CHAPTER-6

Registers and Counters

Digital Design (with an introduction to the Verilog HDL) 6th Edition,
M. Morris Mano, Michael D. Ciletti

Registers

- Clocked sequential circuits
  - a group of flip-flops and combinational gates
  - connected to form a feedback path

    Flip-flops + Combinational gates

    (essential) (optional)
- Circuits that include flip-flops are usually classified by the function they perform
  - Registers
  - Counters
- Register:
  - a group of flip-flops each one of which shares a common clock and is capable of storing one bit of information.
  - The flip-flops hold the binary information, and the gates determine how the information is transferred into the register.
- Counter:
  - a register that goes through a predetermined sequence of binary states

Shift Registers

- A **register** is a digital circuit with two basic functions: **data storage** and **data movement**.
- The **shift capability** of a register permits the movement of data from stage to stage

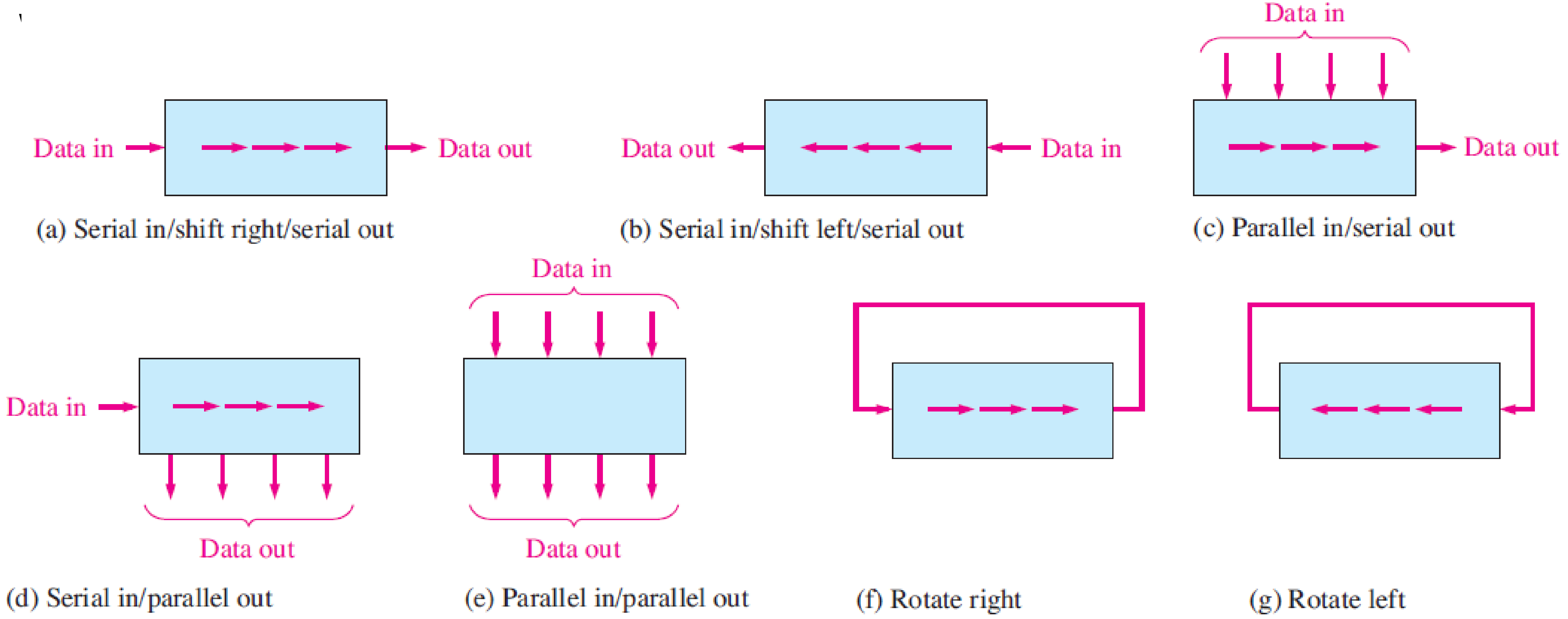


FIGURE 8–2 Basic data movement in shift registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

Registers

- A n-bit register
  - n flip-flops capable of storing n bits of binary information
  - 4-bit register
- The common clock input triggers all flip-flops on the positive edge of each pulse, and the binary data available at the four inputs are transferred into the register.
- The value of (I_3, I_2, I_1, I_0) immediately before the clock edge determines the value of (A_3, A_2, A_1, A_0) after the clock edge.
- When *Clear_b* input goes to 0, all flip-flops are reset asynchronously.
- The transfer of new information into a register is referred to as *loading* or *updating* the register.
- If all the bits of the register are loaded simultaneously with a common clock pulse, we say that the loading is done *in parallel* .
- A clock edge applied to the *C* inputs of the register of Fig. 6.1 will load all four inputs in parallel.

If the contents of the register must be left unchanged…?

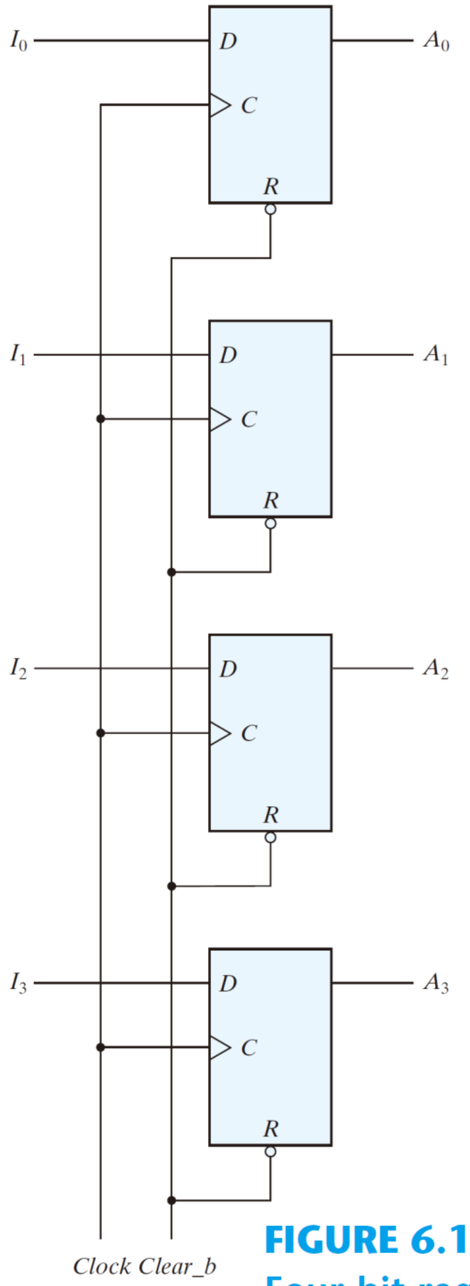


FIGURE 6.1
Four-bit register

4-bit register with parallel load

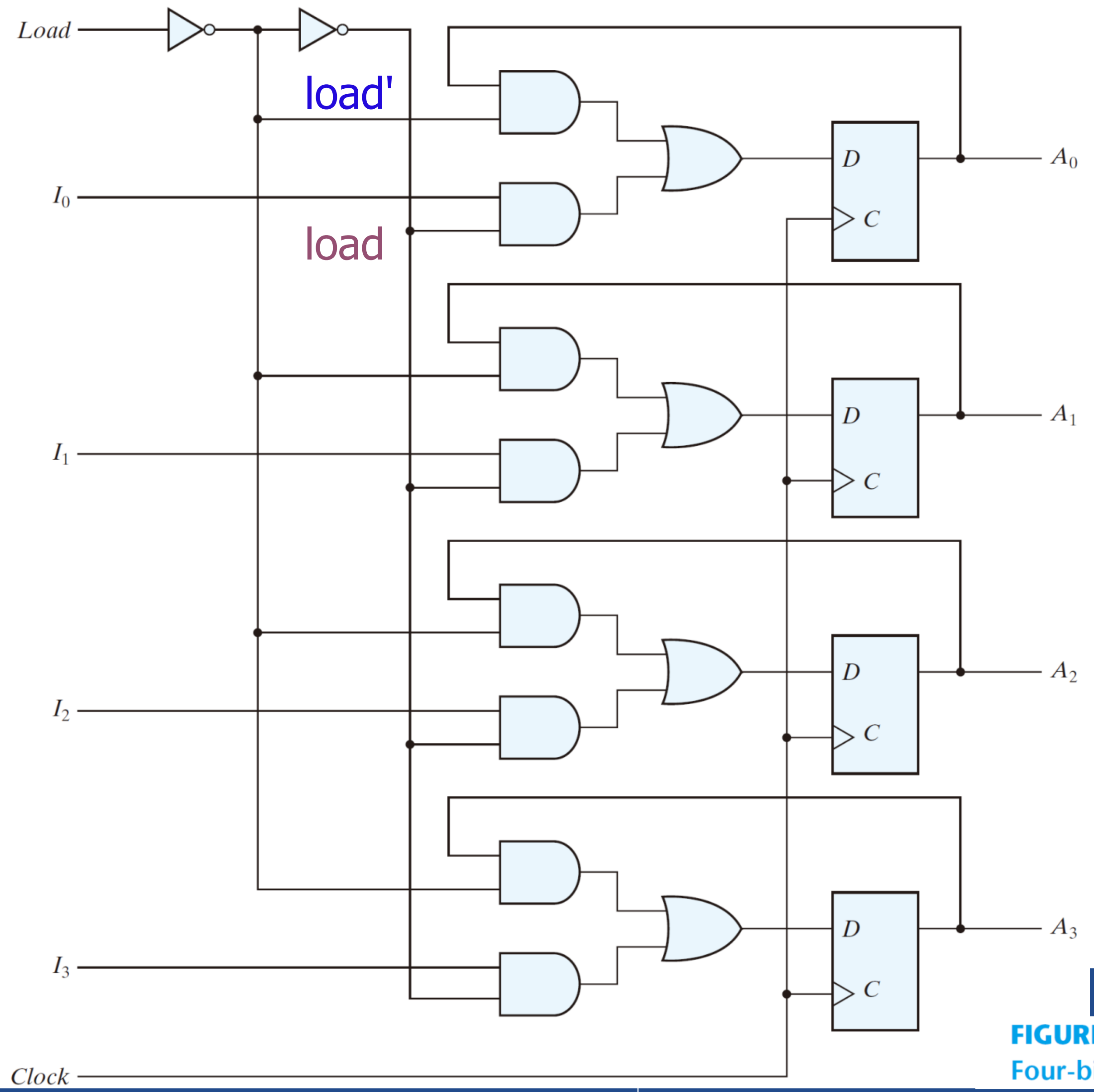


FIGURE 6.2
Four-bit register with parallel load

- The load input to the register determines the action to be taken with each clock pulse.
- When the **load input is 1, the data at the four external inputs are transferred into the register** with the next positive edge of the clock.
- When the **load input is 0, the outputs of the flip-flops are connected to their respective inputs**.
- The load input determines whether the next pulse will accept new information or leave the information in the register intact.

Shift Registers

- Shift register
  - A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a ***shift register***.
- Four-bit shift register

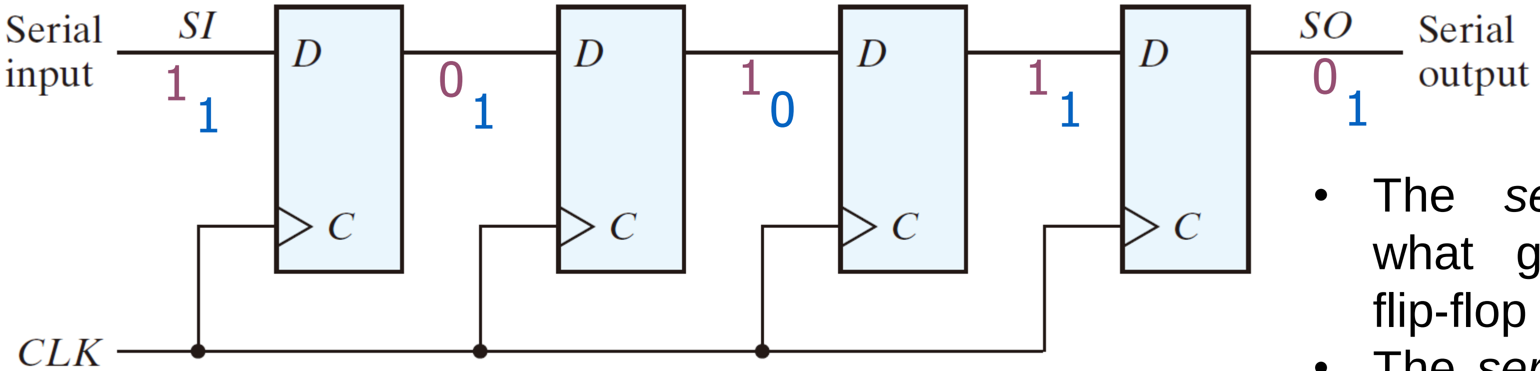
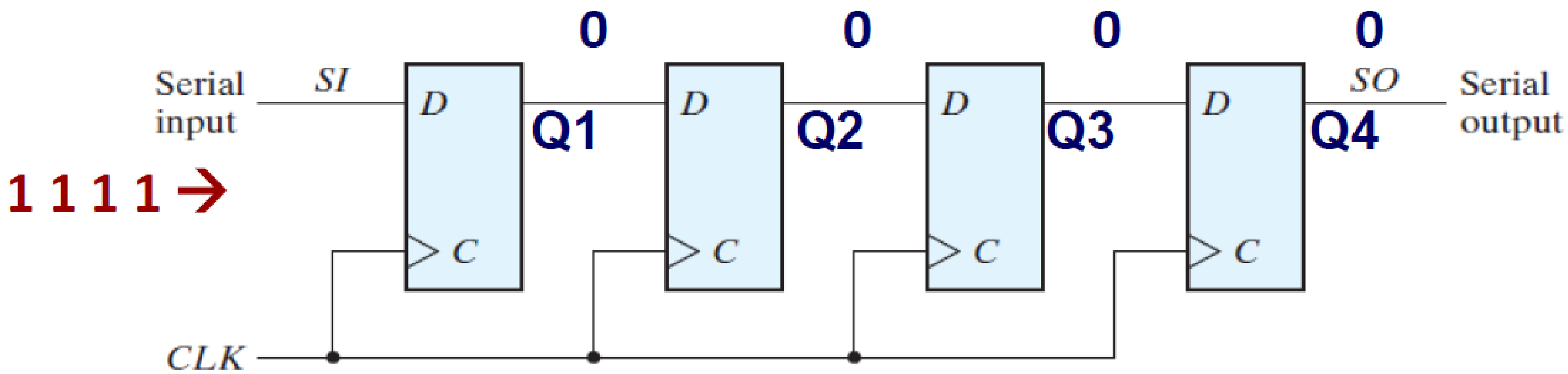


FIGURE 6.3
Four-bit shift register

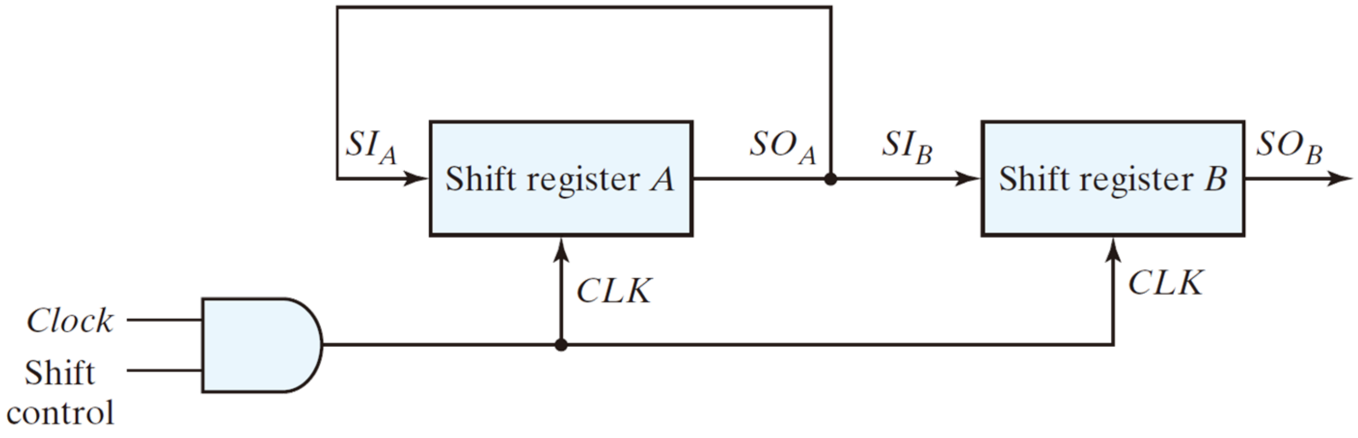
- The *serial input* determines what goes into the leftmost flip-flop during the shift.
- The *serial output* is taken from the output of the rightmost flip-flop.

Shift Registers

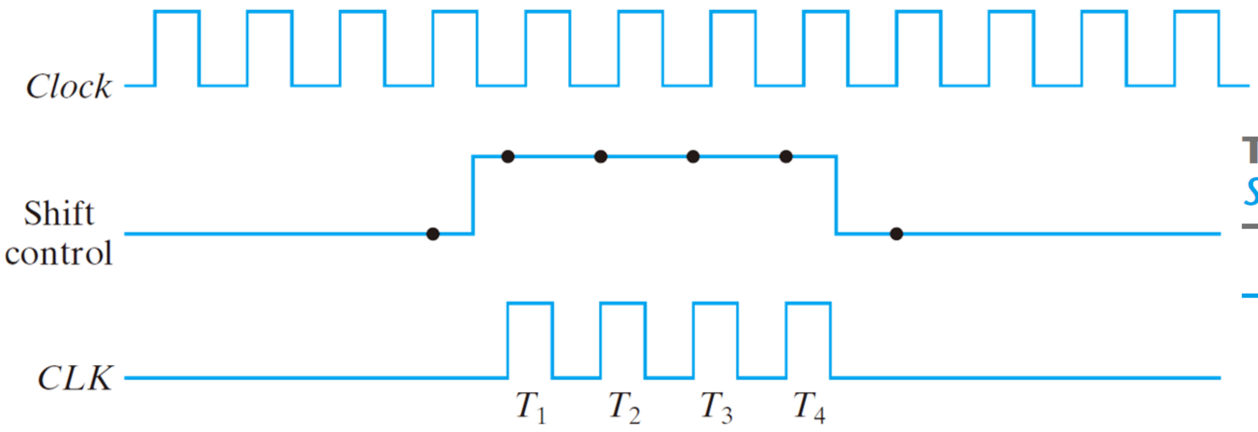


	Q1	Q2	Q3	Q4
Initial values	0	0	0	0
After 1st Posedge	1	0	0	0
After 2nd Posedge	1	1	0	0
After 3rd Posedge	1	1	1	0
After 4th Posedge	1	1	1	1

Example: Serial transfer from reg A to reg B



(a) Block diagram



(b) Timing diagram

FIGURE 6.4
Serial transfer from register *A* to register *B*

- The serial transfer of information from register *A* to register *B* is done with shift registers

- Suppose the shift registers in Fig. 6.4 have four bits each.
- Shift control signal enables the shift registers for a fixed time of four clock pulses in order to pass an entire word.

Table 6.1
Serial-Transfer Example

Timing Pulse	Shift Register *A*				Shift Register *B*			
Initial value	1	0	1	1	0	0	1	0
After T_1	1	1	0	1	1	0	0	1
After T_2	1	1	1	0	1	1	0	0
After T_3	0	1	1	1	0	1	1	0
After T_4	1	0	1	1	1	0	1	1

Serial In/Serial Out Shift Registers

- The serial in/serial out shift register accepts data serially—that is, one bit at a time on a single line. Data bits are entered serially (least-significant bit first).
- It produces the stored information on its output also in serial form.

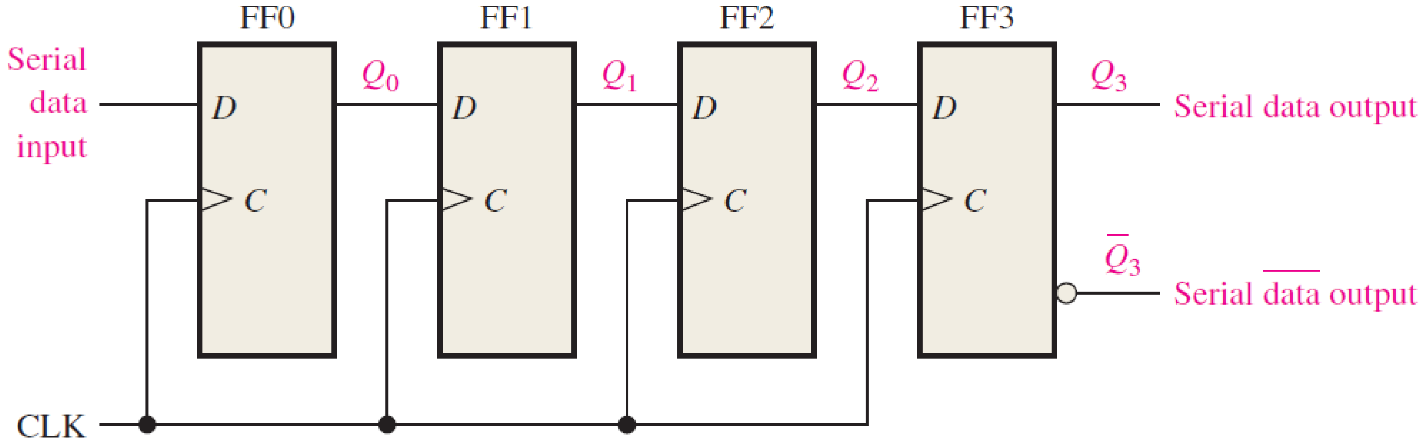


FIGURE 8–3 Serial in/serial out shift register.

FIGURE 8–5 Logic symbol for an 8-bit serial in/serial out shift register.

TABLE 8–1

Shifting a 4-bit code into the shift register in Figure 8–3. Data bits are indicated by a beige screen.

CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	0	0	0	0
1	**0**	0	0	0
2	**1**	**0**	0	0
3	**0**	**1**	**0**	0
4	**1**	**0**	**1**	**0**

TABLE 8–2

Shifting a 4-bit code out of the shift register in Figure 8–3. Data bits are indicated by a beige screen.

CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	**1**	**0**	**1**	**0**
5	0	**1**	**0**	**1**
6	0	0	**1**	**0**
7	0	0	0	**1**
8	0	0	0	0

Serial In/Parallel Out Shift Registers

- Data bits are entered serially (least-significant bit first) into a serial in/parallel out shift register in the same manner as in serial in/serial out registers.
- Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously, rather than on a bit-by-bit basis as with the serial output.

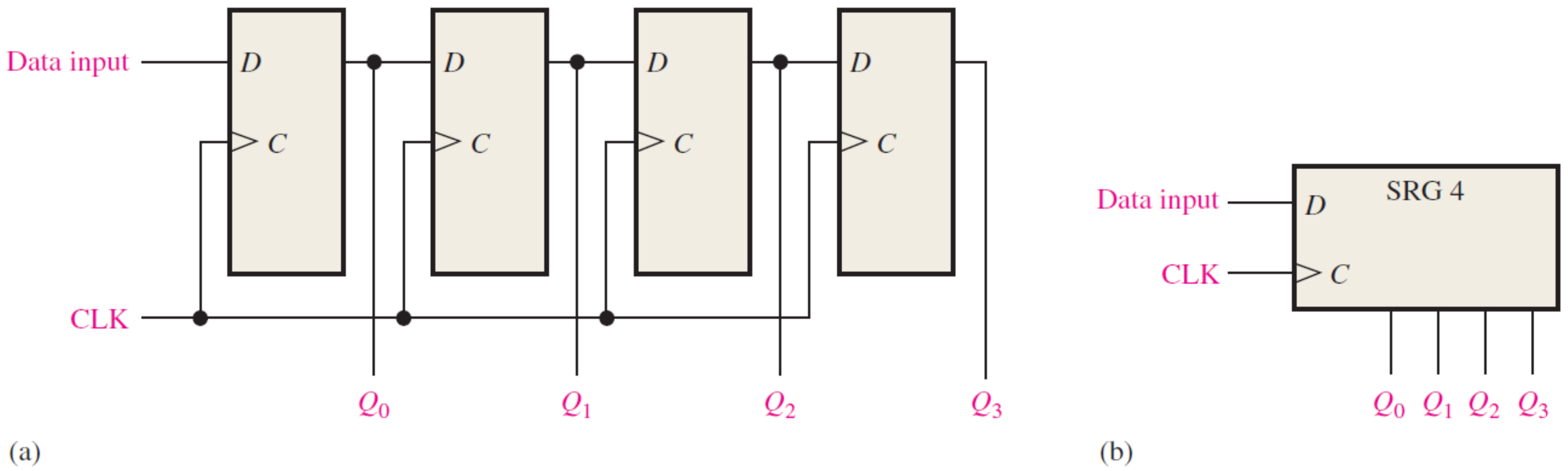


FIGURE 8–6 A serial in/parallel out shift register.

Parallel In/Serial Out Shift Registers

- For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs.

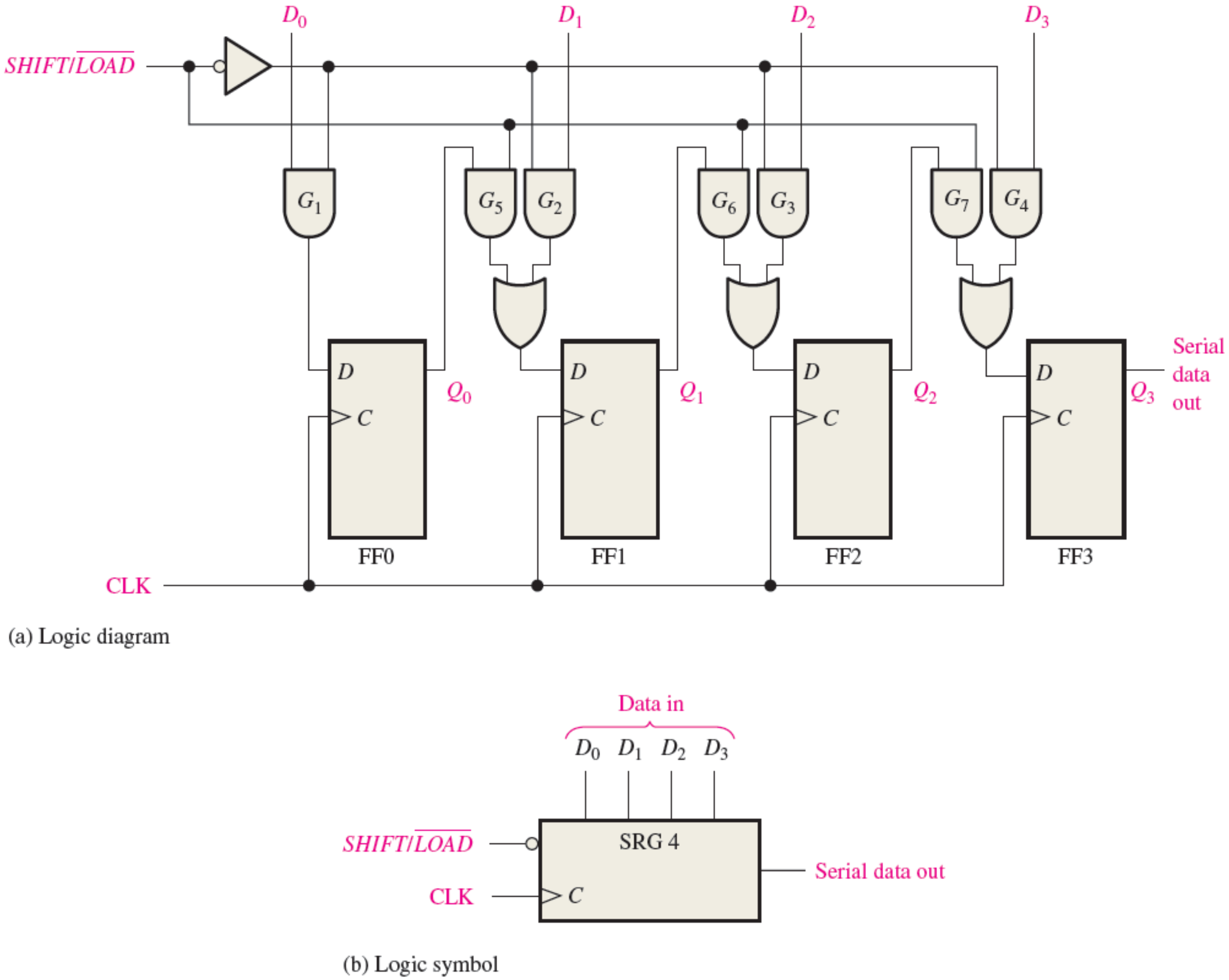


FIGURE 8–10 A 4-bit parallel in/serial out shift register.

- When *SHIFT*/$\overline{LOAD}$ is LOW, gates *G*1 through *G*4 are enabled, allowing each data bit to be applied to the *D* input of its respective flip-flop.
- When a clock pulse is applied, the flip-flops with D = 1 will set and those with D = 0 will reset, thereby storing all four bits simultaneously.
- When SHIFT/$\overline{\text{LOAD}}$ is HIGH, gates G1 through G4 are disabled and gates G5 through G7 are enabled, allowing the data bits to shift right from one stage to the next.
- The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the level on the SHIFT/$\overline{\text{LOAD}}$ input.

Parallel In/Parallel Out Shift Registers

- Immediately following the simultaneous entry of all data bits, the bits appear on the parallel outputs.

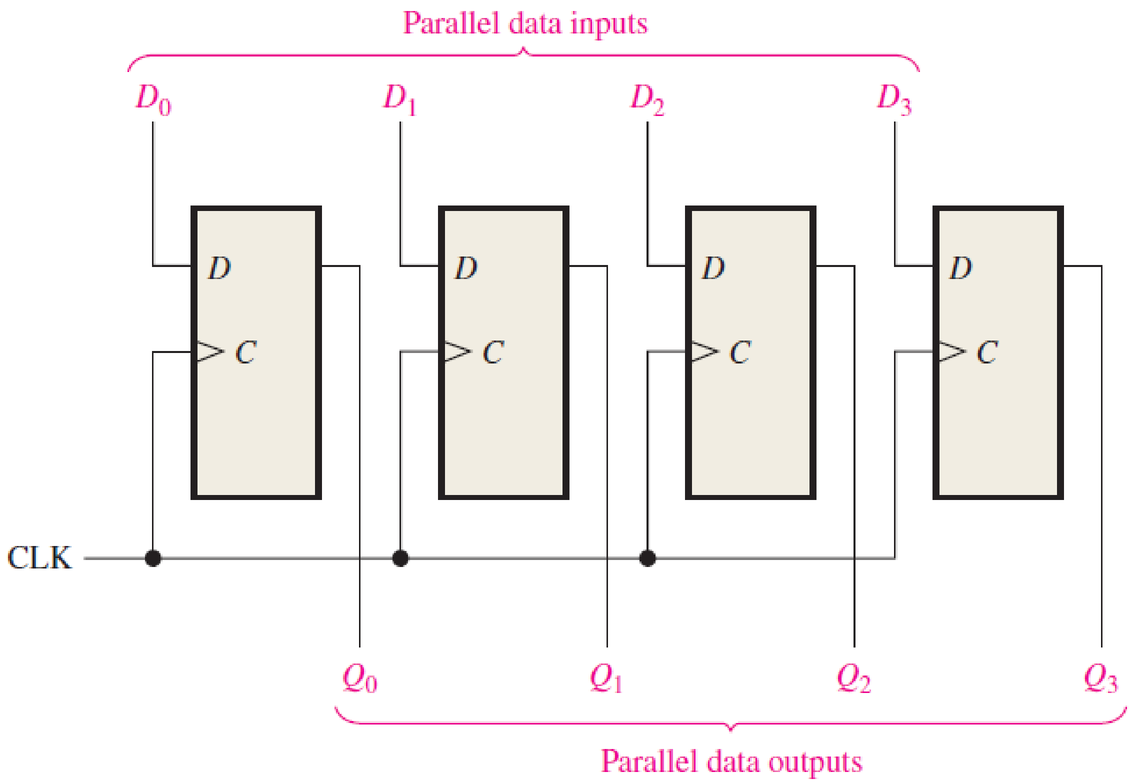


FIGURE 8–14 A parallel in/parallel out register.

Bidirectional Shift Registers

- A bidirectional shift register is one in which the data can be shifted either left or right.
- It can be implemented by using gating logic that enables the transfer of a data bit from one stage to the next stage to the right or to the left, depending on the level of a control line.

RIGHT/~~LEFT~~ is HIGH: G1 to G4 shift right

RIGHT/~~LEFT~~ is LOW: G5 to G8 shift left

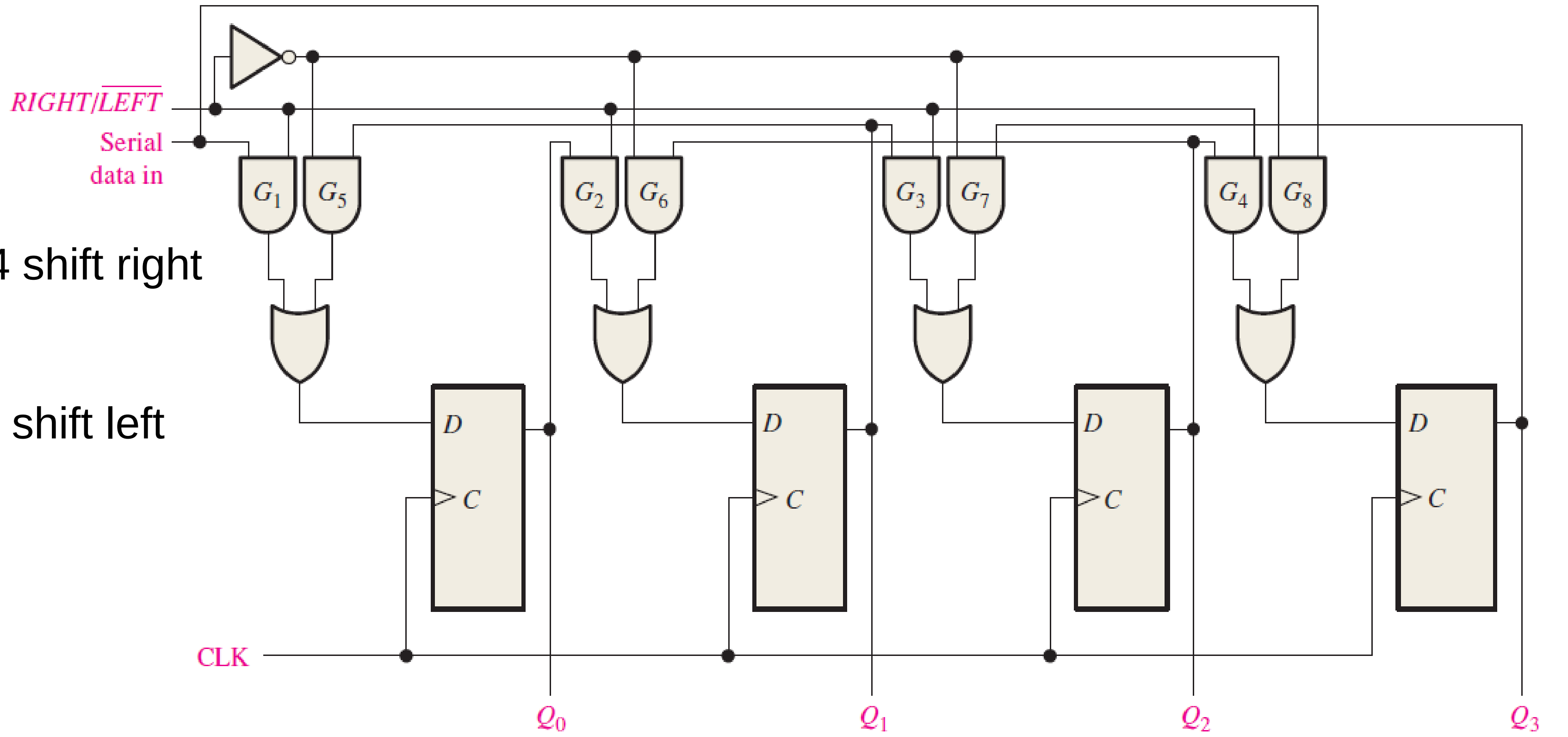


FIGURE 8–17 Four-bit bidirectional shift register.

Serial addition using D flip-flops

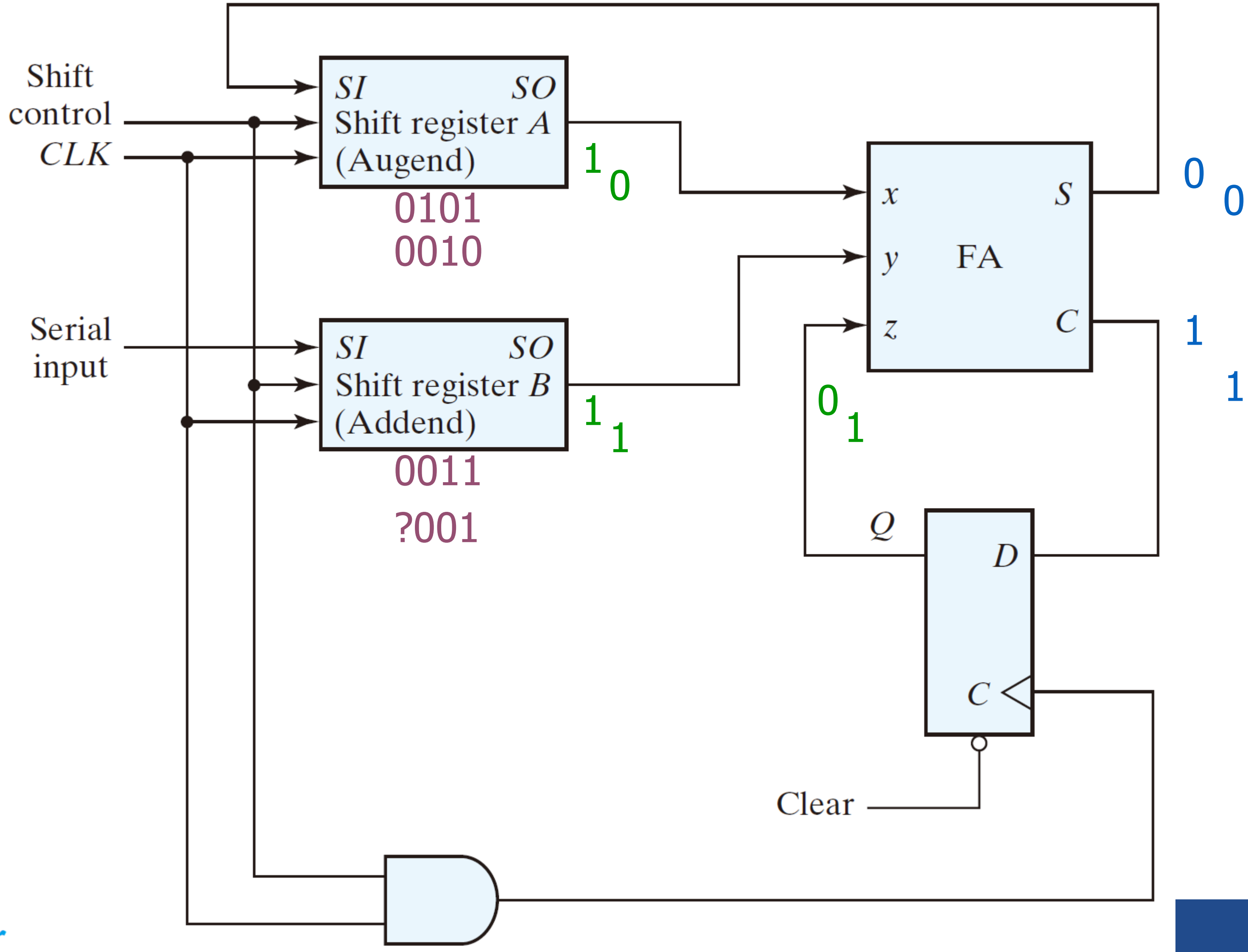


FIGURE 6.5
Serial adder

Universal Shift Register

- Some shift registers provide the necessary input and output terminals for parallel transfer. They may also have both shift-right and shift-left capabilities. The most general shift register has the following capabilities:

1. A *clear* control to clear the register to 0.
2. A *clock* input to synchronize the operations.
3. A *shift-right* control to enable the shift-right operation and the *serial input* and *output* lines associated with the shift right.
4. A *shift-left* control to enable the shift-left operation and the *serial input* and *output* lines associated with the shift left.
5. A *parallel-load* control to enable a parallel transfer and the *n* input lines associated with the parallel transfer.
6. *n* parallel output lines.
7. A control state that leaves the information in the register unchanged in response to the clock. Other shift registers may have only some of the preceding functions, with at least one shift operation.

Universal Shift Register

- A register capable of shifting in one direction only is a ***unidirectional*** shift register.
- One that can shift in both directions is a ***bidirectional*** shift register. If the register has both shifts and parallel-load capabilities, it is referred to as a ***universal shift register***.

Table 6.3
Function Table for the Register of Fig. 6.7

Mode Control		
s_1	s_0	**Register Operation**
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

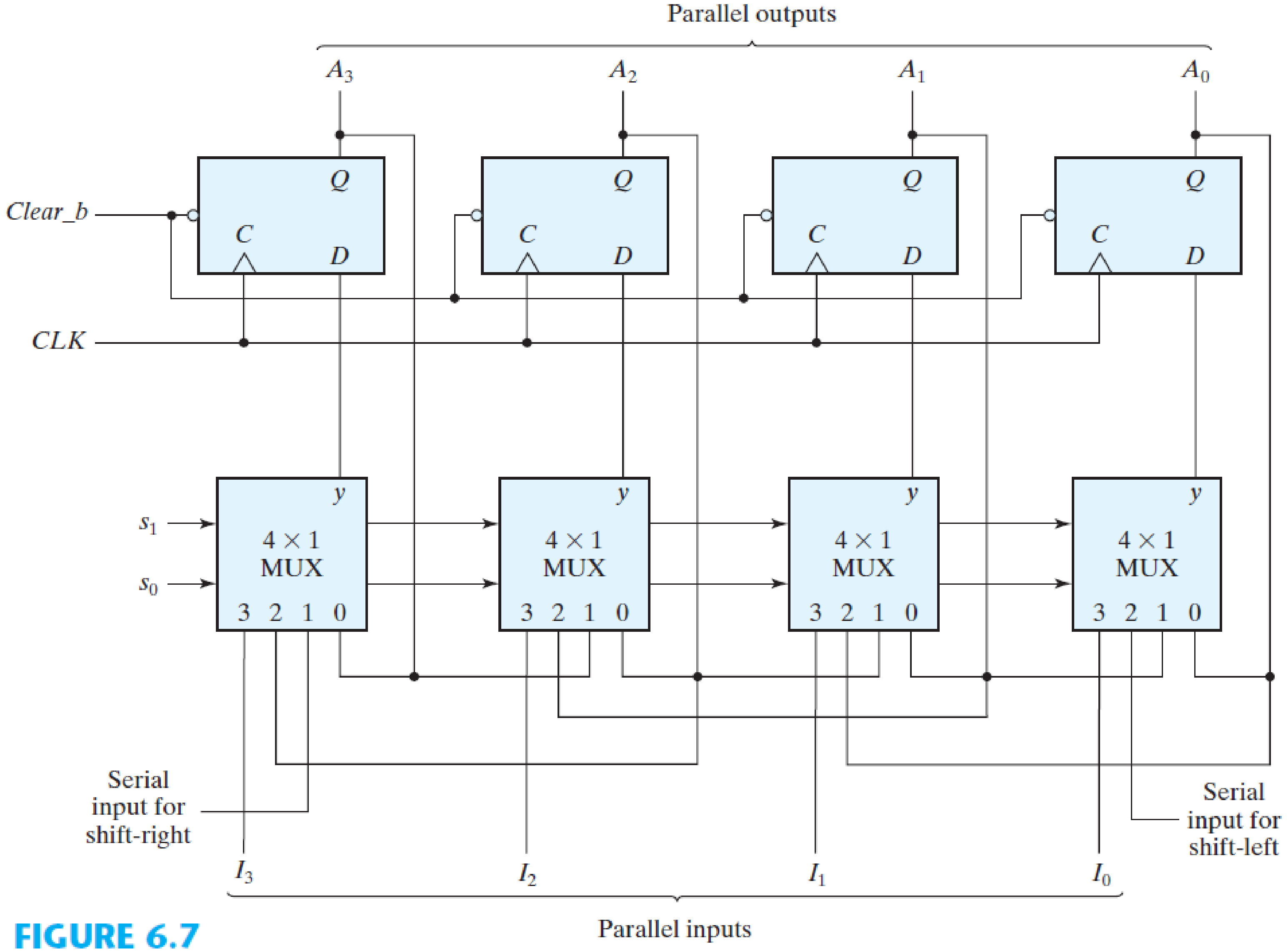
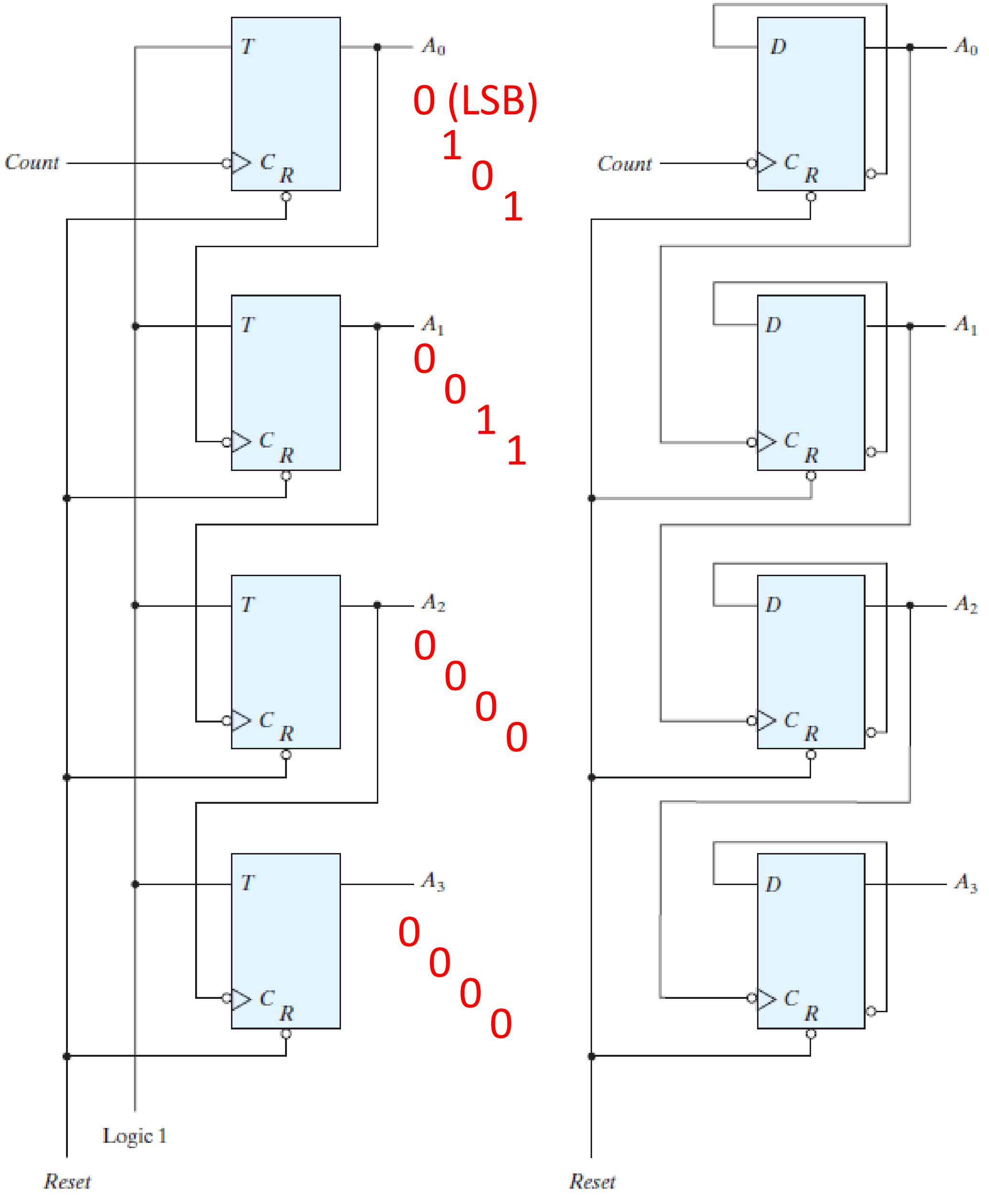


FIGURE 6.7
Four-bit universal shift register

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.
- 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.
- 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 a **synchronous counter**, the C inputs of all flip-flops receive the common clock.

Binary Ripple Counters



(a) With T flip-flops (b) With D flip-flops

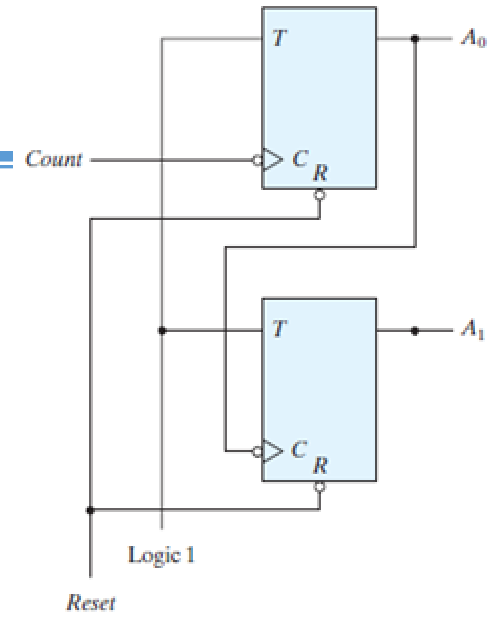
FIGURE 6.8
Four-bit binary ripple counter

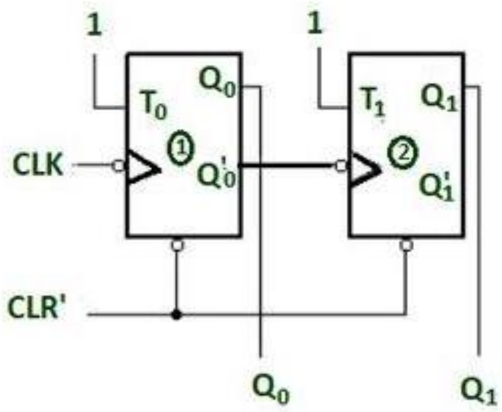
- The count starts with binary 0 and increments by 1 with each count pulse input. After the count of 15, the counter goes back to 0 to repeat the count.
- The least significant bit, A_0, is complemented with each count pulse input.
- Every time that A_0 goes from 1 to 0, it complements A_1.
- Every time 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.

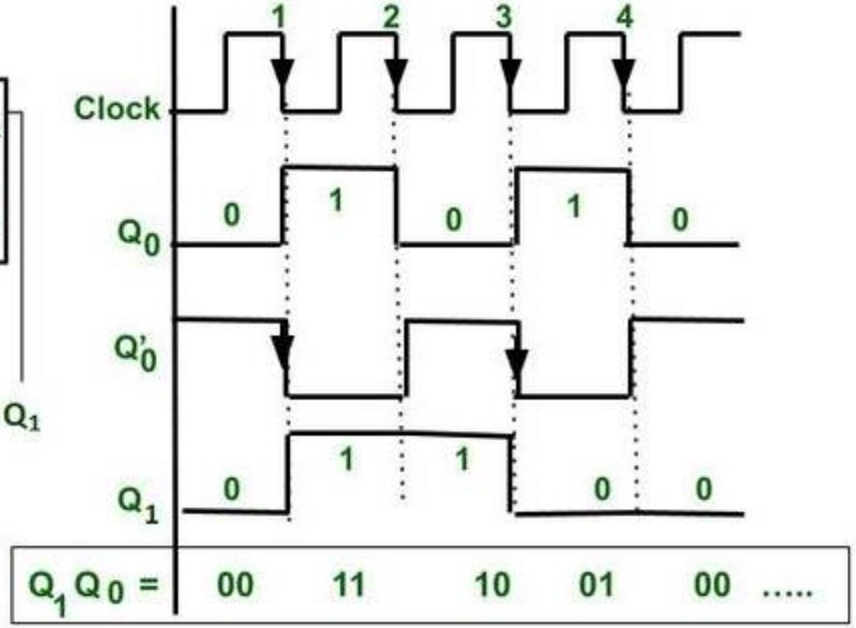
Table 6.4
Binary Count Sequence

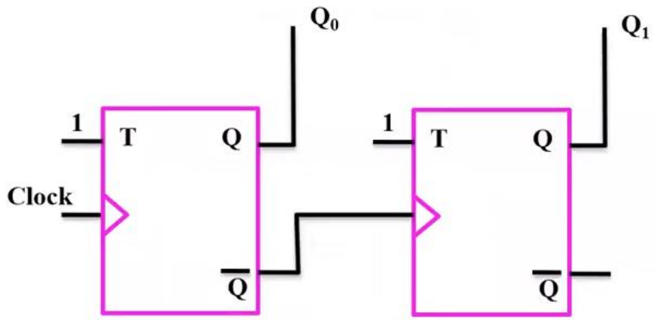
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

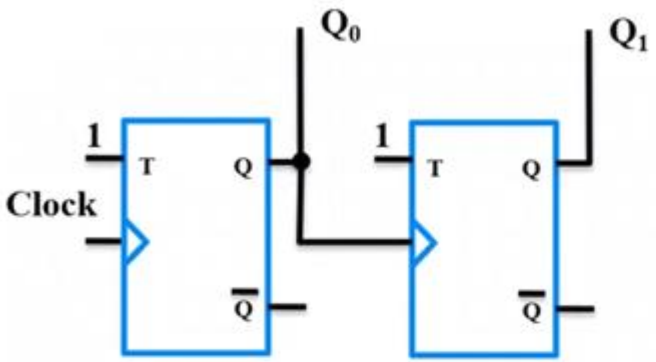
2-bit Binary countdown counter?











Up/Down counter

Synchronous Counters

- Sync counter
  - A common clock triggers all flip-flops simultaneously
- Design procedure
  - apply the same procedure of sync seq ckts
  - Sync counter is simpler than general sync seq ckts

4-bit binary counter

- 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*.
- The counter is enabled by *Count_enable*.

Countdown Binary Counter?

- A bit in any other position is complemented if all lower significant bits are equal to 0.
- Inputs to the AND gates must come from the complemented outputs of the previous flip-flops.

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

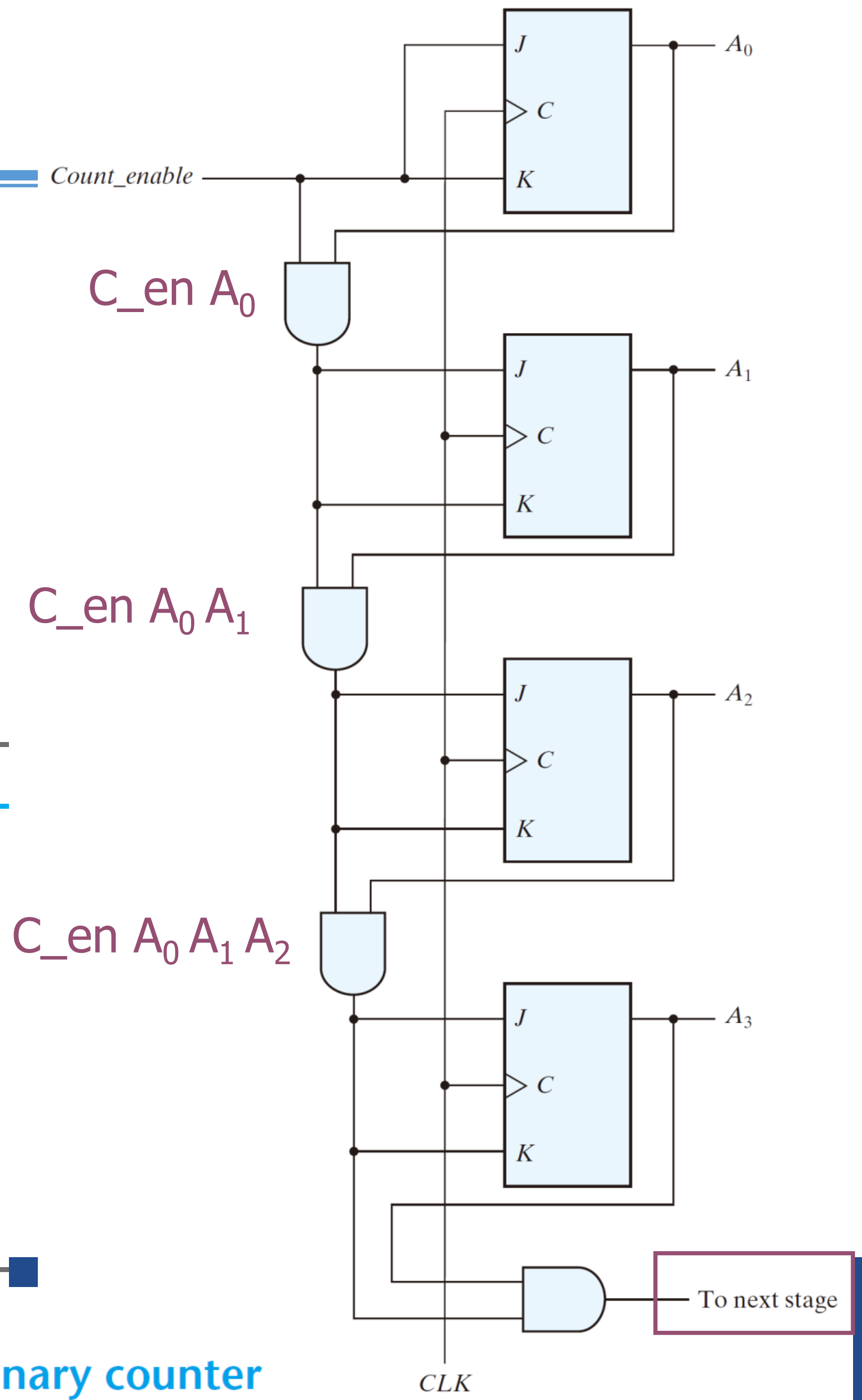
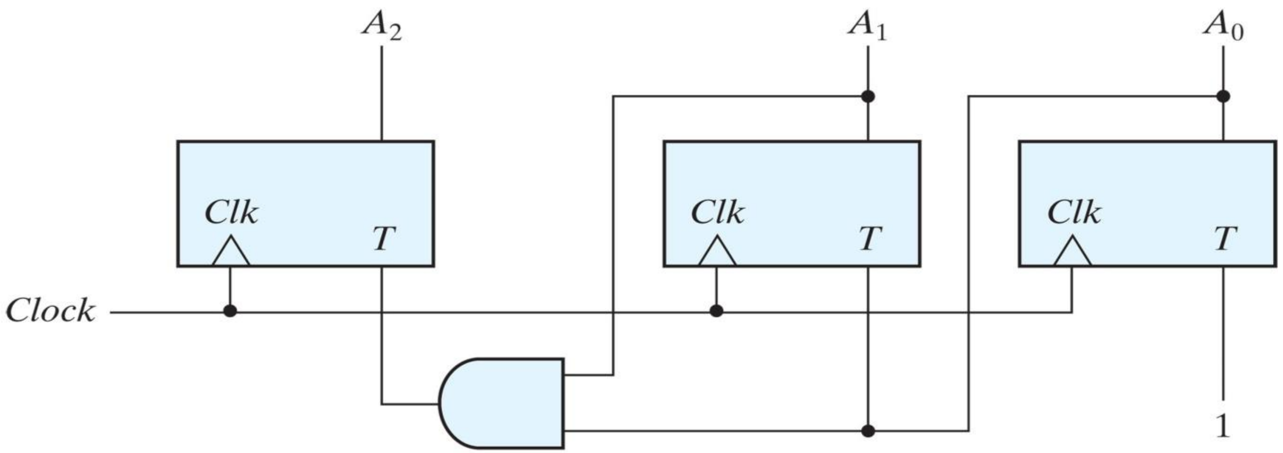
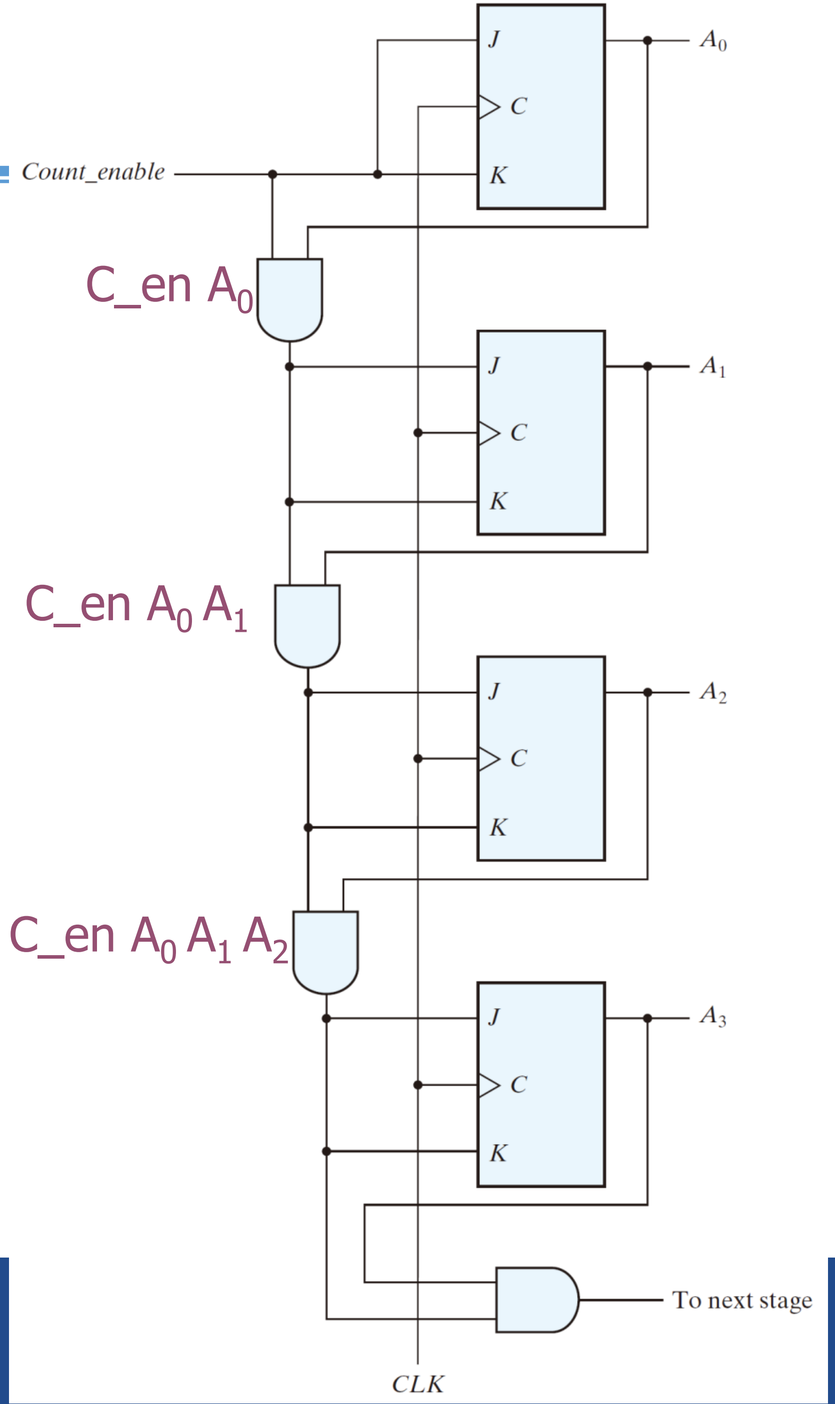


FIGURE 6.12
Four-bit synchronous binary counter



Copyright ©2013 Pearson Education, publishing as Prentice Hall



4-bit up/down binary counter

Up	Down	Operation
1	0	Up counting
0	1	Down counting
0	0	No change
1	1	Up counting

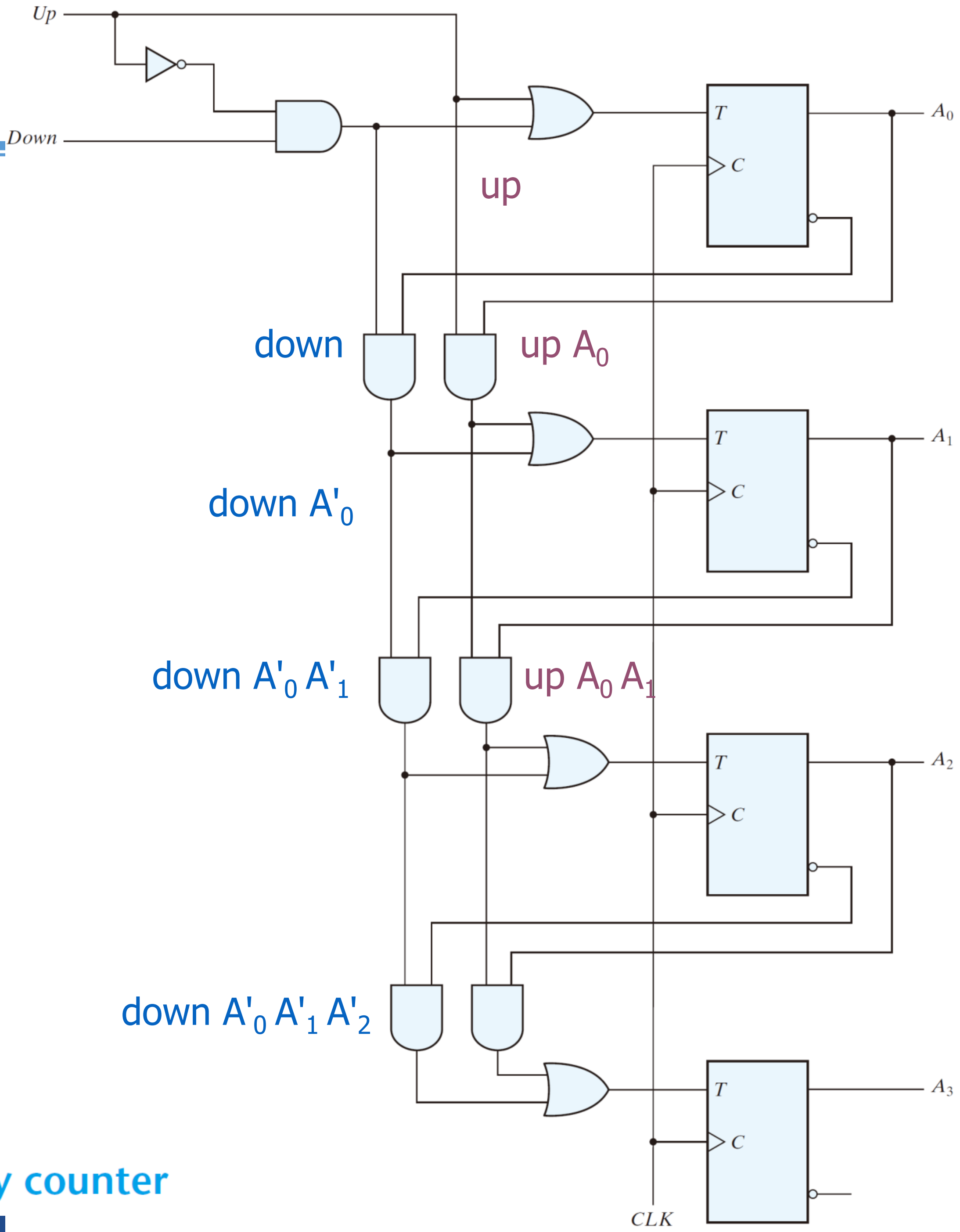


FIGURE 6.13
Four-bit up–down binary counter

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

Simplified functions:

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

Generate any count sequence

- E.g.: BCD counter ⇐ Counter w/ parallel load

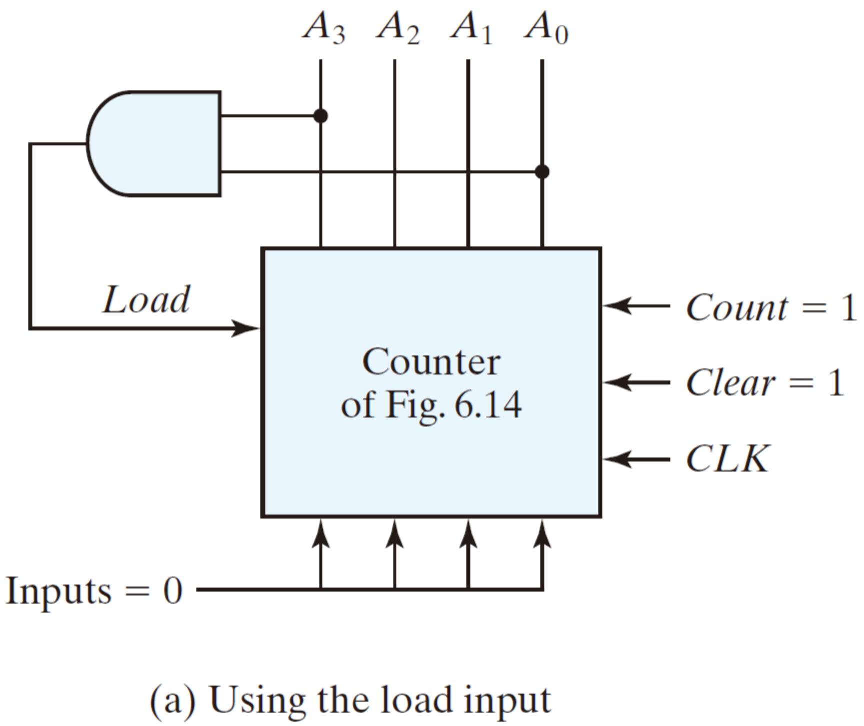


FIGURE 6.15
Two ways to achieve a BCD counter using a counter with parallel load

Counter with Unused States

- *n* flip-flops $\Rightarrow 2^n$ binary states
- Unused states
  - states that are not used in specifying the FSM
  - may be treated as don't-care conditions or
    may be assigned specific next states
- Self-correcting counter
  - Ensure that when a ckt enter one of its unused states, it eventually goes into one of the valid states after one or more clock pulses so it can resume normal operation.

    $\Rightarrow$ Analyze the ckt to determine the next state from an
    unused state after it is designed

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

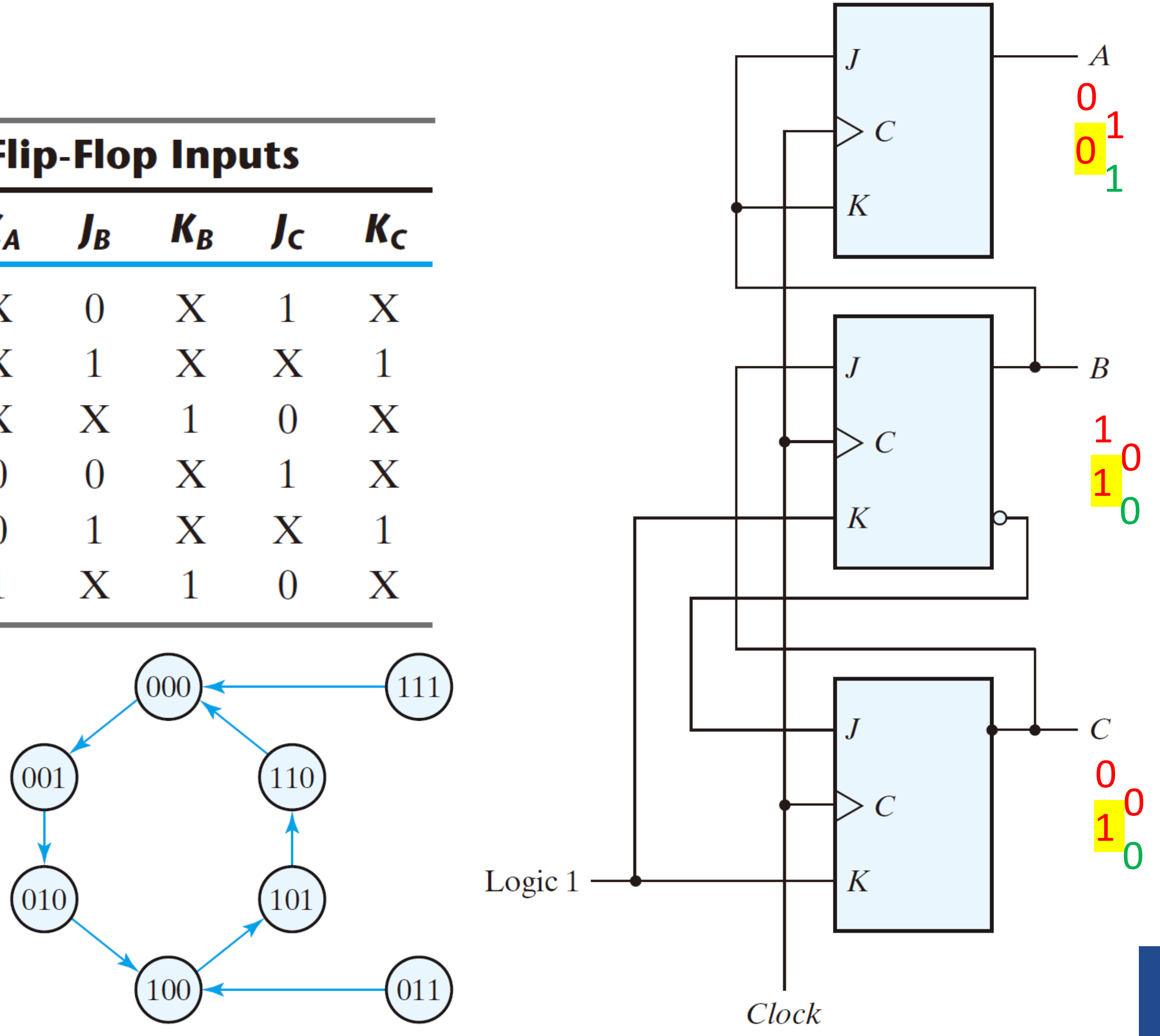
Two unused states: 011 & 111

The simplified flip-flop input equations:

$J_A = B$, $K_A = B$

$J_B = C$, $K_B = 1$

$J_C = B'$, $K_C = 1$



(b) State transition diagram

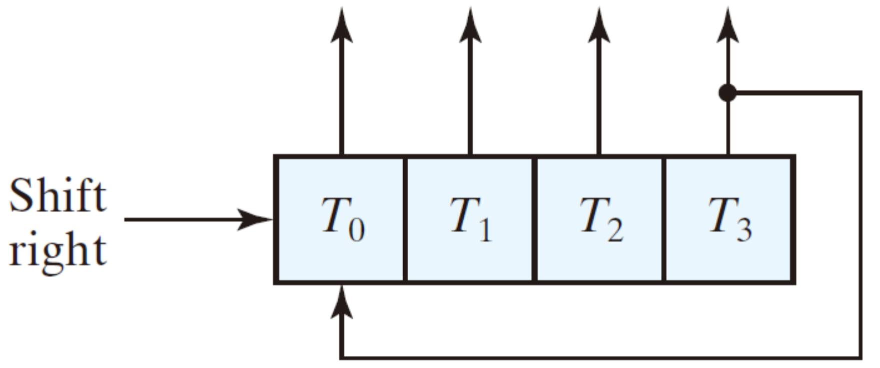
(a) Logic circuit diagram

FIGURE 6.16
Counter with unused states

Ring counter

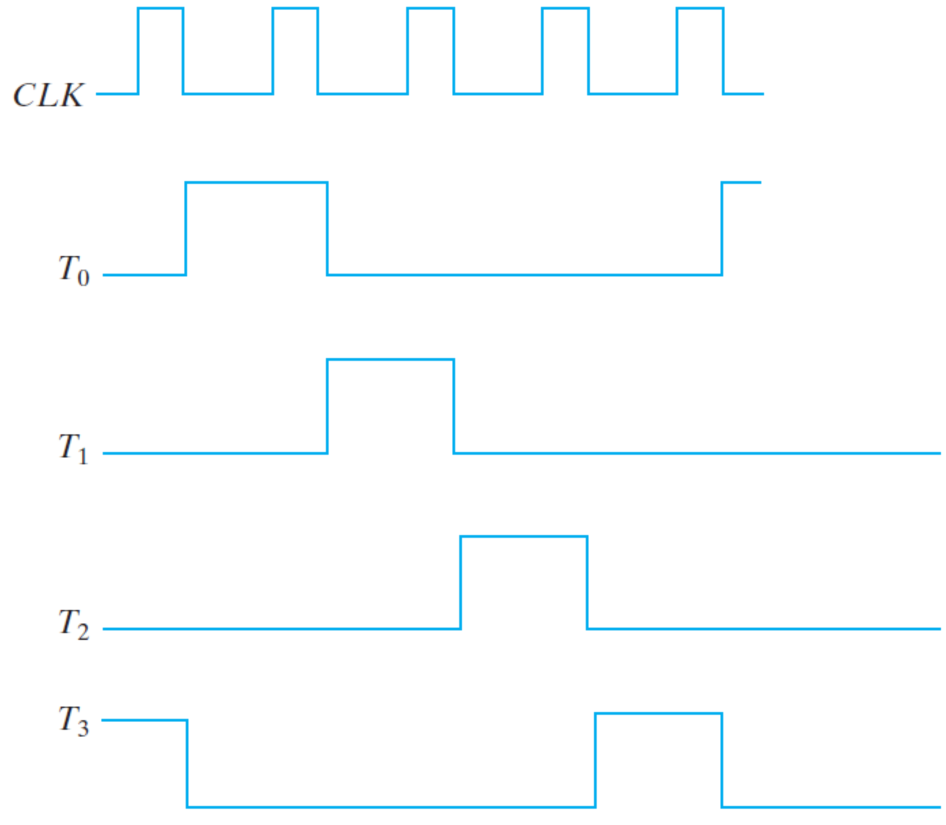
- A ***ring counter*** is a circular shift register w/ only one flip-flop being set at any particular time, all others are cleared (initial value = 1 0 0 … 0)
- The initial value of the register is 1000 and requires Preset/Clear flip-flops.
- The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals.

T_0	T_1	T_2	T_3
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0



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

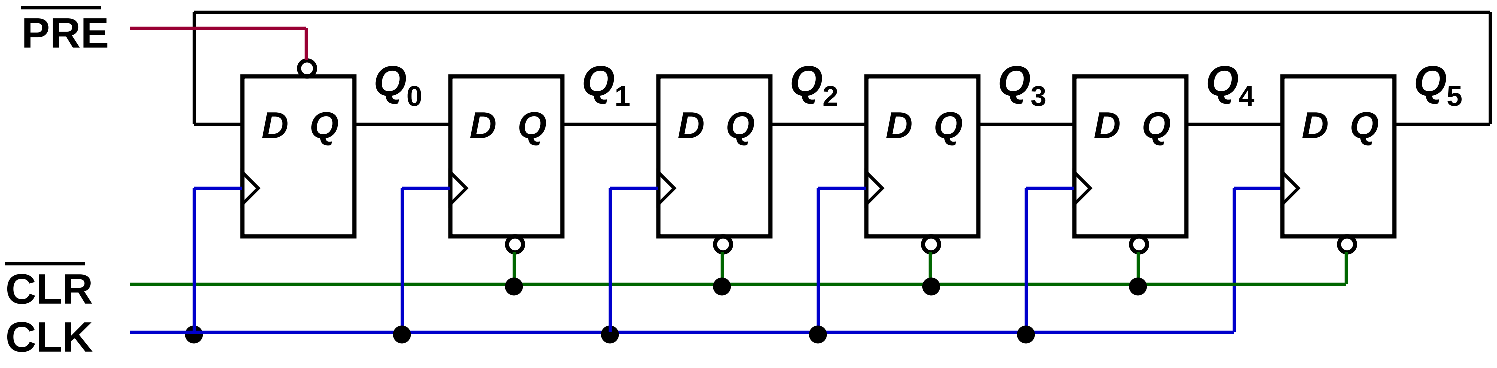
FIGURE 6.17
Generation of timing signals



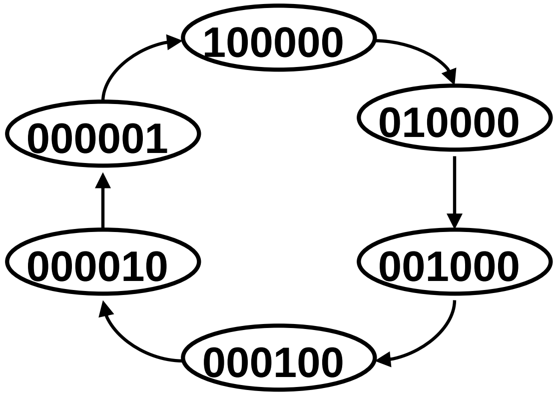
(b) Sequence of four timing signals

Ring counter

- Example: A 6-bit (MOD-6) ring counter.

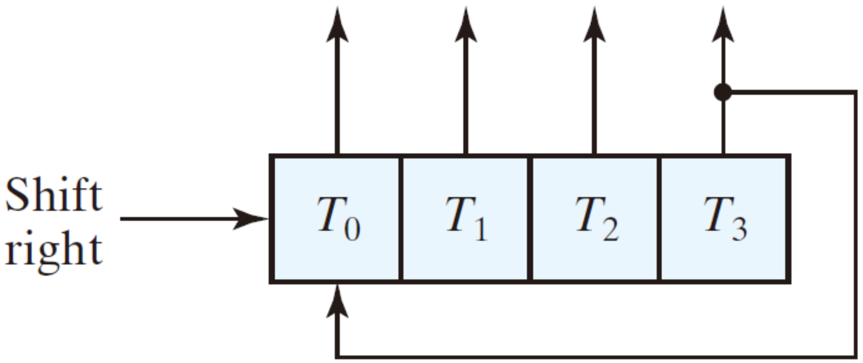


Clock	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

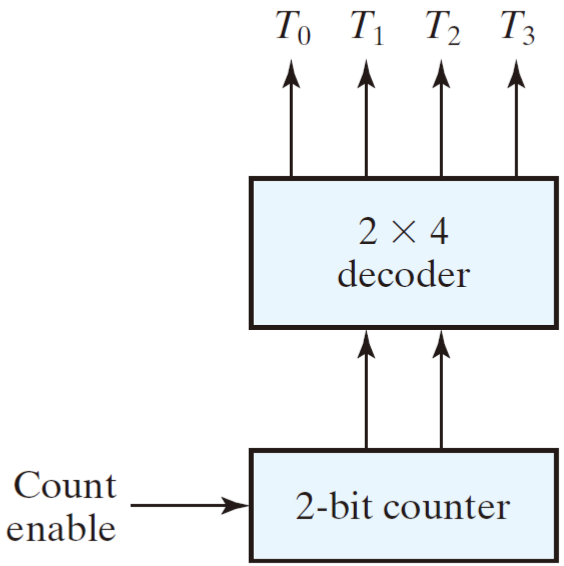


Ring counter

- Application of counters
  - Counters may be used to generate timing signals to control the sequence of operations in a digital system.
- Simple approach for generating 2^n timing signals
  1. a shift register w/ 2^n flip-flops
  2. an n-bit binary counter together w/ an n-to-2^n-line decoder



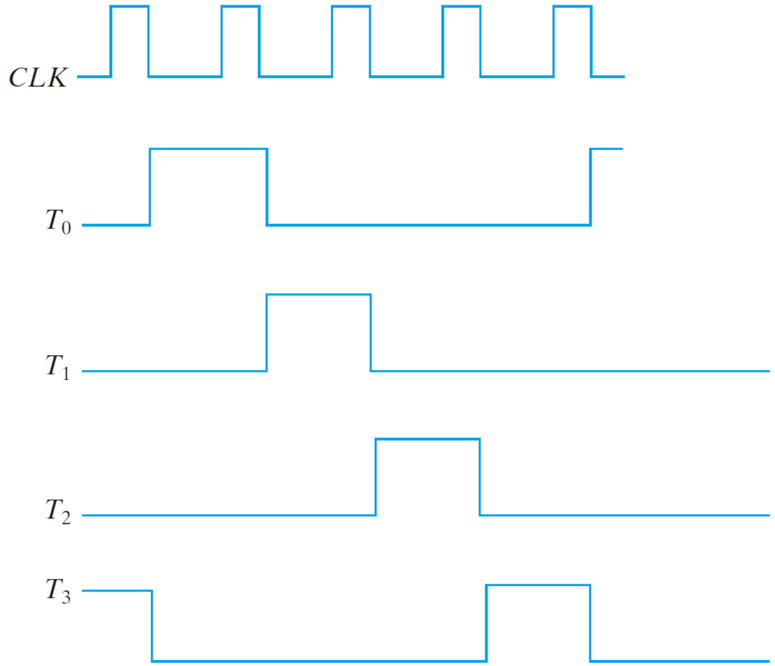
(a) Ring-counter (initial value = 1000)



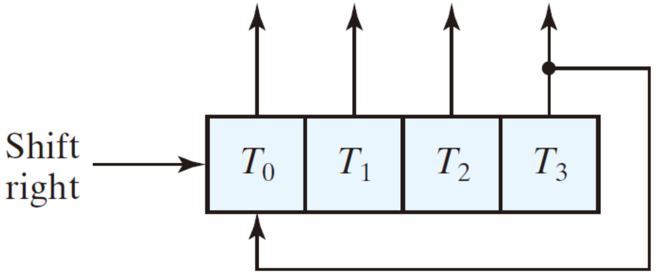
(c) Counter and decoder

FIGURE 6.17
Generation of timing signals

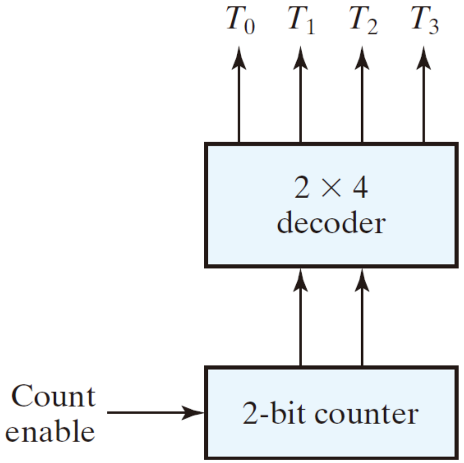
Ring counter



(b) Sequence of four timing signals



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



(c) Counter and decoder

FIGURE 6.17
Generation of timing signals

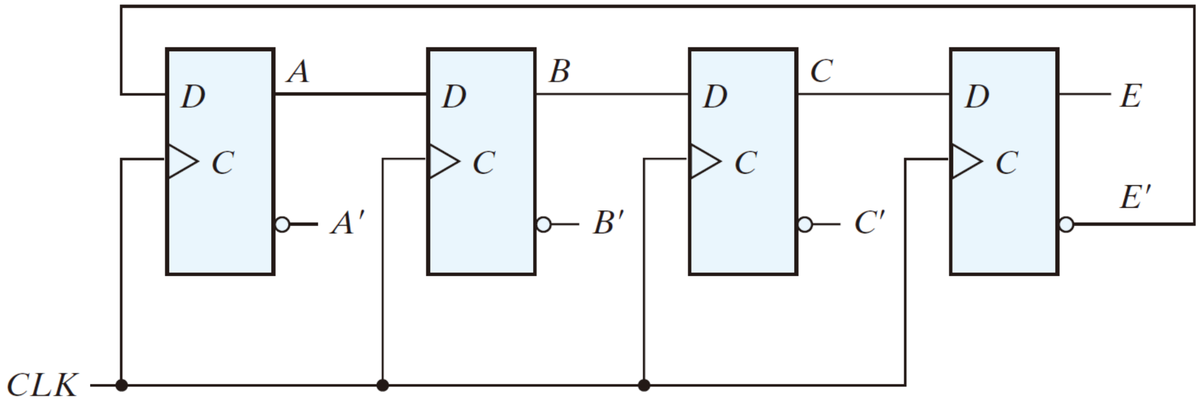
- It is also possible to generate the timing signals with a combination of a shift register and a decoder.
- That way, the number of flip-flops is less than that in a ring counter, and the decoder requires only two-input gates. This combination is called a ***Johnson counter***.

8 timing signals?

Johnson counter

- Ring counter vs. Switch-tail ring counter
  - Ring counter
    - a *k*-bit ring counter circulates a single bit among the flip-flops to provide *k* distinguishable states.
  - Switch-tail ring counter
    - is a circular shift register w/ the complement output of the last flip-flop connected to the input of the first flip-flop
    - a *k*-bit switch-tail ring counter will go through a sequence of 2*k* distinguishable states. (initial value = 0 0 … 0)

Switch-tail ring counter



(a) Four-stage switch-tail ring counter

Sequence number	Flip-flop outputs				AND gate required for output
	A	B	C	E	
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

Johnson counter

- a *k*-bit switch-tail ring counter + 2*k* decoding gates
- provide outputs for 2*k* timing signals
  - E.g.: 4-bit Johnson 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

Valid states:

0, 8, 12, 14, 15, 7, 3, 1, 0,..

Invalid states:

2, 4, 5, 6, 9,10,11,13

- The decoding follows a regular pattern:
  - 2 inputs per decoding gate

Johnson counter

- Disadv. of the switch-tail ring counter
  - if it finds itself in an unused state, it will persist to circulate in the invalid states and never find its way to a valid state.
  - One correcting procedure: $D_C = (A + C)\, B$

- Summary:
  - Johnson counters can be constructed for any # of timing sequences:

    # of flip-flops = 1/2 (the # of timing signals)

    # of decoding gates = # of timing signals

    2-input per gate

The End

Reference:

1. Digital Design (with an introduction to the Verilog HDL) 6th Edition, M. Morris Mano, Michael D. Ciletti