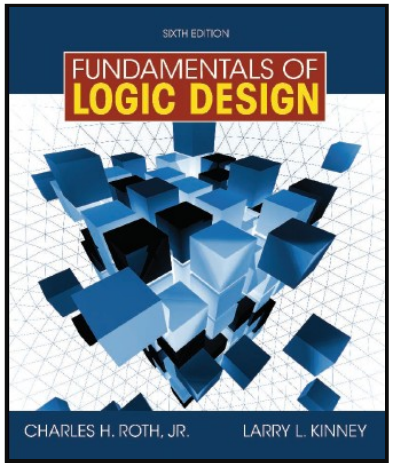
CHAPTER 12

REGISTERS AND COUNTERS



This chapter in the book includes:

Registers and Register Transfers

Several D flip-flops may be grouped together with a common clock to form a register. Because each flip-flop can store one bit of information, a register with four D flip-flops can store four bits of information.

A load signal can be ANDed with the clock to enable and disable loading the registers.

A better approach is to use registers with clock enables if they are available.

Section 12.1 (p. 354)

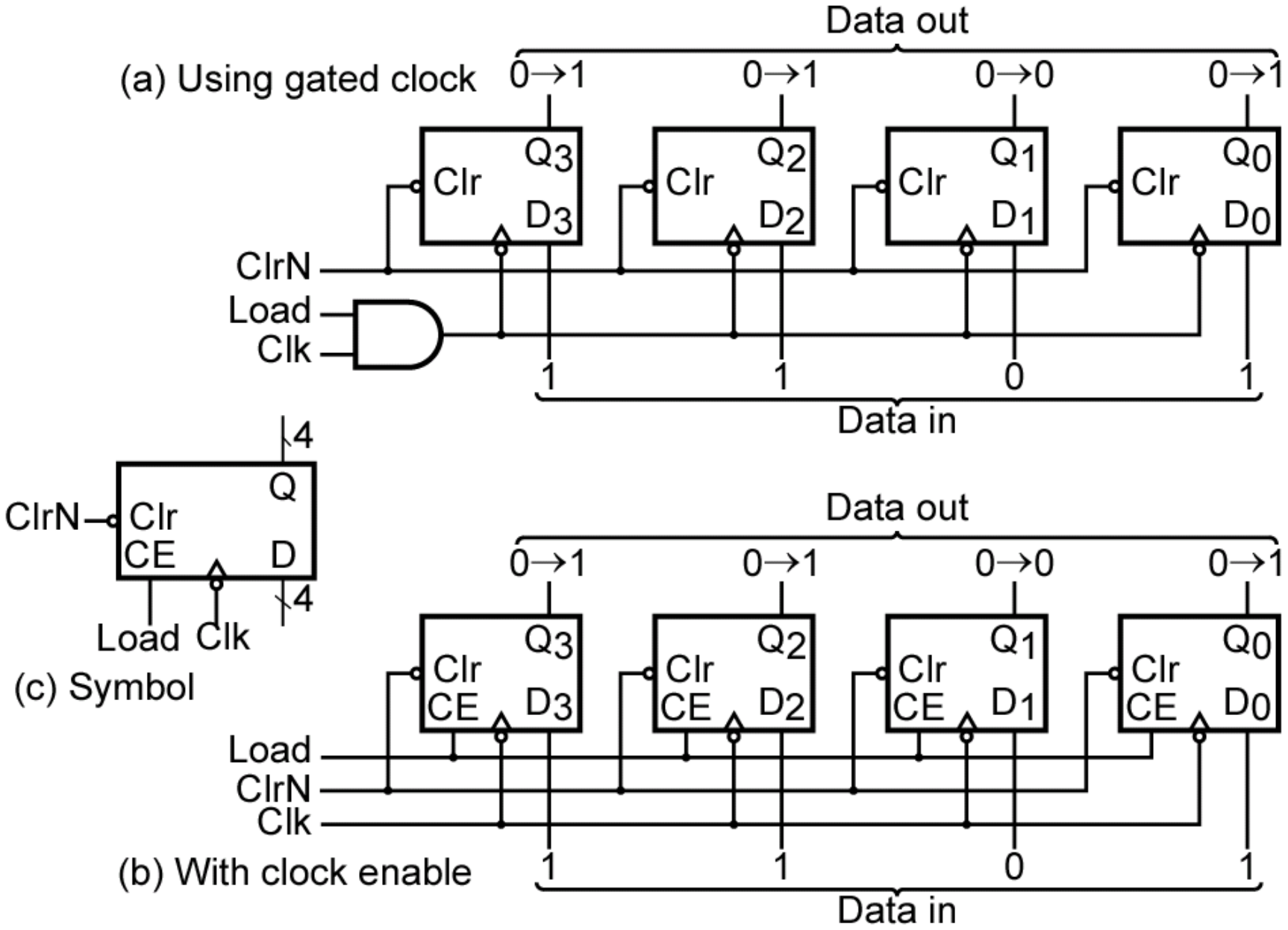


Figure 12-1: **4-Bit D Flip-Flop Registers with Data, Load, Clear, and Clock Inputs**

Data Transfer Between Registers

Transferring data between registers is a common operation in digital systems.

Data can be transferred from the output of one of two registers into a third register using tri-state buffers.

Section 12.1 (p. 355)

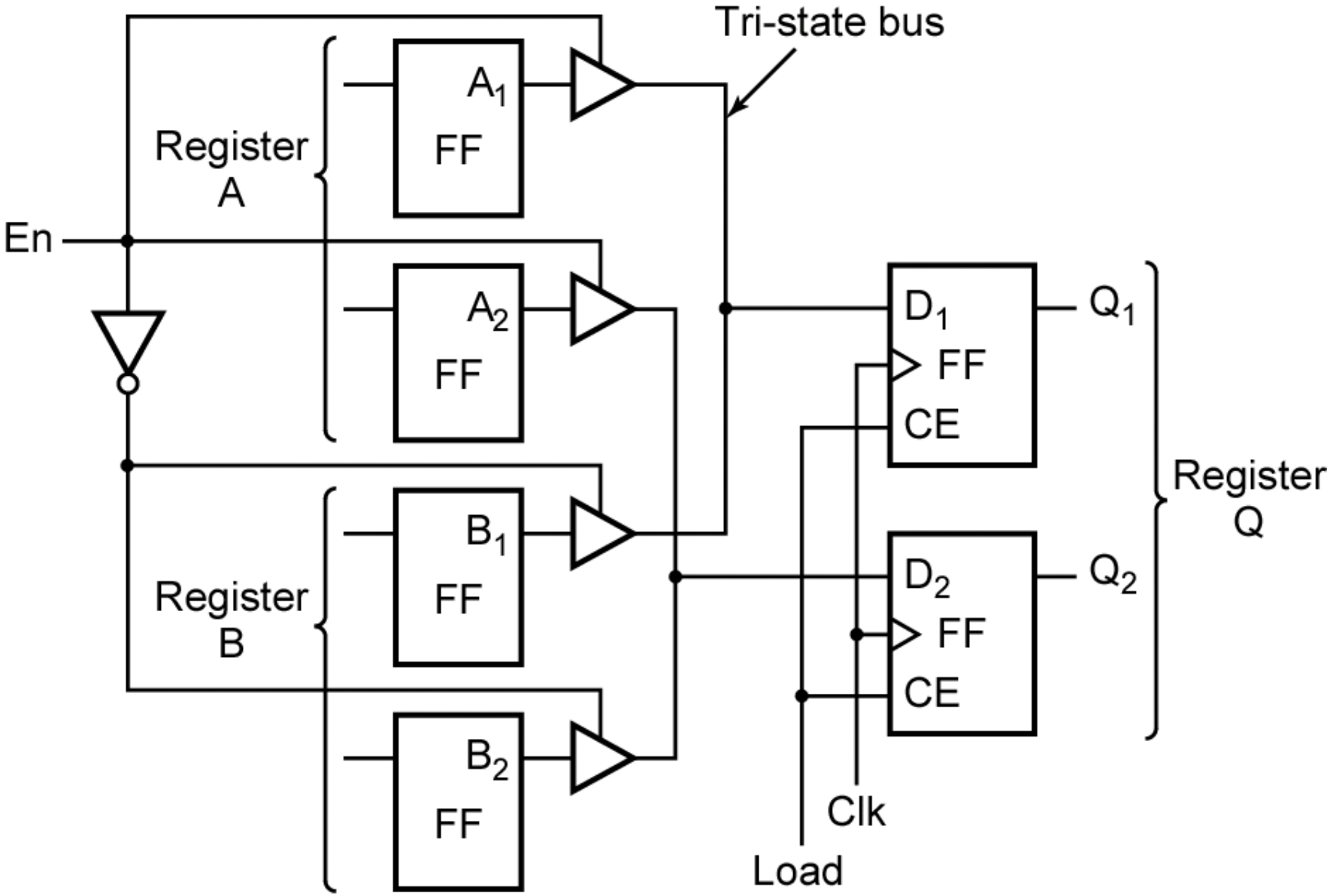


Figure 12-2: **Data Transfer Between Registers**

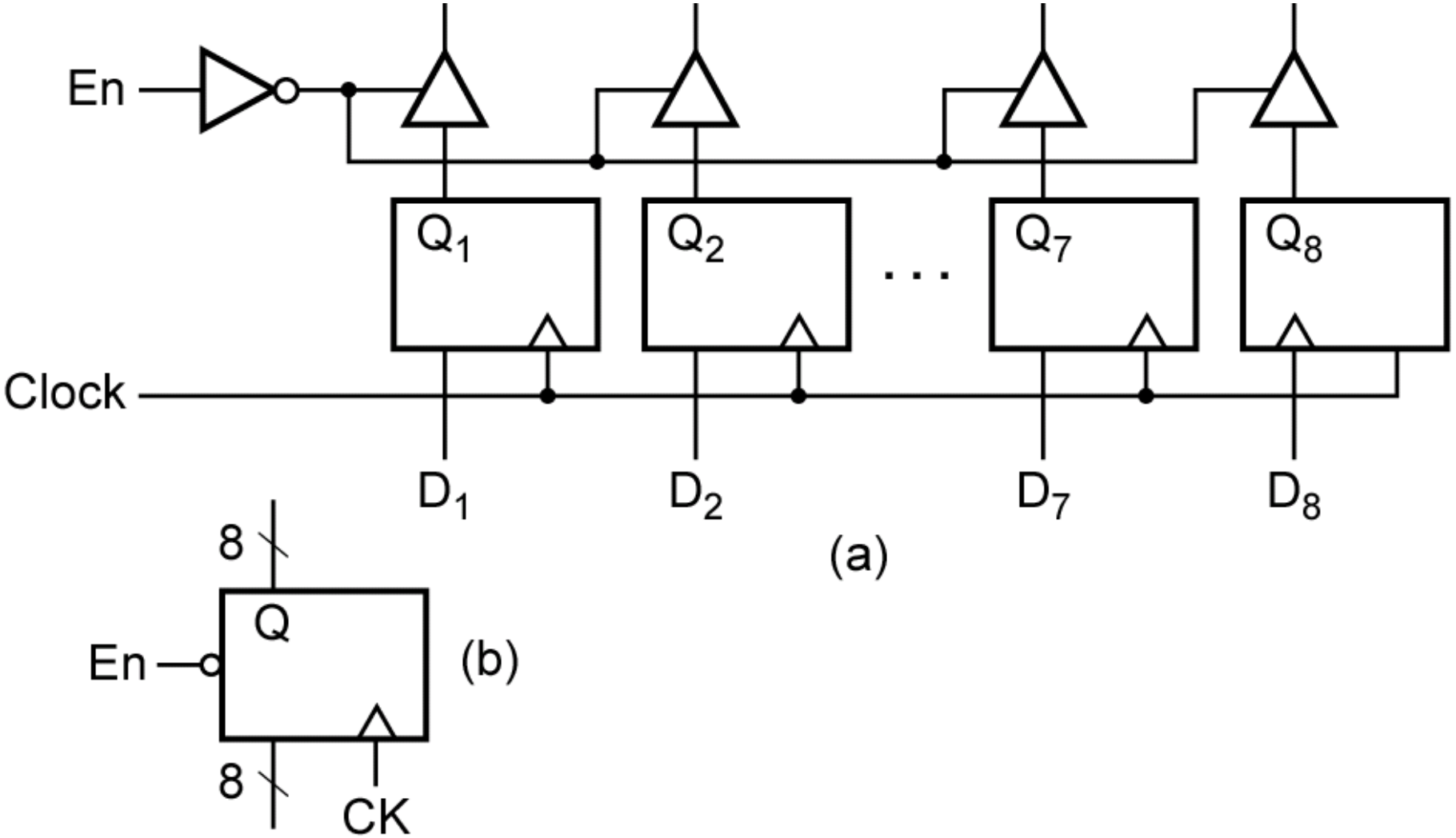


Figure 12-3: **Logic Diagram for 8-Bit Register with Tri-State Output**

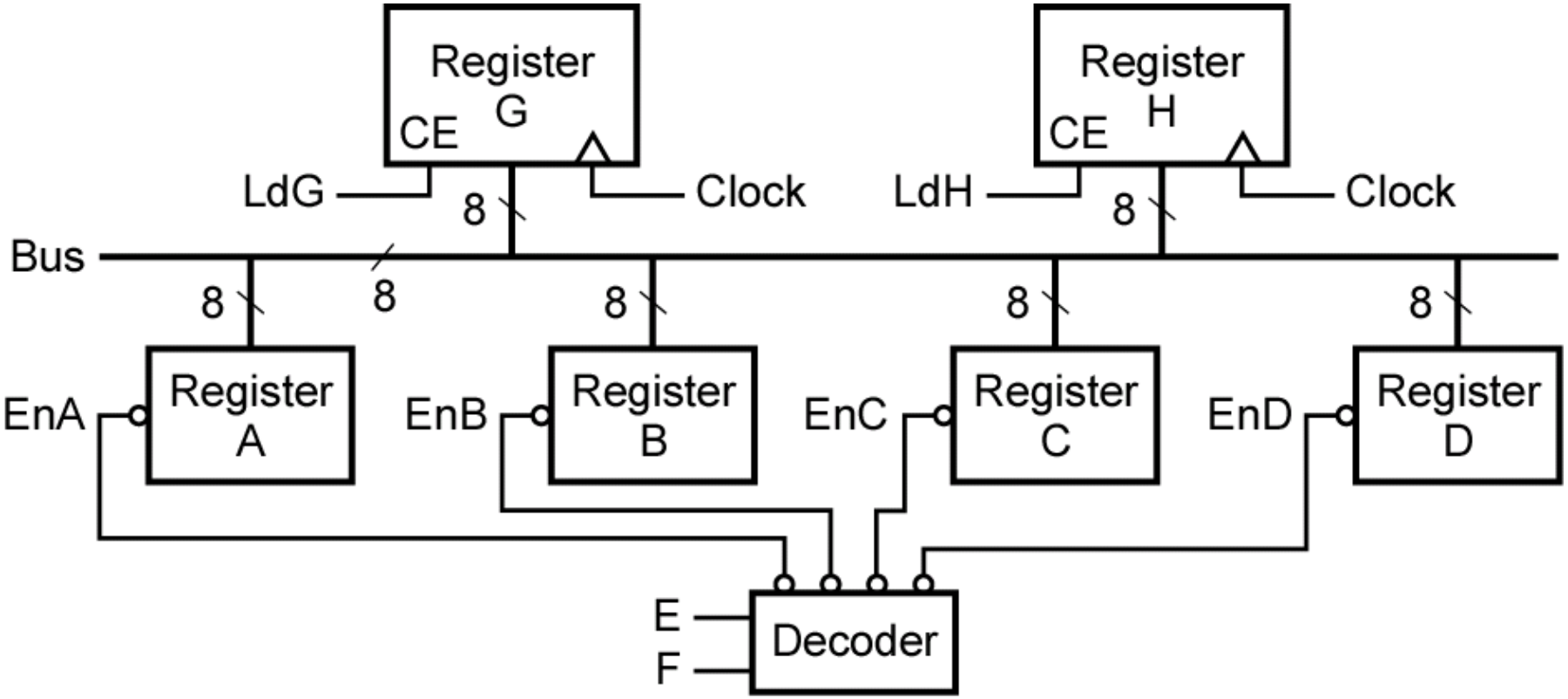


Figure 12-4: **Data Transfer Using a Tri-State Bus**

When *EnA* = 0, the tri-state outputs of register *A* are enabled onto the bus. If *LdG* = 1, these signals on the bus are loaded into register *G* after the rising clock edge (or into register *H* if *LdH* = 1). Similarly, the data in register *B, C,* or *D* is transferred to *G* (or *H*) when *EnB*, *EnC*, or *EnD* is 0, respectively, and *LdG* = 1 (or *LdH* = 1). If *LdG* = *LdH* = 1, both *G* and *H* will be loaded from the bus.

The four enable signals may be generated by a decoder. The operation can be summarized as follows:

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*).

Parallel Adder with Accumulator

In computer circuits, 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.

Section 12.1 (p. 356)

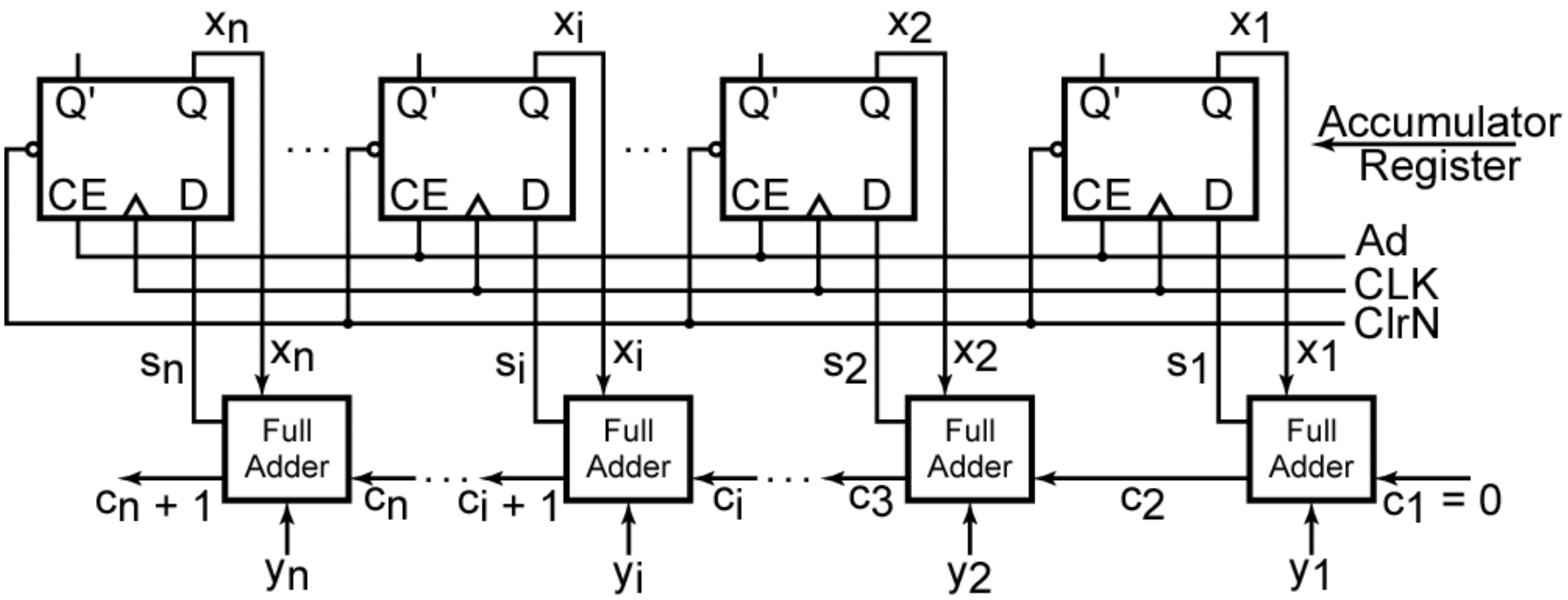


Figure 12-5: **N-Bit Parallel Adder with Accumulator**

Loading Accumulator

Before addition in the previous circuit can take place, the accumulator must be loaded with *X*. This can be accomplished in several ways. The easiest way is to first clear the accumulator using the asynchronous clear inputs on the flip-flops, and then put the *X* data on the *Y* inputs to the adder and add the accumulator in the normal way.

Alternatively, we could add multiplexers at the accumulator inputs so that we could select either the *Y* input data or the adder output to load into the accumulator.

Section 12.1 (p. 357)

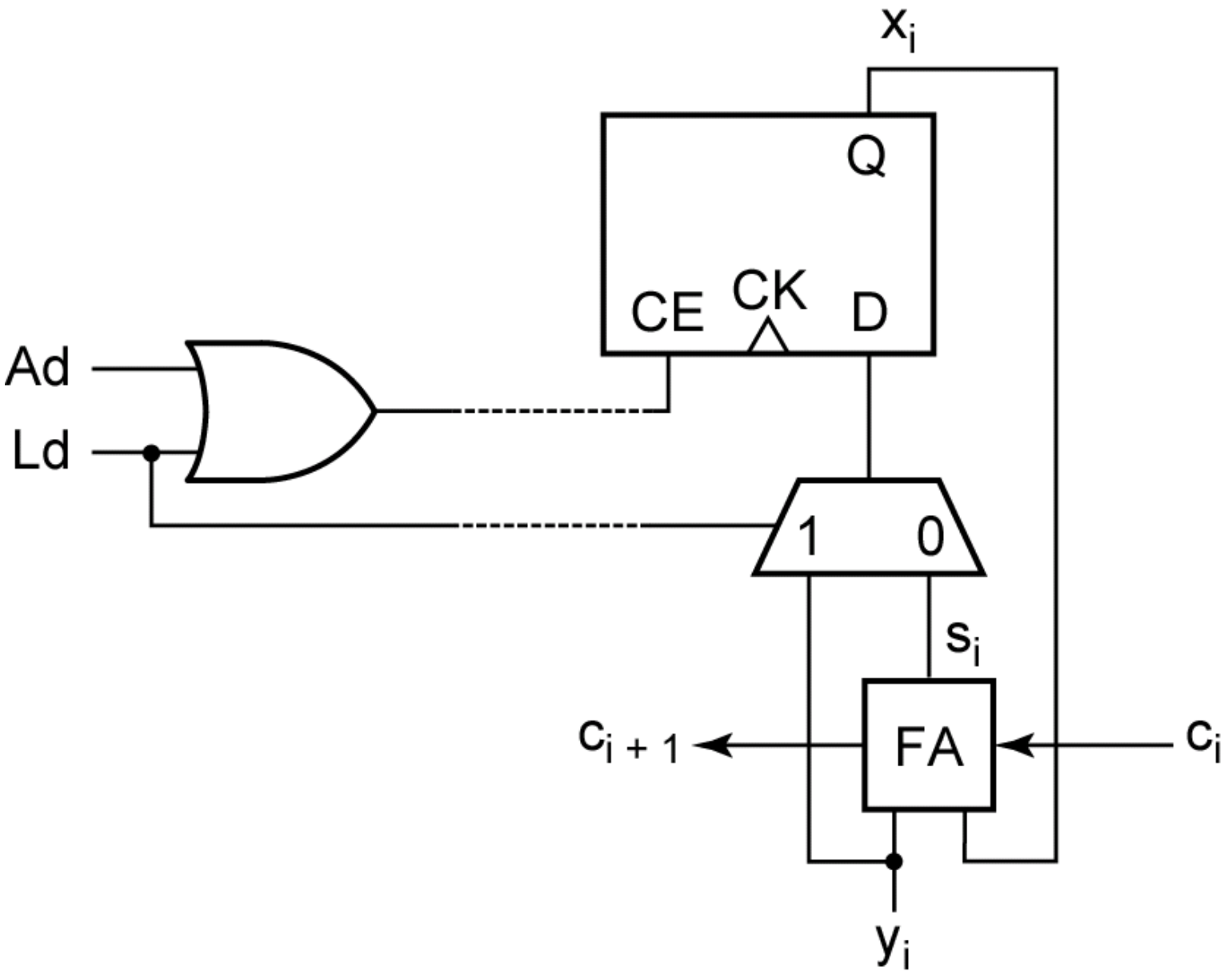
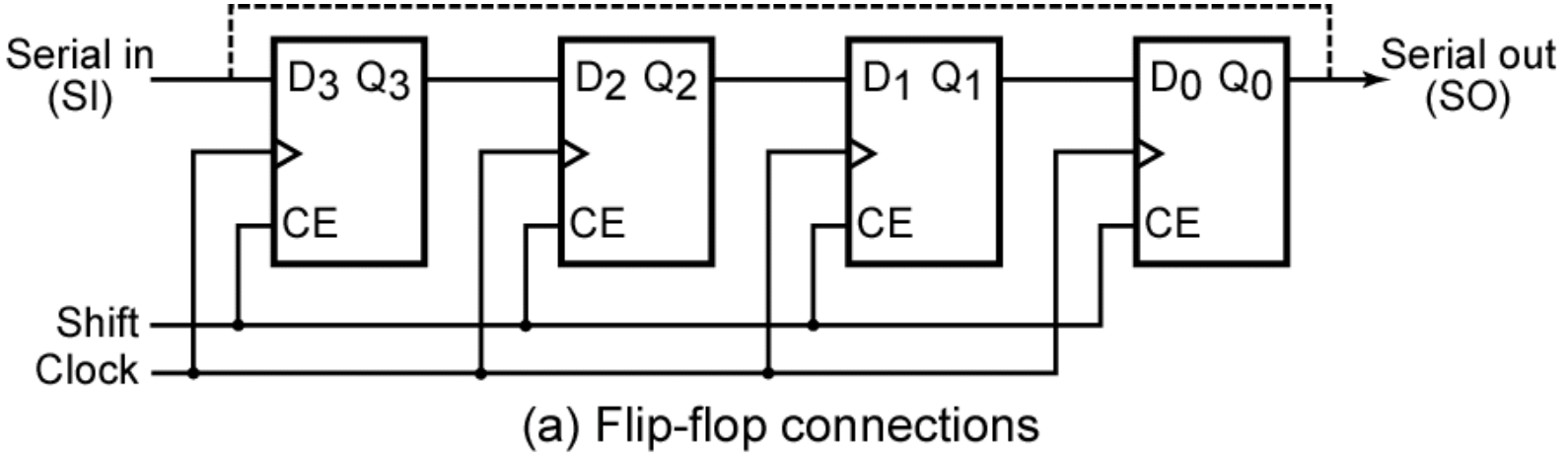
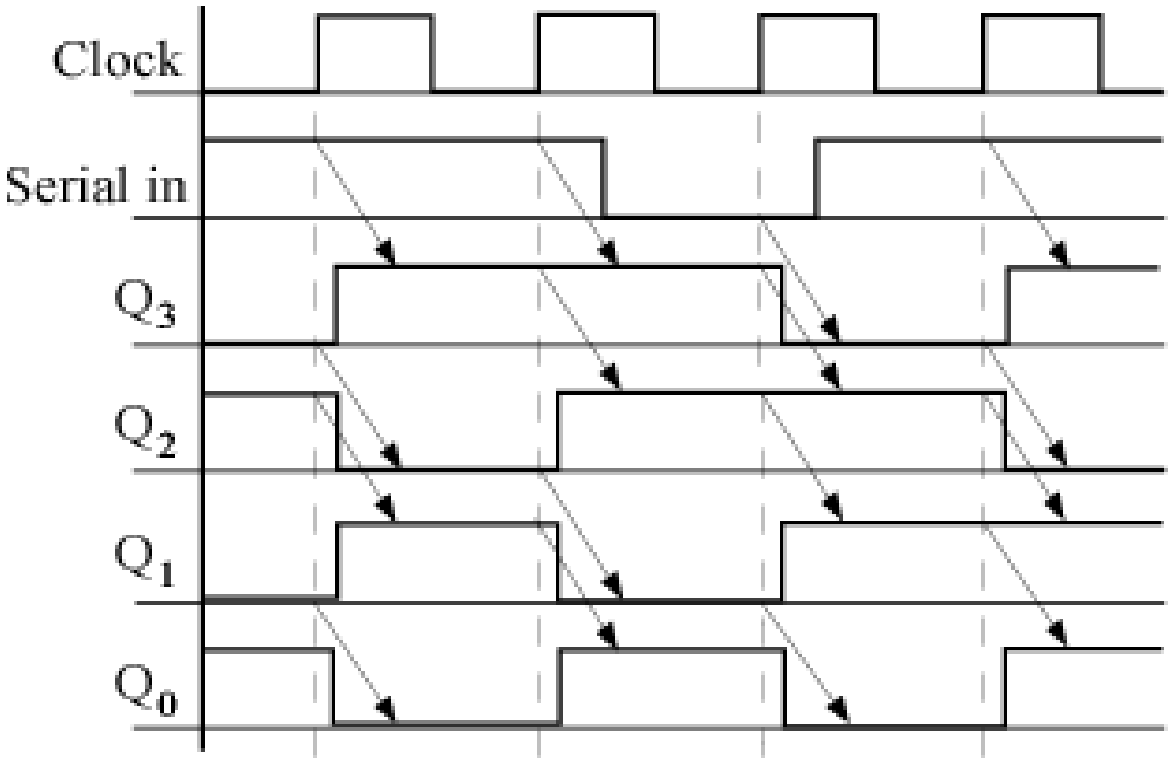


Figure 12-6: **Adder Cell with Multiplexer**



(a) Flip-flop connections

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.



(b) Timing diagram

Figure 12-7: **Right-Shift Register**

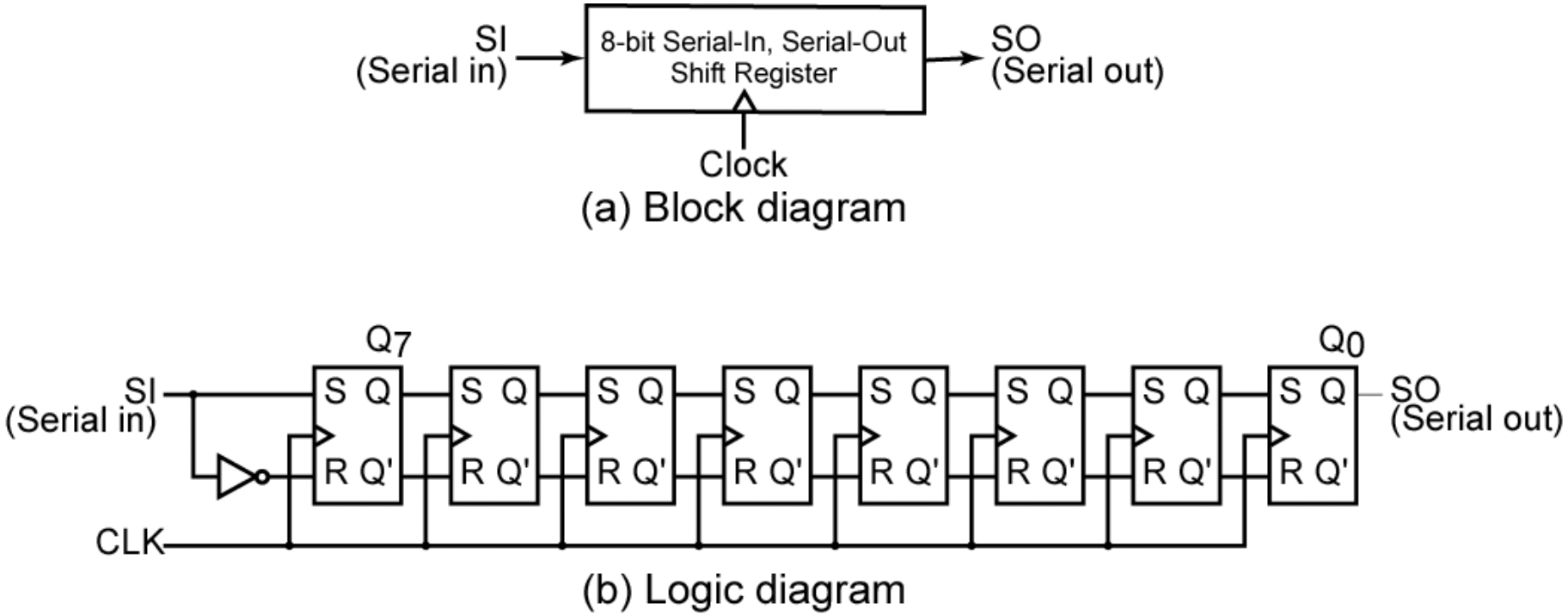


Figure 12-8: **8-Bit Serial-In, Serial-Out Shift Register**

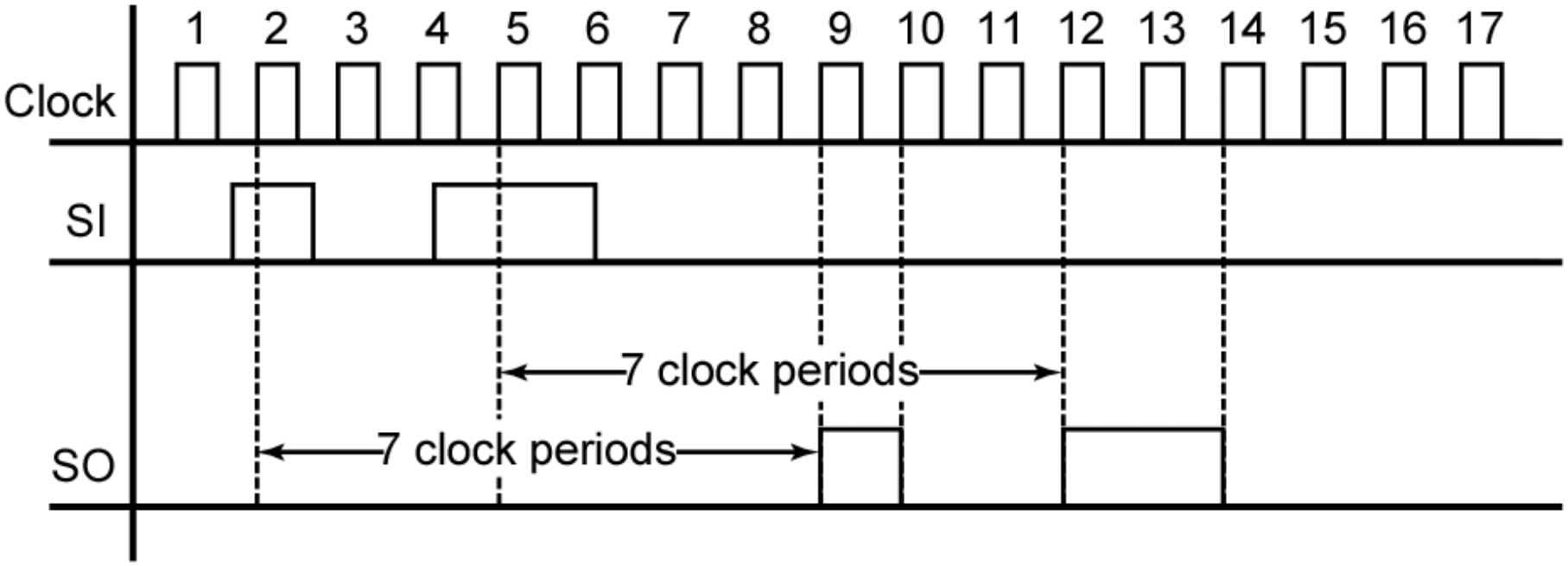


Figure 12-9: **Typical Timing Diagram for Shift Register of Figure 12-8**

Note that the 8th rising edge occurs at the end of the 7th clock period.

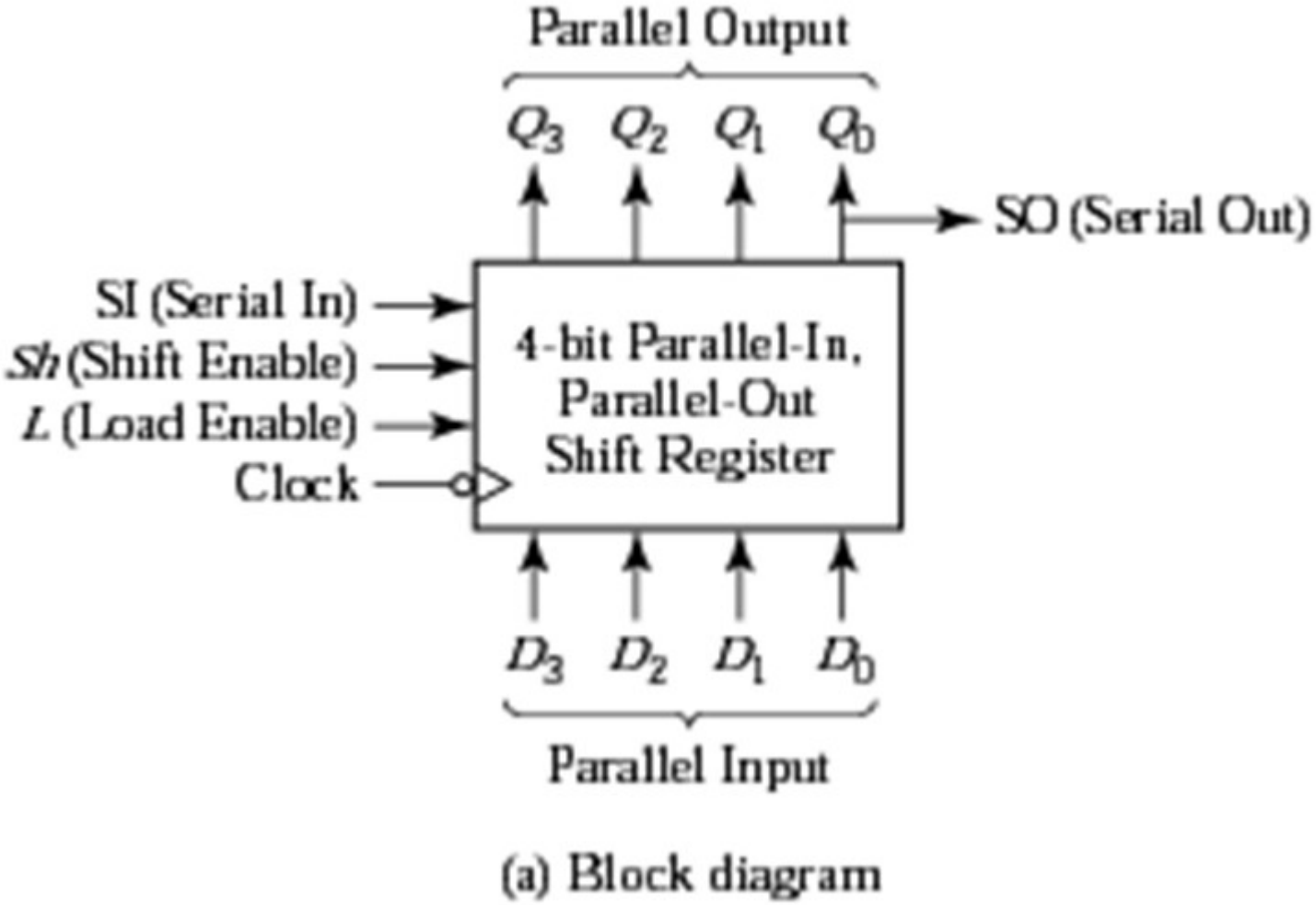
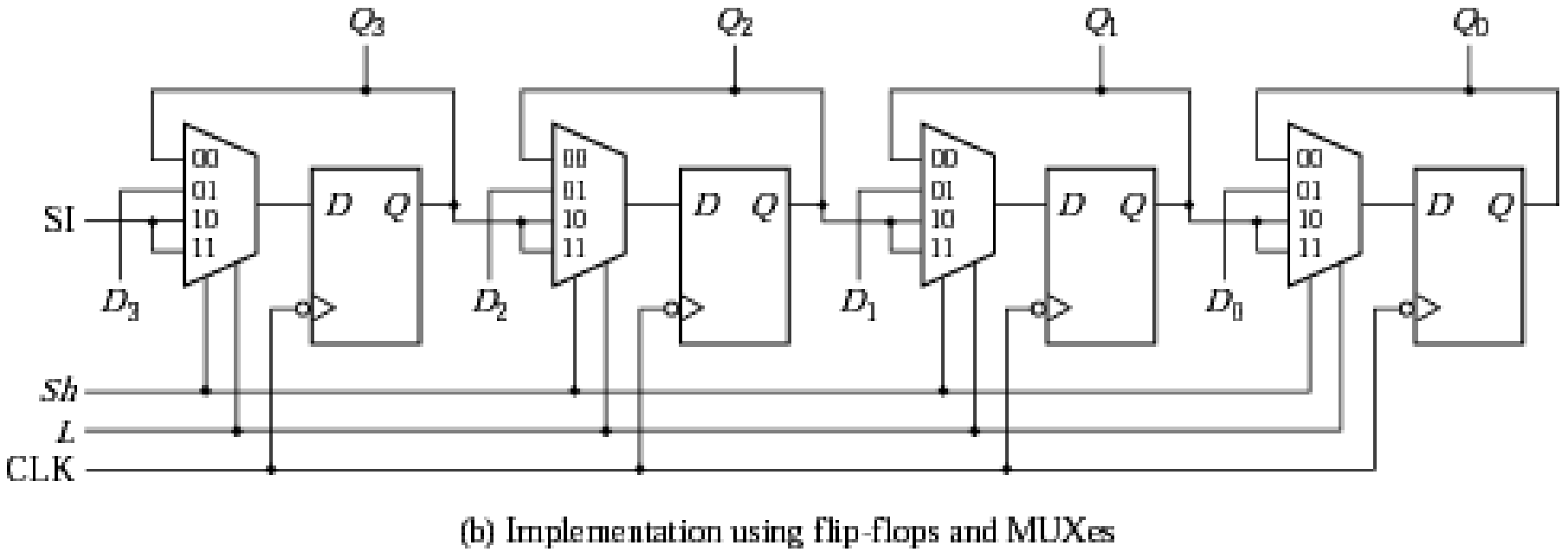


Figure 12-10: **Parallel-In, Parallel-Out, Right Shift Register**



(b) Implementation using flip-flops and MUXes

Figure 12-10: **Parallel-In, Parallel-Out, Right Shift Register**

Table 12-1: Shift Register Operation

Inputs		Next State				Action
Sh (Shift)	Ld (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	
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

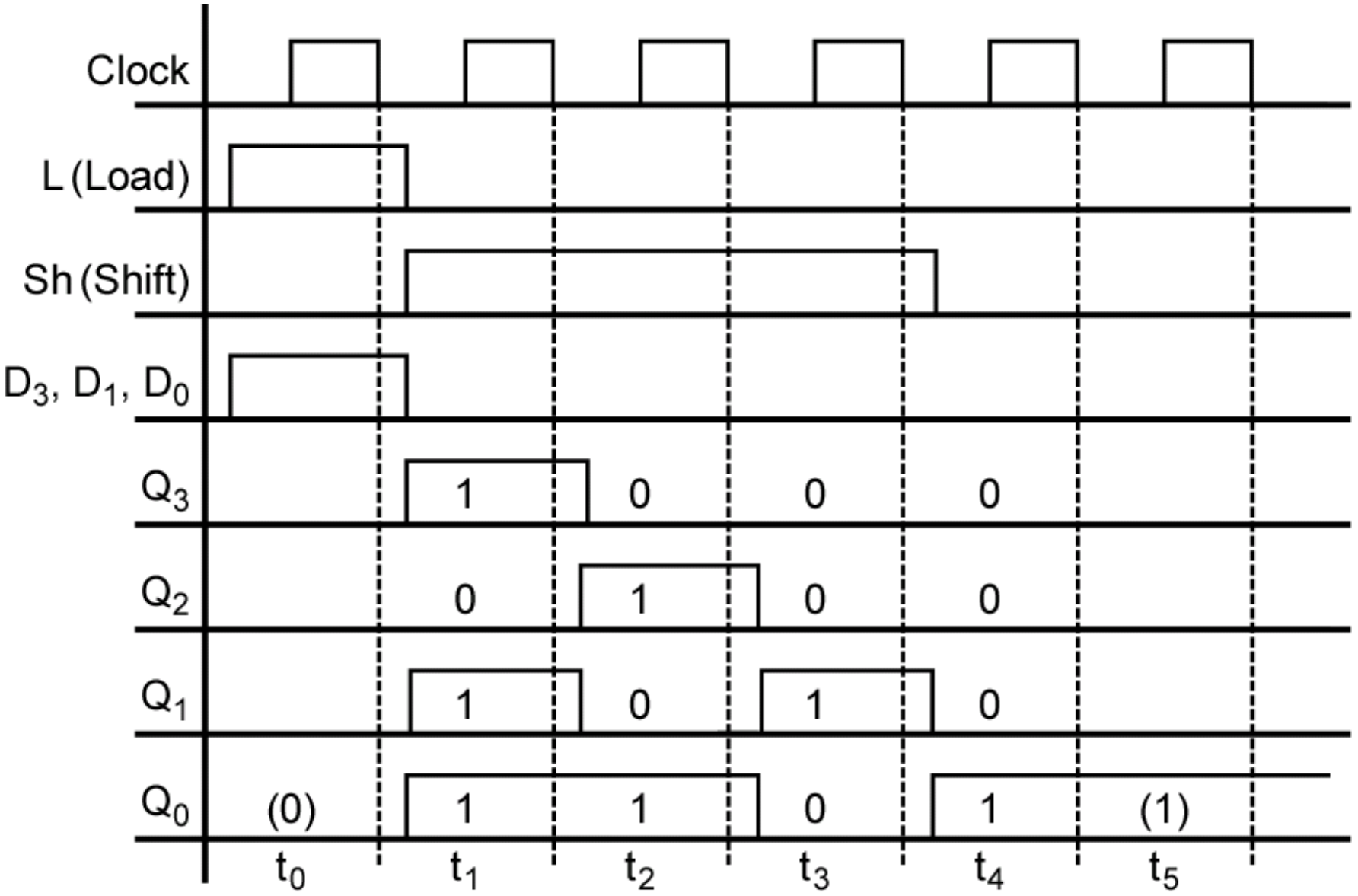
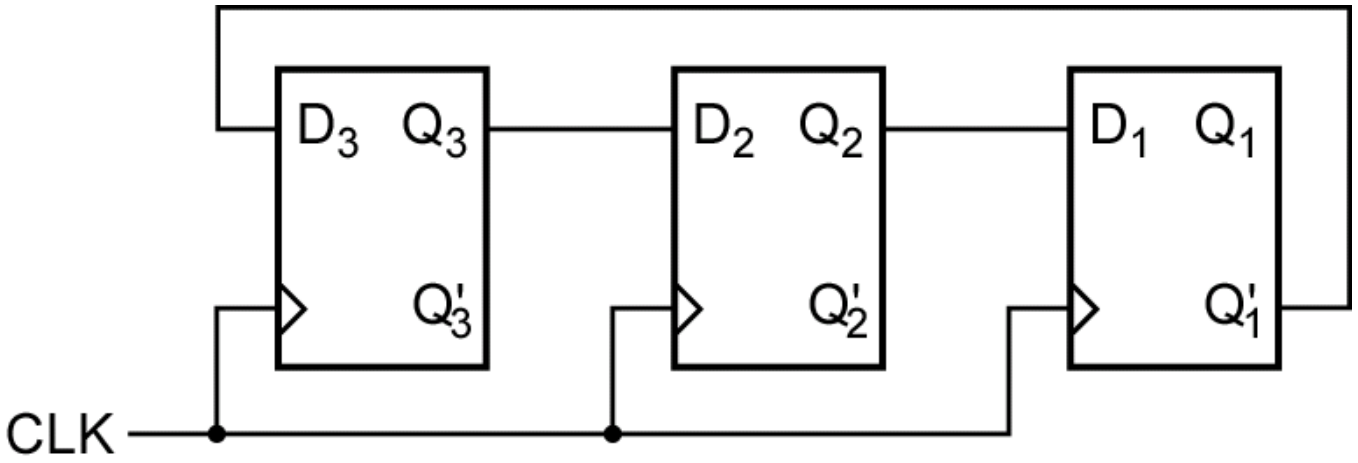
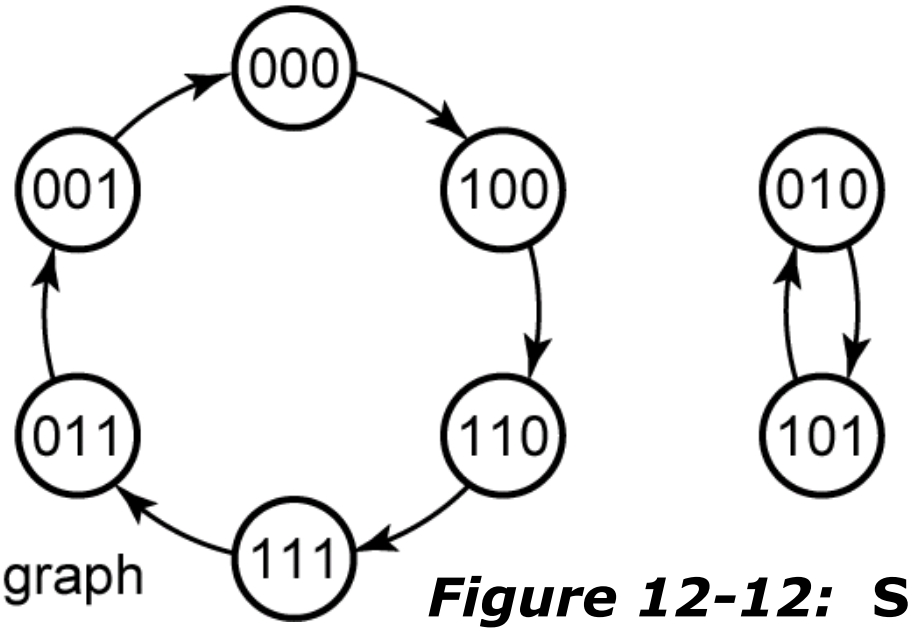


Figure 12-11: **Timing Diagram for Shift Register**



(a) Flip-flop connections



(b) State graph

A circuit that cycles through a fixed sequence of states is called a *counter*.

Figure 12-12: **Shift Register with Inverted Feedback**

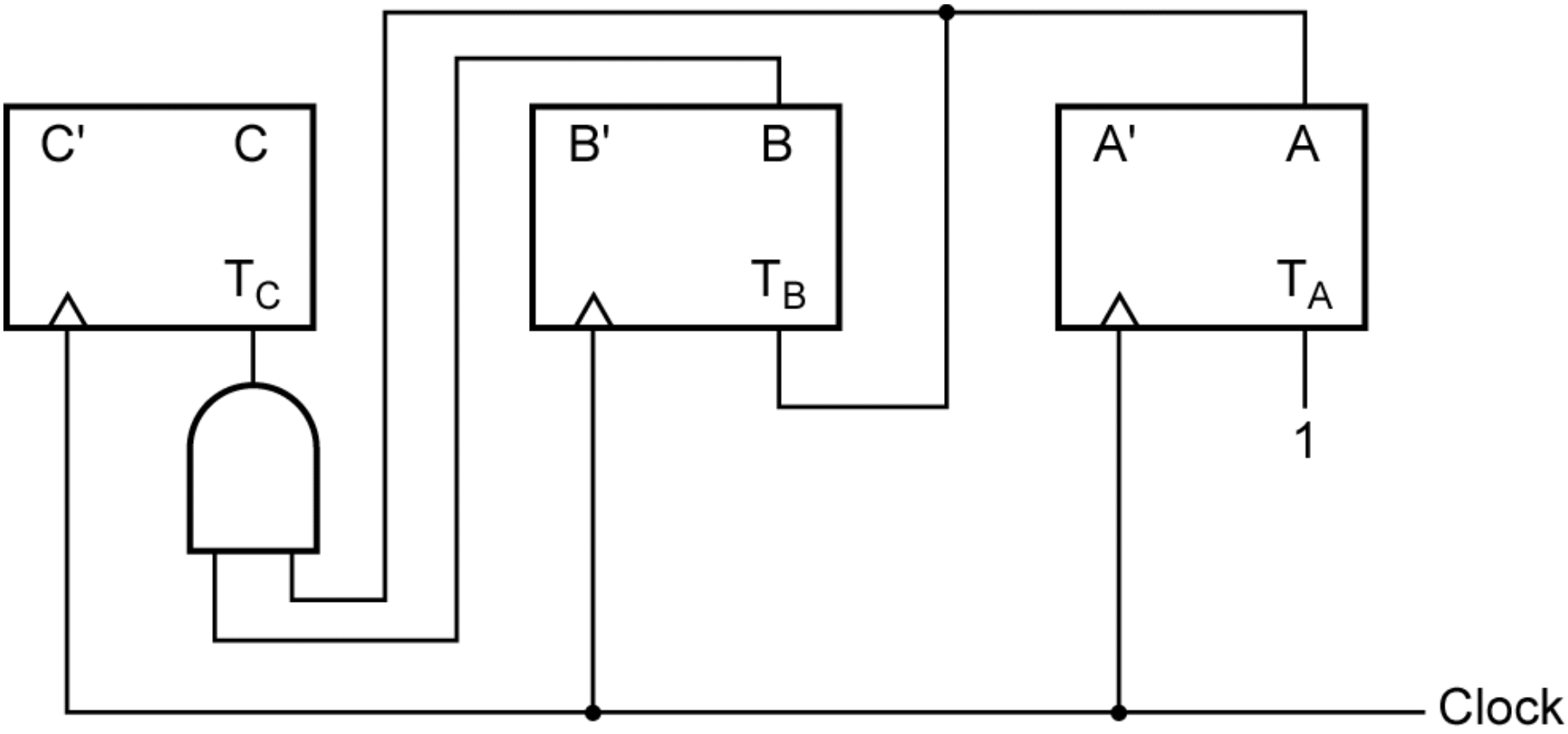


Figure 12-13: **Synchronous Binary Counter**

Design of Binary Counters

1. Create a state graph to count in the desired sequence.
2. Create a state table from the state graph created in (1). We need one flip-flop per bit. Ex: if we need to count from 0 to 7, we need 3 bits, therefore we should use three flip-flops.
3. Derive Karnaugh maps from the state table created in (2) and solve for the inputs to each flip-flop.

Section 12.3 (p. 363)

Table 12-2 State 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

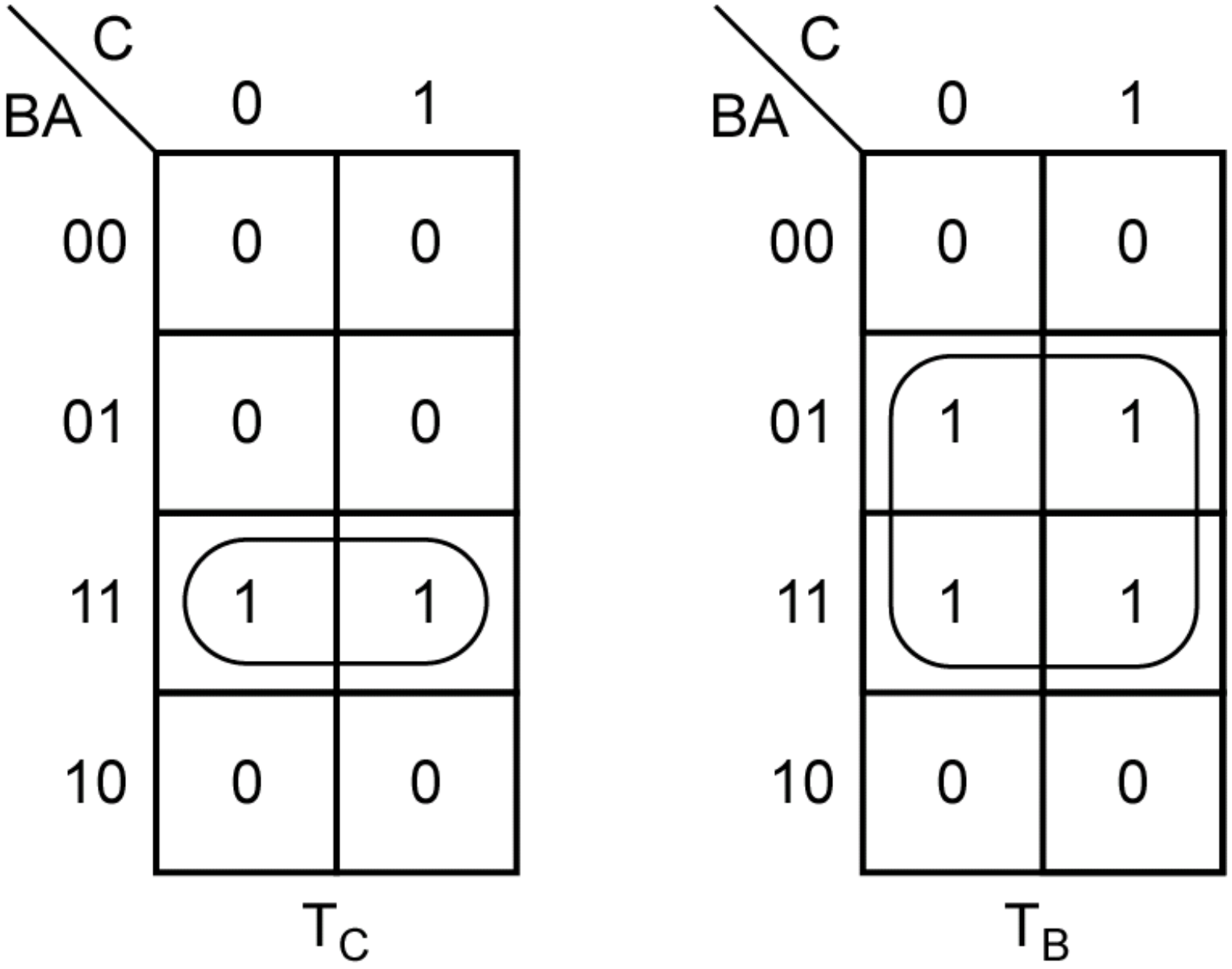


Figure 12-14: **Karnaugh Maps for Binary Counter**

Redesign using D flip-flops

We can redesign the binary counter to use D flip-flops instead of T flip-flops by adding an XOR (exclusive-OR) gate to the inputs of each flip flop.

Section 12.3 (p. 363)

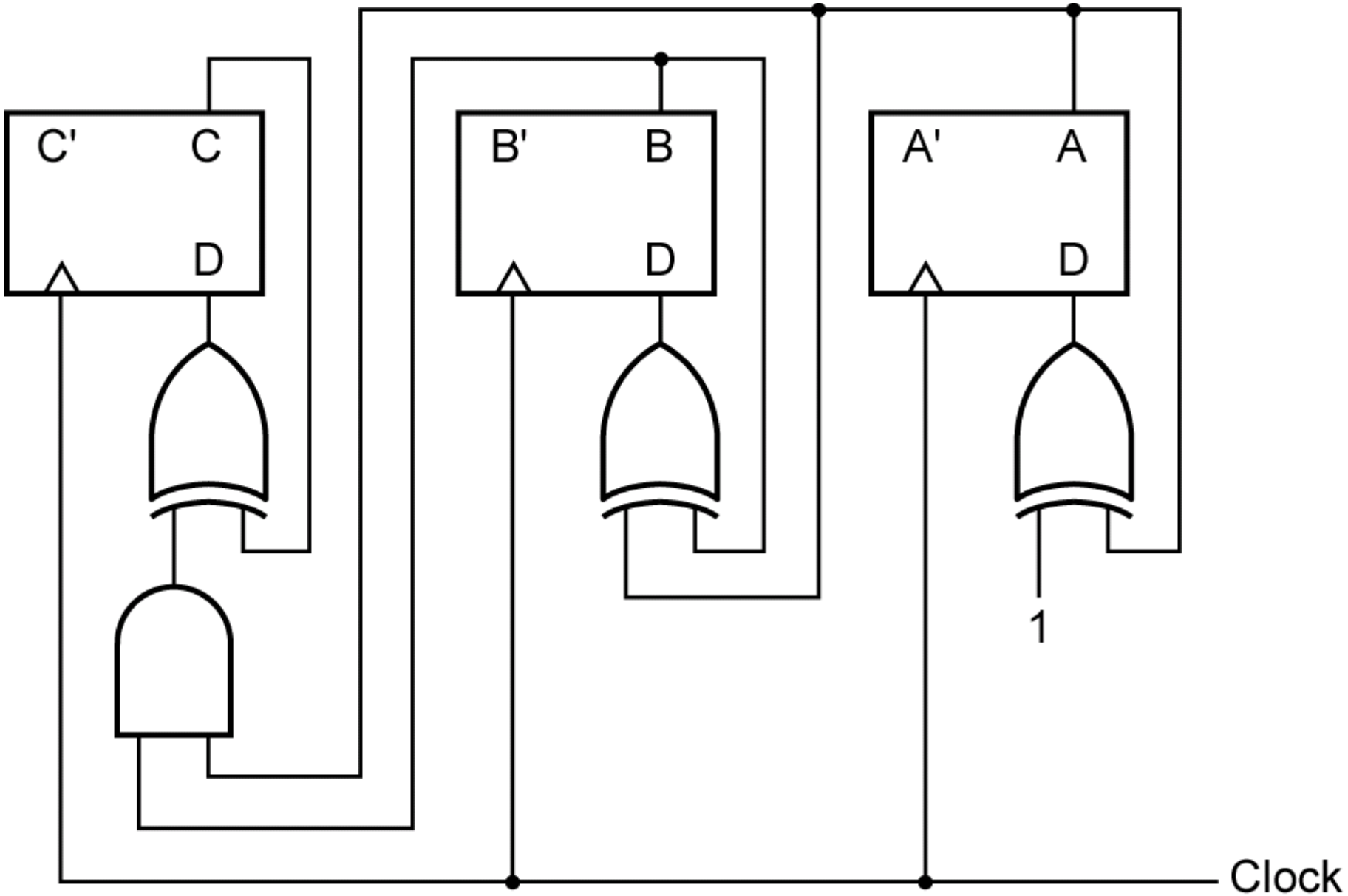


Figure 12-15: **Binary Counter with D Flip-Flops**

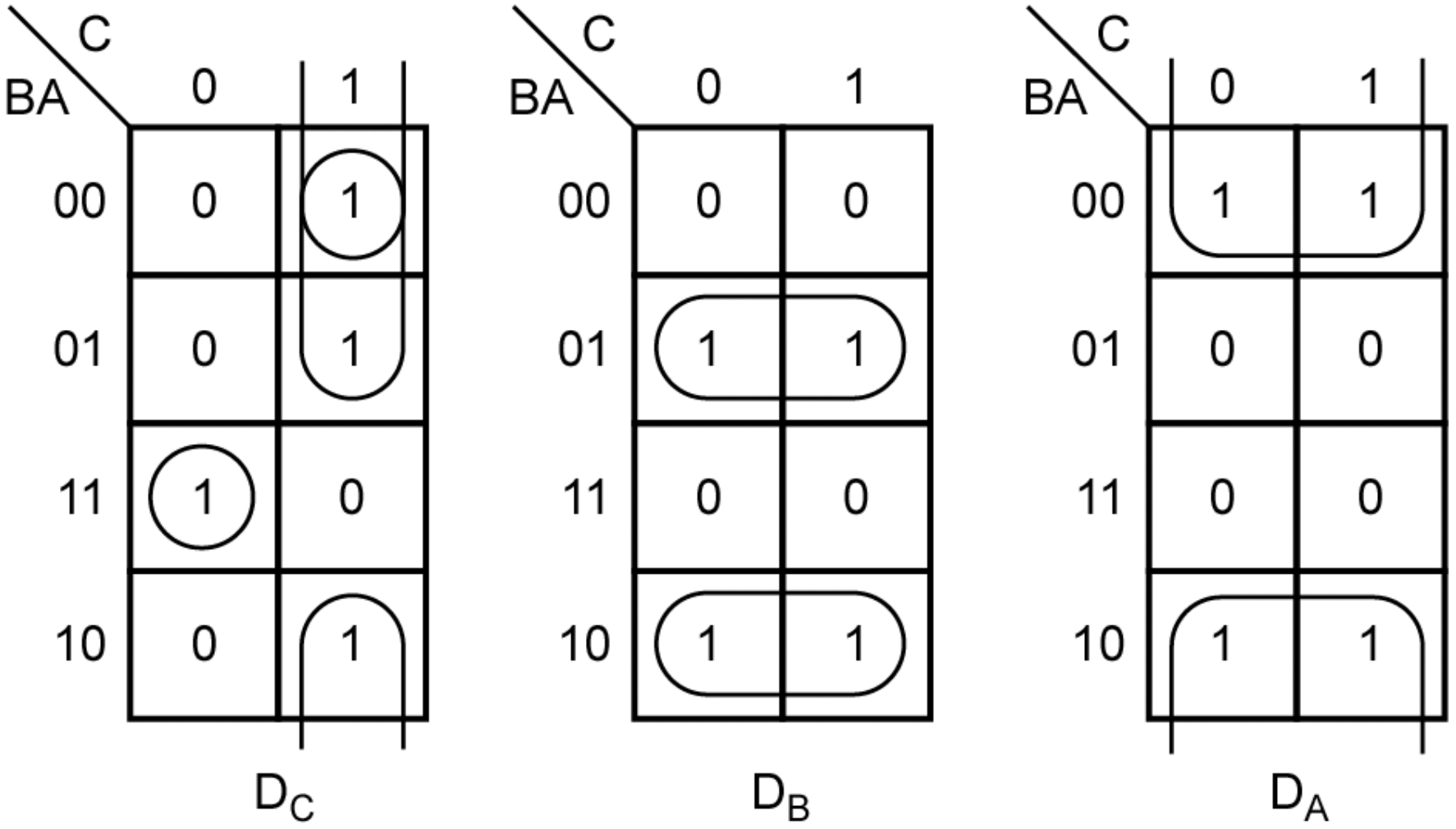
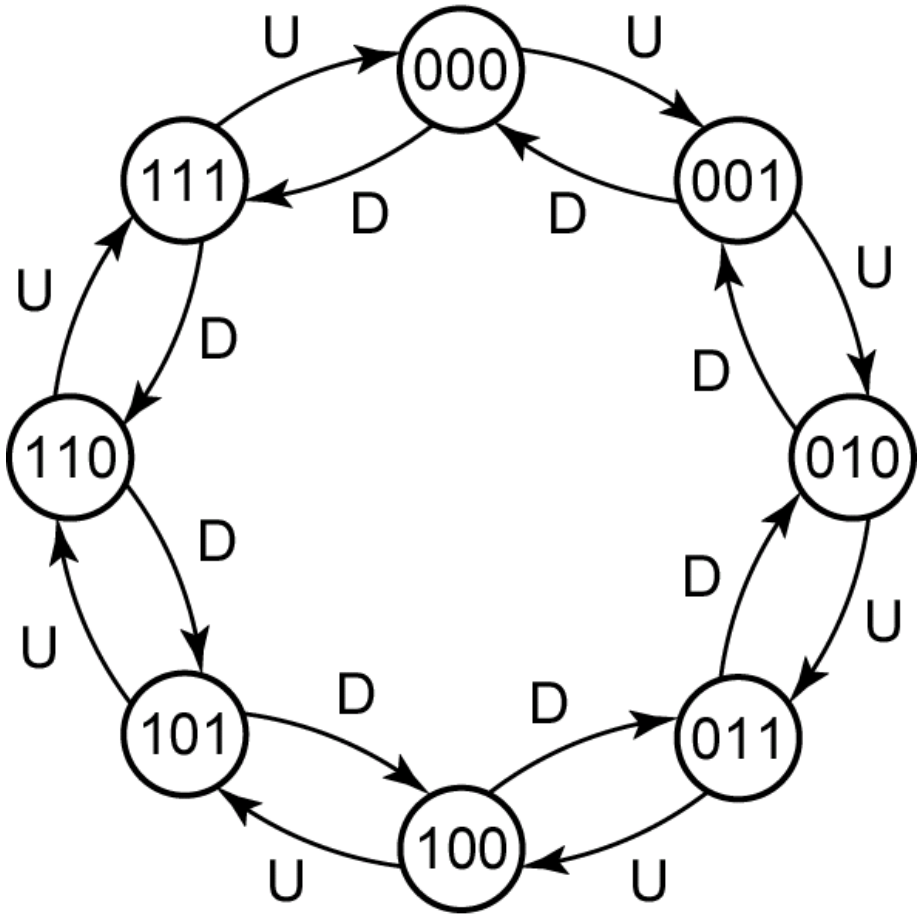


Figure 12-16: **Karnaugh Maps for D Flip-Flops**



CBA	$C^+B^+A^+$	
	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

Figure 12-17: **State Graph and Table for Up-Down Counter**

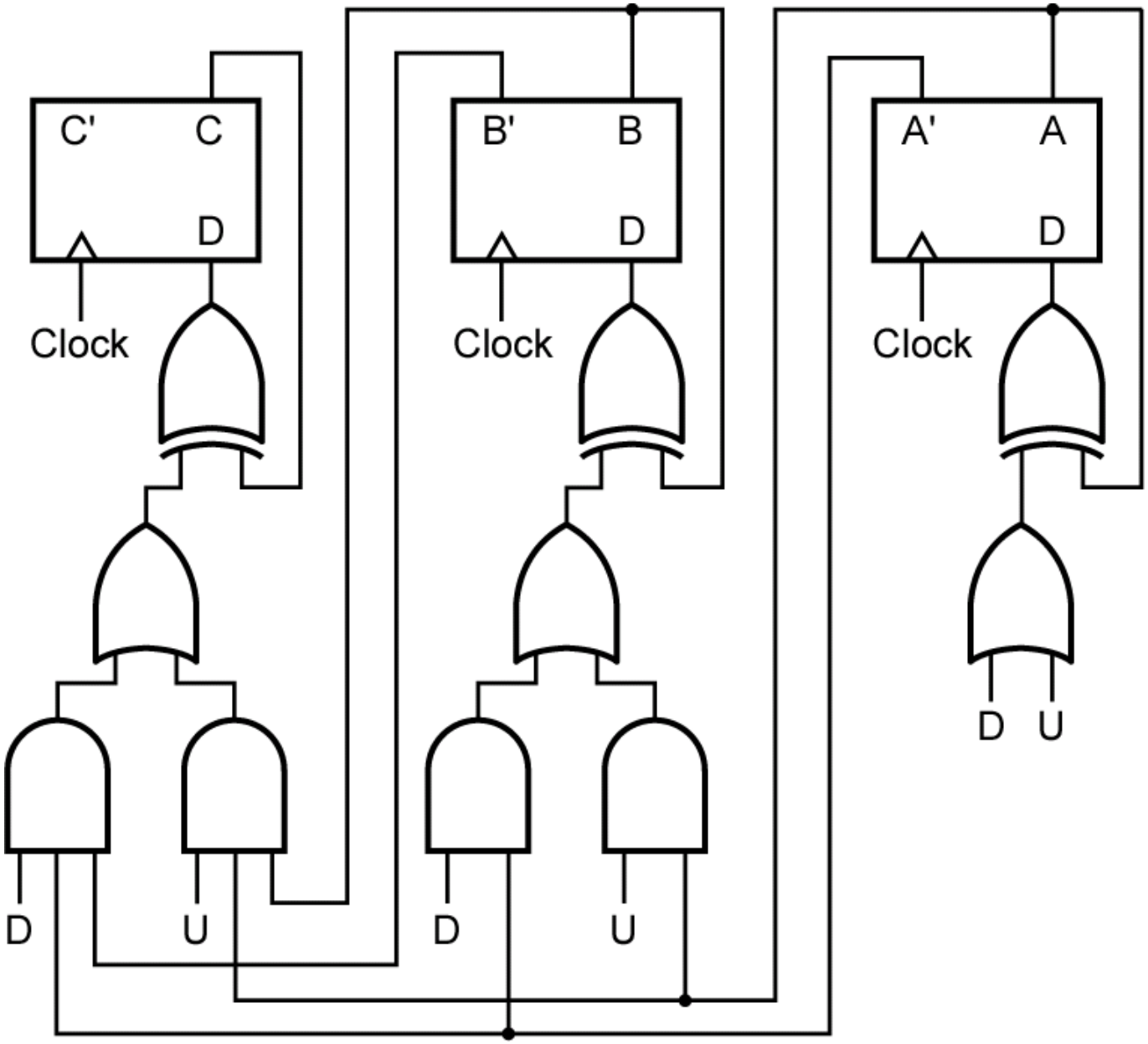
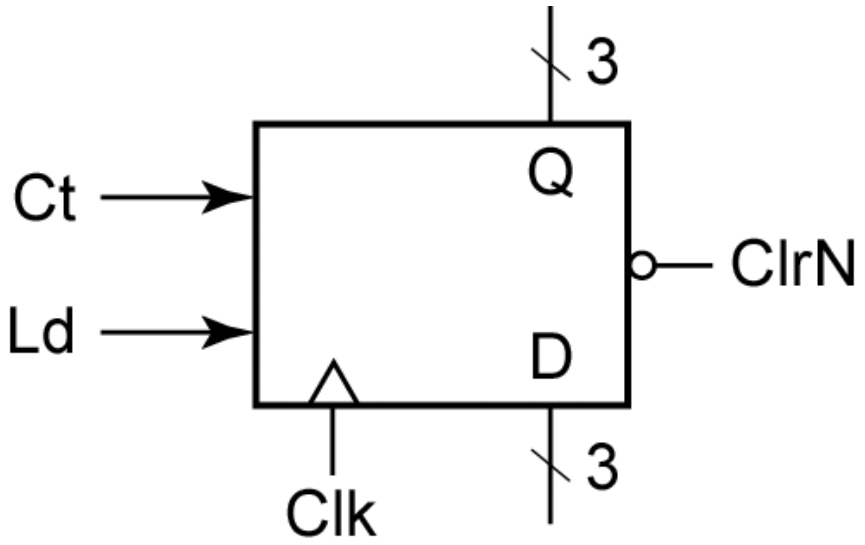


Figure 12-18: **Binary Up-Down Counter**



(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)

Figure 12-19ab: **Loadable Counter with Count Enable**

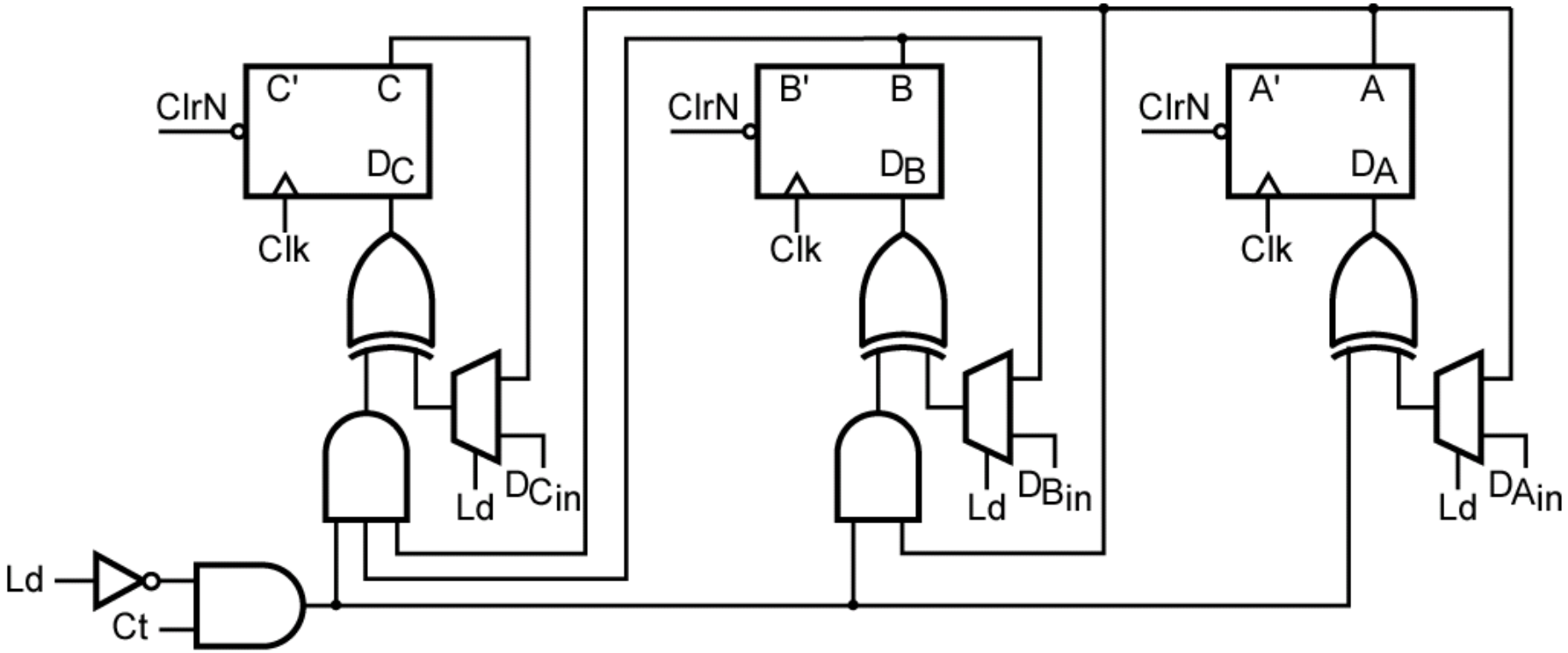


Figure 12-20: **Circuit for Figure 12-19**

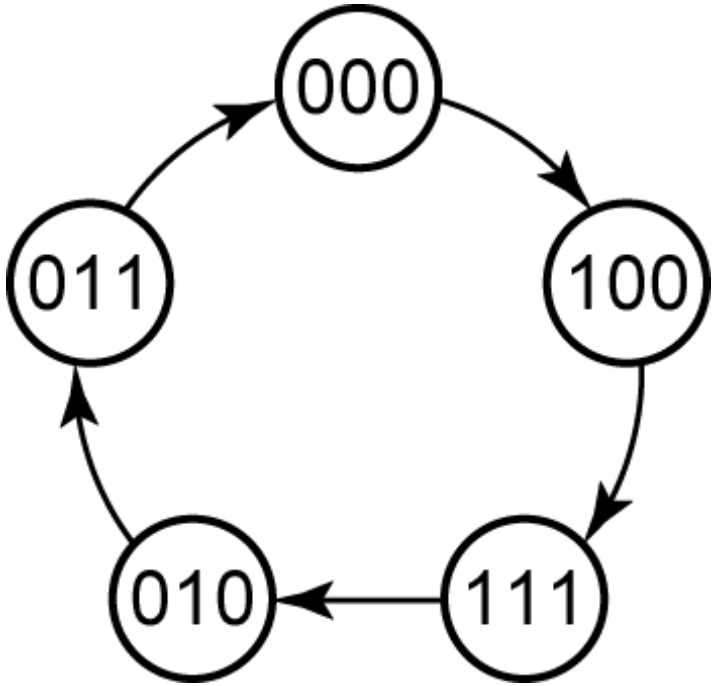


Figure 12-21:
State Graph for Counter

Table 12-3: State Table for Figure 12-21

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

Deriving Equations for T Flip-Flops

We could derive T_C, T_B, and T_A directly from the state table, but it is often more convenient to plot next-state maps showing C^+, B^+, and A^+ as functions of C, B, and A, and then derive T_C, T_B, and T_A from these maps.

Section 12.4 (p. 367)

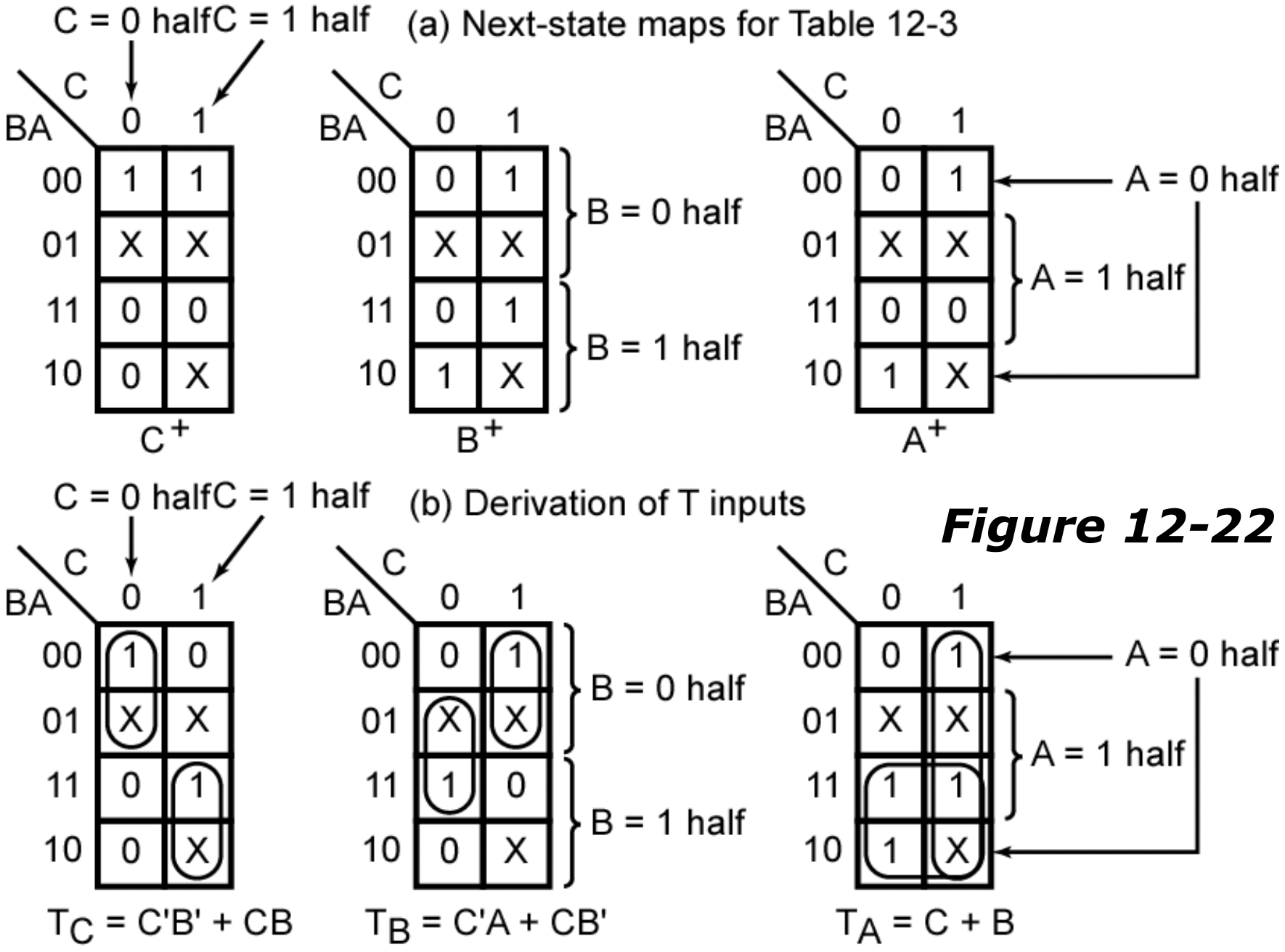


Figure 12-22

Table 12-4. Input for T Flip-Flop

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

$$T = Q^+ \oplus Q$$

Given the present state of a T flip-flop (Q) and the desired next state (Q^+), the T input must be a 1 whenever a change in state is required. Thus, $T = 1$ whenever $Q^+ \neq Q$.

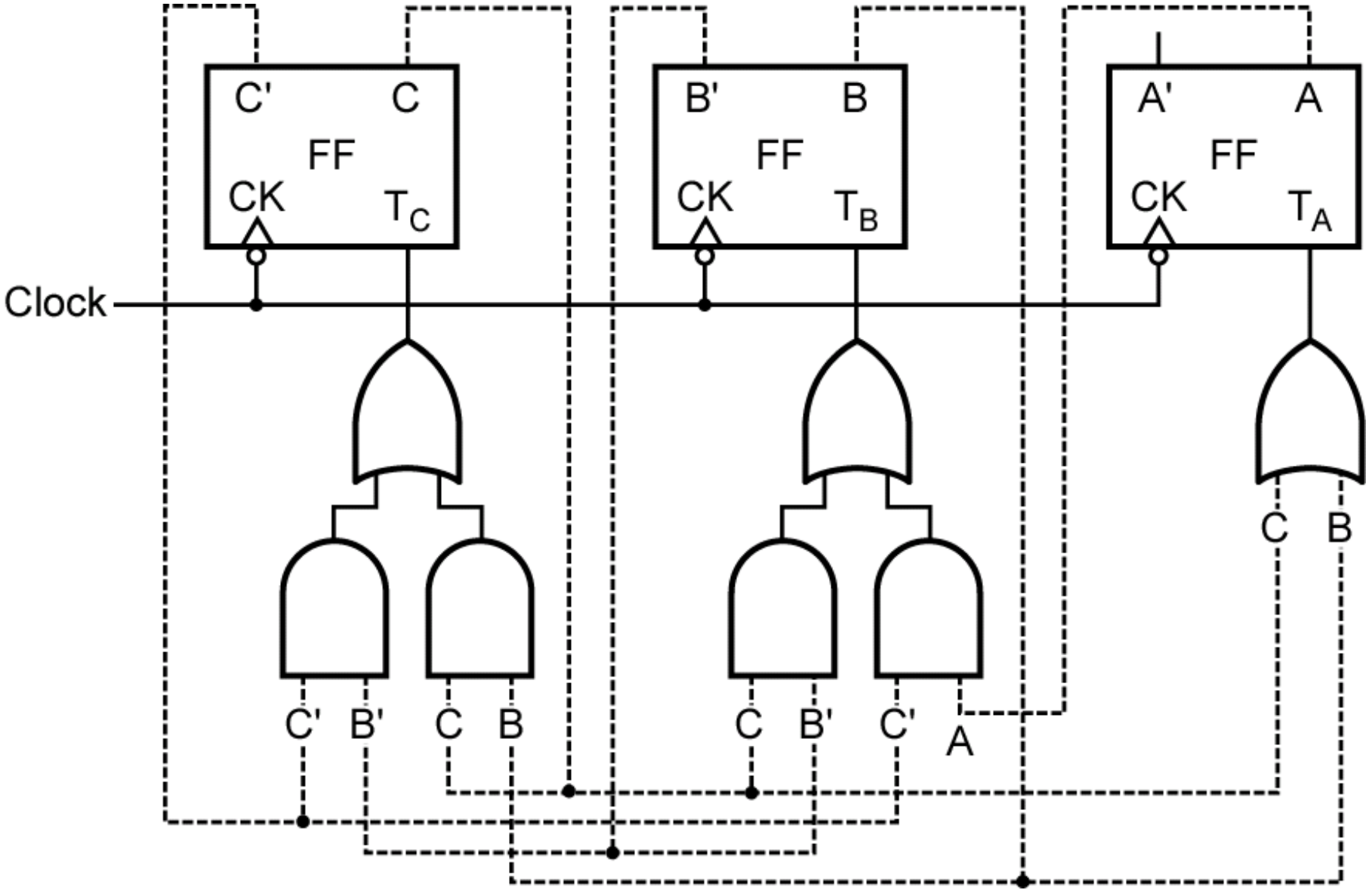


Figure 12-23: **Counter Using T Flip-Flops**

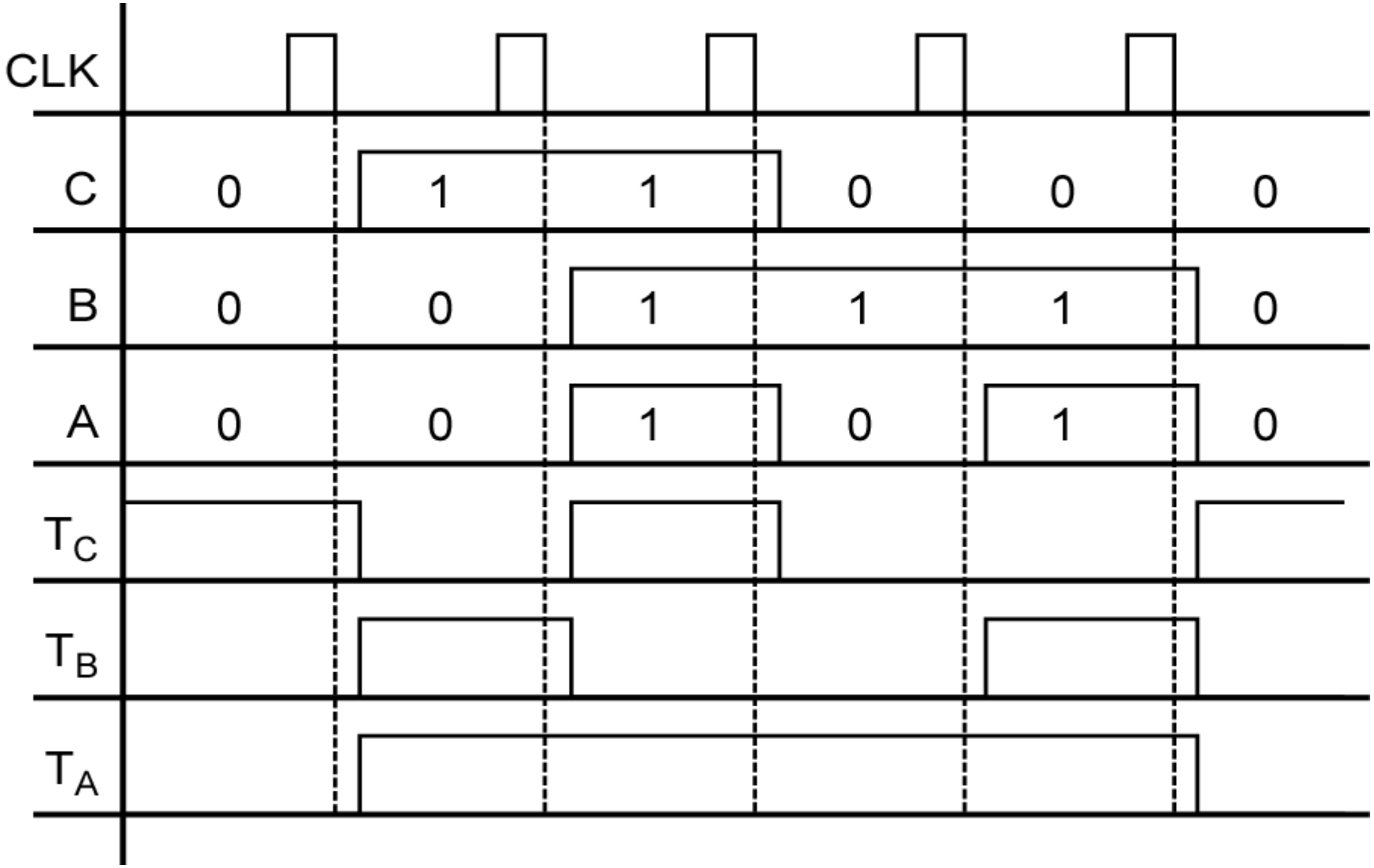


Figure 12-24: **Timing Diagram for Figure 12-23**

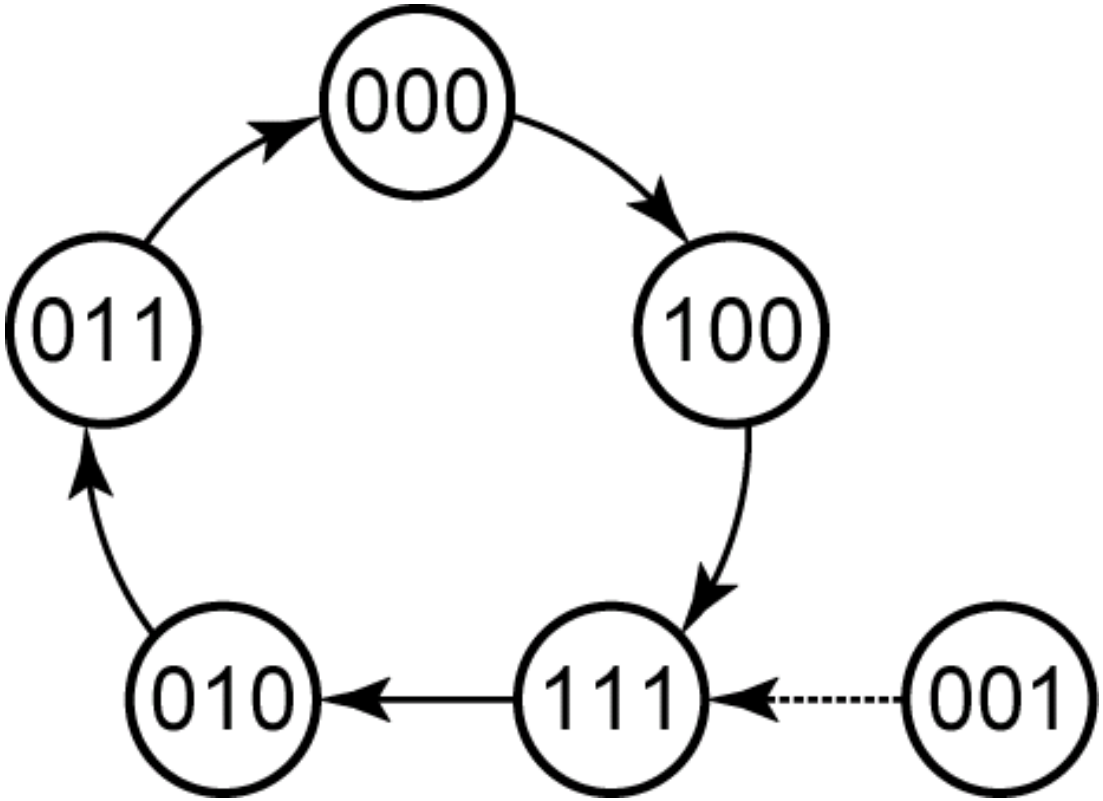


Figure 12-25: **State Graph for Counter**

Although the original state table for the counter is not completely specified, the next states of states 001, 101, and 110 have been specified in the process of completing the circuit design

Deriving Equations for T Flip-Flops

In summary, the following procedure can be used to design a counter using T flip-flops:

1. Form a state 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.

Section 12.4 (p. 370)

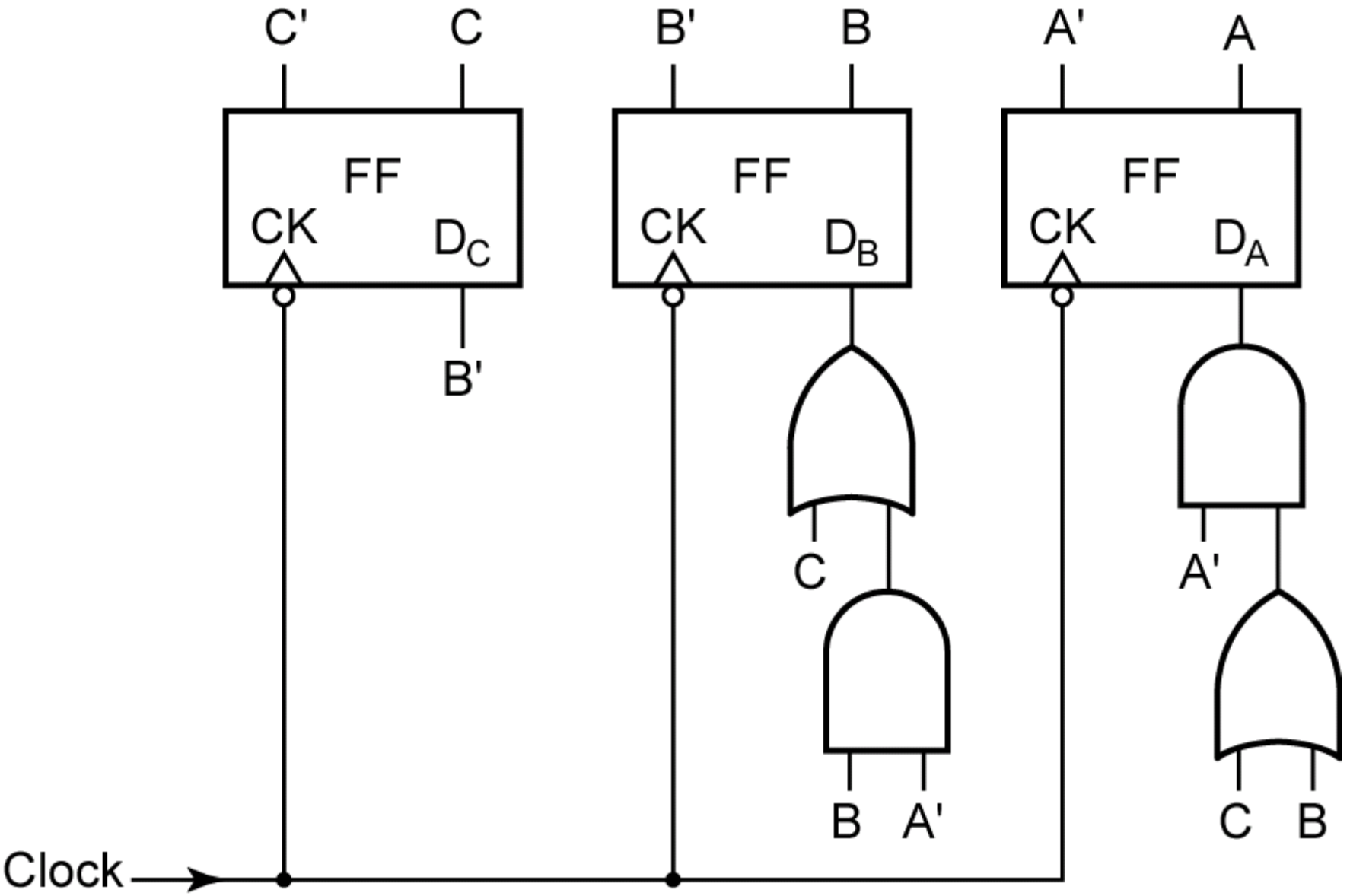


Figure 12-26: **Counter of Figure 12-21 Using D Flip-Flops**

Counter Design Using S-R Flip-Flops

The procedures used to design a counter with S-R flip-flops are similar to the procedures for T flip-flops. However, instead of deriving an input equation for each D or T flip-flop, the S and R input equations must be derived for each S-R flip-flop.

Section 12.5 (p. 371)

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 12-6.
S-R Realization of Figure 12-21

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

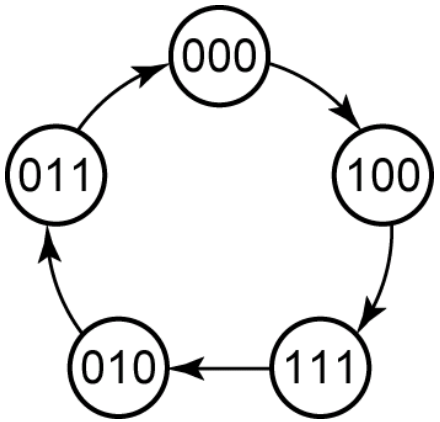


Figure 12-21:
State Graph for Counter

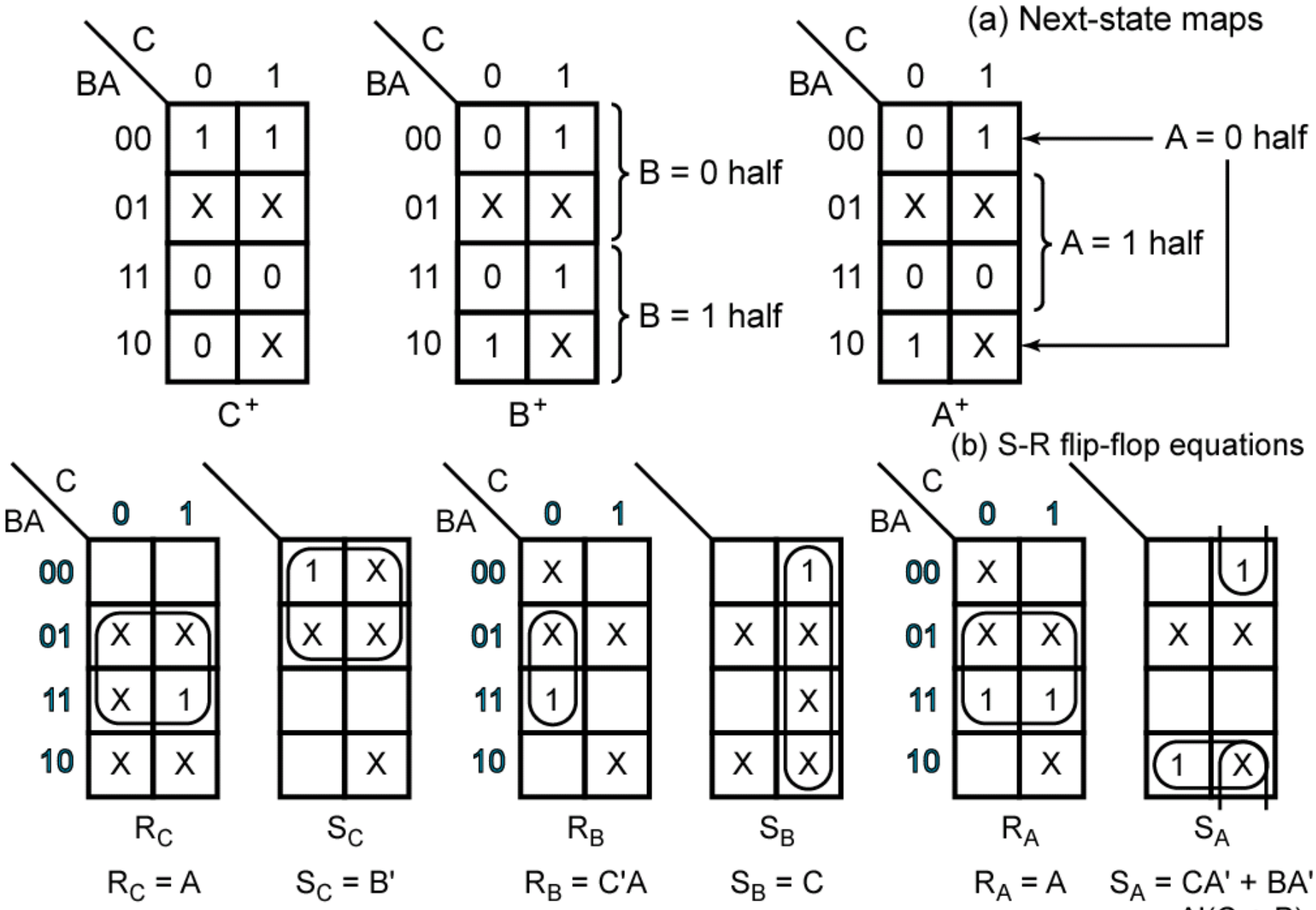
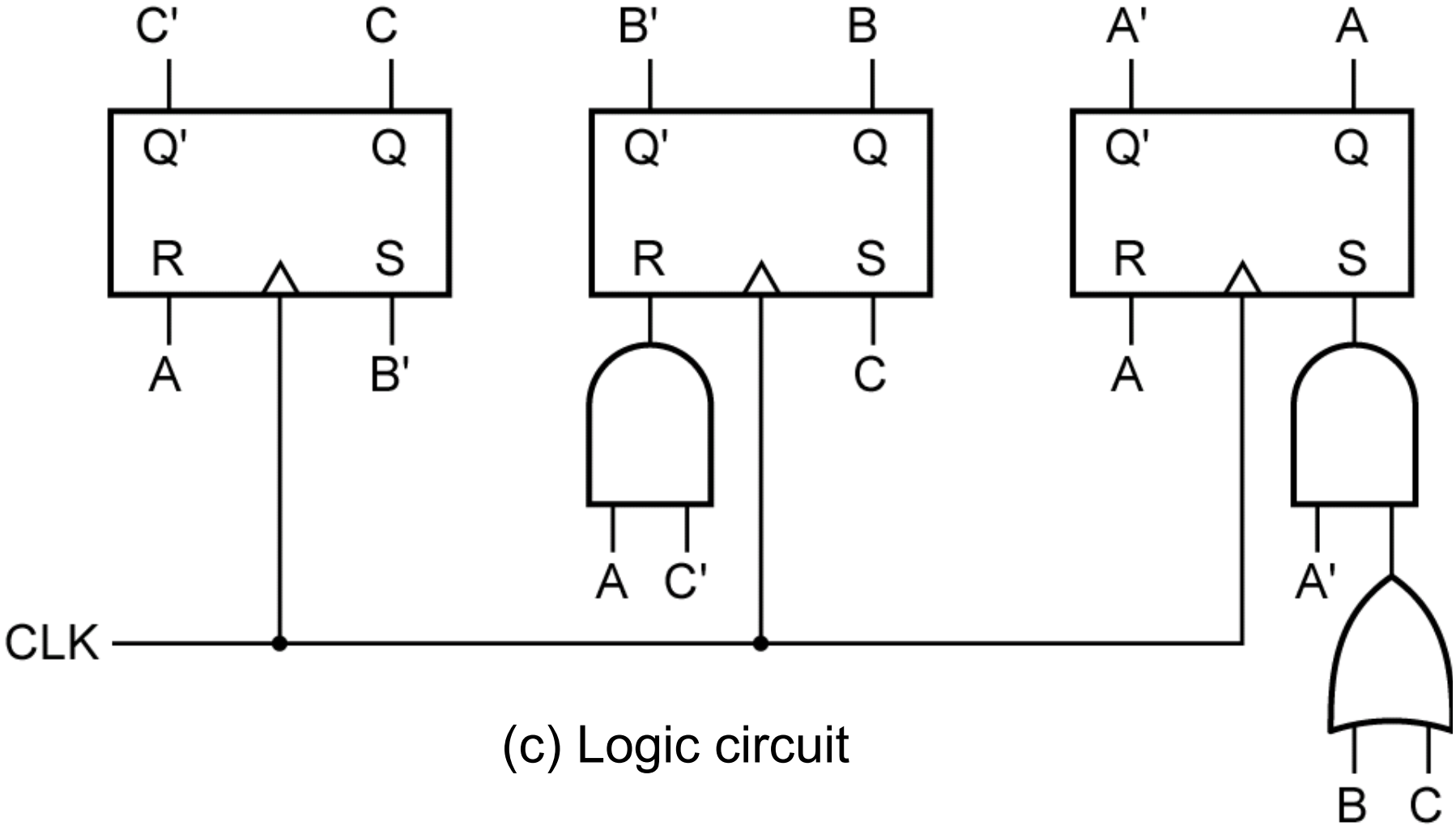


Figure 12-27:

Counter of Figure 12-21 Using S-R Flip-Flops



(c) Logic circuit

Figure 12-27:

Counter of Figure 12-21 Using S-R Flip-Flops

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

Similarly, the same counter can be realized using J-K flip-flops.

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

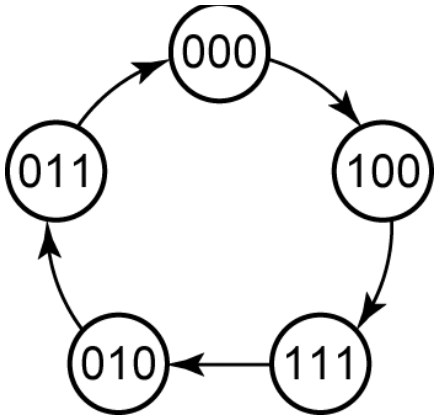


Figure 12-21:
State Graph for Counter

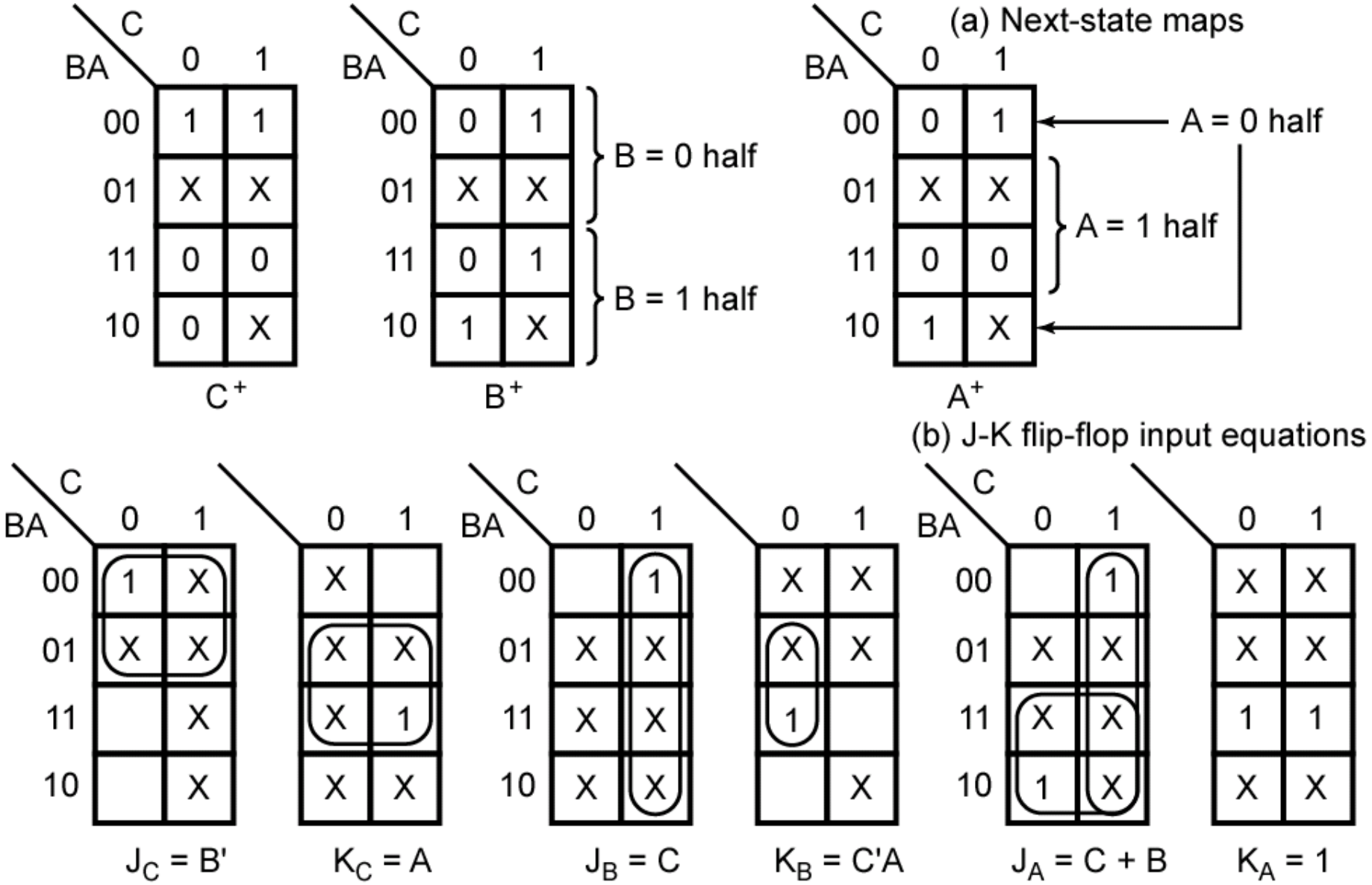
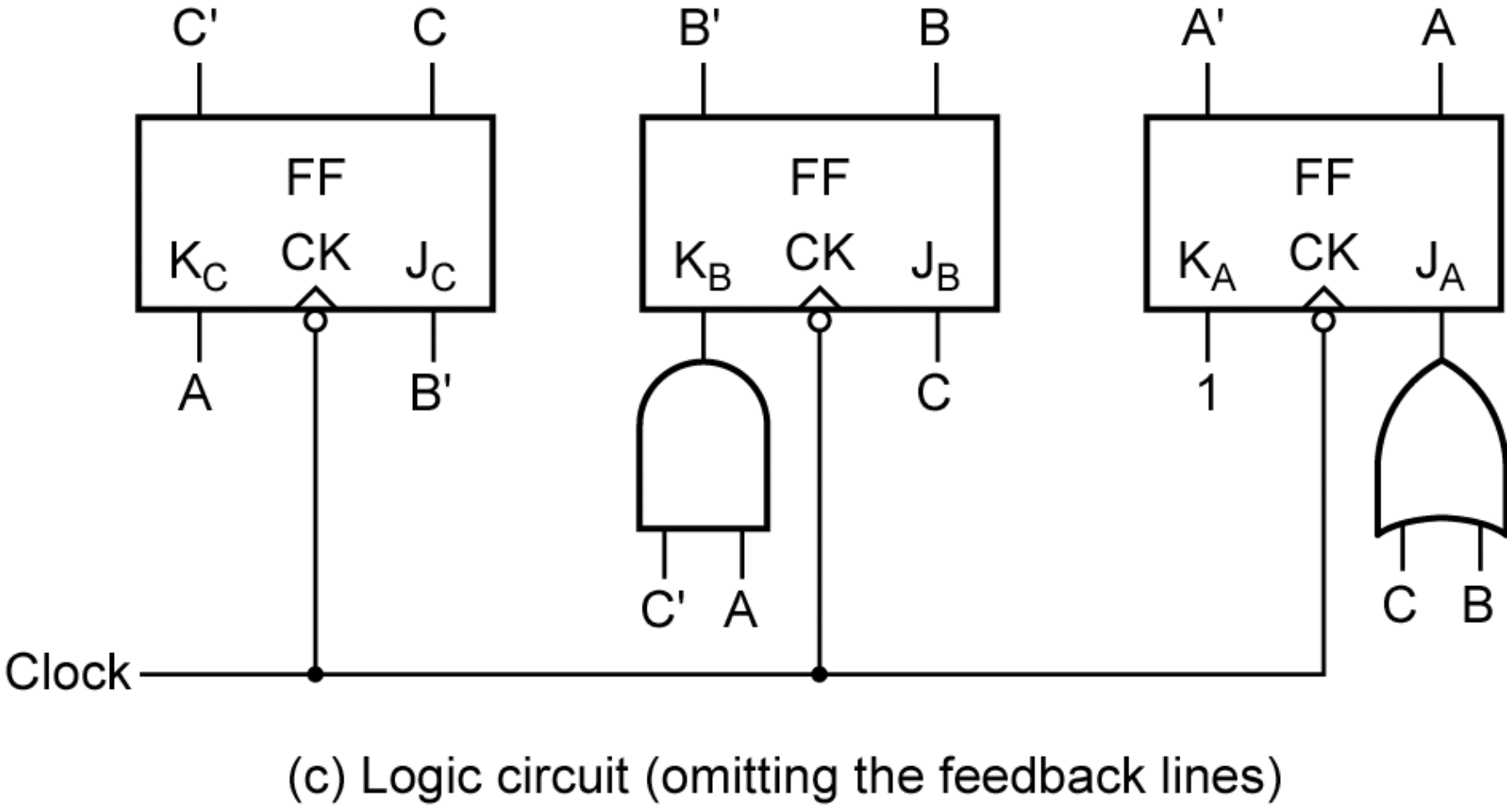


Figure 12-28:
Counter of Figure 12-21 Using J-K Flip-Flops



(c) Logic circuit (omitting the feedback lines)

Figure 12-28:

Counter of Figure 12-21 Using J-K Flip-Flops

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

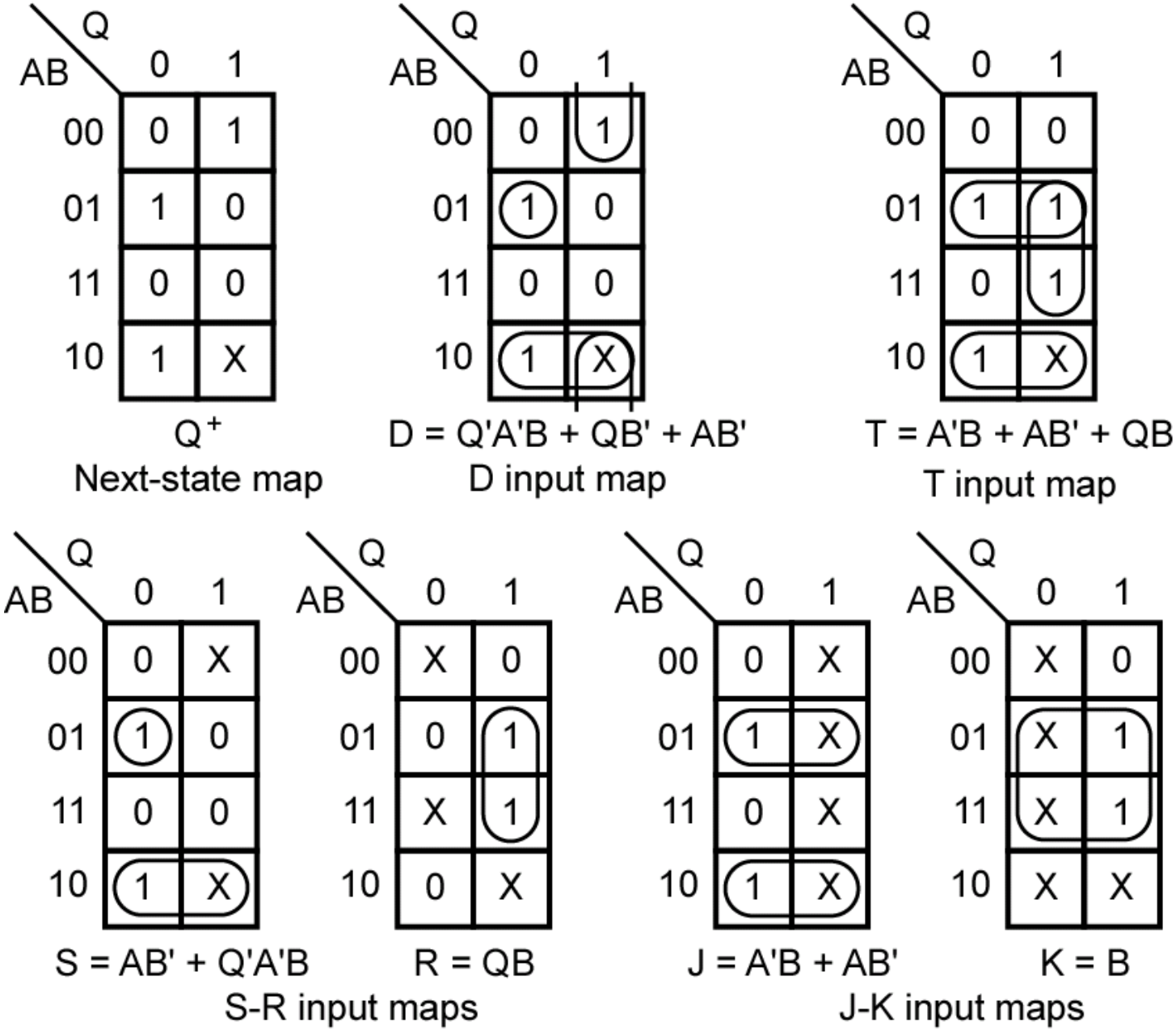
Type of Flip-Flop	Input	$Q = 0$		$Q = 1$		Rules for Forming Input Map From Next-State Map*	
		$Q^+ = 0$	$Q^+ = 1$	$Q^+ = 0$	$Q^+ = 1$	$Q = 0$ Half of Map	$Q = 1$ Half of Map
Delay	D	0	1	0	1	no change	no change
Toggle	T	0	1	1	0	no change	complement
Set-Reset	S	0	1	0	X	no change	replace 1's with X's**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



Example Illustrating the Use of Table 12-9

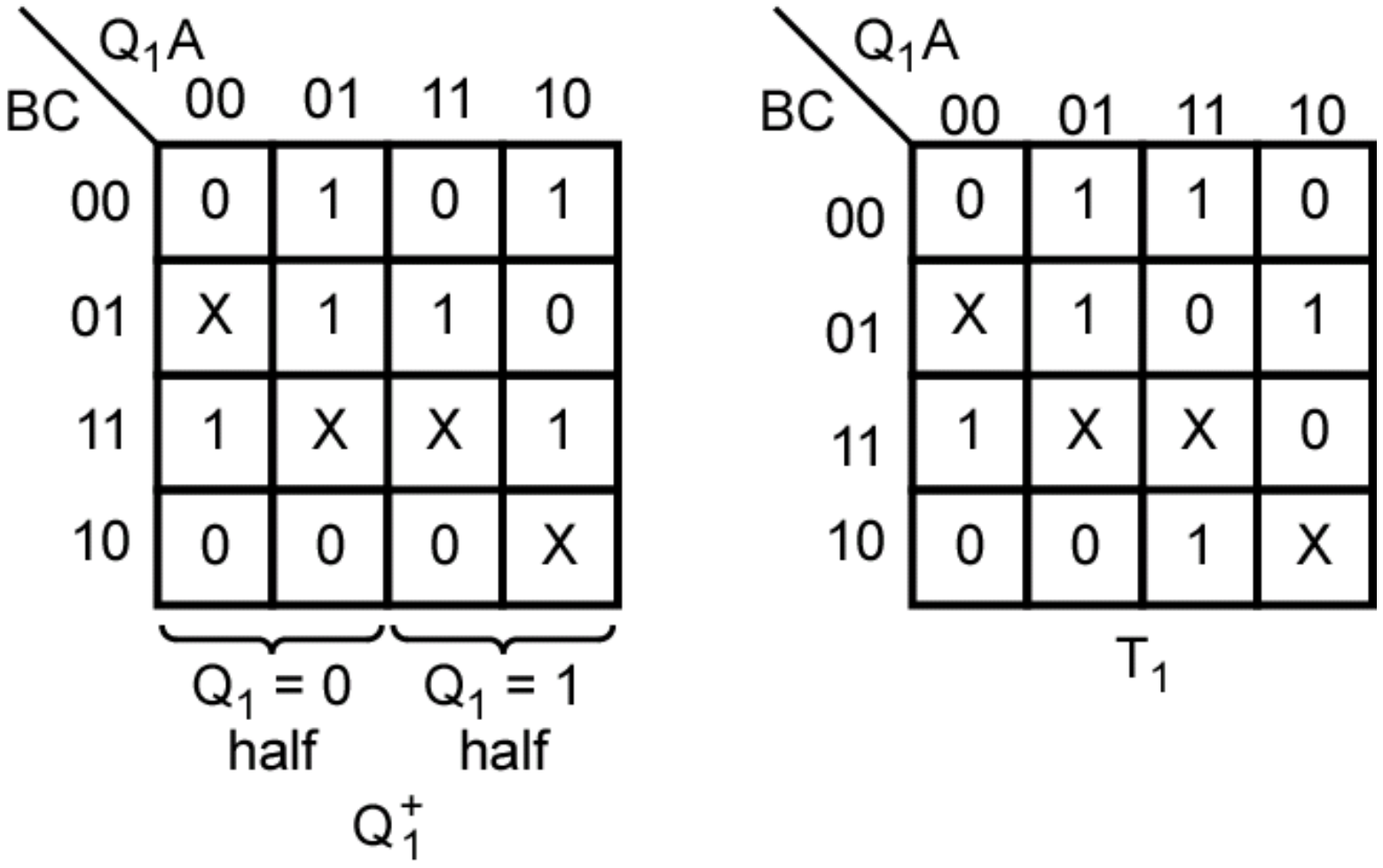
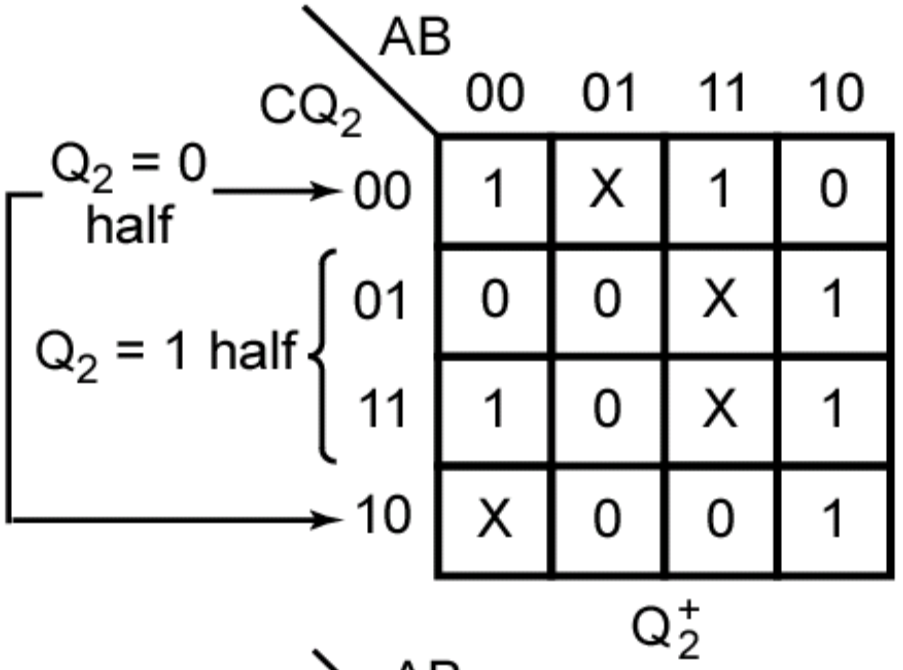
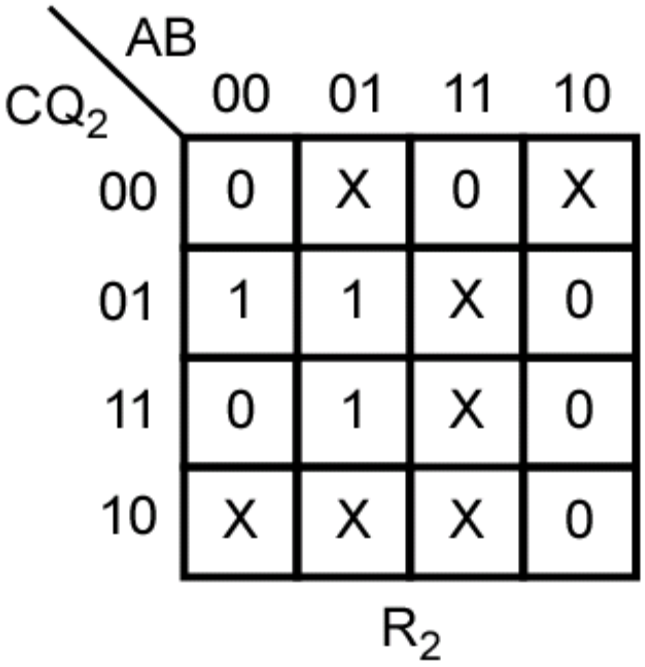


Figure 12-29a:
Derivation of Flip-Flop Input Equations Using 4-Variable Maps

Figure 12-29b:
Derivation of Flip-Flop Input Equations Using 4-Variable Maps





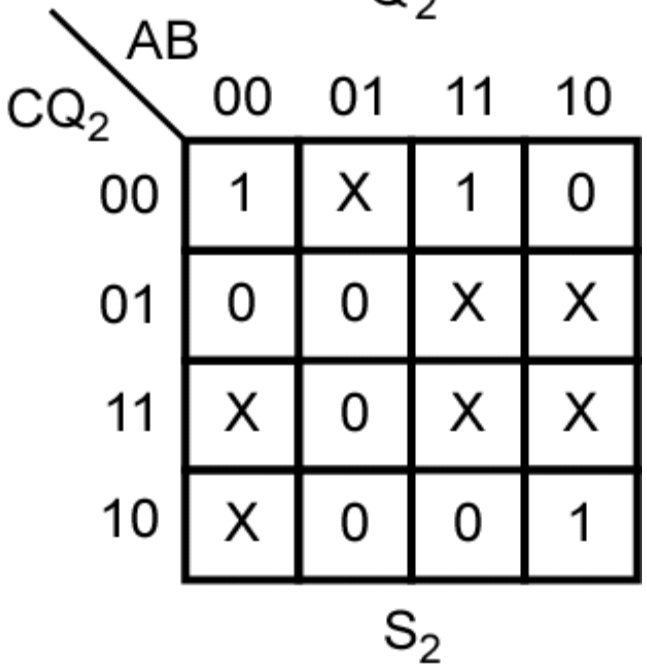


Figure 12-29c:
Derivation of Flip-Flop Input Equations Using 4-Variable Maps