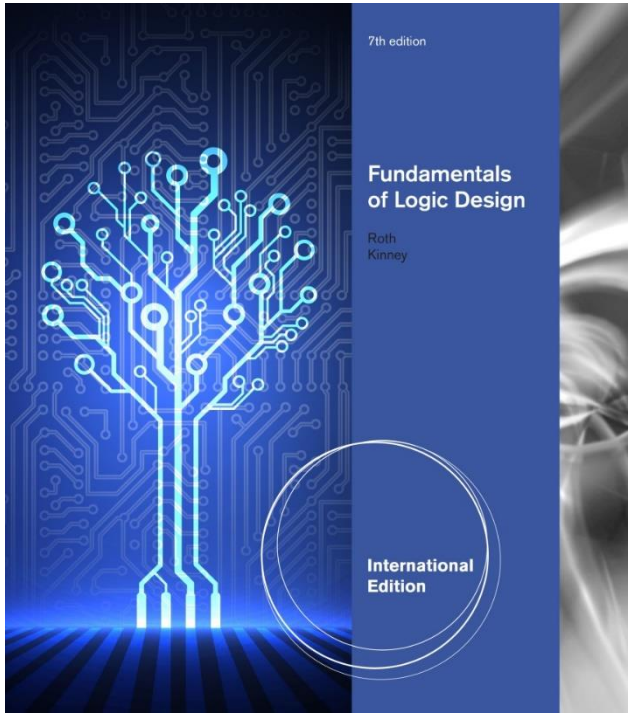


Registers and Counters

Contents

Objectives

- Explain the operation of registers. Show how to transfer data between registers using a tri-state bus.
- Explain the operation of shift registers, show how to build them using flip-flops, and analyze their operation. Construct a timing diagram for a shift register.
- Explain the operation of binary counters, show how to build them using flip-flops and gates, and analyze their operation.
- Given the present state and desired next state of a flip-flop, determine the required flip-flop inputs.
- Given the desired counting sequence for a counter, derive the flip-flop input equations.
- Explain the procedures used for deriving flip-flop input equations.
- Construct a timing diagram for a counter by tracing signals through the circuit.

Registers and Register Transfers

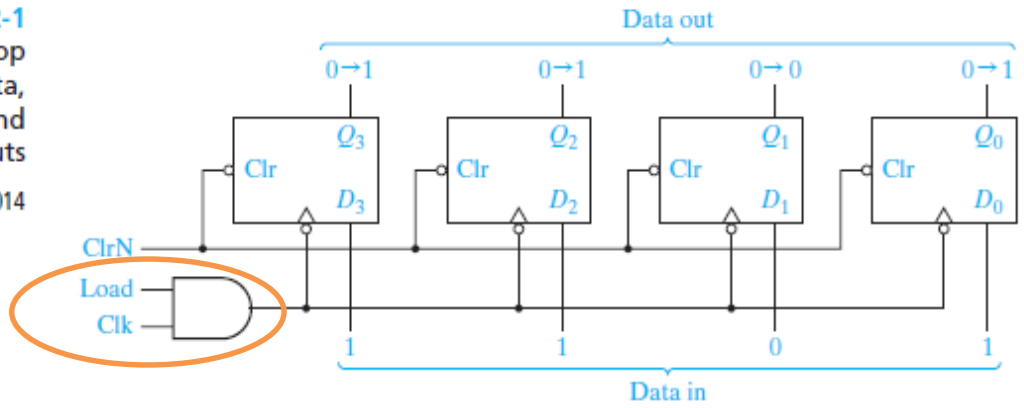
Introduction:

- A **register** consists of a group of flip-flops with a common clock input. Registers are commonly used to store and shift binary data.
- A **counter** is usually constructed from two or more flip-flops which change states in a prescribed sequence when input pulses are received.

4-Bit D Flip-Flop Registers with Data, Load, Clear and Clock Inputs:

- When Load = 1, the clock signal (Clk) is transmitted to the flip-flop clock inputs and the data applied to the *D* inputs will be loaded into the flip-flop on the falling edge of the clock.
- The flip-flops in the register have asynchronous clear inputs that are connected to a common clear signal, ClrN. ClrN is normally 1, and if it is changed momentarily to 0, the *Q* outputs of all four flip-flops will become 0.
- Note that gating the clock with another signal can cause timing problems.

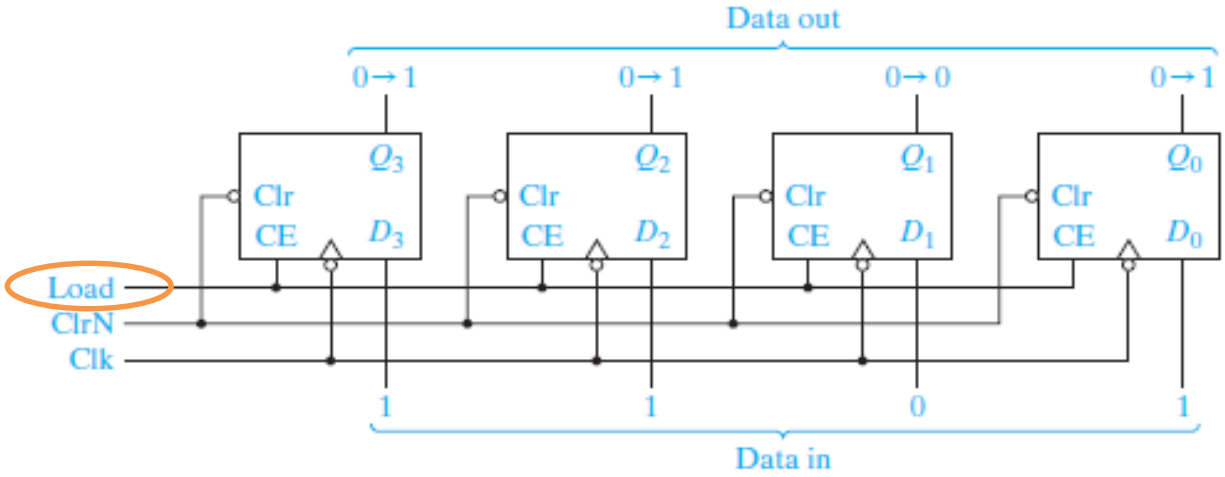
FIGURE 12-1
4-Bit D Flip-Flop Registers with Data, Load, Clear, and Clock Inputs



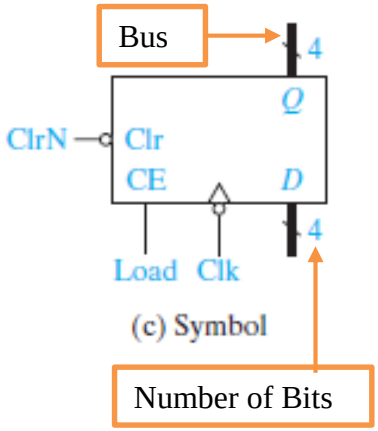
(a) Using gated clock

4-Bit D Flip-Flop Registers with Clock Enable:

- The load signal is connected to all four CE inputs. When Load = 0, the clock is disabled and the register holds its data. When Load is 1, the clock is enabled, and the data applied to the *D* inputs will be loaded into the flip-flops, following the falling edge of the clock.
- Figure (c) shows a symbol for the 4-bit register using bus notation for the *D* inputs and *Q* outputs. A group of wires that perform a common function is often referred to as a bus. A heavy line is used to represent a bus, and a slash with a number beside it indicates the number of bits in the bus.



(b) With clock enable

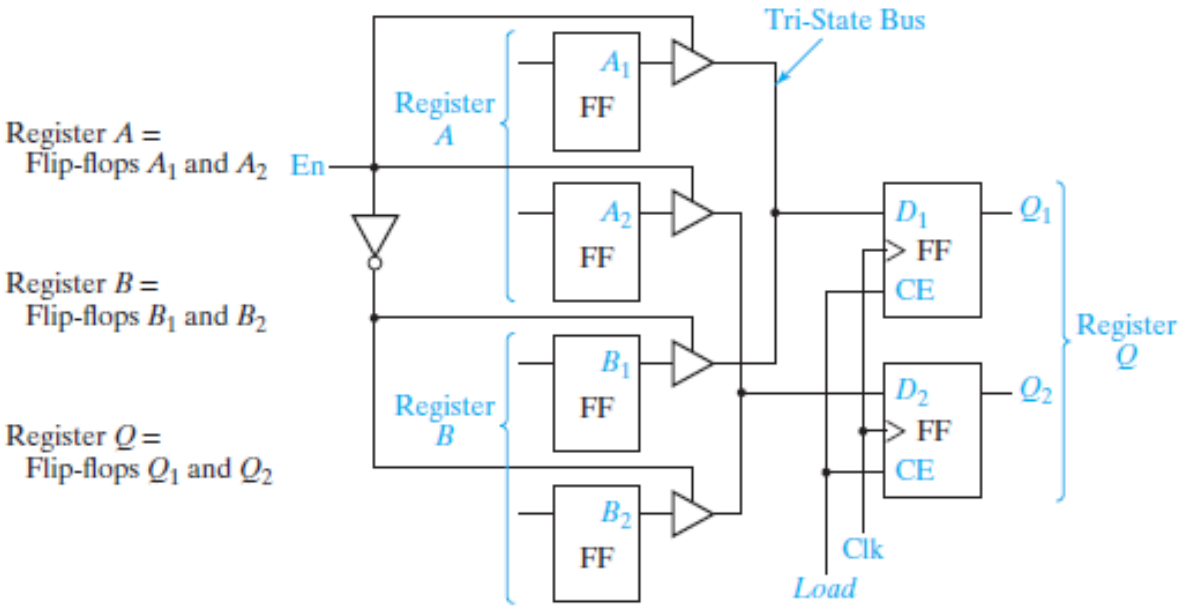


Registers and Register Transfers

Data Transfer Between Registers:

- If En = 1 and Load = 1, the output of register *A* is enabled onto the tri-state bus and the data in register *A* will be stored in *Q* after the rising edge of the clock.
- If En = 0 and Load = 1, the output of register *B* will be enabled onto the tri-state bus and stored in *Q* after the rising edge of the clock.

FIGURE 12-2
Data Transfer
Between Registers

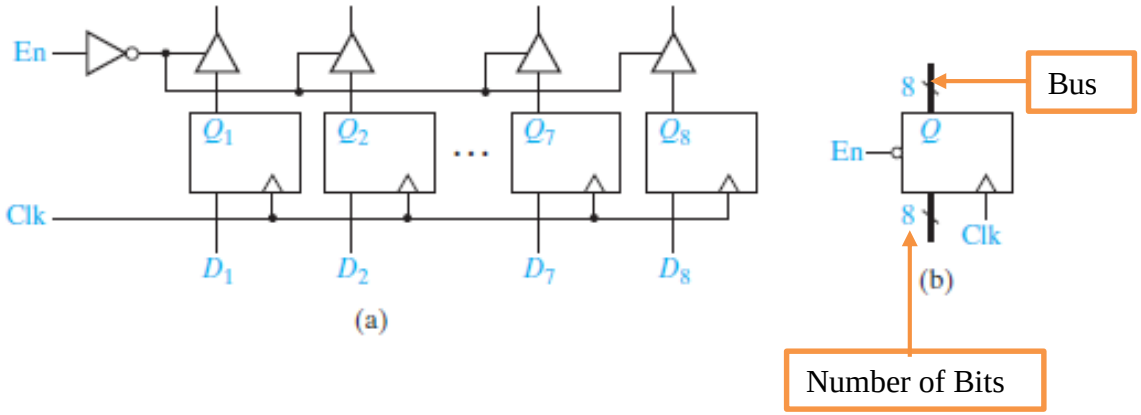


Registers and Register Transfers

8-Bit Register with Tri-State Output:

- Figure 12-3(a) shows an integrated circuit register that contains eight *D* flip-flops with tri-state buffers at the flip-flop outputs.
- These buffers are enabled when En = 0. A symbol for this 8-bit register is shown in Figure 12-3(b).

FIGURE 12-3
Logic Diagram for 8-Bit Register with Tri-State Output

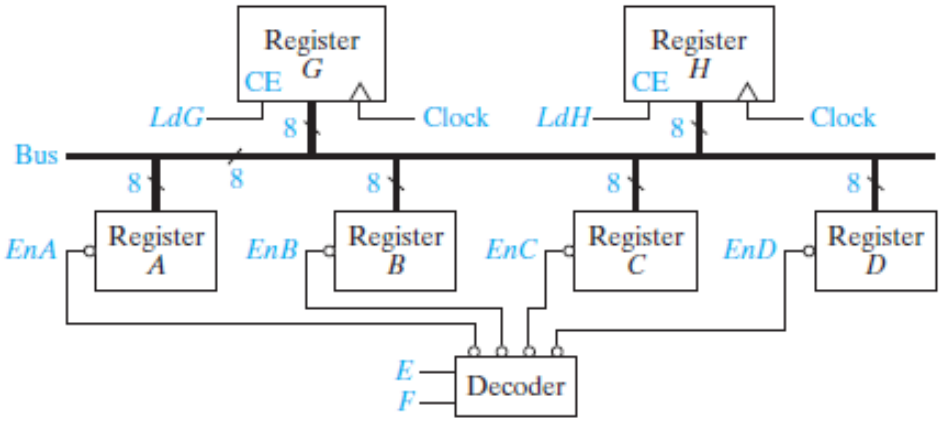


Registers and Register Transfers

Data Transfer Using a Tri-State Bus:

- When *EnA* = 0, the tri-state outputs of register *A* are enabled on to the bus.
- If *LdG* = 1, these signals oh the bus are loaded into register G after the rising clock edge (or into register *H* if *LdH* = 1)
- If *LdG* = *LdH* = 1, both *G* and *H* will be loaded from the bus.

FIGURE 12-4
Data Transfer Using a Tri-State Bus



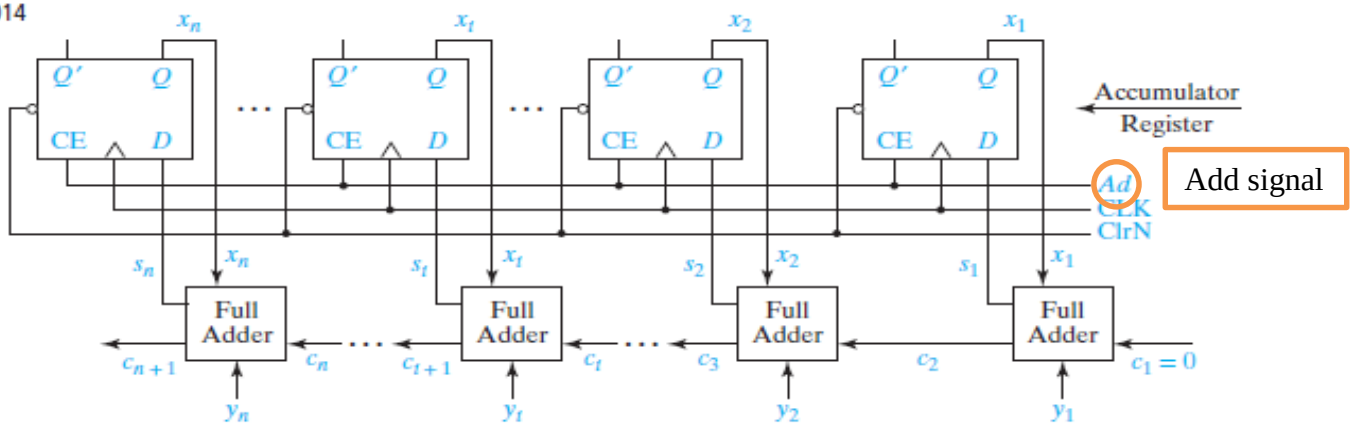
If $EF = 00$, A is stored in G (or H).
If $EF = 01$, B is stored in G (or H).
If $EF = 10$, C is stored in G (or H).
If $EF = 11$, D is stored in G (or H).

Registers and Register Transfers

Parallel Adder with Accumulator:

- It is frequently desirable to store one number in a register of flip-flops (called an **accumulator**) and add a second number to it, leaving the result stored in the accumulator.
- An add signal (*Ad*) is used to load the adder outputs into the accumulator flip-flops on the rising clock edge.
- The next state of the filp-flop can be written as $x_i^+ = s_i$.
- If *Ad* = 1, the number *X* in the accumulator is replaced with the sum of *X* and *Y*, following the rising edge of the clock.

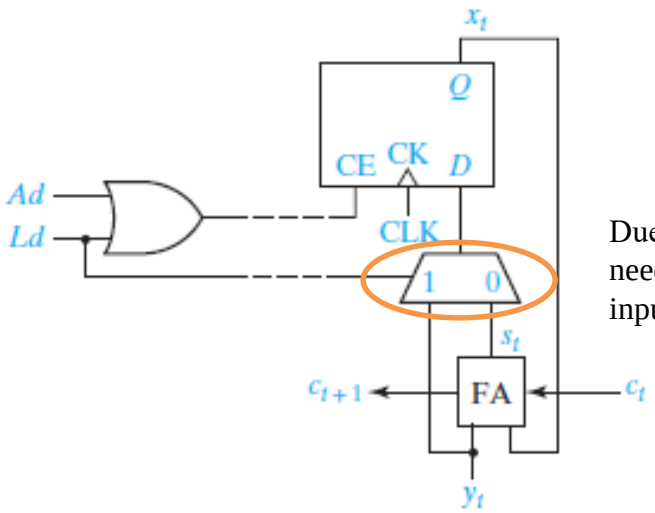
FIGURE 12-5 *n*-Bit Parallel Adder with Accumulator



Adder Cell with Multiplexer:

- Figure 12-6 shows a typical cell of the adder where the accumulator flip-flop can either be loaded directly from y_i or from the sum output s_i through the multiplexer.
- This would eliminate the extra step of clearing the accumulator but would add to the hardware complexity. (Otherwise clear the accumulator using the asynchronous clear inputs on the flip-flops)
- When $Ld = 1$, the multiplexer selects y_i, and y_i is loaded into the accumulator flip-flop (x_i) on the rising clock edge.
- When $Ad = 1$ and $Ld = 0$, the adder output (s_i) is loaded into x_i.
- The *Ad* and *Ld* signals are ORed together to enable the clock when either addition or loading occurs.
- When $Ad = Ld = 0$, the clock is disabled and the accumulator outputs do not change.

FIGURE 12-6
Adder Cell with Multiplexer



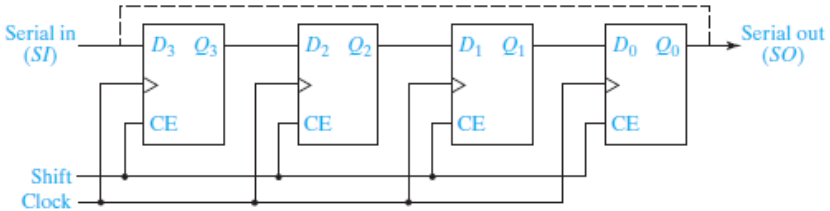
Due to this multiplexer, we don't need to use asynchronous clear inputs on the flip-flops

Shift Registers

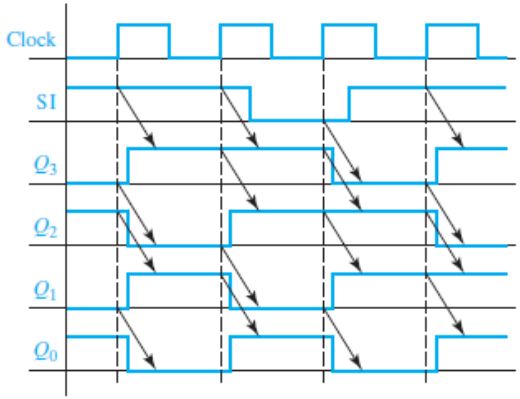
Shift Registers:

- A shift register is a register in which binary data can be stored, and this data can be shifted to the left or right when a shift signal is applied.
- Shifts can be linear or cyclic.
- The figure shows a 4-bit right-shift register with serial input and output constructed from *D* flip-flops.
- When Shift = 1, the clock is enabled and shifting occurs on the rising clock edge. When Shift = 0, no shifting occurs and the data in the register is unchanged.

FIGURE 12-7
Right-Shift Register



(a) Flip-flop connections



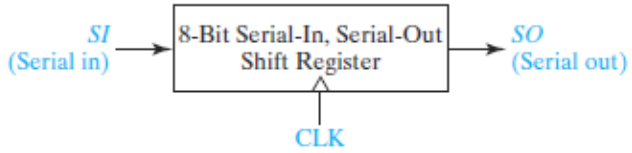
(b) Timing diagram

Shift Registers

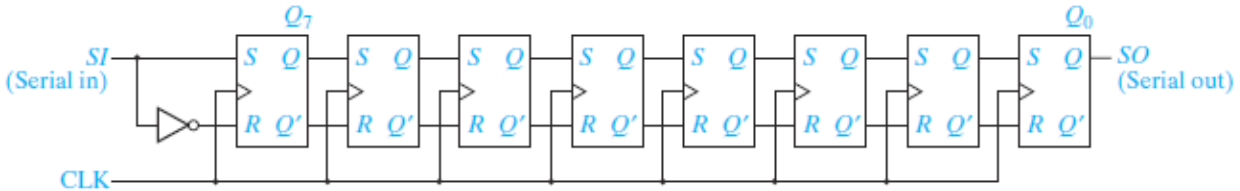
Serial-In Serial-Out Shift Registers:

- **Serial in** means that data is shifted into the first flip-flop one bit at a time, and the flip-flops cannot be loaded in parallel.
- **Serial out** means that data can only be read out of the last flip-flop and the outputs from the other flip-flops are not connected to terminals of the integrated circuit.
- The inputs to the first flip-flops are $S = SI$ and $R = SI'$.

FIGURE 12-8
8-Bit Serial-In, Serial-Out Shift Register



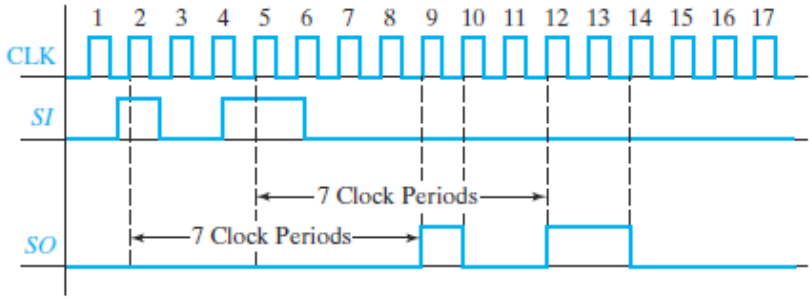
(a) Block diagram



(b) Logic diagram

- If $SI = 1$, a 1 is shifted into the register when it is clocked, and if $SI = 0$, a 0 is shifted in.

FIGURE 12-9
Typical Timing Diagram for Shift Register of Figure 12-8



Shift Registers

Parallel-In Parallel-Out Shift Register:

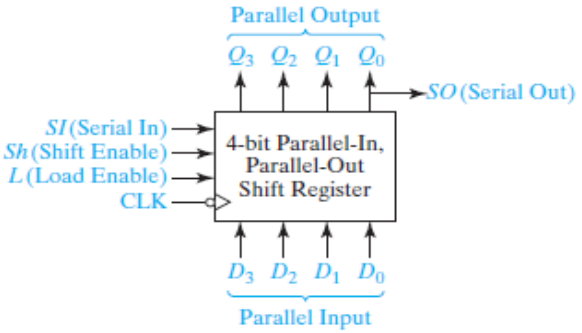
- Parallel-in implies that all four bits can be loaded at the same time, and parallel-out implies that all bits can be read out at the same time.
- The shift register has two control inputs, shift enable (*Sh*) and load enable (*L*).

- If *Sh* = 1 (and *L*=1, or *L*=0), clocking the register causes the serial input (*SI*) to be shifted into the first flip-flop, while the data in flip-flops Q_3, Q_2, Q_1 are shifted right.
- If *Sh* = 0 and *L* = 1, clocking the shift register will cause the four data inputs (D_3, D_2, D_1, D_0) to be loaded in parallel into the flip-flops.
- If *Sh* = *L* = 0, clocking the register causes no change of state.

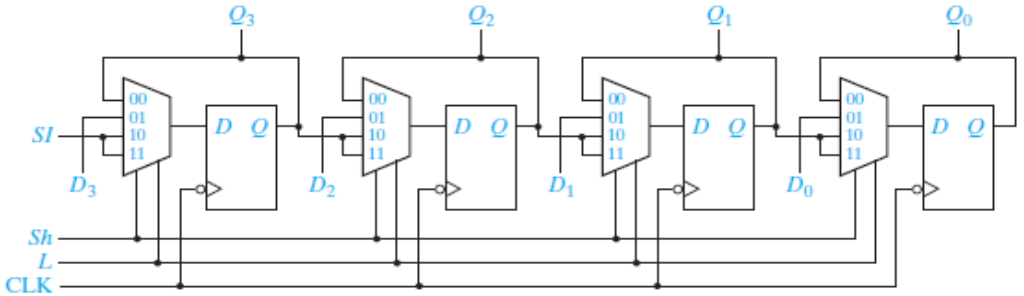
TABLE12-1 Shift Register Operation

Inputs		Next State				
Sh (Shift)	*L* (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	Action
0	0	Q_3	Q_2	Q_1	Q_0	No change
0	1	D_3	D_2	D_1	D_0	Load
1	X	*SI*	Q_3	Q_2	Q_1	Right shift

FIGURE 12-10 Parallel-In, Parallel-Out Right-Shift Register



(a) Block diagram



(b) Implementation using flip-flops and MUXes

Shift Registers

Parallel-In Parallel-Out Shift Register:

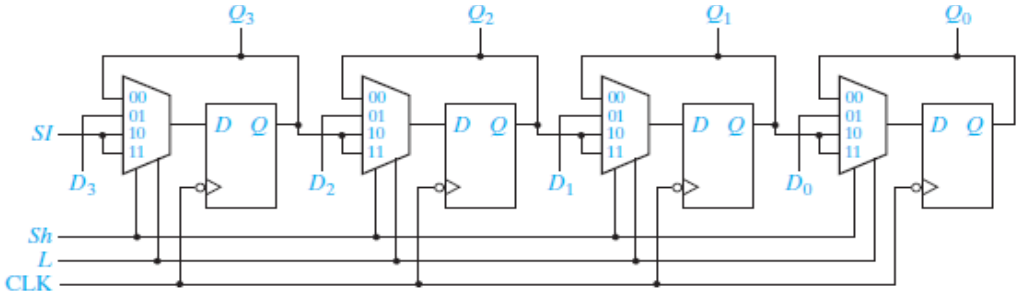
- For the first flip-flop, when $Sh = L = 0$, the flip-flop Q_3 output is selected by the MUX, so $Q_3^+ = Q_3$ and no state change occurs.
- When $Sh = 0$ and $L = 1$, the data input D_3 is selected and loaded into the flip-flop.
- When $Sh = 1$ and $L = 0, 1$, SI is selected and loaded into the flip-flop. The second MUX selects Q_2, D_2 or Q_3, etc.
- The next-state equations for the flip-flops are

$$Q_3^+ = Sh' \cdot L' \cdot Q_3 + Sh' \cdot L \cdot D_3 + Sh \cdot SI$$
$$Q_2^+ = Sh' \cdot L' \cdot Q_2 + Sh' \cdot L \cdot D_2 + Sh \cdot Q_3$$
$$Q_1^+ = Sh' \cdot L' \cdot Q_1 + Sh' \cdot L \cdot D_1 + Sh \cdot Q_2$$
$$Q_0^+ = Sh' \cdot L' \cdot Q_0 + Sh' \cdot L \cdot D_0 + Sh \cdot Q_1$$

Shift Registers

Conversion of Parallel Data to Serial Data:

- The output from the last flip-flop (Q_0) serves as a serial output as well as one of the parallel outputs.
  - ✓ The first clock pulse loads data into the shift register in parallel.
  - ✓ Assuming that the register is initially clear ($Q_3Q_2Q_1Q_0 = 0000$), that the serial input is $SI = 0$ throughout, and that the data inputs $D_3D_2D_1D_0 = 1011$ during the load time t_0.
  - ✓ Shifting occurs at the end of t_1, t_2, and t_3 and the serial output can be read during these clock times. During t_4, $Sh = L = 0$, so no state change occurs.



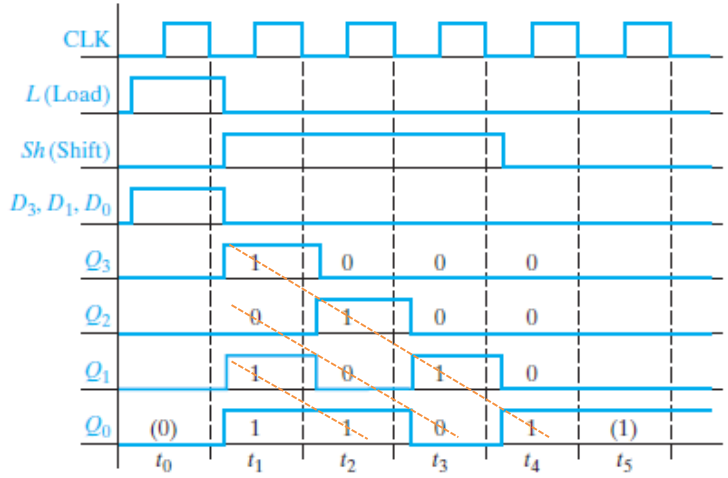


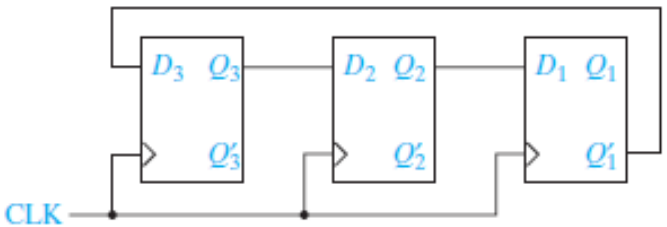
FIGURE 12-11
Timing Diagram for Shift Register

Shift Registers

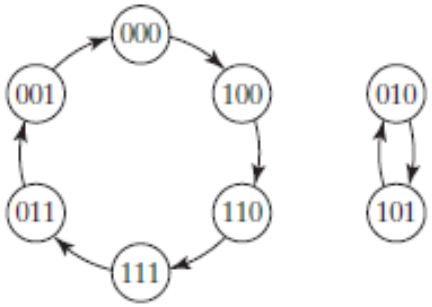
Johnson Counter:

- A circuit that cycles through a fixed sequence of states is called a **counter**.
- A shift register with inverted feedback is called a **Johnson counter** or a **twisted ring counter**.
- If the register is in state 010, then a shift pulse takes it to 101 and vice versa; therefore, we have a secondary loop on the transition graph
- If the feedback is not converted (e.g., Q_1 connected to D_3), the shift register counter is called a **ring counter**.

FIGURE 12-12
Shift Register with Inverted Feedback



(a) Flip-flop connections



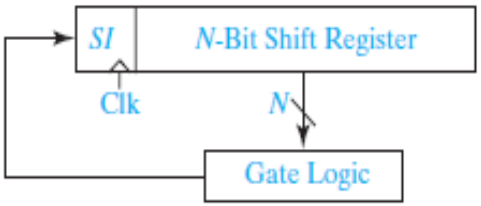
(b) Transition graph

Shift Registers

General Form of a Shift Register Counter:

- The bit being shifted into the leftmost stage can be a general function of the shift register contents.
- If the gate logic only contains exclusive-OR gates, the counter is called a linear (feedback) shift register counter.
- It can be shown that, for each integer *n*, there exists a linear *n*-bit shift register counter that generates a count cycle of length 2^n -1; all states are included except for the all 0's state.
- Linear shift register counters have many applications, including as random number generators and as encoders and decoders for linear error-correcting codes.

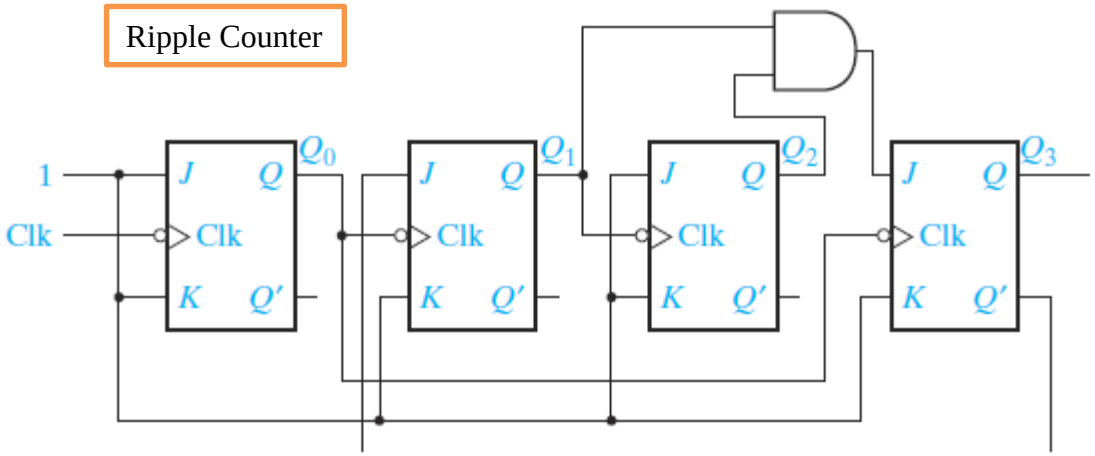
FIGURE 12-13
General Shift Register Counter



Design of Binary Counters

Synchronous and Ripple Counters:

- For synchronous counters, the operation of the flip-flops is synchronized by a common clock pulse so that when several flip-flops must change state, the state changes occur simultaneously.
- Ripple counters are those in which the state change of one flip-flop triggers another flip-flop.

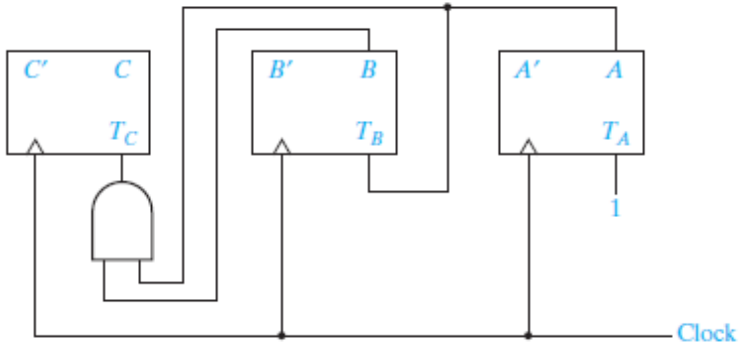


Design of Binary Counters

Binary Counters Using 3 T Flip-Flops to Count Clock Pulses:

- We assume that all the flip-flops change state a short time following the rising edge of the input pulse.
- The state of the counter is determined by the states of the individual flip-flops; for example, if flip-flop *C* is in state 0, *B* in state 1, and *A* in state 1, the state of the counter is 011.
- Initially, assume that all flip-flops are set to the 0 state. When a clock pulse is received, the counter will change to state 001; when a second pulse is received, the state will change to 010, etc. When 111 is reached, the counter resets to the 000 state.
- The sequence of flip-flop states is *CBA* = 000, 001, 010, 011, 100, 101, 110, 111, (repeat) 000, … (Design by inspection)
  - ✓ Because *A* changes state on every rising clock edge, T_A must equal 1.
  - ✓ *B* changes state only if *A* = 1. Therefore, *A* is connected to T_B , so that if *A* = 1, B will change state when a rising clock edge occurs.
  - ✓ *C* changes state when a rising clock edge occurs only if *B* and *A* are both 1.

FIGURE 12-14
Synchronous Binary Counter



Present State			Next State		
C	*B*	*A*	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Design of Binary Counters

Design of Binary Counter Using Transition Tables:

- This table shows the present state of flip-flops *C*, *B*, and *A* (before a clock pulse is received) and the corresponding next state (after the clock pulse is received).
- A third column in the table is used to derive the inputs for T_C, T_B, and T_A.
  - ✓ Whenever the entries in the *A* and A^+ columns differ, flip-flop *A* must change state and T_A must be 1.
  - ✓ Similarly, if *B* and B^+ differ, *B* must change state so T_B must be 1.

Karnaugh Maps for Binary Counters:

- Figure 12-15 shows the Karnaugh maps for T_C and T_B, from which $T_C = BA$ and $T_B = A$. These equations yield the same circuit derived previously for Figure 12-14.

TABLE12-2
Transition Table for Binary Counter

Present State			Next State			Flip-Flop Inputs		
C	*B*	*A*	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

FIGURE 12-15
Karnaugh Maps for Binary Counter

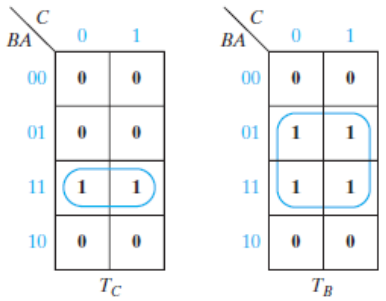
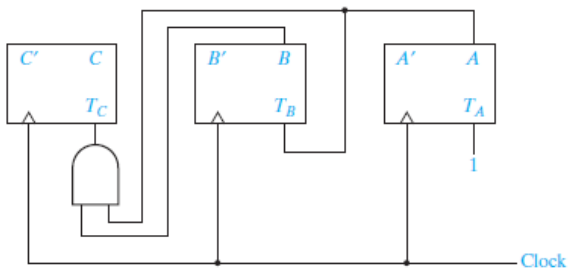


FIGURE 12-14
Synchronous Binary Counter

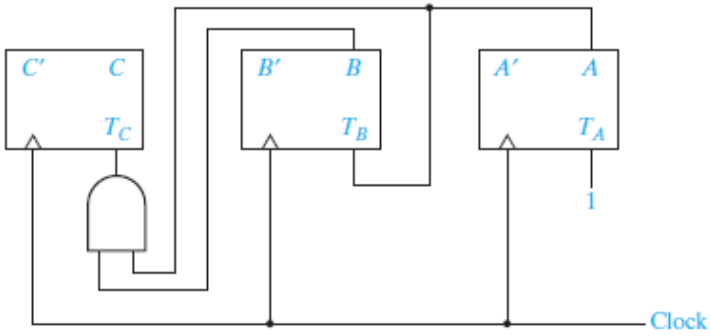


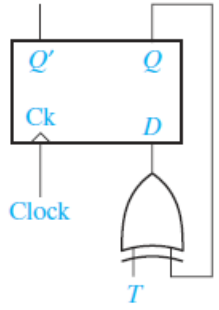
Design of Binary Counters

Binary Counter with D Flip-Flops:

- We must convert each *D* flip-flop to a *T* flip-flop by adding an XOR (exclusive-OR) gate, as shown in Figure 11-28(b). Figure 12-16 shows the resulting counter circuit.

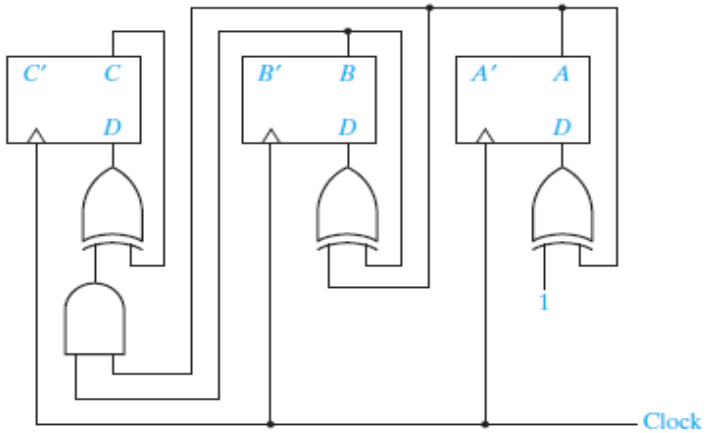
FIGURE 12-14
Synchronous Binary Counter





(b) Conversion of *D* to *T*

FIGURE 12-16
Binary Counter with D Flip-Flops



Design of Binary Counters

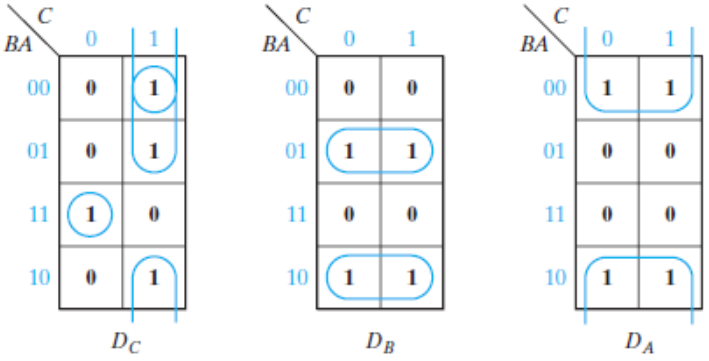
Binary Counter with D Flip-Flops:

We can also derive the D flip-flop inputs for the binary counter starting with its transition table (Table 12-2). For a D flip-flop, $Q^+ = D$. By inspection of the table, $Q_A{}^+ = A'$, so $D_A = A'$. The maps for $Q_B{}^+$ and $Q_C{}^+$ are plotted in Figure 12-17. The D input equations derived from the maps are

$$\begin{aligned} D_A &= A^+ = A' \\ D_B &= B^+ = BA' + B'A = B \oplus A \\ D_C &= C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA \end{aligned} \tag{12-2}$$

which give the same logic circuit as was obtained by inspection.

FIGURE 12-17
Karnaugh Maps for D Flip-Flops



Present State			Next State		
C	B	A	C^+	B^+	A^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

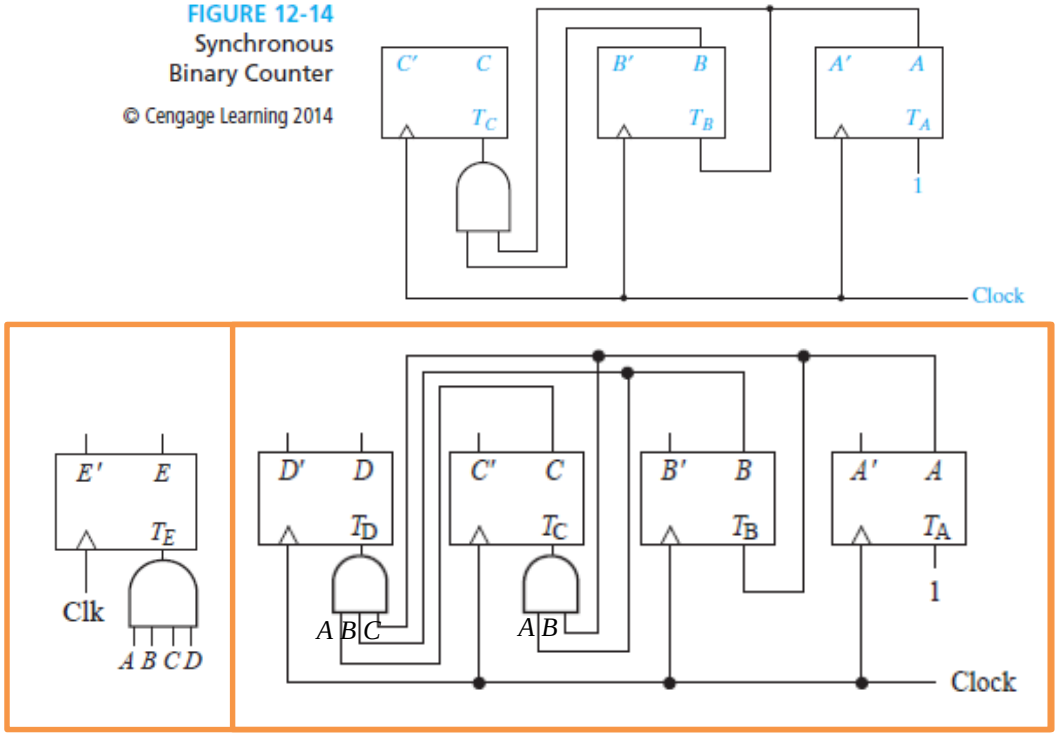
Design of Binary Counters

Binary Counter with T Flip-Flops:

- **3-bit, 4-bit, and 5-bit binary counters with T flip-flops**

Present State $D\,C\,B\,A$	Next State $D^+C^+B^+A^+$	Flip-Flop Inputs $T_D T_C T_B T_A$
0 0 0 0	0 0 0 1	0 0 0 1
0 0 0 1	0 0 1 0	0 0 1 1
0 0 1 0	0 0 1 1	0 0 0 1
0 0 1 1	0 1 0 0	0 1 1 1
0 1 0 0	0 1 0 1	0 0 0 1
0 1 0 1	0 1 1 0	0 0 1 1
0 1 1 0	0 1 1 1	0 0 0 1
0 1 1 1	1 0 0 0	1 1 1 1
1 0 0 0	1 0 0 1	0 0 0 1
1 0 0 1	1 0 1 0	0 0 1 1
1 0 1 0	1 0 1 1	0 0 0 1
1 0 1 1	1 1 0 0	0 1 1 1
1 1 0 0	1 1 0 1	0 0 0 1
1 1 0 1	1 1 1 0	0 0 1 1
1 1 1 0	1 1 1 1	0 0 0 1
1 1 1 1	0 0 0 0	1 1 1 1

FIGURE 12-14
Synchronous Binary Counter



5-bit binary counter

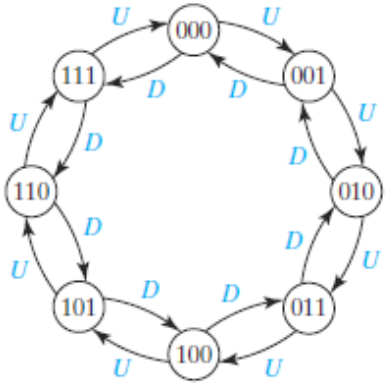
4-bit binary counter

Design of Binary Counters

Binary Up-Down Counter:

- The transition graph and table for up-down counter are shown below.

FIGURE 12-18
Transition Graph and Table for Up-Down Counter

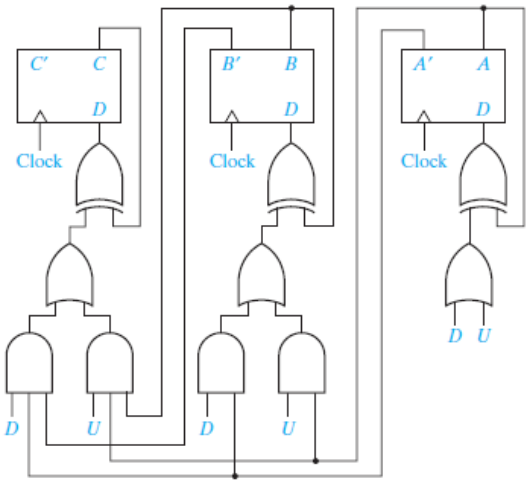


	$C^+B^+A^+$	
CBA	U	D
000	001	111
001	010	000
010	011	001
011	100	010
100	101	011
101	110	100
110	111	101
111	000	110

- The up-down counter can be implemented using D flip-flops and gates as shown below. The corresponding logic equations are

$$D_A = A^+ = A \oplus (U + D)$$
$$D_B = B^+ = B \oplus (UA + DA')$$
$$D_C = C^+ = C \oplus (UBA + DB'A')$$

FIGURE 12-19
Binary Up-Down Counter

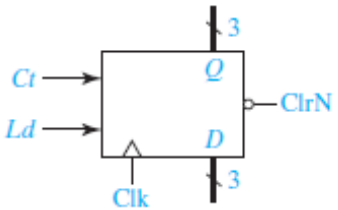


Design of Binary Counters

Loadable Counter with Count Enable:

- The loadable counter has two control signals *Ld* (load) and *Ct* (Count).

FIGURE 12-20
Loadable Counter with Count Enable



(a)

ClrN	*Ld*	*Ct*	C^+	B^+	A^+	
0	X	X	0	0	0	
1	1	X	D_C	D_B	D_A	(load)
1	0	0	*C*	*B*	*A*	(no change)
1	0	1	Present state + 1			

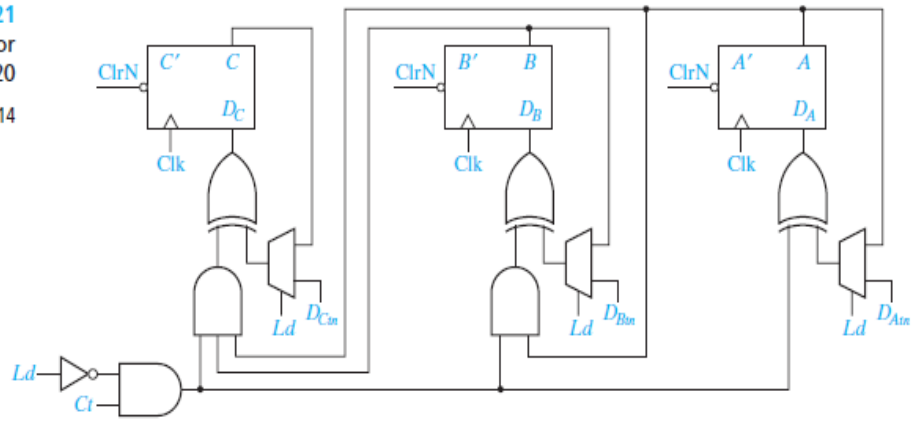
(b)

- The next-state equations for the counter of Figure 12-21 are

$$A^+ = D_A = (Ld'\cdot A + Ld\cdot D_{Ain}) \oplus Ld'\cdot Ct$$
$$B^+ = D_B = (Ld'\cdot B + Ld\cdot D_{Bin}) \oplus Ld'\cdot Ct\cdot A$$
$$C^+ = D_C = (Ld'\cdot C + Ld\cdot D_{Cin}) \oplus Ld'\cdot Ct\cdot B\cdot A$$

When *Ld* = 0 and *Ct* = 1, these equations reduce to $A^+ = A'$, $B^+ = B \oplus A$, and $C^+ = C \oplus BA$, which are the equations previously derived for a 3-bit counter.

FIGURE 12-21
Circuit for Figure 12-20



Counters for Other Sequences

Counter for Other Sequences (Example):

- We will design a counter for the transition table shown in Table 12-3 using T Flip-Flops.

FIGURE 12-22
Transition Graph for Counter

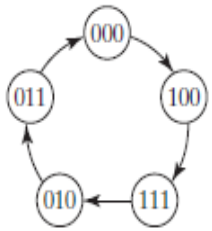
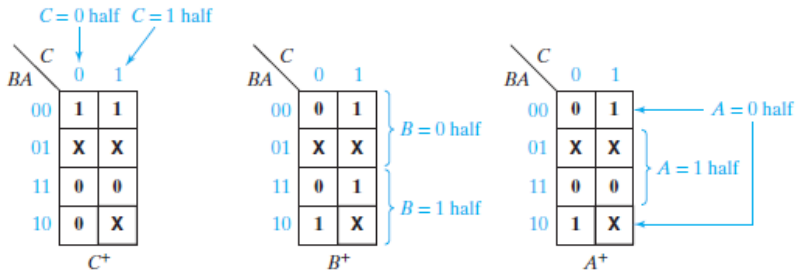


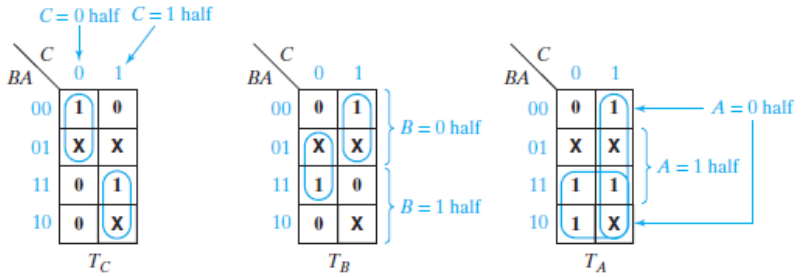
TABLE 12-3
Transition Table for Figure 12-22

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	–	–	–
1	1	0	–	–	–
1	1	1	0	1	0

FIGURE 12-23



(a) Next-state maps for Table 12-3

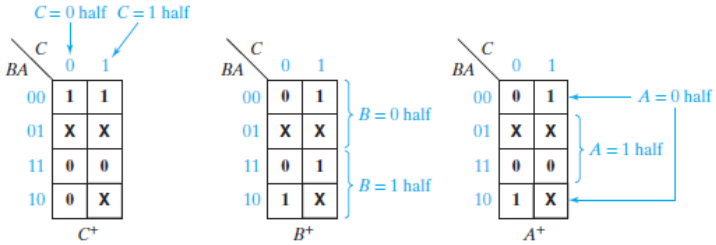


(b) Derivation of T inputs

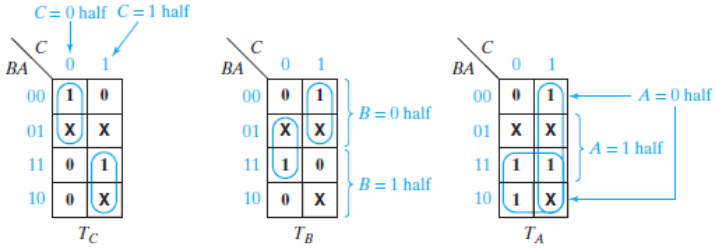
Counters for Other Sequences

Determination of Flip-Flop inputs from Nest-State Equations

FIGURE 12-23



(a) Next-state maps for Table 12-3



$T_C = C'B' + CB$ $T_B = C'A + CB'$ $T_A = C + B$

(b) Derivation of T inputs

TABLE 12-9
Determination of Flip-Flop Input Equations from Next-State Equations Using Karnaugh Maps

Type of Flip-Flop	Input	$Q = 0$		$Q = 1$		Rules for Forming Input Map From Next-State Map*	
		$Q^+ = 0$	$Q^+ = 1$	$Q^+ = 0$	$Q^+ = 1$	$Q = 0$ Half of Map	$Q = 1$ Half of Map
Delay	D	0	1	0	1	no change	no change
Toggle	T	0	1	1	0	no change	complement
Set-Reset	S	0	1	0	X	no change	replace 1's with X's**
	R	X	0	1	0	replace 0's with X's**	complement
J-K	J	0	1	X	X	no change fill	fill in with X's
	K	X	X	1	0	in with X's	complement

Q^+ means the next state of Q
X is a don't-care
*Always copy X's from the next-state map onto the input maps first.
**Fill in the remaining squares with 0's.

Counters for Other Sequences

Input for T Flip-Flop:

- Q represents the present state of the flip-flop (C, B, or A) under consideration, and Q^+ represents the next state (C^+, B^+, or A^+) of the same flip-flop.
- T=1 whenever a state of change is required.

TABLE 12-4
Input for
T Flip-Flop

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

$T = Q^+ \oplus Q$

Counter Using T Flip-Flops and Timing Diagram:

FIGURE 12-24
Counter Using
T Flip-Flops

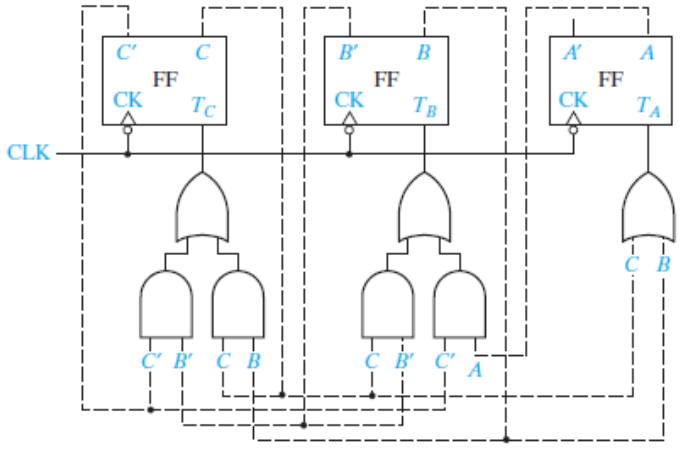
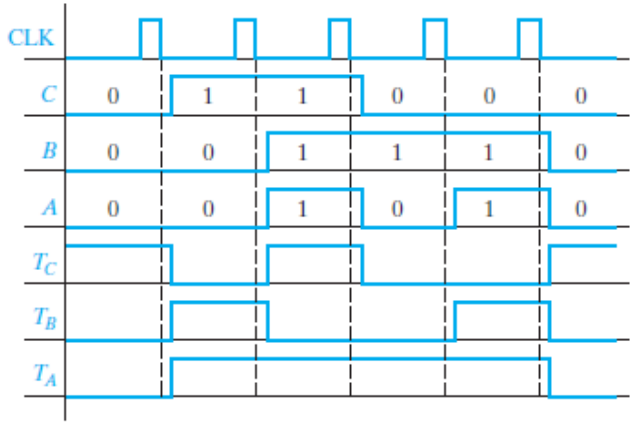


FIGURE 12-25
Timing Diagram
for Figure 12-24



Counters for Other Sequences

Procedure for Designing a Counter Using T Flip-Flops:

1. Form a transition table which gives the next flip-flop states for each combination of present flip-flop states.
2. Plot the next-state maps from the table.
3. Plot a T input map for each flip-flop. When filling in the T_Q map, T_Q must be 1 whenever $Q^+ \neq Q$. This means that the T_Q map can be formed from the Q^+ map by complementing the $Q = 1$ half of the map and leaving the $Q = 0$ half unchanged.
4. Find the T input equations from the maps and realize the circuit.

TABLE 12-9
Determination of Flip-Flop Input Equations from Next-State Equations Using Karnaugh Maps

Type of Flip-Flop	Input	$Q = 0$		$Q = 1$		Rules for Forming Input Map From Next-State Map*	
		$Q^+ = 0$	$Q^+ = 1$	$Q^+ = 0$	$Q^+ = 1$	$Q = 0$ Half of Map	$Q = 1$ Half of Map
Delay	D	0	1	0	1	no change	no change
Toggle	T	0	1	1	0	no change	complement
Set-Reset	S	0	1	0	X	no change	replace 1's with X's**
	R	X	0	1	0	replace 0's with X's**	complement
J-K	J	0	1	X	X	no change fill	fill in with X's
	K	X	X	1	0	in with X's	complement

Q^+ means the next state of Q
X is a don't-care
*Always copy X's from the next-state map onto the input maps first.
**Fill in the remaining squares with 0's.

Counters for Other Sequences

Counter Design using D Flip-Flops:

- For a D flip-flop, $Q^+ = D$, so the D input map is identical with the next-state map.

FIGURE 12-22
Transition Graph for Counter

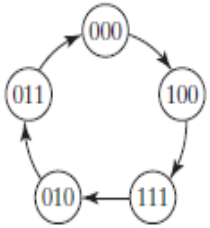
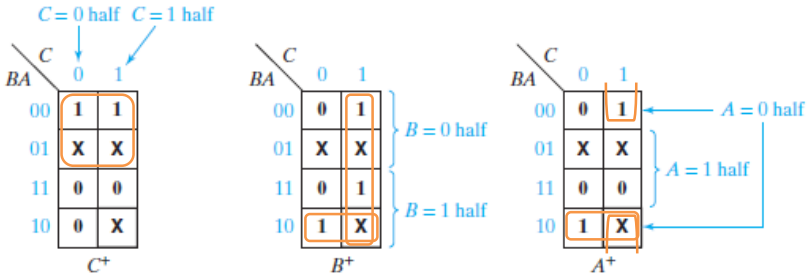


TABLE 12-3
Transition Table for Figure 12-22

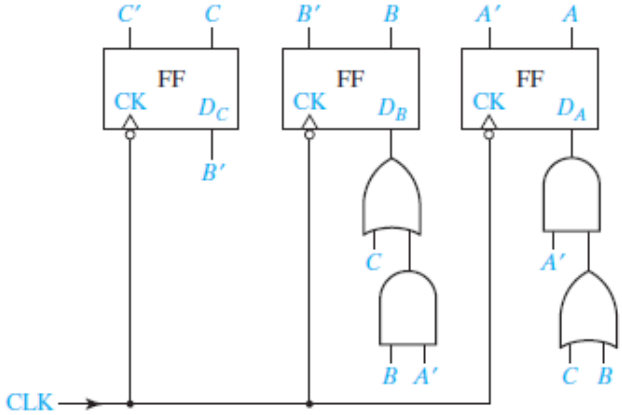
C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	–	–	–
1	1	0	–	–	–
1	1	1	0	1	0

FIGURE 12-23



$D_C = C^+ = B'$
$D_B = B^+ = C + BA'$
$D_A = A^+ = CA' + BA' = A'(C+B)$

FIGURE 12-27
Counter of Figure 12-22 Using D Flip-Flops

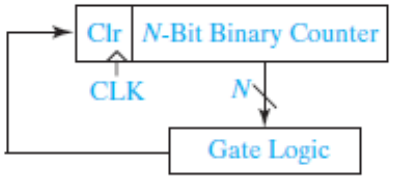


Counters for Other Sequences

Binary Counter with Clear:

- Counters and shift registers with clear, preset, or parallel load capability can also be used to generate non-binary count cycles.
- Consider a binary counter with a clear input as shown Figure 12-28.

FIGURE 12-28
Binary Counter with Clear



- Synchronous Clear: Clr = Q_3Q_0
- Asynchronous Clear: Clr = Q_3Q_1

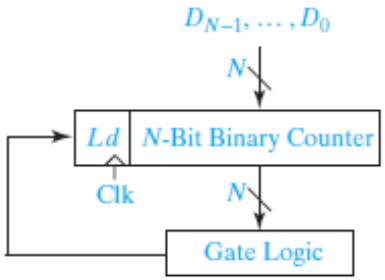
Binary Counter with Parallel Load:

- A decimal counter that counts using the excess-3 code
  - ✓ The counter must cycle through states 3 to 12. The logic must cycle through 3 through 12.
  - ✓ The logic must generate *Ld* when the counter is in state 12 and the parallel inputs must be 0011.

$$D_3 = 0, D_2 = 0, D_1 = 1, D_0 = 1$$
$$Ld = Q_3Q_2$$

  - ✓ States 0, 1, 2, 13, 14 and 15 are don't cares.

FIGURE 12-29
Binary Counter with Parallel Load



Counter Design Using S-R and J-K Flip-Flops

Procedure for Counter Design Using S-R Flip-Flops:

- Instead of deriving an input equation for each D or T flip-flop, the S and R input equations must be derived.

FIGURE 12-22
Transition Graph for Counter

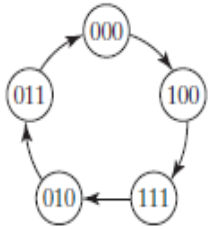


TABLE 12-5
S-R Flip-Flop Inputs

(a)

S	R	Q	Q^+	
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	–	inputs not allowed
1	1	1	–	

(b)

Q	Q^+	S	R
0	0	0	0
		0	1
0	1	1	0
1	0	0	1
1	1	0	0
		1	0

(c)

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

TABLE 12-3
Transition Table for Figure 12-22

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	–	–	–
1	1	0	–	–	–
1	1	1	0	1	0

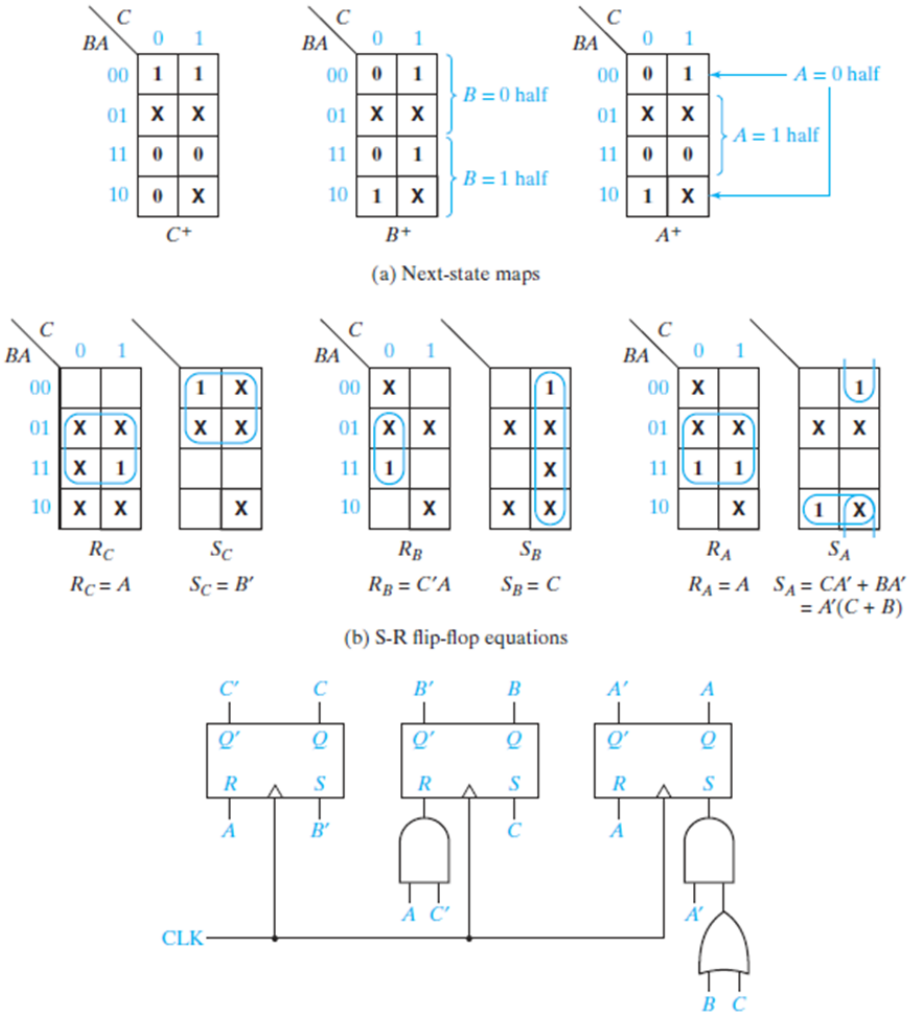
TABLE 12-6

C	B	A	C^+	B^+	A^+	S_C	R_C	S_B	R_B	S_A	R_A
0	0	0	1	0	0	1	0	0	X	0	X
0	0	1	–	–	–	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	–	–	–	X	X	X	X	X	X
1	1	0	–	–	–	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1

Counter Design Using S-R and J-K Flip-Flops

Counter Using S-R Flip Flops:

FIGURE 12-30
Counter of Figure 12-22 Using S-R Flip-Flops



Counter Design Using S-R and J-K Flip-Flops

Procedure for Counter Design Using J-K Flip- Flops:

- The procedure used to design a counter with J-K flip-flops is very similar to that used for S-R flip-flops, except that J and K can be 1 simultaneously, in which case the flip-flop changes state.

FIGURE 12-22
Transition Graph for Counter

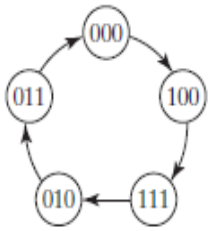


TABLE 12-3
Transition Table for Figure 12-22

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	–	–	–
1	1	0	–	–	–
1	1	1	0	1	0

TABLE 12-7
J-K Flip-Flop Inputs

(a)

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(b)

Q	Q^+	J	K
0	0	0	0
		0	1
0	1	1	0
		1	1
1	0	0	1
		1	1
1	1	0	0
		1	0

(c)

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

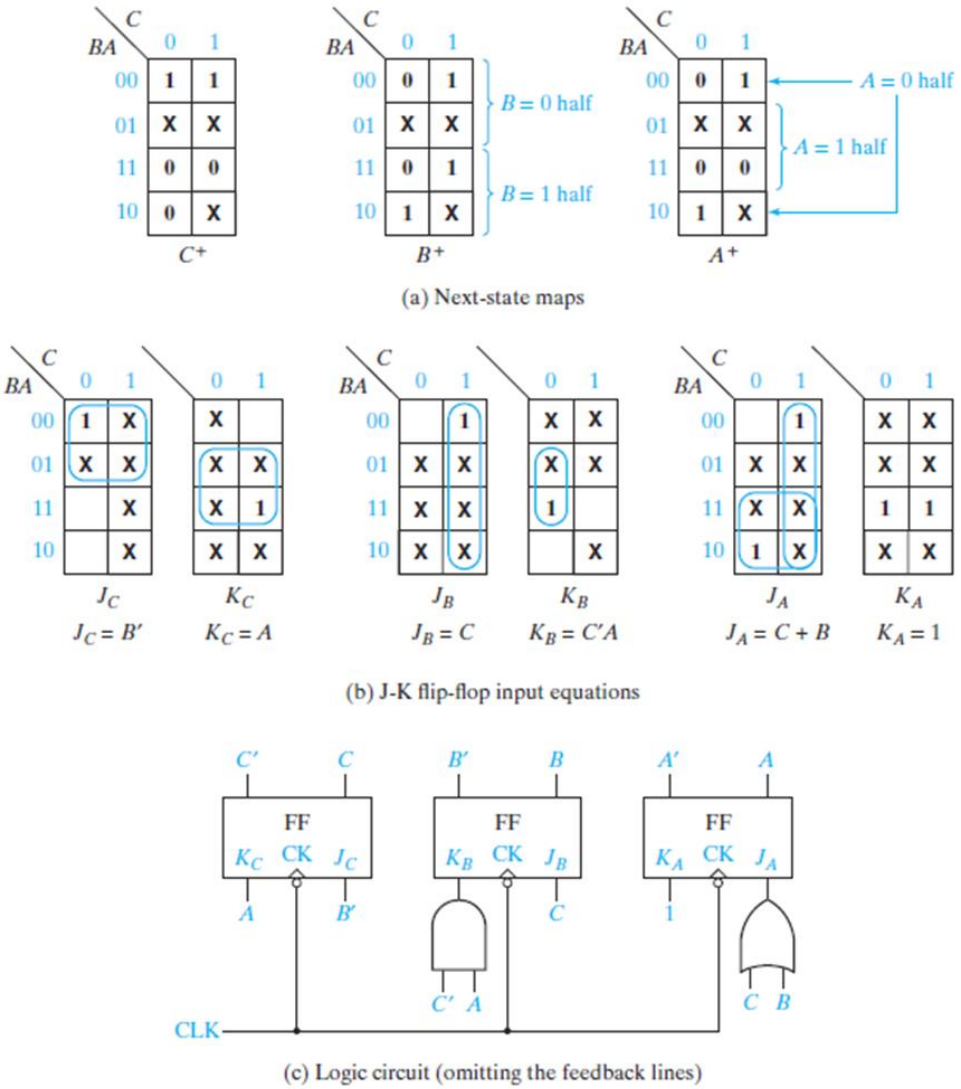
TABLE 12-8

C	B	A	C^+	B^+	A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	–	–	–	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	–	–	–	X	X	X	X	X	X
1	1	0	–	–	–	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1

Counter Design Using S-R and J-K Flip-Flops

Counter Design Using J-K Flip-Flop:

FIGURE 12-31
Counter of Figure 12-22 Using J-K Flip-Flops



(a) Next-state maps

(b) J-K flip-flop input equations

(c) Logic circuit (omitting the feedback lines)

Derivation of Flip-Flop Input Equation - Summary

Summary:

- The input equation for the flip-flops in a sequential circuit may be derived from the next-state equations by using truth tables or by using Karnaugh maps.
- For the D flip-flop, the input is the same as the next state.
- For the T flip-flop, the input is 1 whenever a state change is required.
- For the S-R flip-flop, *S* is 1 whenever the flip-flop must be set to 1 and *R* is 1 when it must be reset to 0.
- For a J-K flip-flop, the J and K inputs are the same as S and R, respectively, except that when one input is 1 the other input is X.

TABLE 12-9
Determination of Flip-Flop Input Equations from Next-State Equations Using Karnaugh Maps

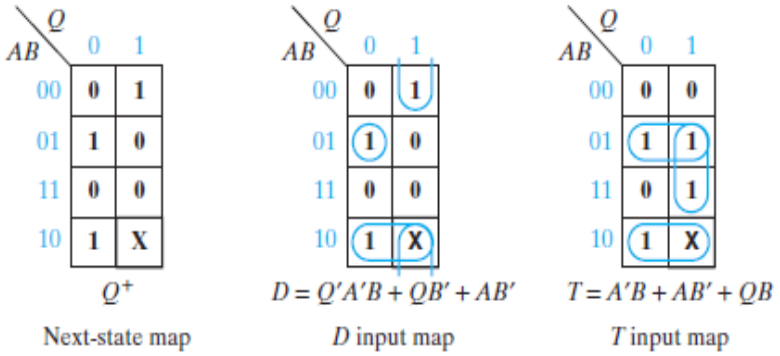
Type of Flip-Flop	Input	$Q = 0$		$Q = 1$		Rules for Forming Input Map From Next-State Map*	
		$Q^+ = 0$	$Q^+ = 1$	$Q^+ = 0$	$Q^+ = 1$	$Q = 0$ Half of Map	$Q = 1$ Half of Map
Delay	*D*	0	1	0	1	no change	no change
Toggle	*T*	0	1	1	0	no change	complement
Set-Reset	*S*	0	1	0	X	no change	replace 1's with X's**
	R	X	0	1	0	replace 0's with X's**	complement
J-K	*J*	0	1	X	X	no change fill	fill in with X's
	K	*X*	X	1	0	in with X's	complement

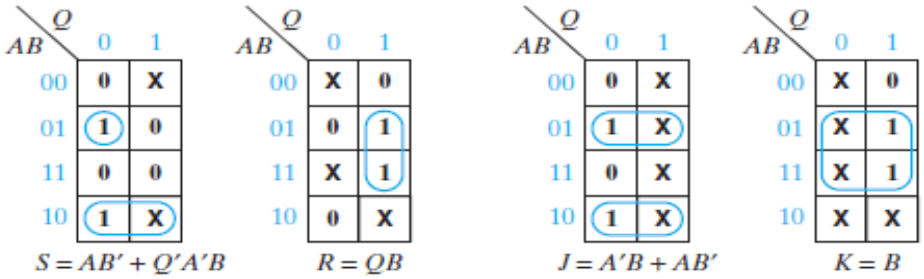
Q^+ means the next state of Q
X is a don't-care
*Always copy X's from the next-state map onto the input maps first.
**Fill in the remaining squares with 0's.

Derivation of Flip-Flop Input Equation - Summary

Example 1:

Example (illustrating the use of Table 12-9)





For the S-R flip-flop, note that when $Q = 0$, $R = X$ if $Q^+ = 0$; and when $Q = 1$, $R = 1$ if $Q^+ = 0$. Therefore, to form the R map from the Q^+ map, replace 0's with X's on the $Q = 0$ half of the map and replace 0's with 1's on the $Q = 1$ half (and fill in 0's for the remaining entries). Similarly, to form the S map from the Q^+ map, copy the 1's on the $Q = 0$ half of the map, and replace the 1's with X's on the $Q = 1$ half.