Chapter 6: Registers and Counters

6.1 Registers

6.2 Shift Registers

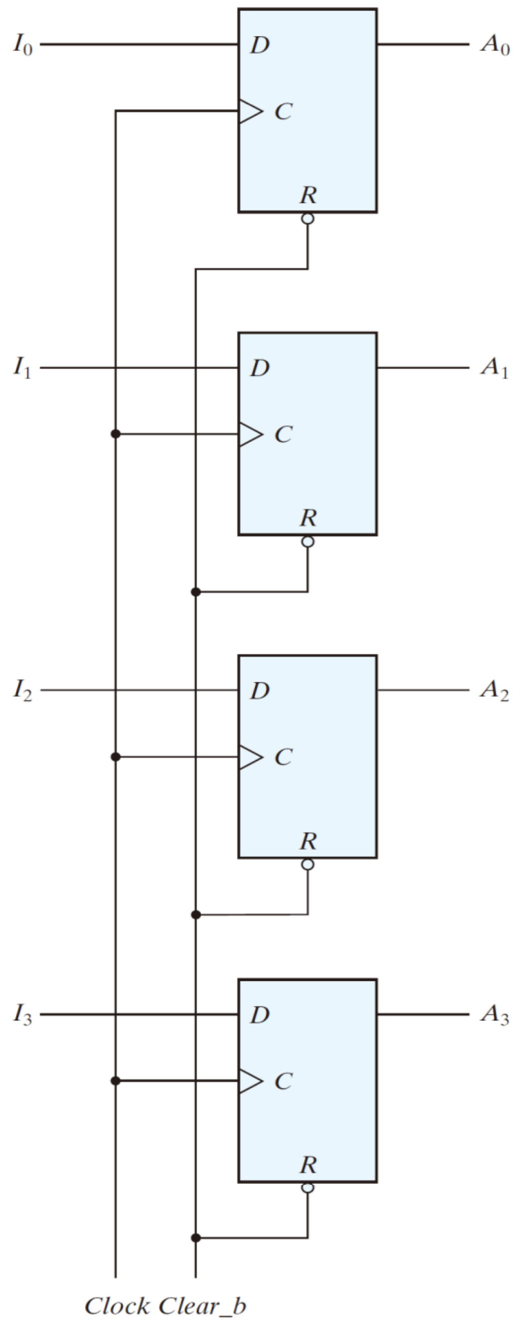
6.3 Ripple Counters

6.4 Synchronous Counters

6.5 Other Counters

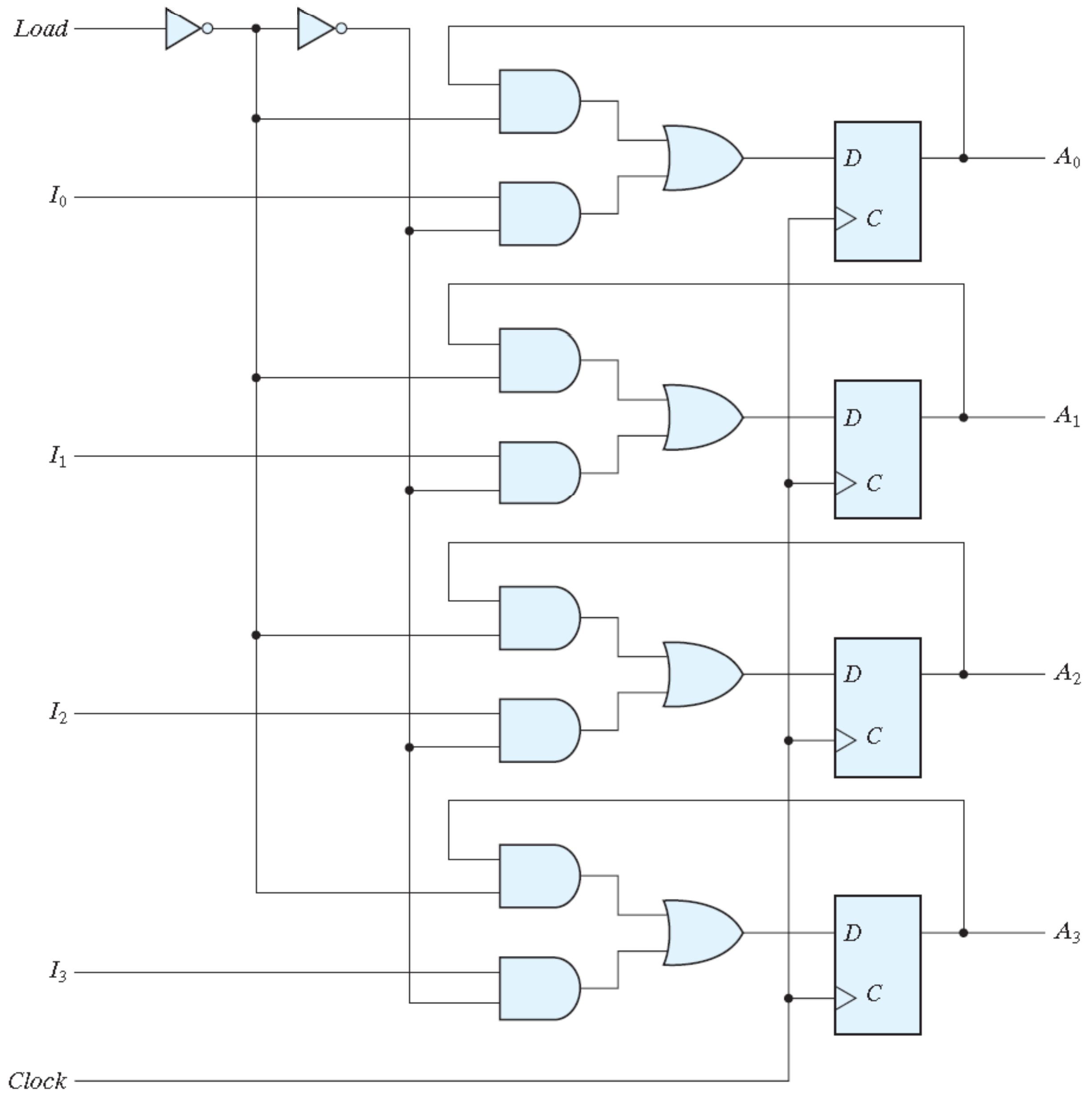
6.1 Registers

- A *register* is a group of FFs, each FF shares a common clock and is capable of storing one bit of information.
- An *n*-bit register consists of a group of *n* FFs capable of storing *n* bits of binary information.
- A register may have combinational gates that perform certain data-processing tasks.
- The simplest register is contains only FFs.
- Example: four *D*-FFs form a 4-bit data storage register.
- The clock positive edge triggers all FFs, and the binary data at the four inputs are transferred into the register.
- (I_3, I_2, I_1, I_0) determines (A_3, A_2, A_1, A_0) at the clock edge, and the four outputs can be sampled at any time.
- *Clear_b* is active-low to reset FFs asynchronously.
- The *R* inputs must be maintained at logic 1 (i.e., de-asserted) during normal clocked operation.



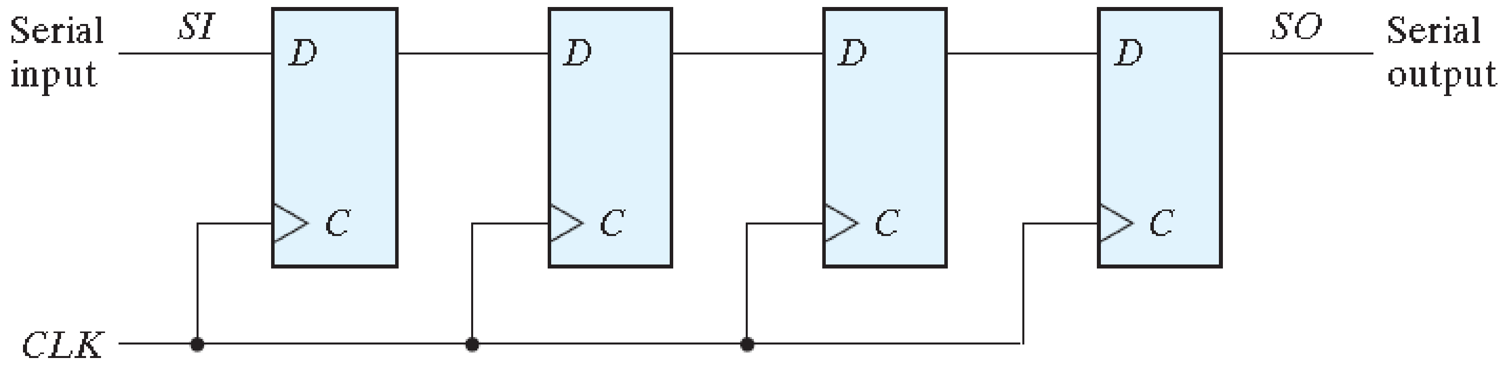
4-bit Register with Parallel Load

- The transfer of new information into a register is referred to as *loading* or *updating*.
- If all the bits of the register are loaded simultaneously with a common clock pulse, it is called as *parallel* loading.
- Two-channel mux selects either the data bus or the output of the register to the register.
- *Load* is for the selection of mux's.



6-2 Shift Registers

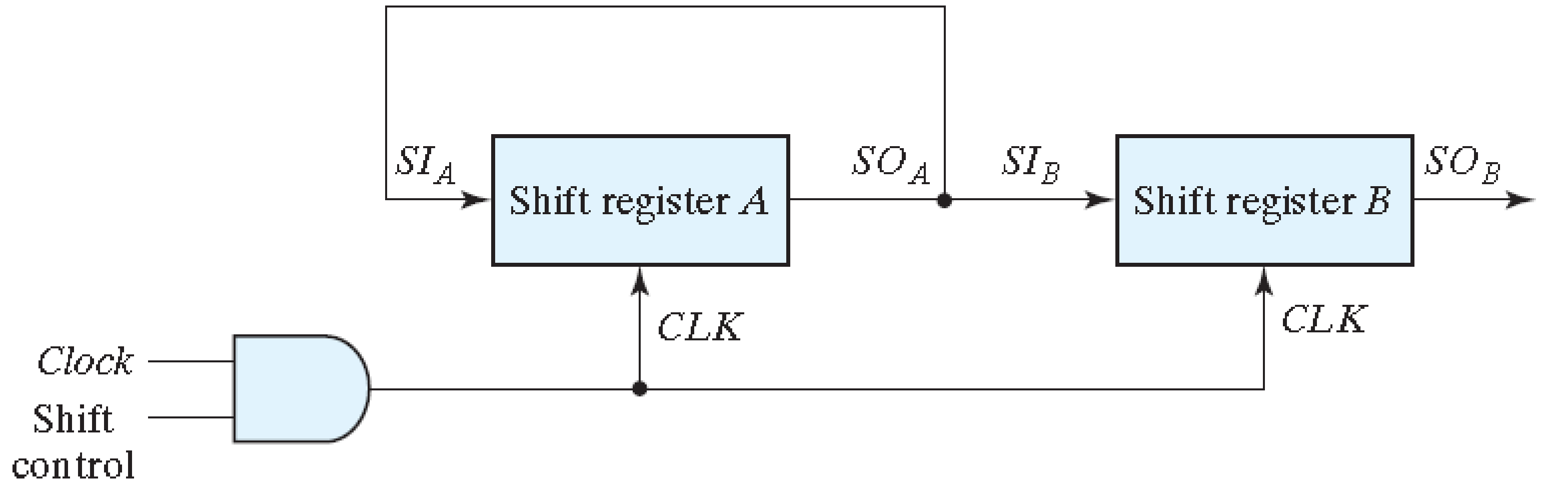
- *Shift register* is a cascaded chain of FFs with the output of each FF connected to the input of the next FF.
- It is capable of shifting the binary information held in each FF to its neighboring FF, in a selected direction.
- All FFs receive common clock to activate the shift of data.
- The simplest possible shift register uses only FFs, as shown below.



- This shift register is unidirectional (left-to-right).
- The *serial input* determines what goes into the leftmost FF during the shift.
- The *serial output* is taken from the output of the rightmost FF.

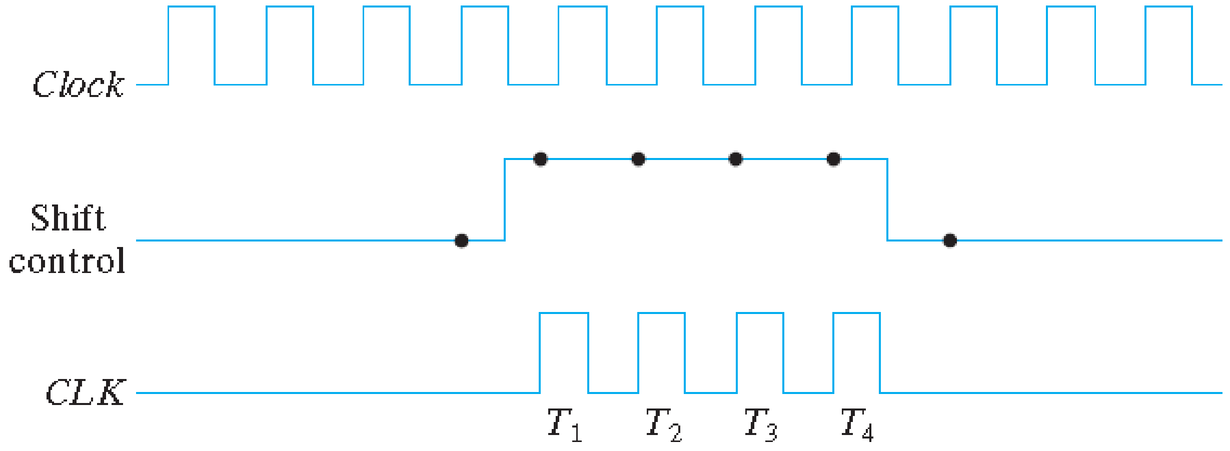
Serial Transfer

- Serial mode: information is transferred and manipulated one bit at a time by shifting the bits out of the source register and into the destination register.
- Serial transfer is done with shift registers, as shown below.



- Rotation: circulating connection of register *A* prevent the loss of information.

- *Gated clock*: the AND gate allows *Clock* pulses to pass into the *CLK* terminals only when the *Shift control* is active.



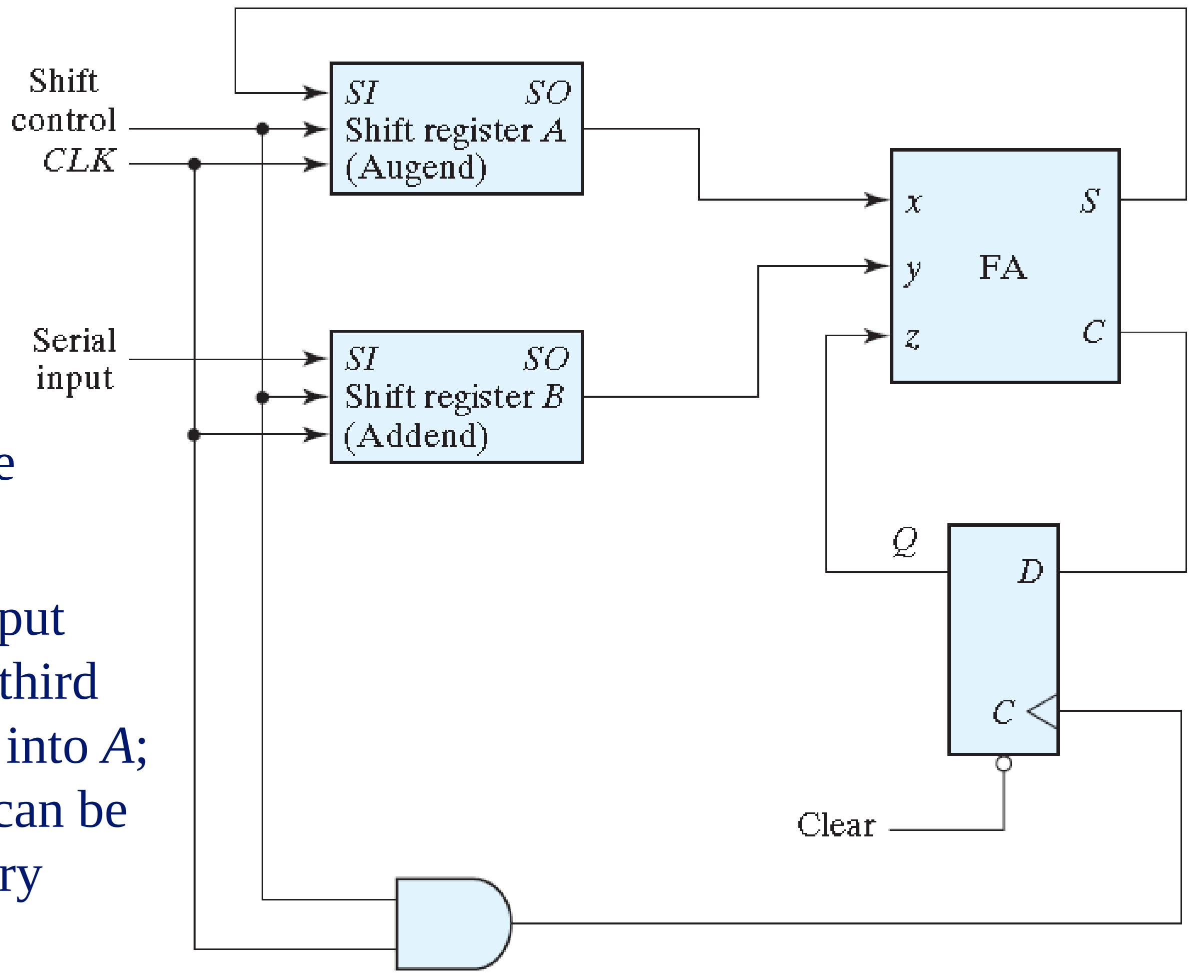
- Assume the binary content of *A* before the shift is 1011 and that of *B* is 0010.
- The serial transfer from *A* to *B* occurs in four steps.

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

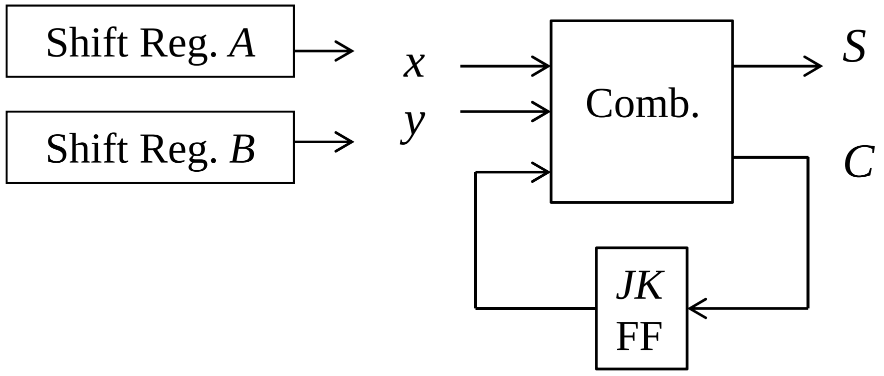
Serial Adder

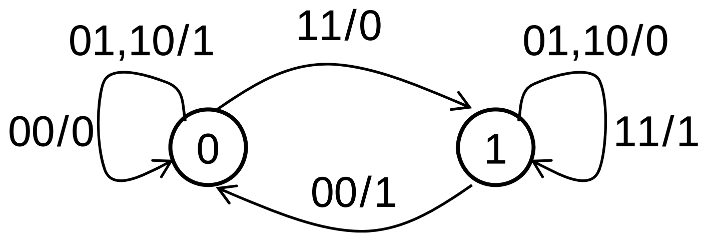
- Shift registers A and B store the two binary numbers to be added.
- A single full-adder (FA) adds one pair of bits at a time.
- The *D* FF temporarily stores the carry out for the next bit addition.



- The sum bit from the *S* output could be transferred into a third shift register or by shifting into *A*; while the serial input of *B* can be used to transfer a new binary number.

Redesign the Serial Adder with *JK* Flip-flops





- The present state of Q is the present value of the carry.

State Table for Serial Adder

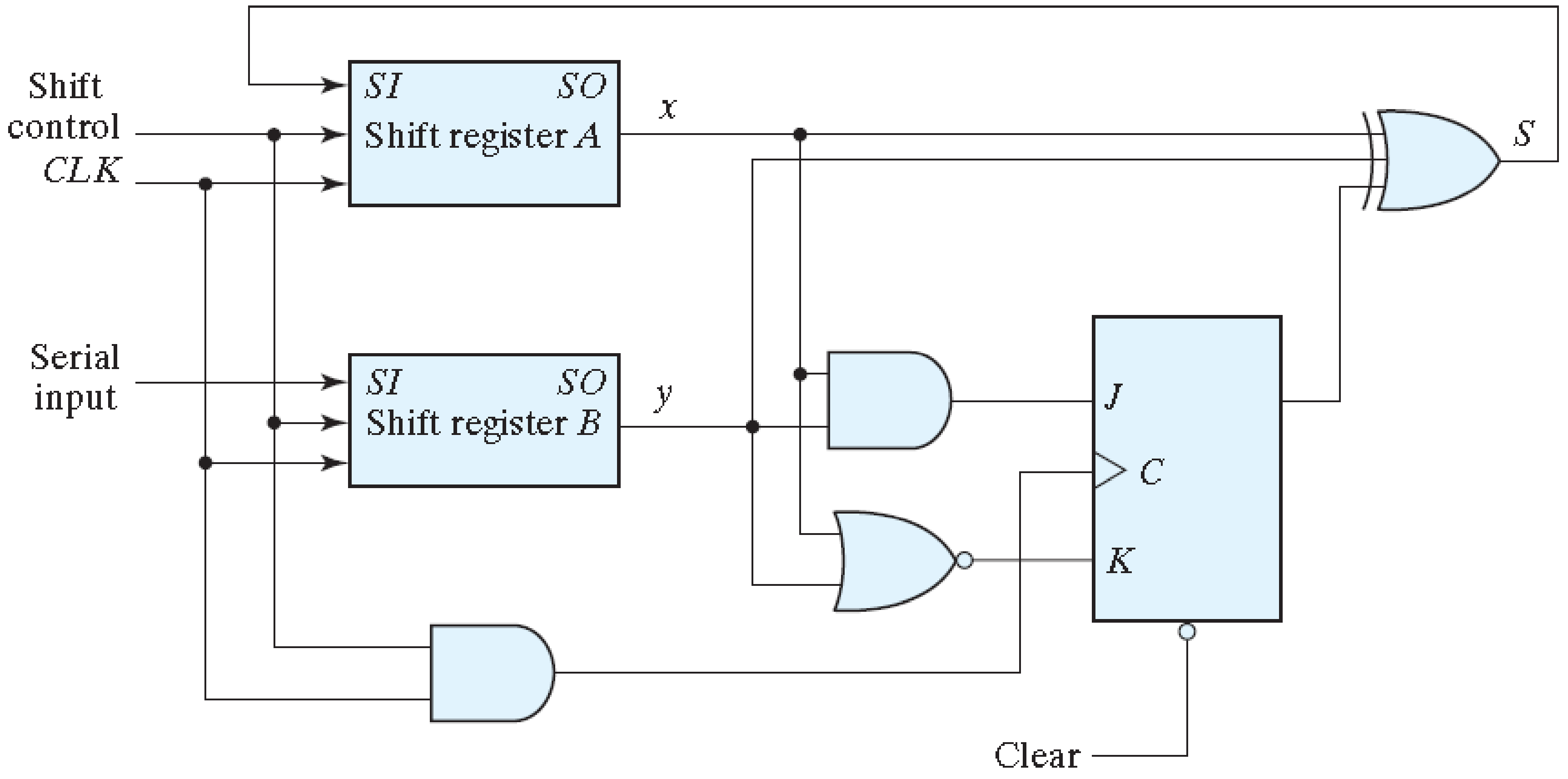
Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	x	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

***JK* Flip-Flop**

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

$$J_Q = xy, \qquad K_Q = x'y' = (x+y)', \qquad S = x \oplus y \oplus Q$$

$J_Q = xy, \qquad K_Q = x'y' = (x+y)', \qquad S = x \oplus y \oplus Q$



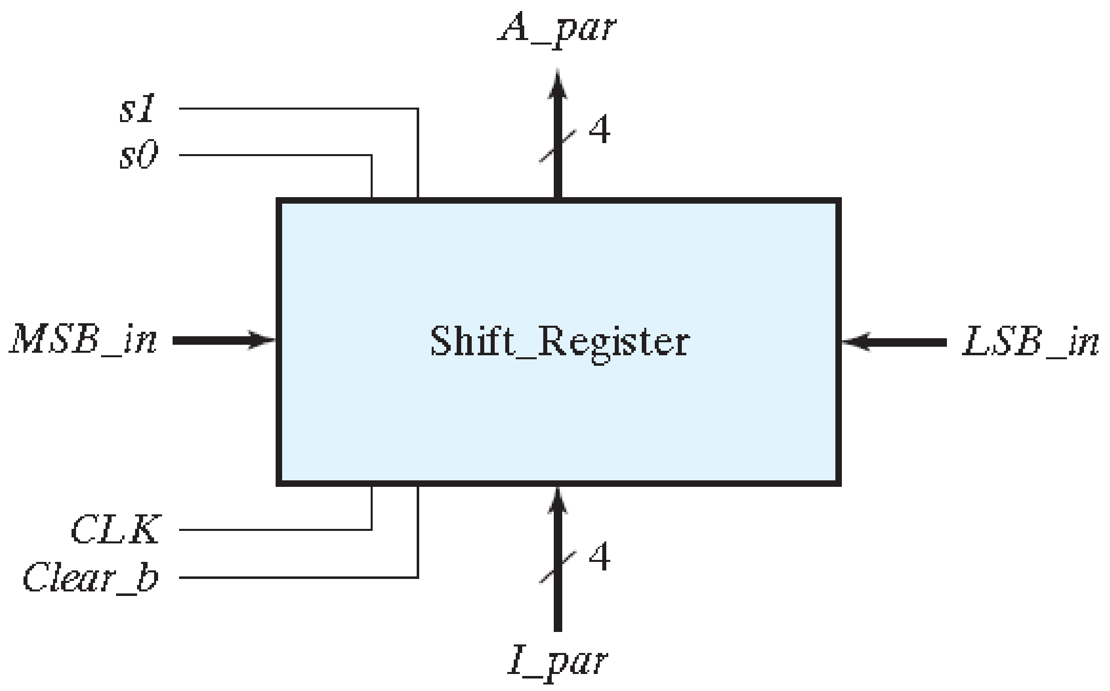
Universal Shift Register

- 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 parallel*** input lines.
  6. ***n parallel outpu***t 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.

- A *universal shift register* can shift in both directions (*bidirectional shift*), and has both shifts and parallel-load capabilities.

4-bit Universal Shift Register



Function Table for the Register of Fig. 6.7

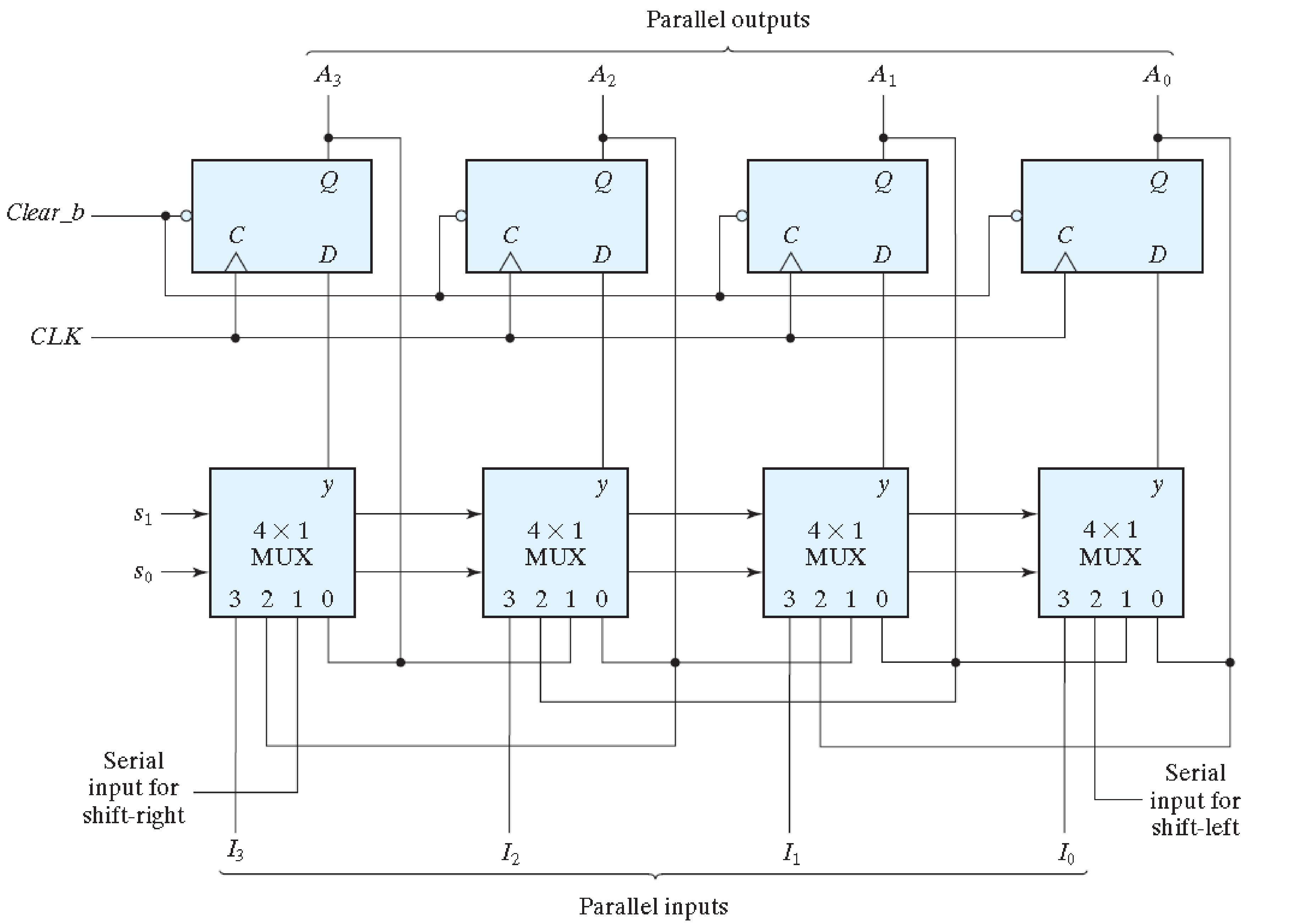
Mode Control		
s_1	s_0	**Register Operation**
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

SISO: serial in serial out

SIPO: serial in parallel out

PIPO: parallel in serial out

PISO: parallel in serial out


Parallel outputs
A_3
A_2
A_1
A_0
Clear_b
CLK
Q
C
D
s_1
s_0
y
4 × 1 MUX
3 2 1 0
Serial input for shift-right
Serial input for shift-left
I_3
I_2
I_1
I_0
Parallel inputs

6-3 Ripple Counters

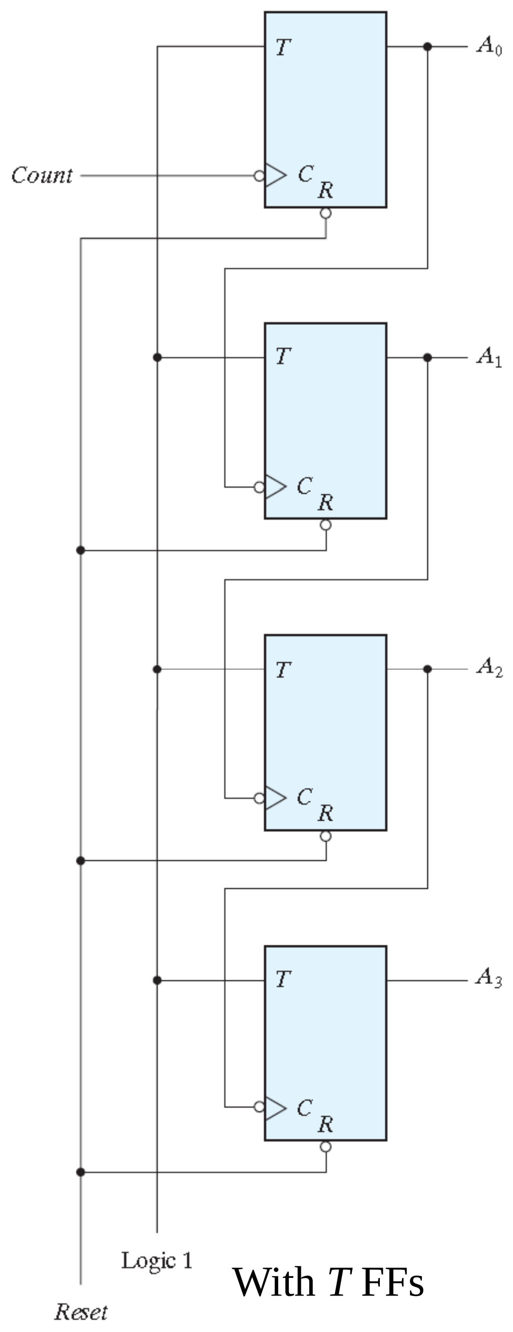
- Counter is a register that goes through a prescribed sequence of states upon the application of input pulses.
- A counter that follows the binary number sequence is called a *binary counter* .
- *n*-bit binary counter, consists of *n* FFs, can count in binary from 0 through 2^n - 1.
- Counters have two categories: *ripple counters* and *synchronous counters*.
- Ripple counter: the *C* input of some or all FFs are triggered, not by the common clock pulses, but rather by the transition that occurs in other FF outputs.
- Synchronous counter: the *C* inputs of all FFs receive the common clock.
- Here, the binary and BCD ripple counters are introduced, and synchronous counters are presented in the next two sections.

Binary Ripple Counter

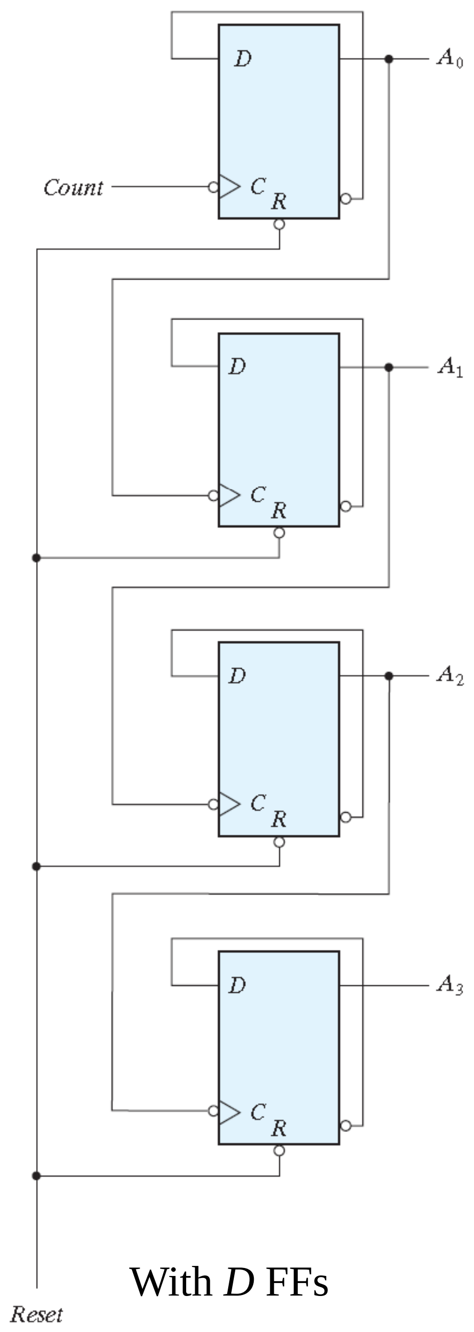
- A binary ripple counter consists of a series connection of complementing FFs, with the output of each FF connected to the *C* input of the next higher order FF.
- The FF holding the least significant bit receives the incoming count pulses.
- Complementing FF: (1) *JK* FF with the *J* and *K* inputs tied together, (2) *T* FF, (3) *D* FF with the complement output.
- Example: 4-bit binary ripple counter
  - Binary count sequence: 4-bit
  - 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.

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



With T FFs



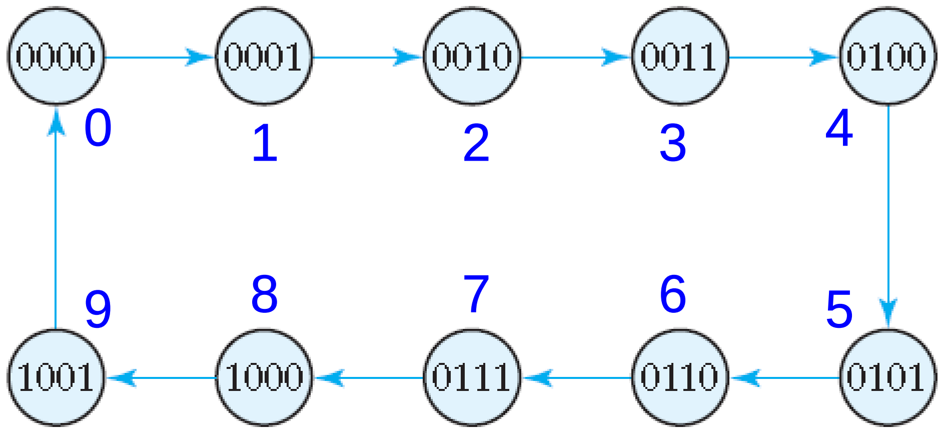
With D FFs

Binary Count Sequence

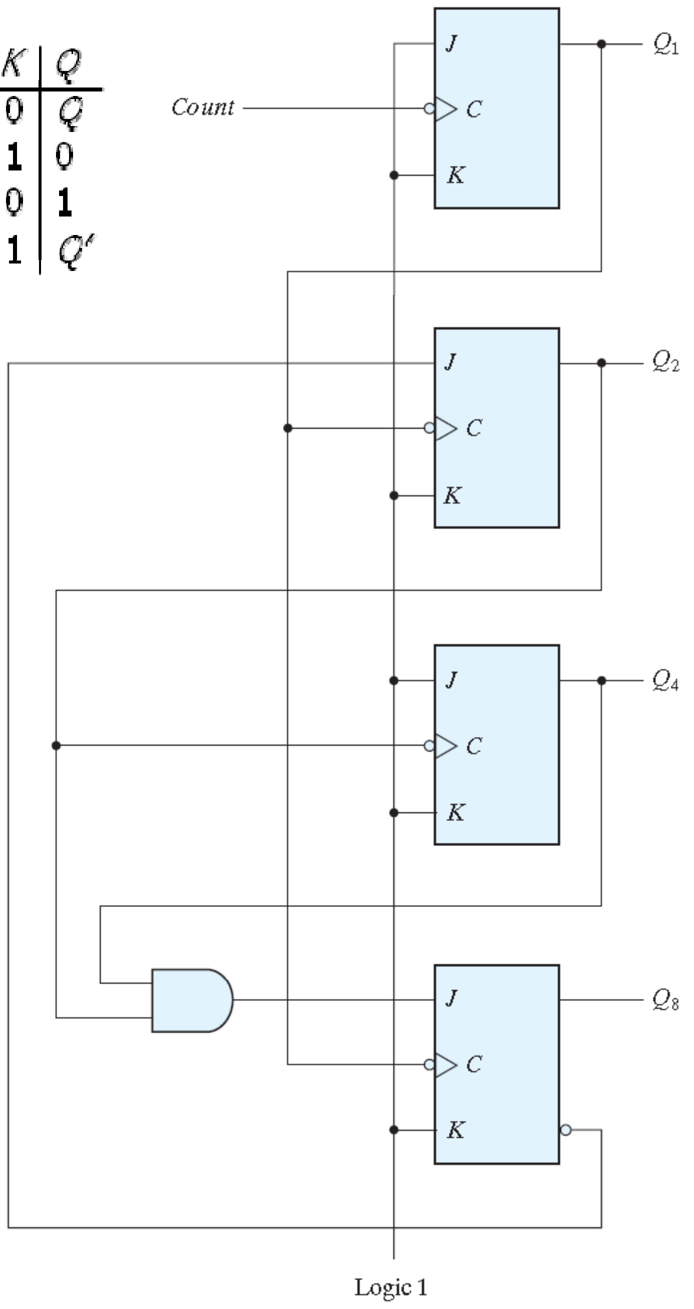
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

BCD Ripple Counter

- State diagram of a decimal BCD counter



J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	Q'



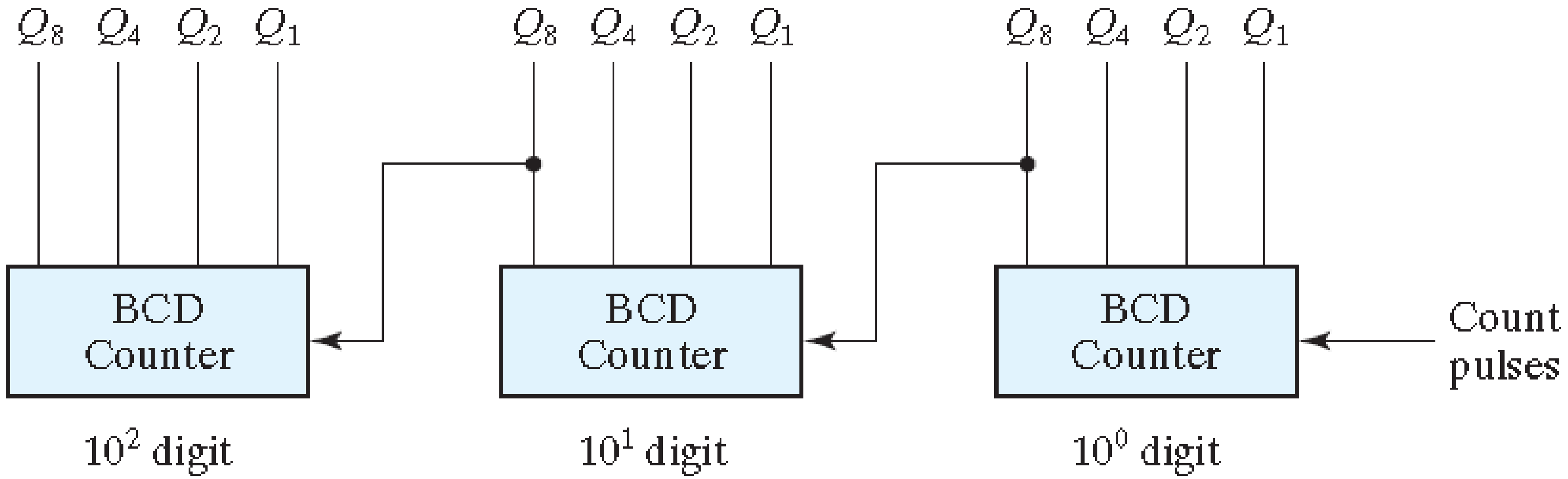
Q_1: changes state after each clock pulse.

Q_2: complements every time Q_1 goes from 1 to 0, as long as $Q_8 = 0$. When Q_8 becomes 1, Q_2 remains at 0.

Q_4: complements every time Q_2 goes from 1 to 0.

Q_8: remains at 0 as long as Q_2 or Q_4 is 0. When both Q_2 and Q_4 become 1, Q_8 complements when Q_1goes from 1 to 0. Q_8 is cleared on the next transition of Q_1.

Three-decade BCD Counter

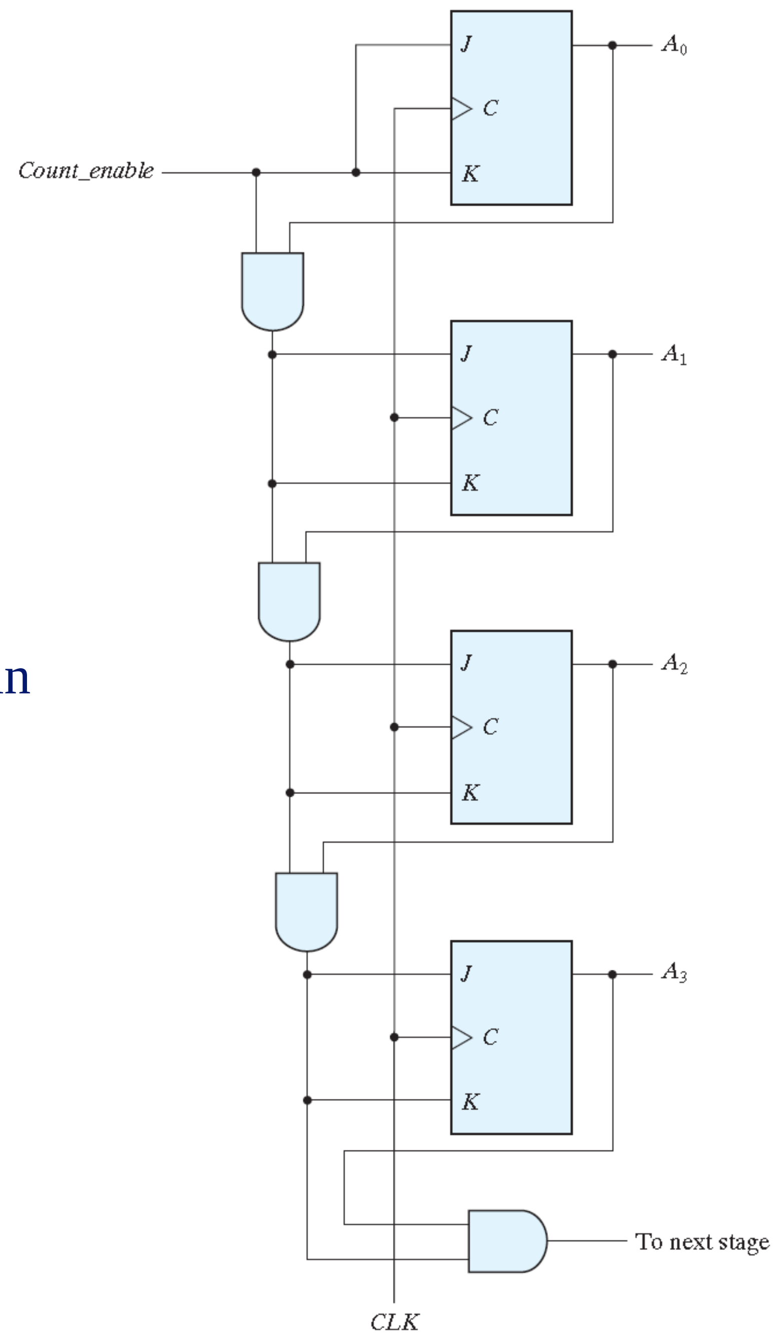


6-4 Synchronous Counters

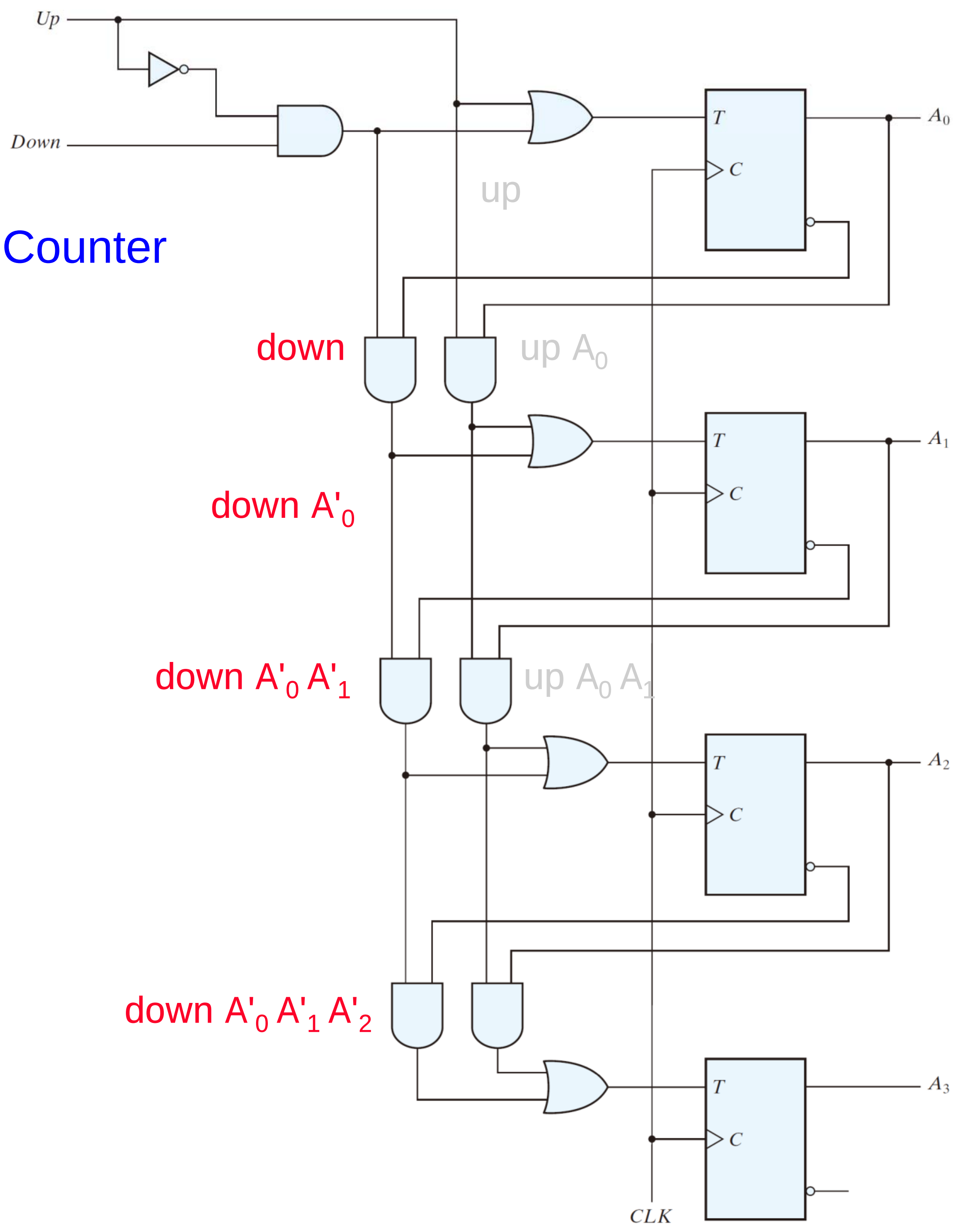
- A common clock triggers all FFs simultaneously.
- If $T = 0$ or $J = K = 0$, the FF does not change state.
- If $T = 1$ or $J = K = 1$, the FF complements.

Binary Counter

- The FF in the least significant position is complemented with every pulse, and the FF in any other position is complemented when all the bits in the lower significant positions are equal to 1.
- Synchronous binary counters have a regular pattern and can be constructed with complementing FFs and gates.



4-bit Up/down Binary Counter



BCD Counters

State Table for BCD Counter

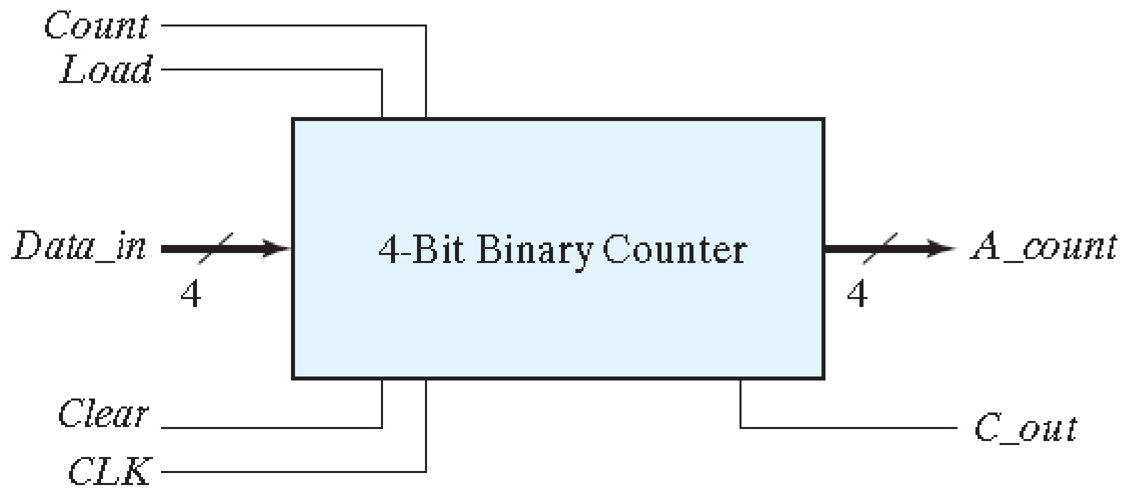
Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1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

- The FF input equations can be simplified by means of maps.
- The unused states for minterms 10 to 15 are taken as don't-care terms.
- The simplified functions are

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

$y = Q_8 Q_1$

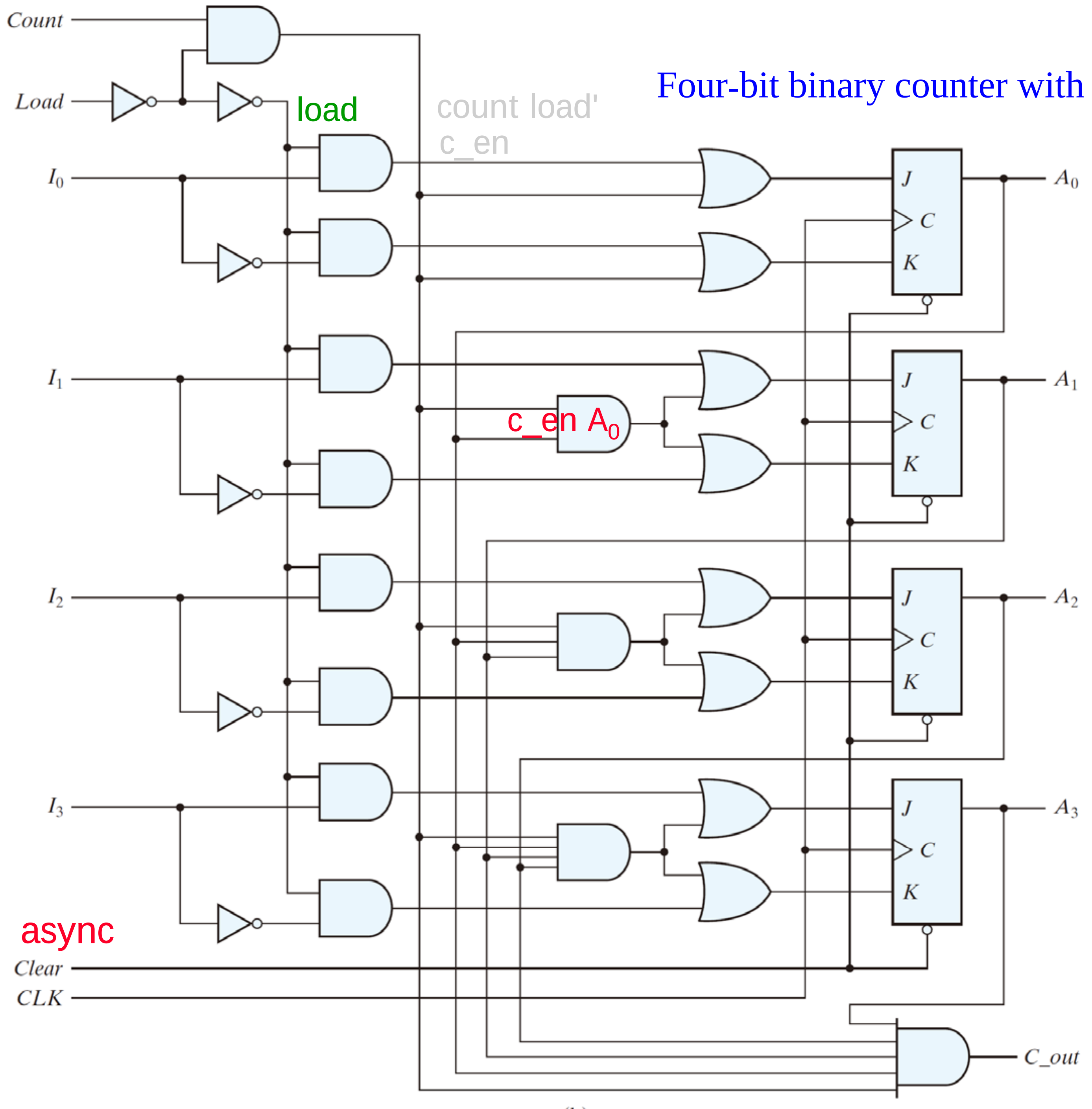
Binary Counter with Parallel Load



Function Table for the Counter of Fig. 6.14

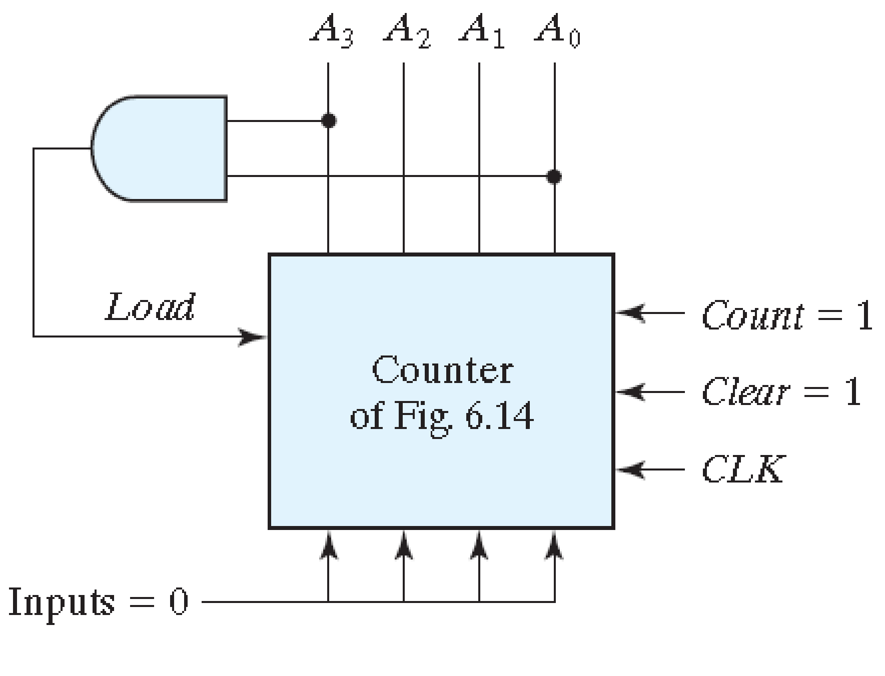
Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

Four-bit binary counter with parallel load

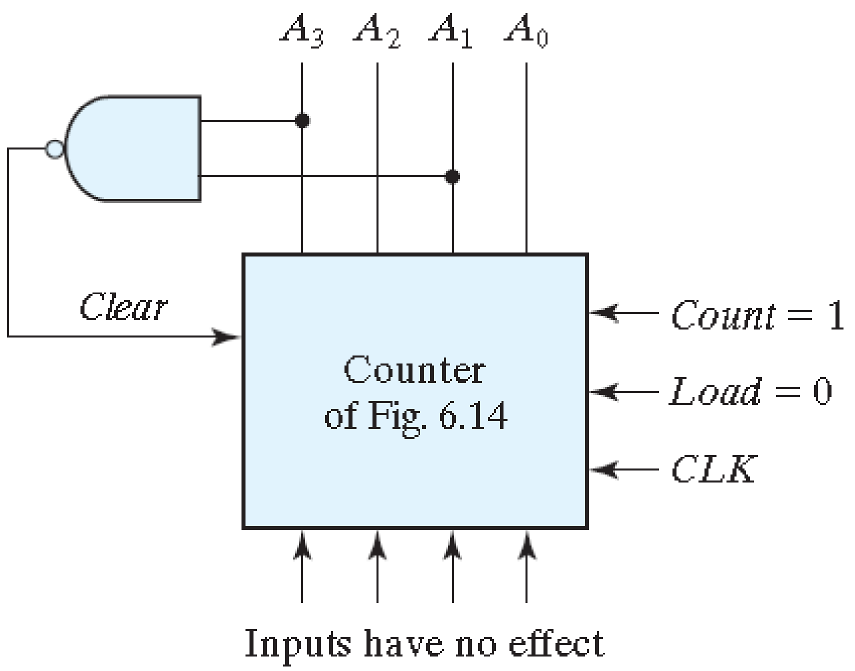


(b)

- A counter with a parallel load can be used to generate any desired count sequence.
- Two ways to achieve a BCD counter using a counter with parallel load.



(a) Using the load input



(b) Using the clear input

6-5 Other Counters

- Counters can be designed to generate any desired sequence of states.
- A divide-by-*N* counter (a.k.a. a modulo-*N* counter) is a counter that goes through a repeated sequence of *N* states; and the sequence may follow the binary count or any other arbitrary sequence.
- A few examples of non-binary counters are presented here.

Counter with Unused States

- *n* FFs has 2^n binary states, while state table may have unused state.
- The unused states may be treated as don't-care or assigned specific next states.
- Once the circuit is designed and constructed, outside interference may cause the circuit to enter one of the unused states.
- Ensure that the circuit eventually goes into one of the valid states so that it can resume normal operation; and avoid the sequential circuit circulates among unused states.
- If the unused states are treated as don't-care conditions, then once the circuit is designed, it must be investigated to determine the effect of the unused states.
- The next state from an unused state can be determined from the circuit designed.

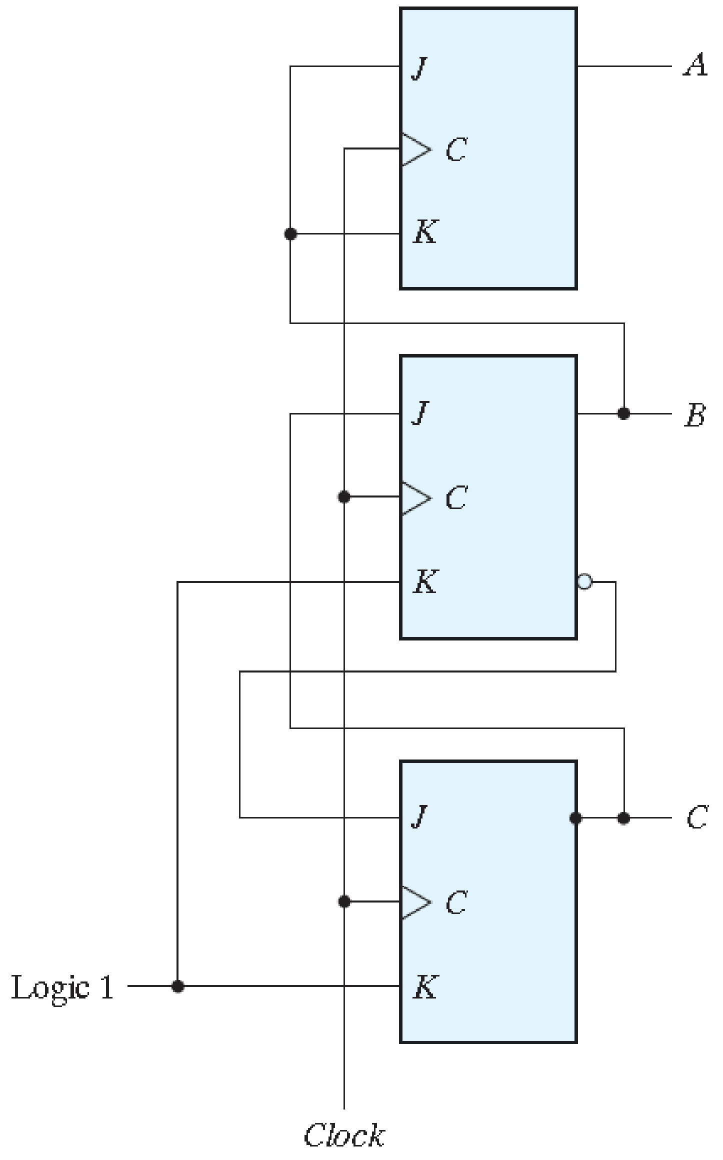
State Table for Counter

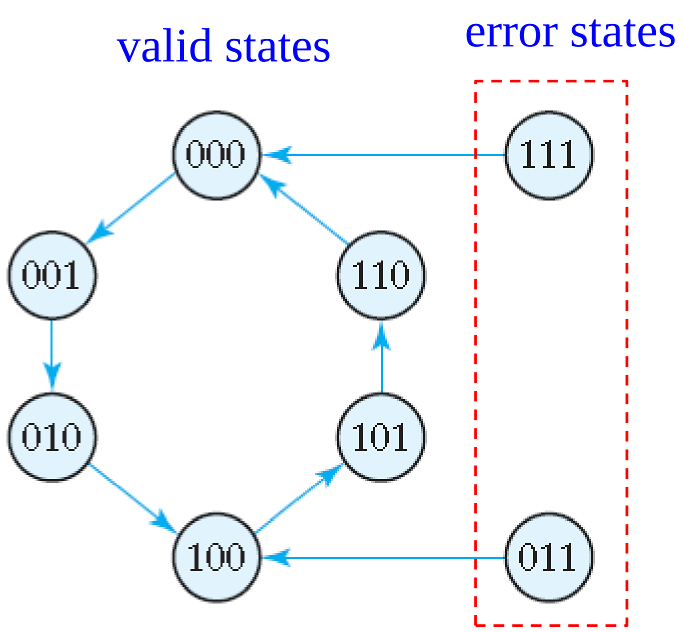
Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

- Six repeated sequence of states, with FFs *B* and *C* repeating the binary count 00, 01, 10, and FF *A* alternating between 0 and 1 every three counts.
- The sequence of the counter is not straight binary, and two states, 011 and 111, are not included in the count.
- The choice of *JK* FFs results in the FF input conditions listed in the table. Inputs K_B and K_C have only 1's and X's in their columns, so these inputs are always equal to 1.
- The other FF input equations can be simplified by using minterms 3 and 7 as don't-care conditions. The simplified equations are

$$J_A = B \qquad K_A = B, \quad J_B = C \qquad K_B = 1, \quad J_C = B' \quad K_C = 1$$

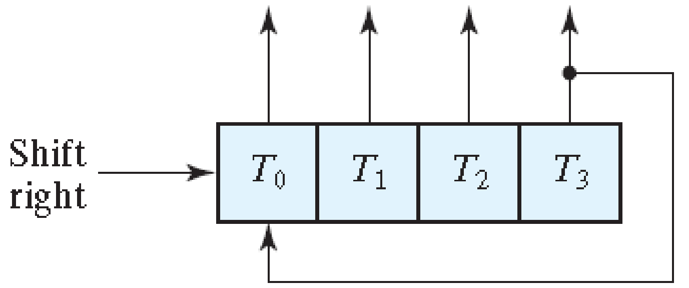
Self-correcting Counter





Ring Counter

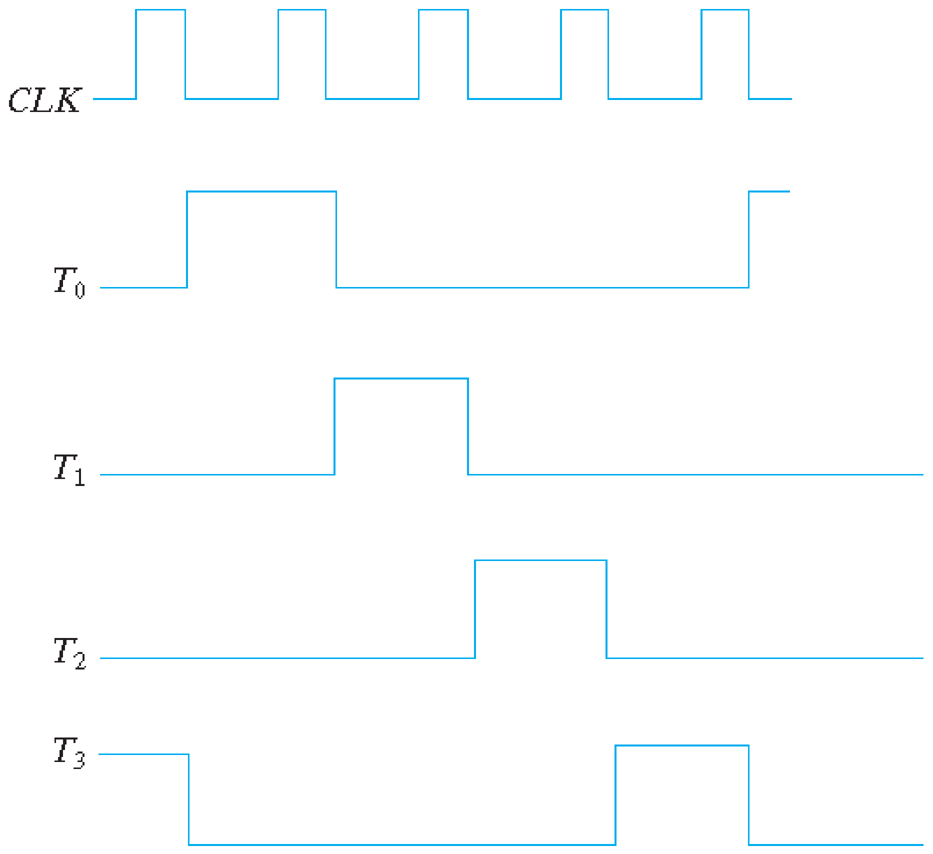
- R*ing counter*: a circular shift register with only one FF being set at any particular time; all others are cleared.
- The single bit is shifted from one FF to the next.
- Example:



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

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

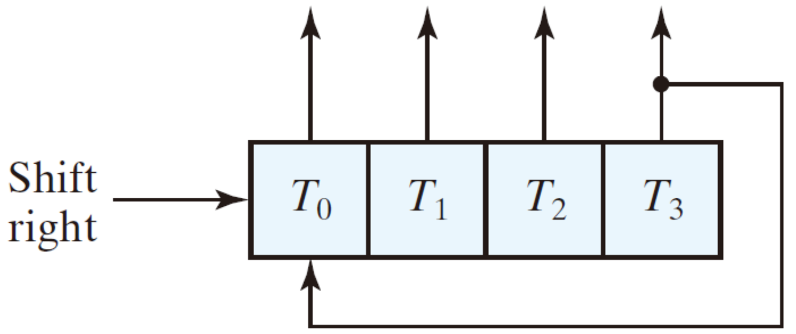
… … …



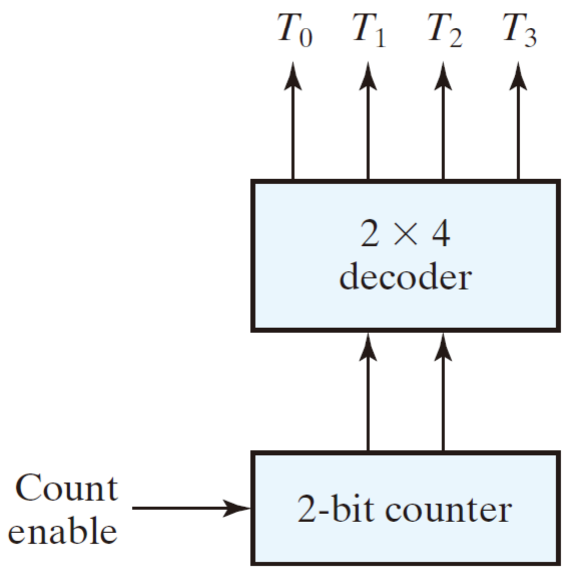
(b) Sequence of four timing signals

Generation of Timing Signals

- Application of counters
  - Counters may be used to generate timing signals to control the sequence of operations in a digital system.
- Approaches for generation of 2^n timing signals
  1. a shift register with 2^n FFs
  2. an n-bit binary counter together with an n-to-2^n-line decoder



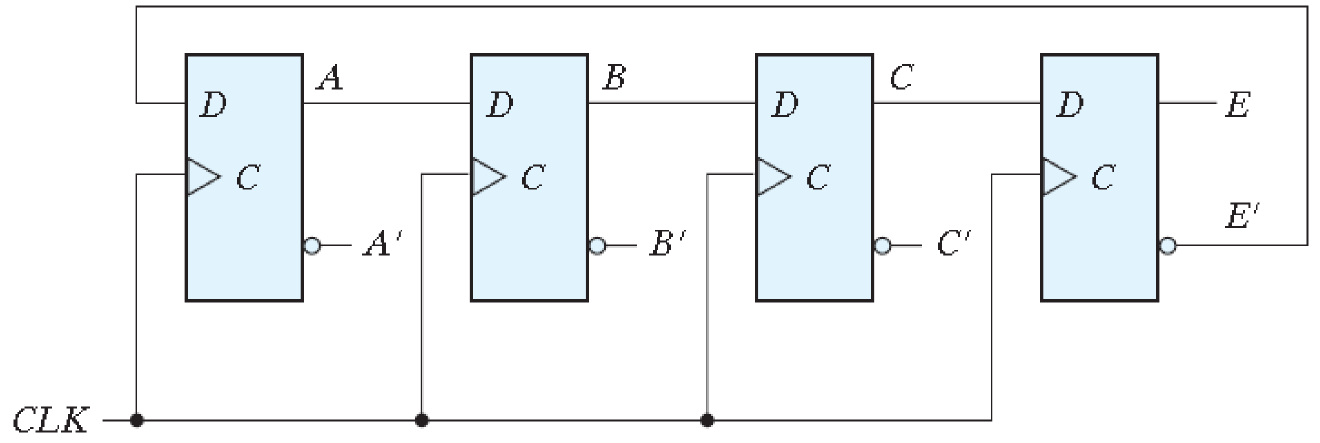
(a) Ring-counter (initial value = 1000)



(c) Counter and decoder

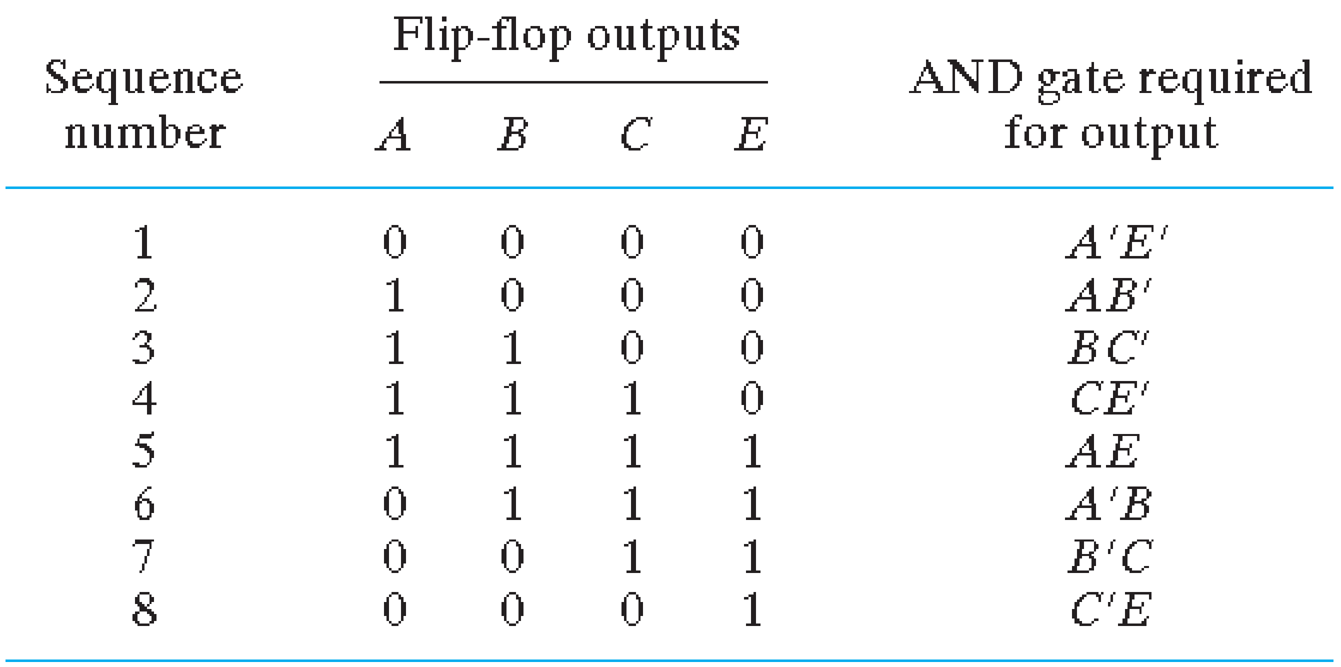
Johnson Counter

- A *k*-bit ring counter circulates a single bit among the FFs to provide *k* distinguishable states.
- The number of states is doubled for a *switch-tail* ring counter.
- A switch-tail ring counter is a circular shift register with the complemented output of the last FF connected to the input of the first FF.
- In general, a *k*-bit switch-tail ring counter will go through a sequence of 2*k* states.
- Starting from all 0's, each shift operation inserts 1's from the left until the register is filled with all 1's.
- In the next sequences, 0's are inserted from the left until the register is again filled with all 0's.



Construction of a Johnson Counter

- 2*k* decoding gates are needed to provide outputs for 2 *k* timing signals.
- The decoding gates are the eight AND gates listed in the table.
- Since each gate is enabled during one particular state sequence, the outputs of the gates generate eight timing signals in succession.



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	$C'E$