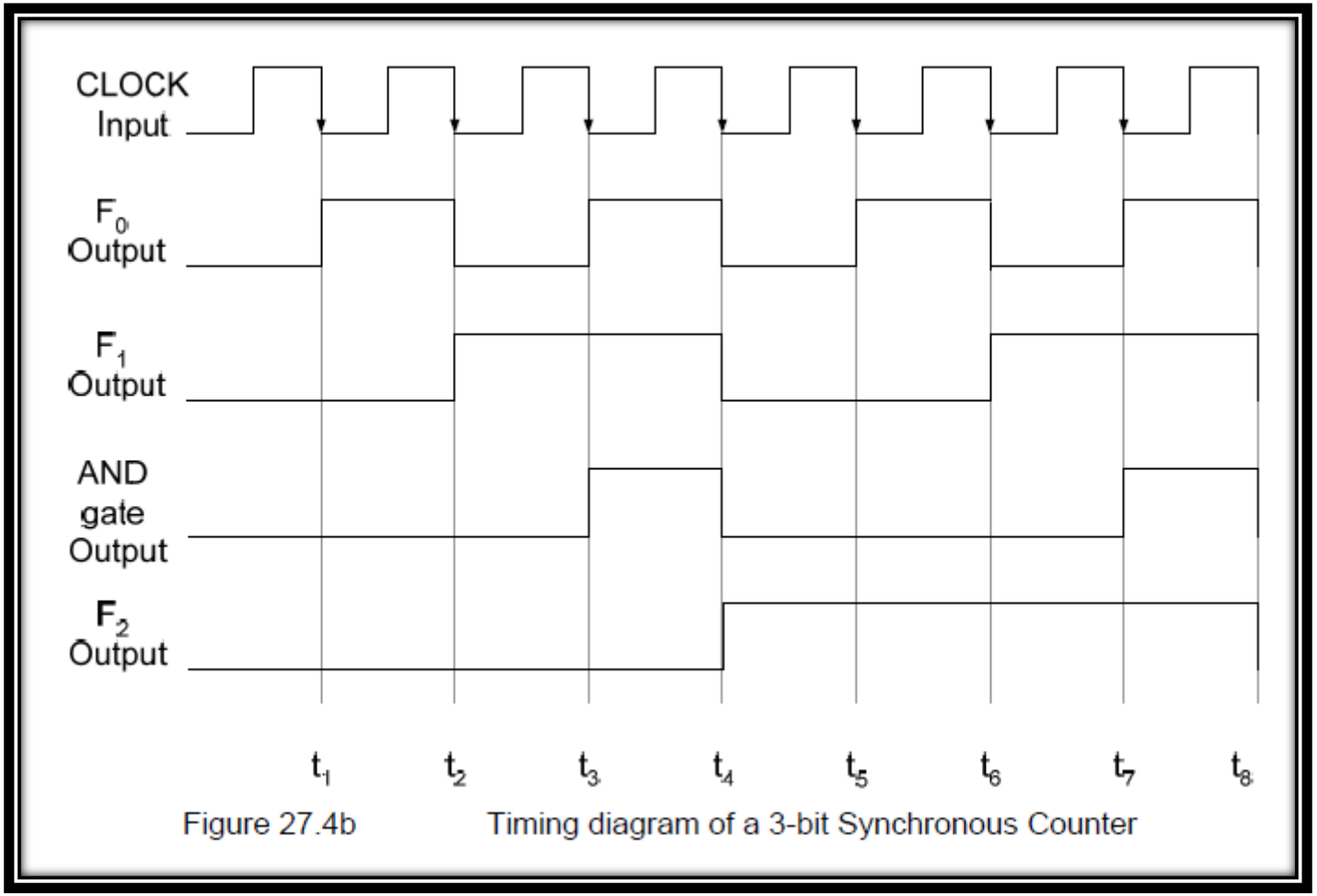


Figure 27.4b Timing diagram of a 3-bit Synchronous Counter

Binary Counter

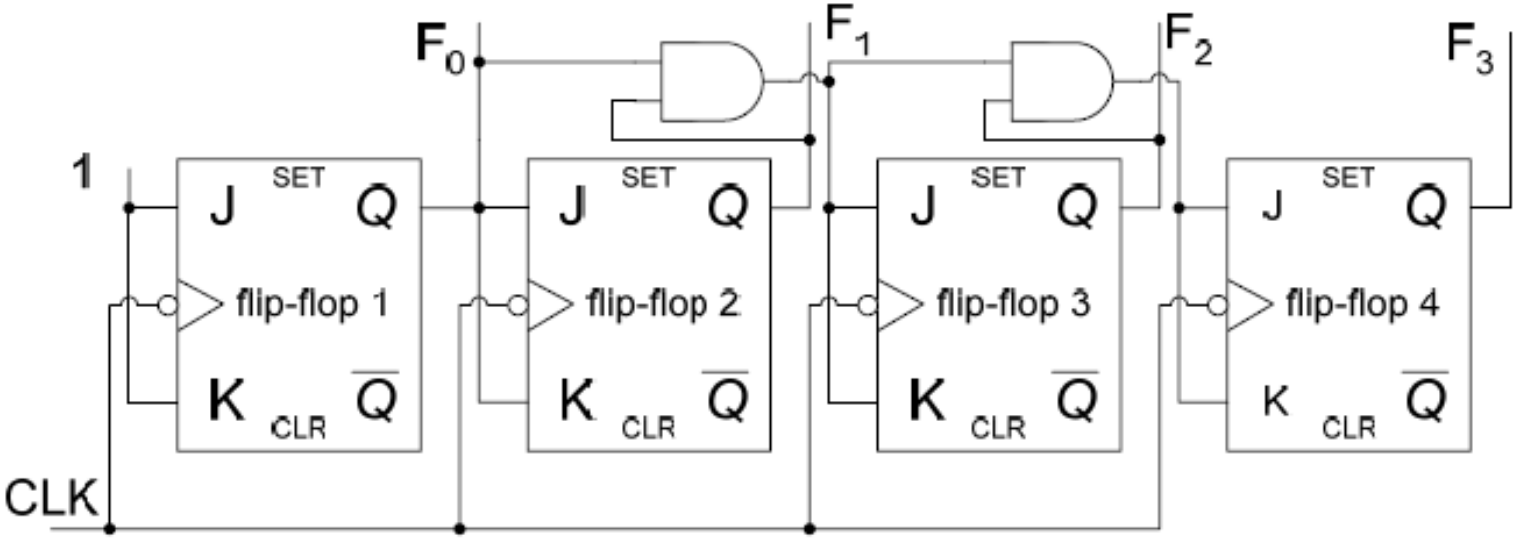


Figure 27.5 4-bit Synchronous Binary Counter

Binary Counter

- Synchronous binary counters have a regular pattern and can be constructed with complementing flip-flops and gates.
- The regular pattern can be seen from the four-bit counter depicted in Fig. 6.12 .
- The *C* inputs of all flip-flops are connected to a common clock.
- The counter is enabled by *Count_enable*.
- If the enable input is 0, all *J* and *K* inputs are equal to 0 and the clock does not change the state of the counter.
- The first stage, *A*0, has its *J* and *K* equal to 1 if the counter is enabled.

Binary Counter

- The other *J* and *K* inputs are equal to 1 if all previous least significant stages are equal to 1 and the count is enabled.
- The chain of AND gates generates the required logic for the *J* and *K* inputs in each stage.
- The counter can be extended to any number of stages, with each stage having an additional flip-flop and an AND gate that gives an output of 1 if all previous flip-flop outputs are 1.

Binary Counter

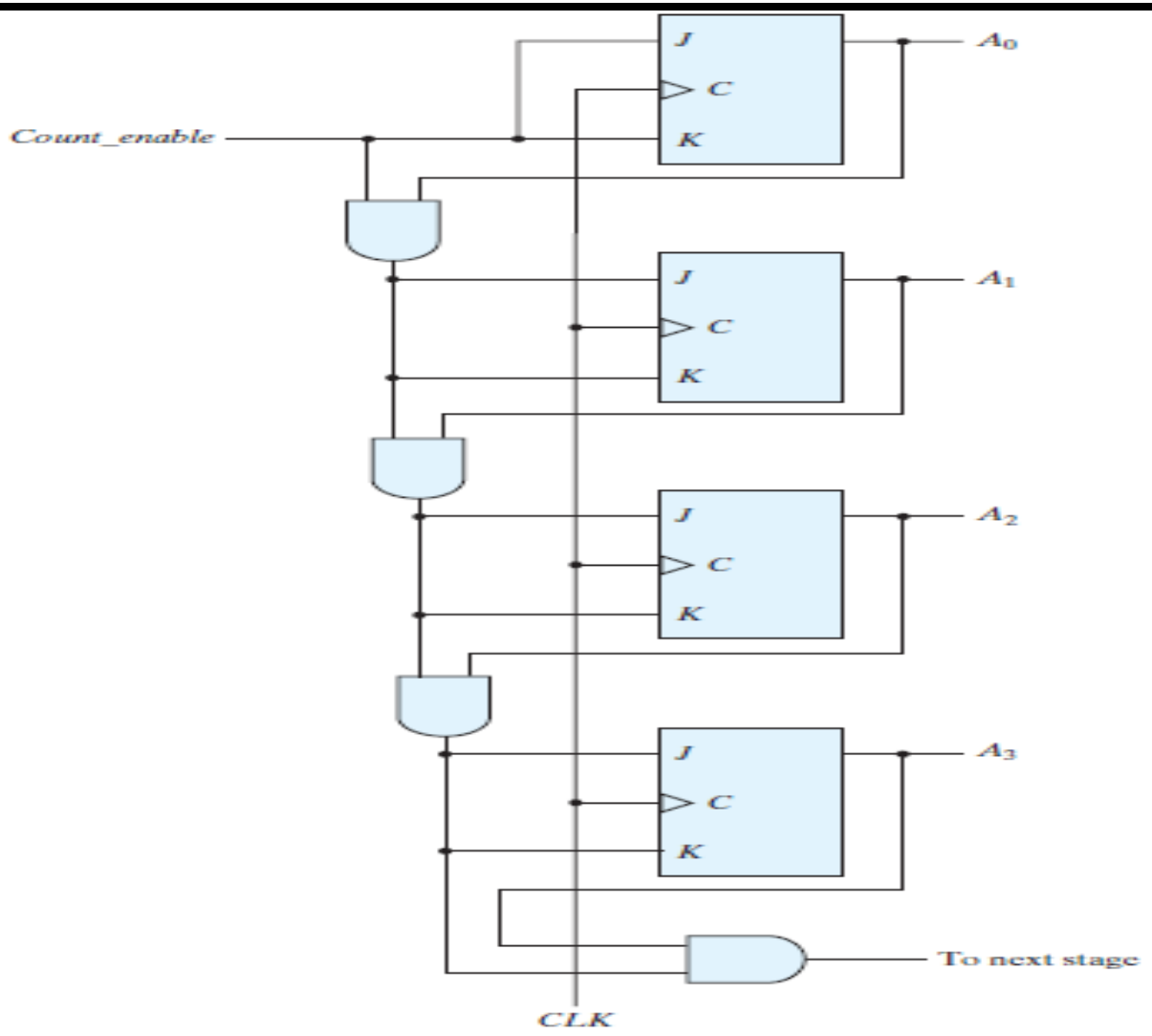


FIGURE 6.12
Four-bit synchronous binary counter

Up–Down Binary Counter

- A synchronous countdown binary counter goes through the binary states in reverse order, from 1111 down to 0000 and back to 1111 to repeat the count.
- It is possible to design a countdown counter in the usual manner, but the result is predictable by inspection of the downward binary count.
- The bit in the least significant position is complemented with each pulse.
- *A bit in any other position is complemented if all lower significant bits are equal to 0.*
- For example, the next state after the present state of 0100 is 0011.

Up–Down Binary Counter

- The least significant bit is always complemented.
- The second significant bit is complemented because the first bit is 0.
- The third significant bit is complemented because the first two bits are equal to 0.
- But the fourth bit does not change, because not all lower significant bits are equal to 0.
- The two operations can be combined in one circuit to form a counter capable of counting either up or down.
- The circuit of an up–down binary counter using *T* flip-flops is shown in Fig. 6.13 .

Up–Down Binary Counter

- It has an up control input and a down control input.
- When the up input is 1, the circuit counts up, since the *T* inputs receive their signals from the values of the previous normal outputs of the flip-flops.
- When the down input is 1 and the up input is 0, the circuit counts down, since the complemented outputs of the previous flip-flops are applied to the *T* inputs.
- When the up and down inputs are both 0, the circuit does not change state and remains in the same count.
- When the up and down inputs are both 1, the circuit counts up.
- This set of conditions ensures that only one operation is performed at any given time.

Up–Down Binary Counter

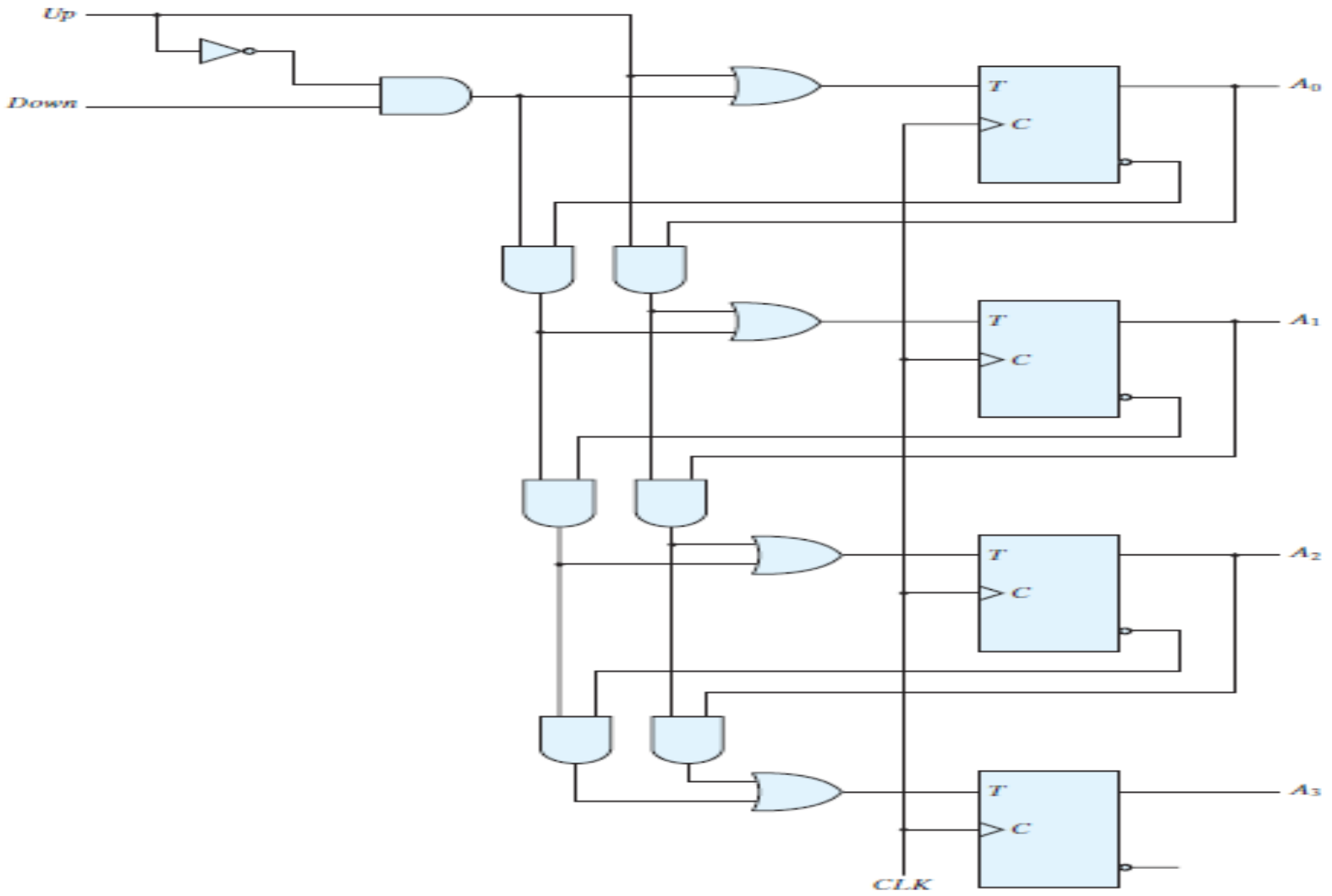


FIGURE 6.13
Four-bit up–down binary counter

BCD Counter

- A BCD counter counts in binary-coded decimal from 0000 to 1001 and back to 0000.
- Because of the return to 0 after a count of 9, a BCD counter does not have a regular pattern, unlike a straight binary count.
- To derive the circuit of a BCD synchronous counter, it is necessary to go through a sequential circuit design procedure.
- The state table of a BCD counter is listed in Table 6.5 .
- The input conditions for the *T* flip-flops are obtained from the present- and next-state conditions.
- Also shown in the table is an output *y*, which is equal to 1 when the present state is 1001.

BCD Counter

- In this way, *y* can enable the count of the next-higher significant decade while the same pulse switches the present decade from 1001 to 0000.
- The simplified functions are:

$$T_{Q1} = 1$$
$$T_{Q2} = Q'_8 Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

BCD Counter

Table 6.5
State Table for BCD Counter

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

Binary Counter with Parallel Load

- Counters employed in digital systems quite often require a parallel-load capability for transferring an initial binary number into the counter prior to the count operation.
- Figure 6.14 shows the top-level block diagram symbol and the logic diagram of a four-bit register that has a parallel load capability and can operate as a counter.
- When equal to 1, the input load control disables the count operation and causes a transfer of data from the four data inputs into the four flip-flops.
- If both control inputs are 0, clock pulses do not change the state of the register.

Binary Counter with Parallel Load

- The operation of the counter is summarized in Table 6.6 .
- The four control inputs— *Clear*, *CLK*, *Load*, and *Count* — determine the next state.
- The *Clear* input is asynchronous and, when equal to 0, causes the counter to be cleared regardless of the presence of clock pulses or other inputs.
- This relationship is indicated in the table by the X entries, which symbolize don't-care conditions for the other inputs.
- The *Clear* input must be in the 1 state for all other operations.
- With the *Load* and *Count* inputs both at 0, the outputs do not change, even when clock pulses are applied.

Binary Counter with Parallel Load

- A *Load* input of 1 causes a transfer from inputs *I*0 - *I*3 into the register during a positive edge of *CLK* .
- The input data are loaded into the register regardless of the value of the *Count* input, because the *Count* input is inhibited when the *Load* input is enabled.
- The *Load* input must be 0 for the *Count* input to control the operation of the counter.

Table 6.6
Function Table for the Counter of Fig. 6.14

Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

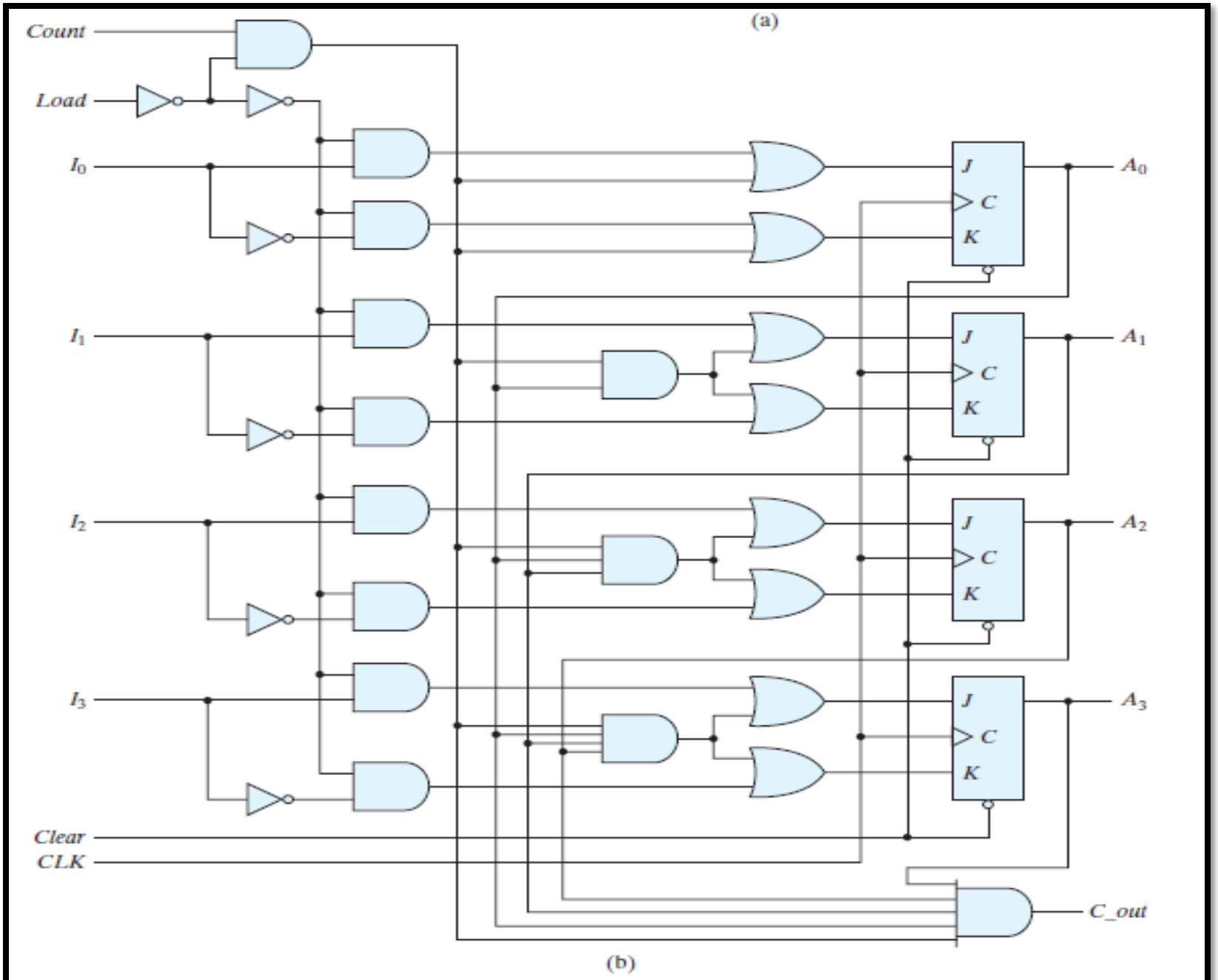


FIGURE 6.14
Four-bit binary counter with parallel load

OTHER COUNTERS

- Counters can be designed to generate any desired sequence of states.
- A divide-by- *N* counter (also known as a modulo- *N* counter) is a counter that goes through a repeated sequence of *N* states.
- The sequence may follow the binary count or may be any other arbitrary sequence.
- Counters are used to generate timing signals to control the sequence of operations in a digital system.
- Counters can also be constructed by means of shift registers.
- In this section, we present a few examples of non-binary counters.

Counter with Unused States

- A circuit with *n* flip-flops has 2^n binary states.
- There are occasions when a sequential circuit uses fewer than this maximum possible number of states.
- States that are not used in specifying the sequential circuit are not listed in the state table.
- In simplifying the input equations, the unused states may be treated as don't-care conditions.
- It is important to realize that once the circuit is designed and constructed, outside interference during its operation may cause the circuit to enter one of the unused states.
- In that case, it is necessary to ensure that the circuit eventually goes into one of the valid states so that it can resume normal operation.

Counter with Unused States

- As an illustration, consider the counter specified in Table 6.7.
- The count has a repeated sequence of six states, with flip-flops *B* and *C* repeating the binary count 00, 01, 10, and flip-flop *A* alternating between 0 and 1 every three counts.
- The simplified equations are
- $J_A = B$ $K_A = B$
- $J_B = C$ $K_B = 1$
- $J_C = B$ $K_C = 1$
- The logic diagram of the counter is shown in Fig. 6.16 (a). Since there are two unused states, we analyze the circuit to determine their effect.

Counter with Unused States

- If the circuit happens to be in state 011 because of an error signal, the circuit goes to state 100 after the application of a clock pulse.
- This action may be determined from an inspection of the logic diagram by noting that when *B* = 1, the next clock edge complements *A* and clears *C* to 0, and when *C* = 1, the next clock edge complements *B.*
- In a similar manner, we can evaluate the next state from present state 111 to be 000.

Counter with Unused States

Table 6.7
State Table for Counter

Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

Counter with Unused States

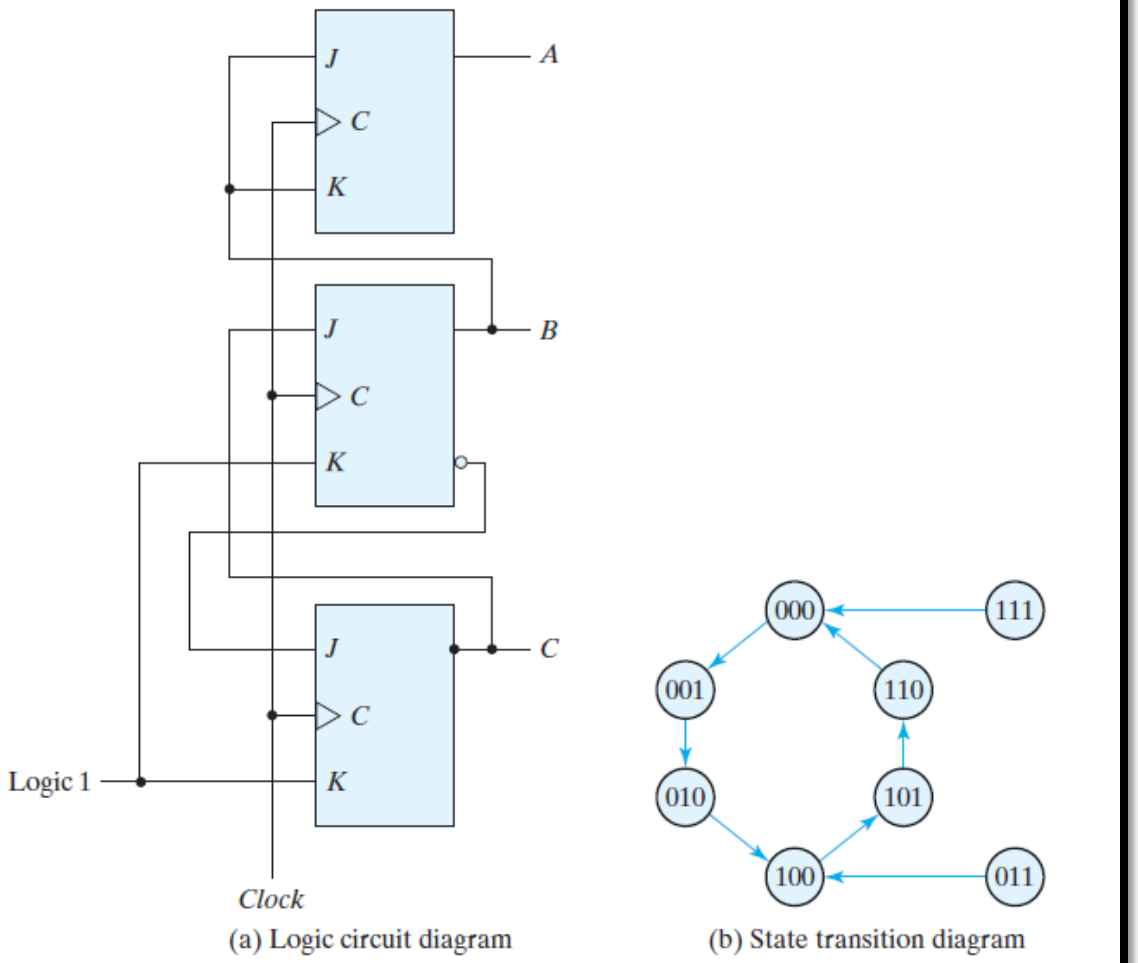


FIGURE 6.16
Counter with unused states

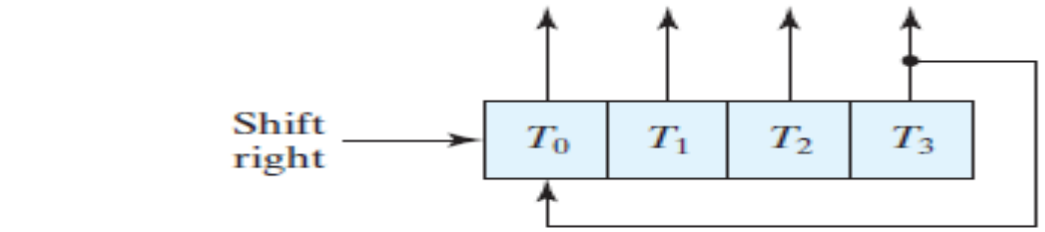
Ring Counter

- Timing signals that control the sequence of operations in a digital system can be generated by a shift register or by a counter with a decoder.
- A *ring counter* is a circular shift register with only one flip-flop being set at any particular time; all others are cleared.
- The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals.
- Figure 6.17 (a) shows a four-bit shift register connected as a ring counter.
- The initial value of the register is 1000 and requires Preset/Clear flip-flops.

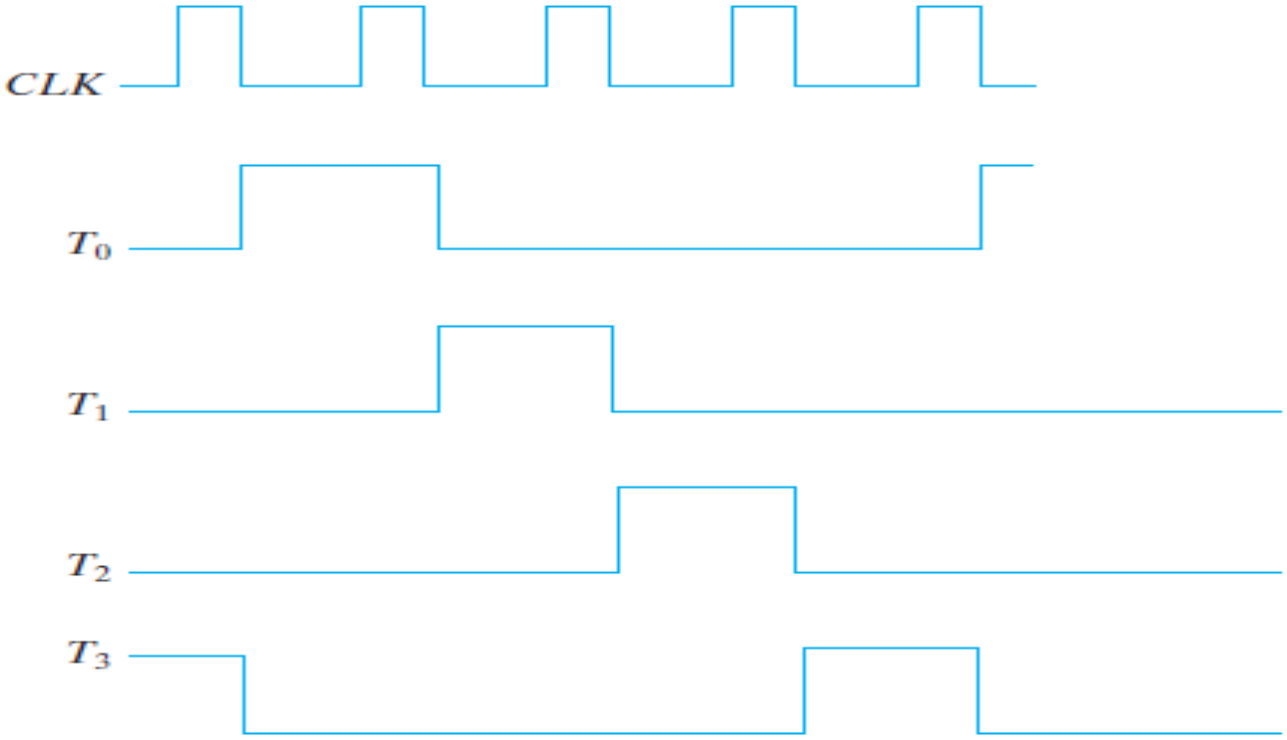
Ring Counter

- The single bit is shifted right with every clock pulse and circulates back from *T*3 to *T*0.
- Each flip-flop is in the 1 state once every four clock cycles and produces one of the four timing signals shown in Fig. 6.17 (b).
- Each output becomes a 1 after the negative-edge transition of a clock pulse and remains 1 during the next clock cycle.

Ring Counter



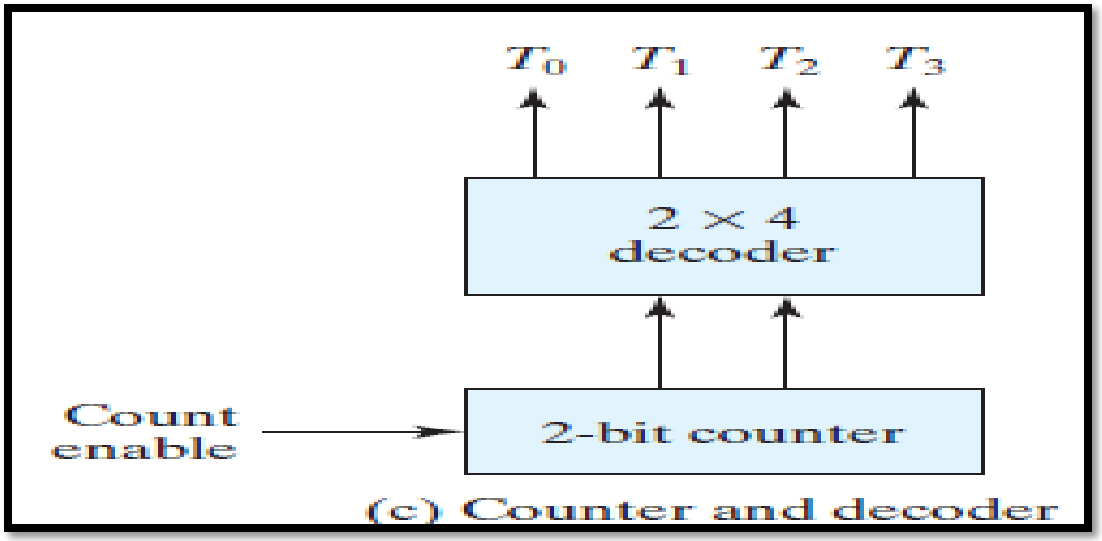
(a) Ring-counter (initial value = 1000)



(b) Sequence of four timing signals

Ring Counter

- For an alternative design, the timing signals can be generated by a two-bit counter that goes through four distinct states.
- The decoder shown in Fig. 6.17 (c) decodes the four states of the counter and generates the required sequence of timing signals. To generate 2^n timing signals, we need either a shift register with 2^n flip-flops or an n -bit binary counter together with an n -to-2^n -line decoder.



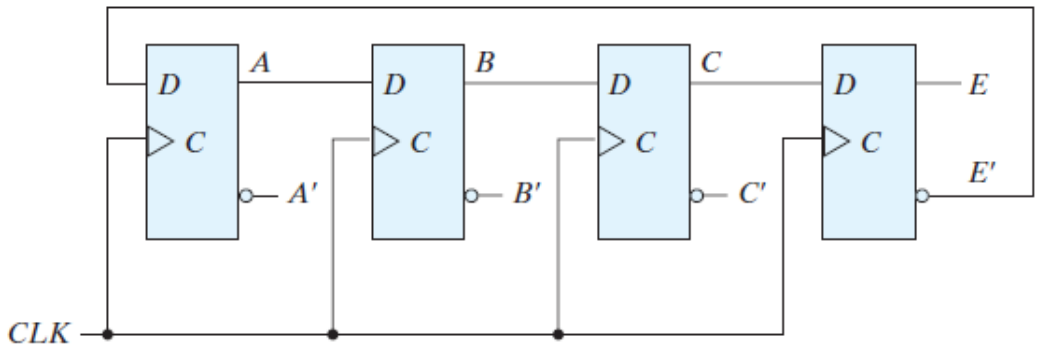
Johnson Counter

- A *k* -bit ring counter circulates a single bit among the flip-flops to provide *k* distinguishable states.
- The number of states can be doubled if the shift register is connected as a *switch-tail* ring counter.
- A switch-tail ring counter is a circular shift register with the complemented output of the last flip-flop connected to the input of the first flip-flop.
- Figure 6.18 (a) shows such a shift register.
- The circular connection is made from the complemented output of the rightmost flip-flop to the input of the leftmost flip-flop.

Johnson Counter

- The register shifts its contents once to the right with every clock pulse, and at the same time, the complemented value of the *E* flip-flop is transferred into the *A* flip-flop.
- Starting from a cleared state, the switch-tail ring counter goes through a sequence of eight states, as listed in Fig. 6.18 (b).
- In general, a *k* -bit switch-tail ring counter will go through a sequence of 2*k* states.
- A Johnson counter is a *k* -bit switch-tail ring counter with 2*k* decoding gates to provide outputs for 2 *k* timing signals.
- The decoding gates are not shown in Fig. 6.18 , but are specified in the last column of the table.

Johnson Counter



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs A	B	C	E	AND gate required for output
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

(b) Count sequence and required decoding

FIGURE 6.18
Construction of a Johnson counter