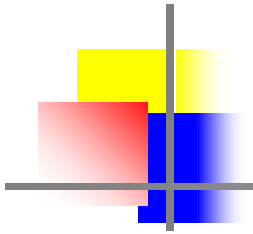
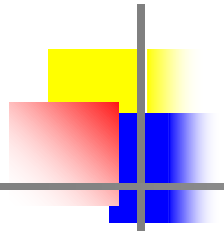
Ch. 06. Registers and Counters

2019. 5. 13.

K-S. Sohn



Contents

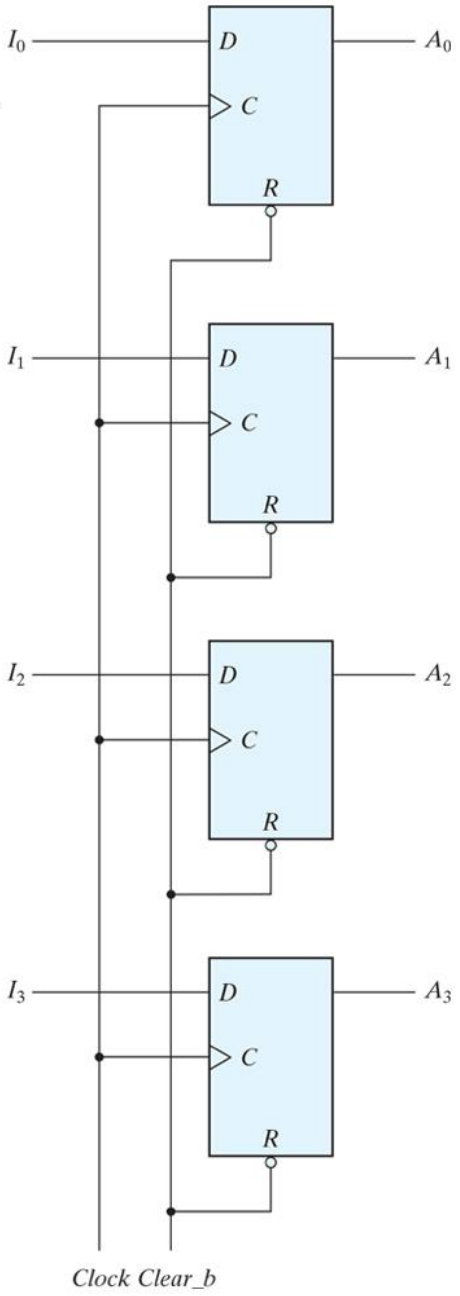


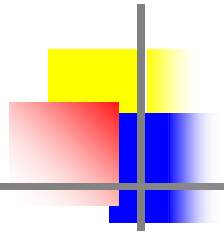
Registers and Counters

- Register
  - A clocked sequential logic
  - n-bit register ⇒ n flip-flops
- Counter
  - A register with determined sequence of states
  - Ripple counter
    - Cascaded clock signal ⇒ ripple effect
  - Synchronous counter
    - Common clock signal

A Basic Register

- 4-bit register with 4 D flip-flops
  - Signals
    - R : reset with clear signal
    - C : clock signal input
  - Data input(4bit binary) : I_0~I_4
  - Data output(4bit binary) : A_0~A_4
  - State transition
    - Each bit of data output is changed at the clock's positive edge whenever the corresponding data input changes.
    - How to parallelism and maintain stability

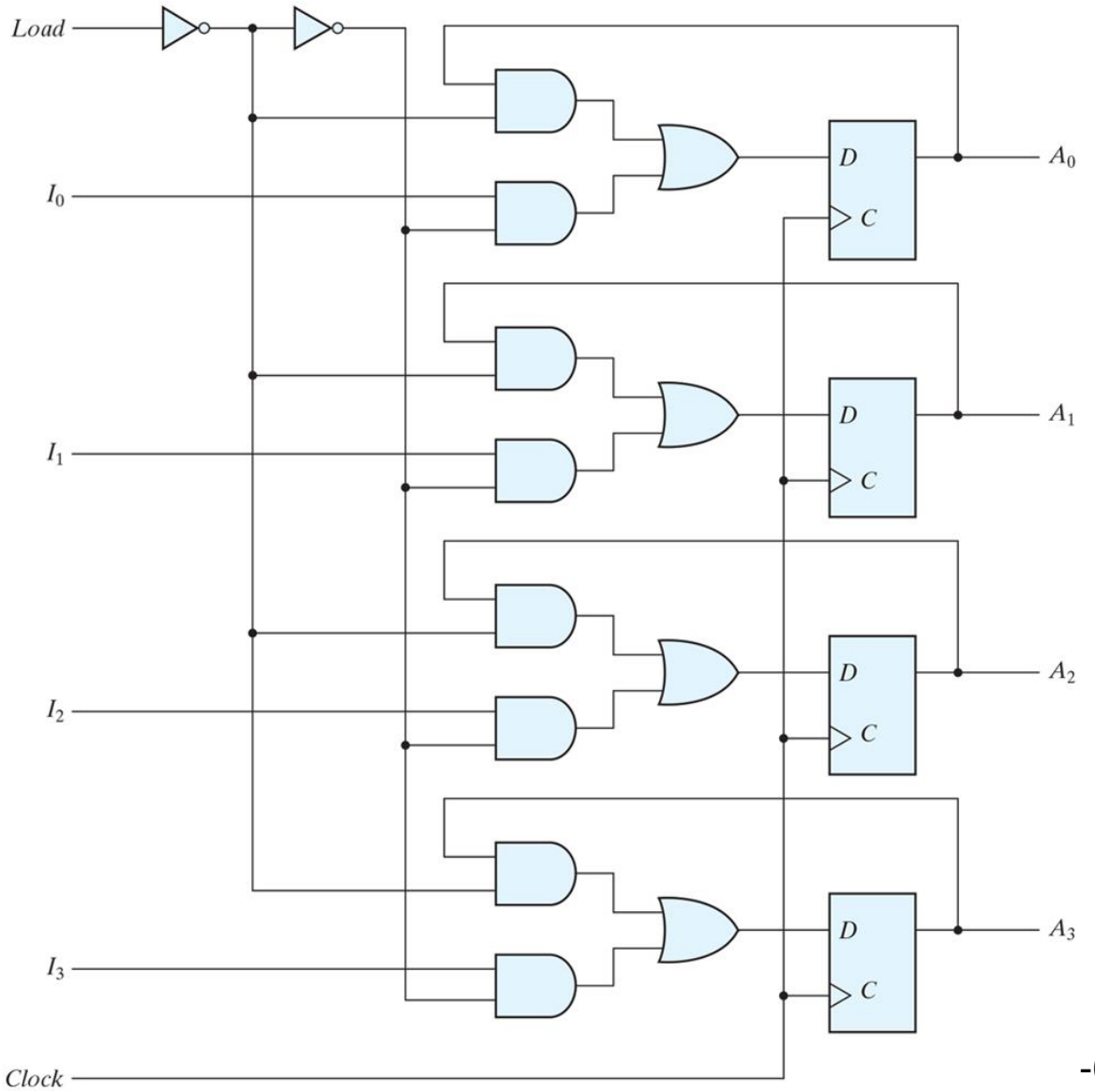




Parallel Data Loading

- Objectives
  - Preventing the prematured input data from putting into the register
- Definition
  - All input data bits are loaded to the register in a clock time when the load is needed.
- Control methods for parallel loading
  - Data input control
    - Violation of parallelism due to the delay on the paths of input data
  - Clock control
    - Violation of synchronism due to the delay on the paths of clock signal

Four-bit Register with Parallel Load



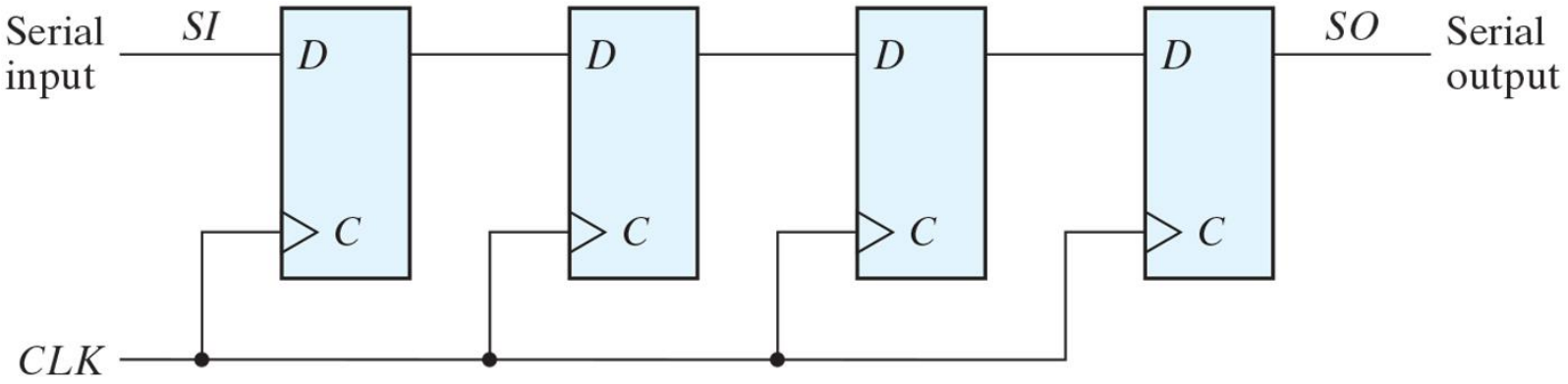
Four-bit Register with Parallel Load

- 4-bit register with parallel load
  - Load signal
  - Input equations
    - $D_i = L_i'A_i + L_iI_i$
- Derivation of input equations

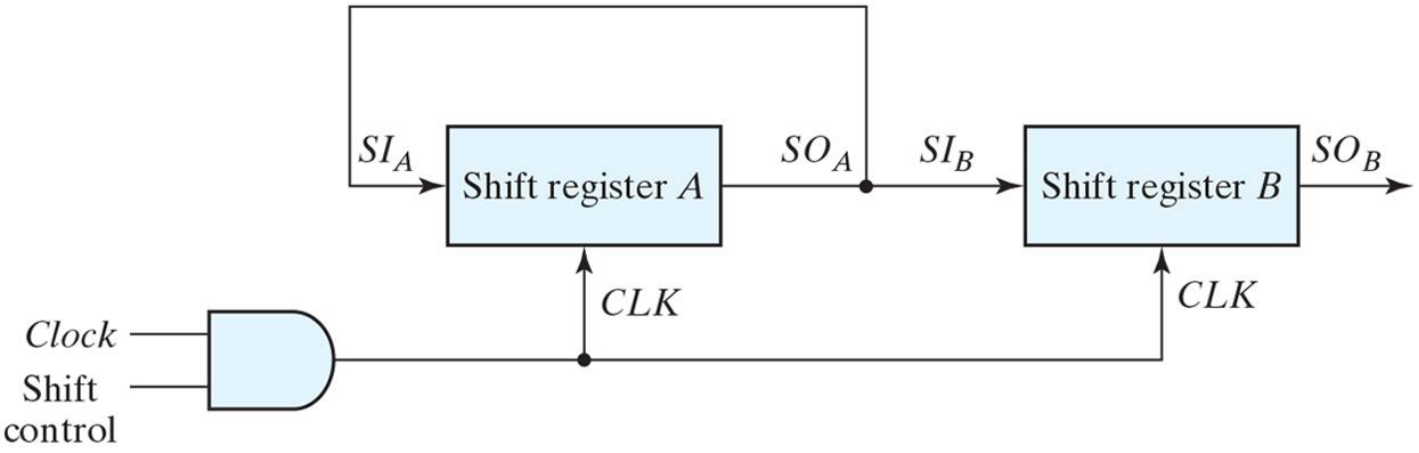
$A_i(t)$	I_i	L_i	$A_i(t+1)$	D
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

Shift Registers

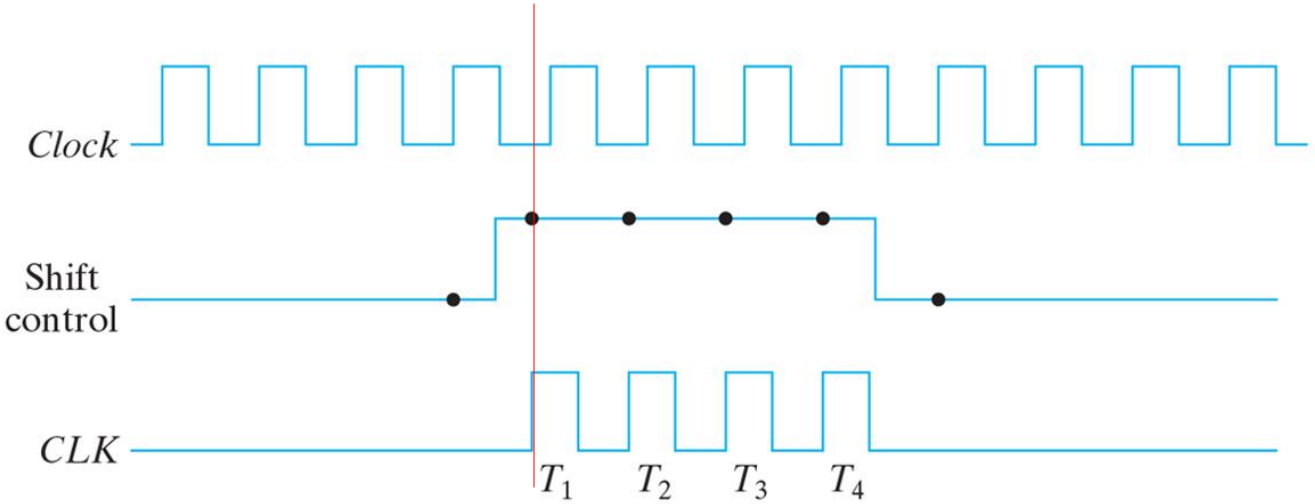
- A simple 4-bit shift register
  - Serial input and serial output (SISO)



Serial Transfer



(a) Block diagram



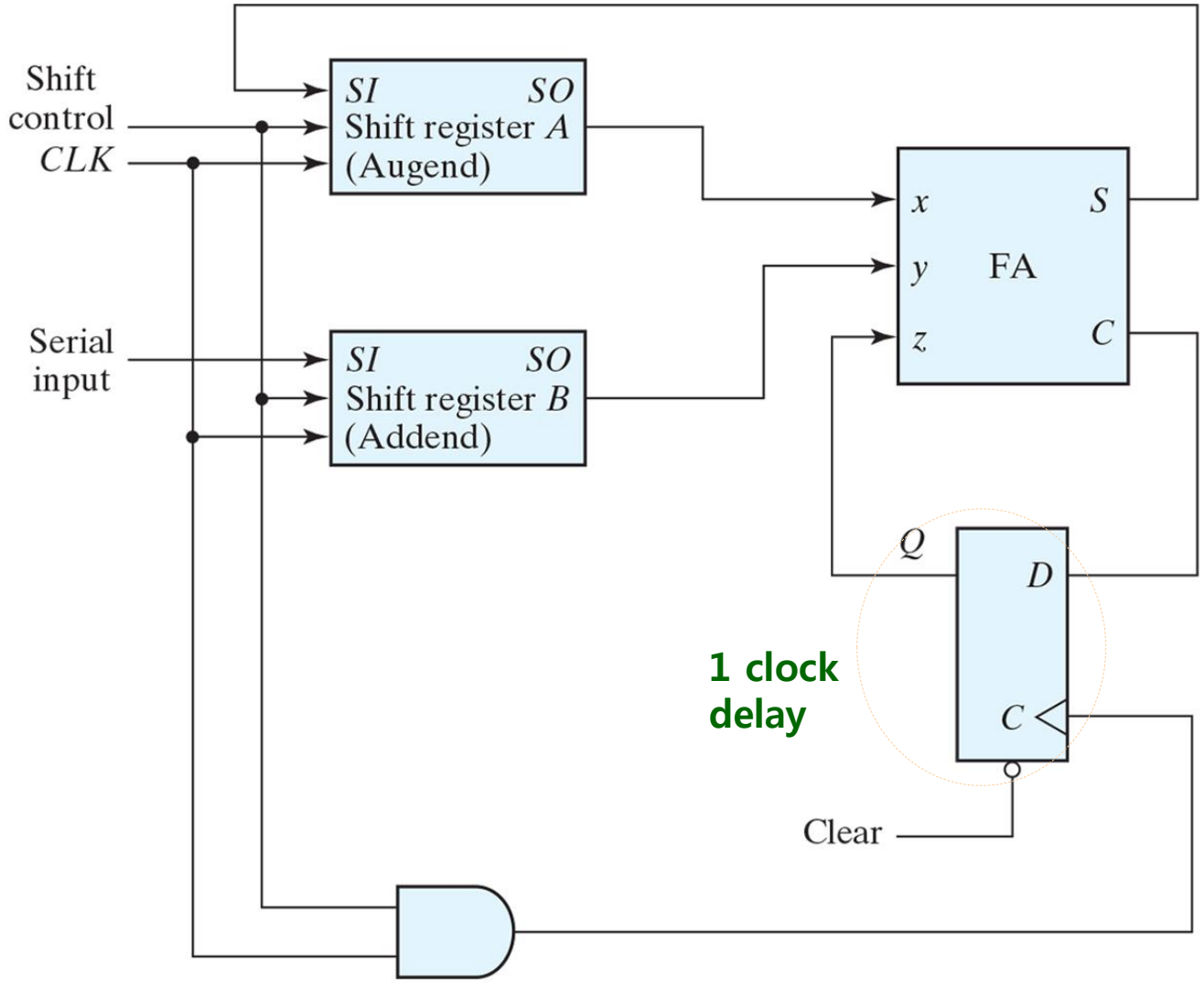
(b) Timing diagram

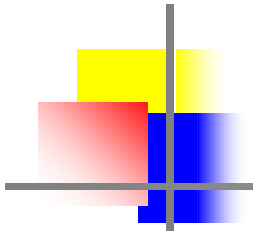
Serial Transfer

- Serial transfer between registers
  - Source register : shift register A
  - Destination register : shift register B
- Option: preventing reg-A from losing its content
  - loopback
- State transition table

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 Addition





Serial Addition

- 4bit binary number addition
- 1 FA instead 4 FAs but slower
- augend and addend are stored in register A and B, respectively
- result(sum) is stored in register A
- When control=1, bit by bit addition is executed serially.
- The number of clock ticks when control = 1 should exactly equal to the number of bits of data.
- The length of register A and B should be same.

Serial Adder

- State table

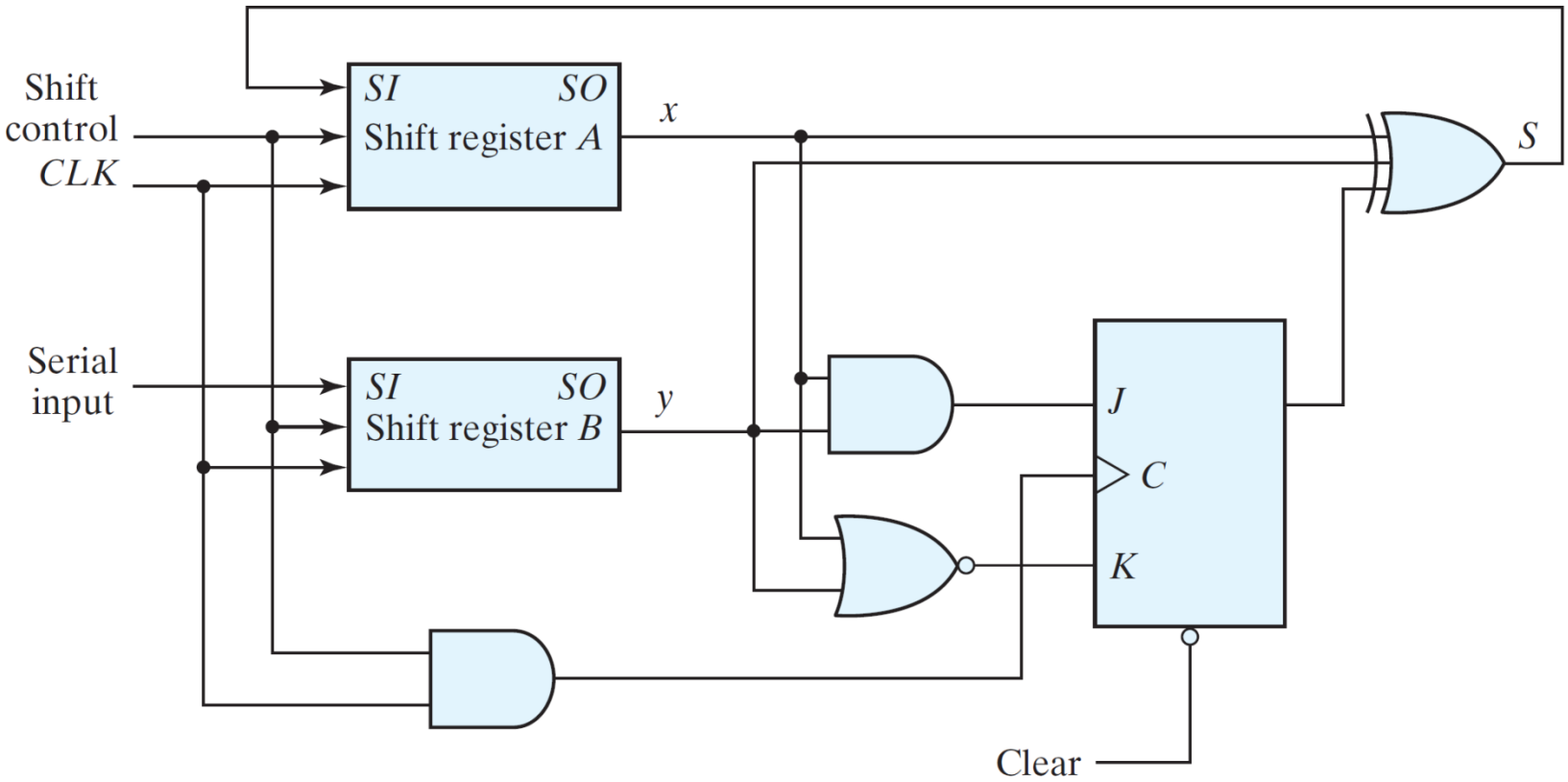
Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	x	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

$$J_Q = xy$$

$$K_Q = x'y' = (x+y)'$$

$$S = x \oplus y \oplus Q$$

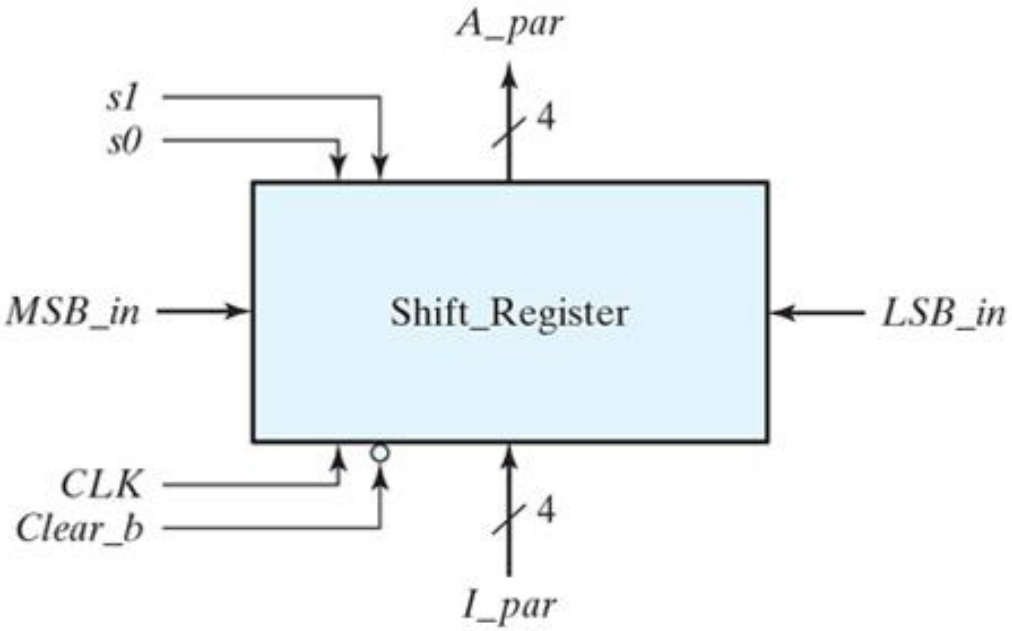
Serial Adder with JK F-F



Universal Shift Register

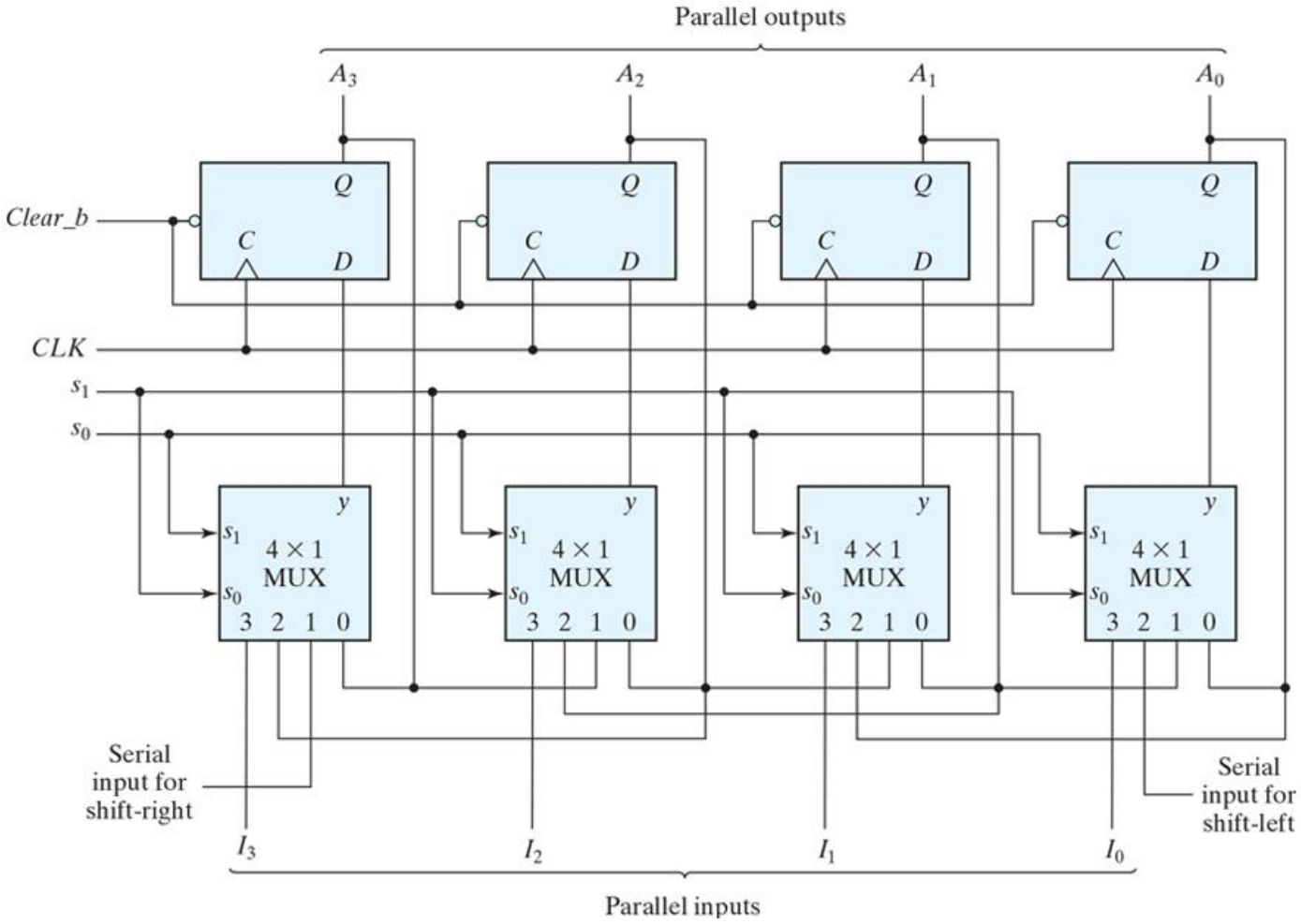
- Capabilities (or requirements)
  - Extracting intermediate flip-flop outputs
  - Parallel load
  - Bidirectional(left / right) shift operation
  - Preservation of register contents for some clock ticks
  - Clear

S_0S_1	control
00	Preserving
01	Shift right
10	Shift left
11	Parallel load



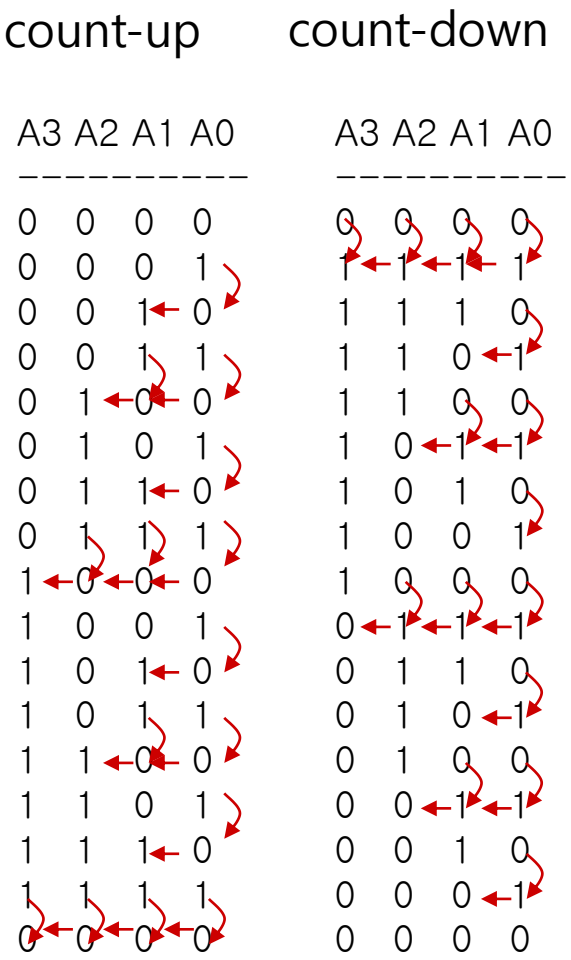
Four-bit Universal Shift Register

S_0S_1	control
00	Preserving
01	Shift right
10	Shift left
11	Parallel load



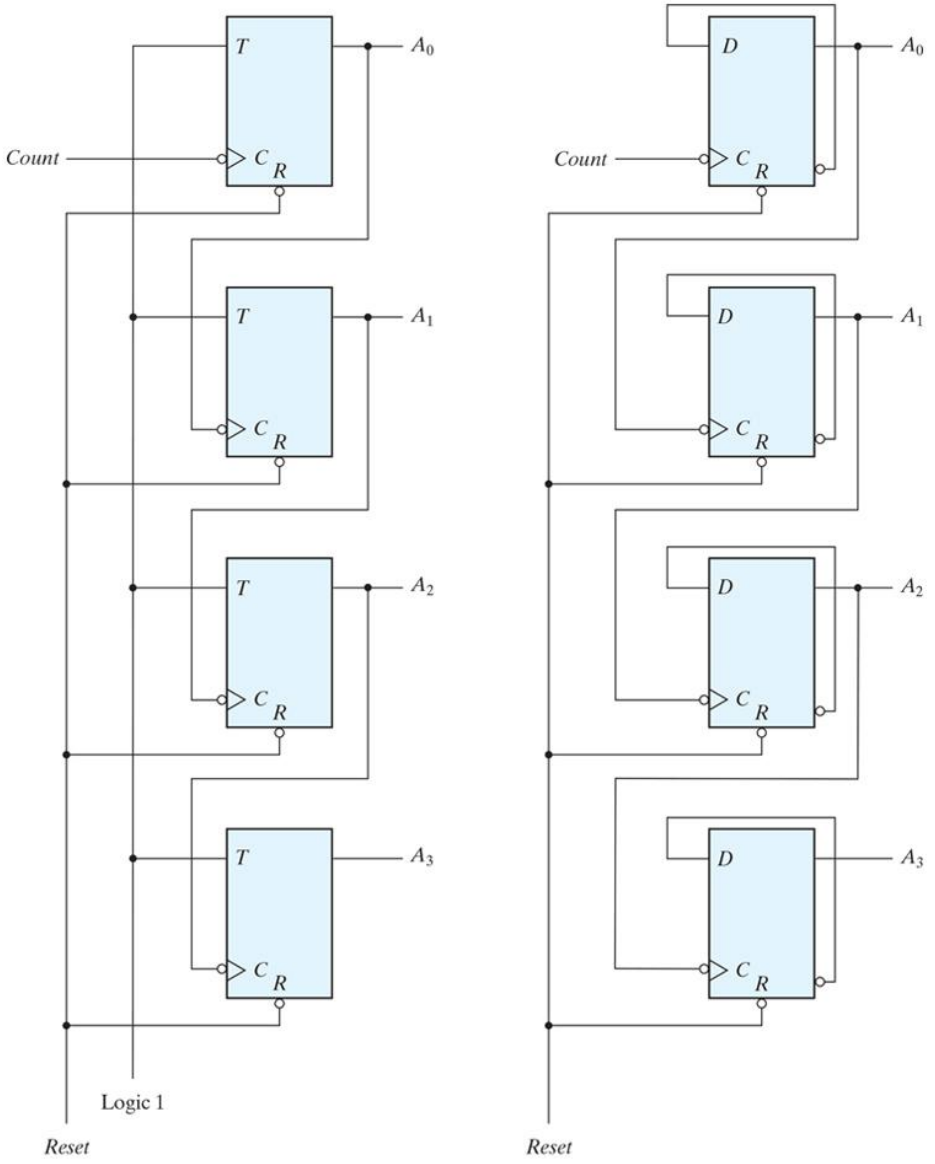
Binary Ripple Counter

- Binary count-up ripple counter
  - A serise of complementing flip-flops
  - Negative-edge clock triggering
- Binary count-down ripple counter
  - A serise of complementing flip-flops
  - Positive-edge clock triggering



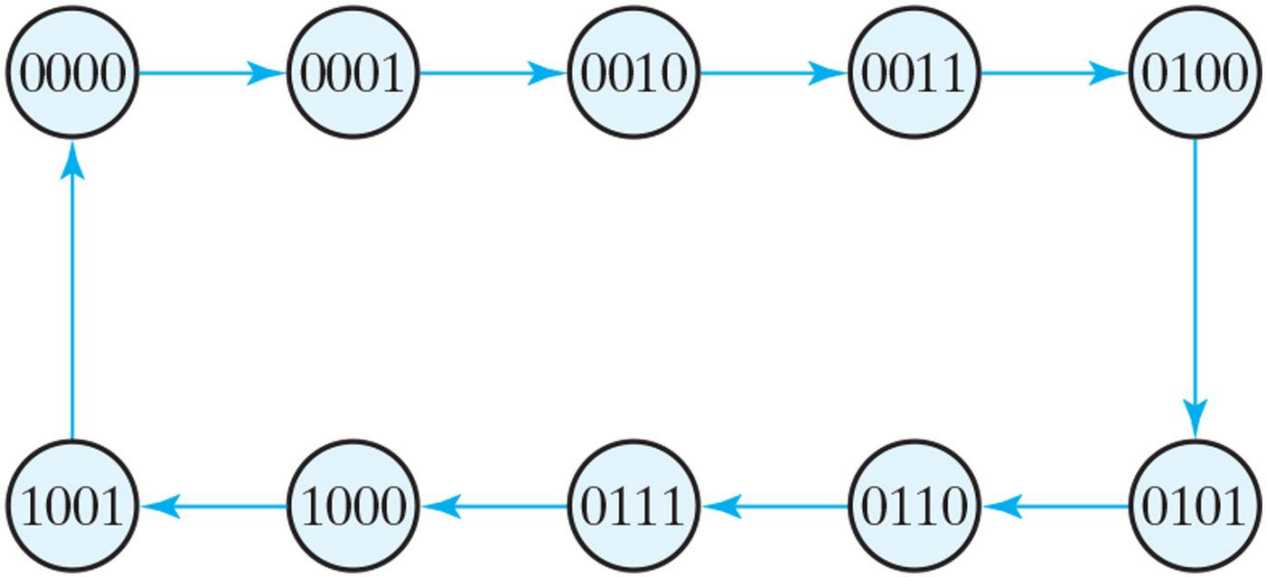
Four-bit Binary Ripple Counter

- Implementation with T F-F and D F-F



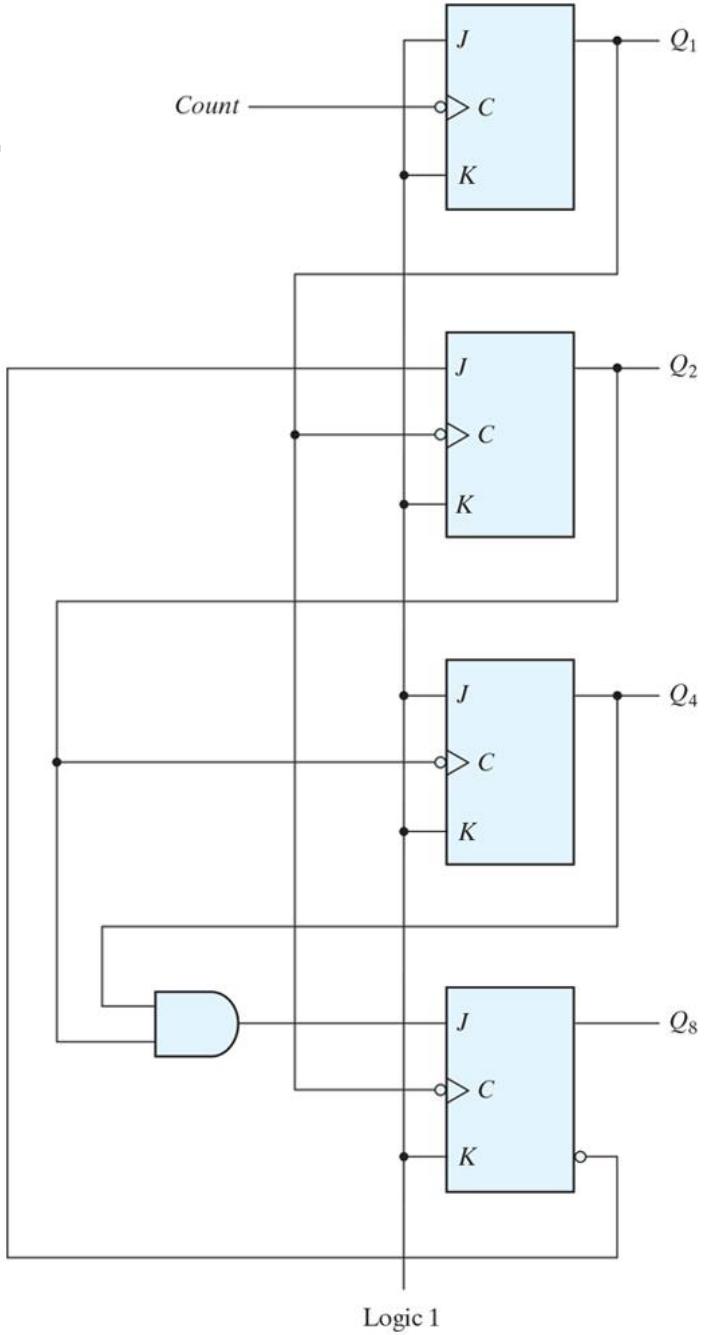
BCD Ripple Counter

- State transition diagram



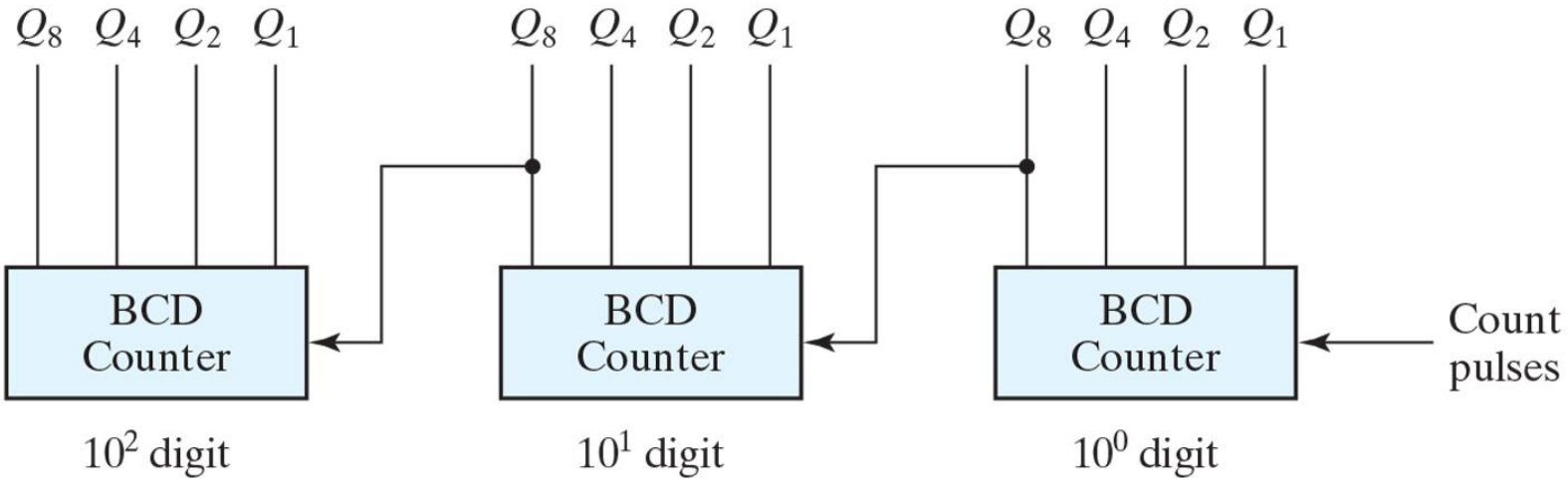
BCD Ripple Counter

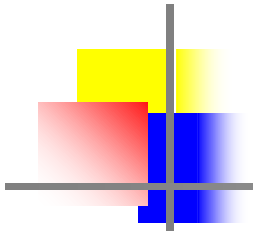
- Block diagram



Three-digit BCD Counter

- Block diagram





Synchronous Counter

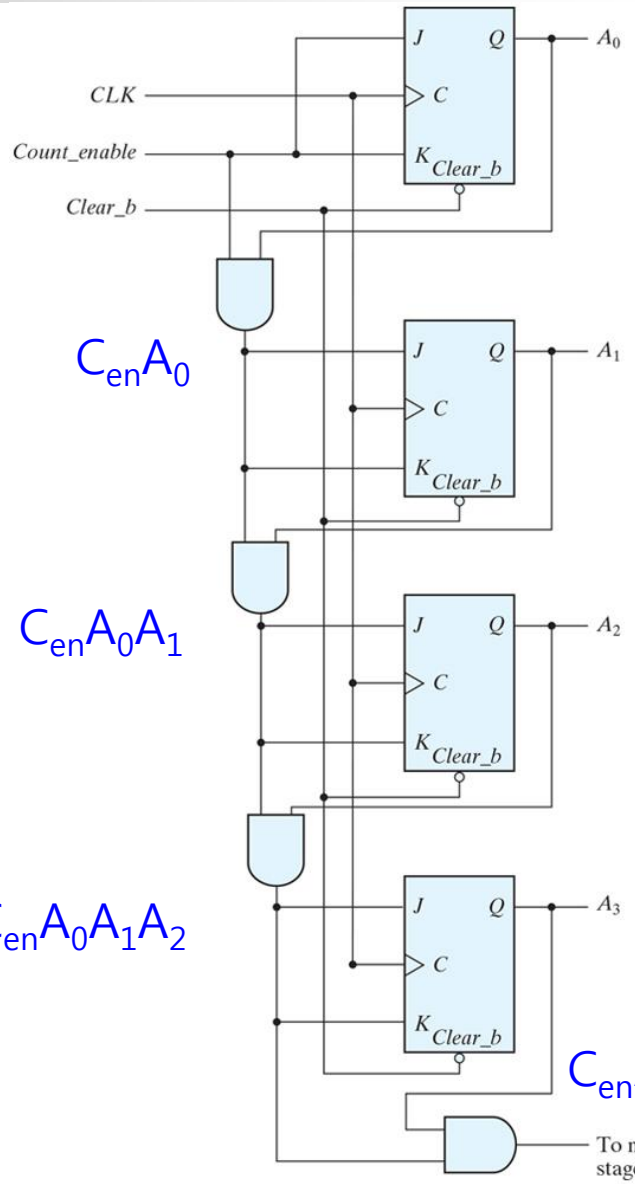
- Synchronism
  - A common clock triggers all flip-flops simultaneously in synchronous counters
  - c.f. : One flip-flop is triggered at a time in succession in a ripple counter
- Design of synchronous counters
  - Determine the number of bits and number system (binary, BCD, gray, …)
  - State diagram
  - State table
  - Flip-flop input equations
  - Logic diagram of the counter

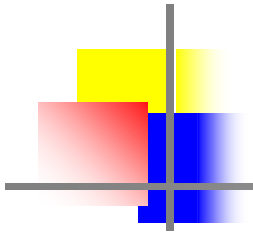
Synchronous Binary Counter

- Main idea for the design of sync. binary counter
  - The flip-flop in the least significant position is complemented with every counter pulse.
  - A flip-flop in any other position is complemented when all the bits in the lower significant positions are equal to 1.
- Implementation with JK flip-flops
  - The first state A_0 has its J and K equal to 1 if the counter is enabled
  - The other J and K inputs are equal to 1 if all previous least significant state are equal to 1 and the count is enabled.

Synchronous Binary Counter

- Four-bit block diagram





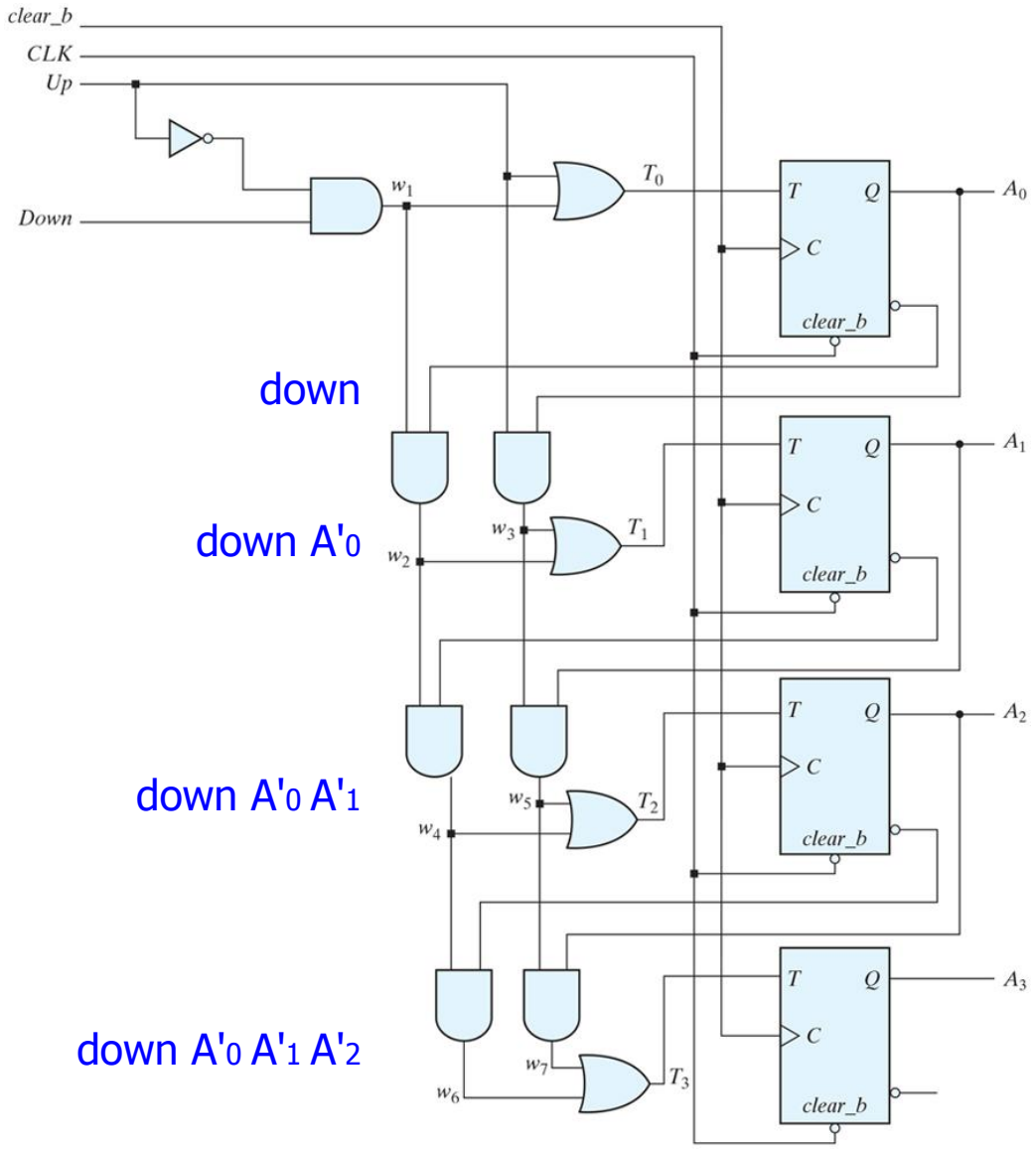
Sync. Count Down Binary Counter

- Main idea
  - The bit in the least significant position is complemented with each down counter pulse.
  - A bit in any other position is complemented if all lower significant bits are equal to 0.

- Implementation with JK flip-flops
  - Complement output of a flip-flop is provided to J and K inputs of the next flip-flop through AND gate.

Sync. Binary Counter

- Four-bit up/down binary counter block diagram



Sync. Binary Counter

- State table

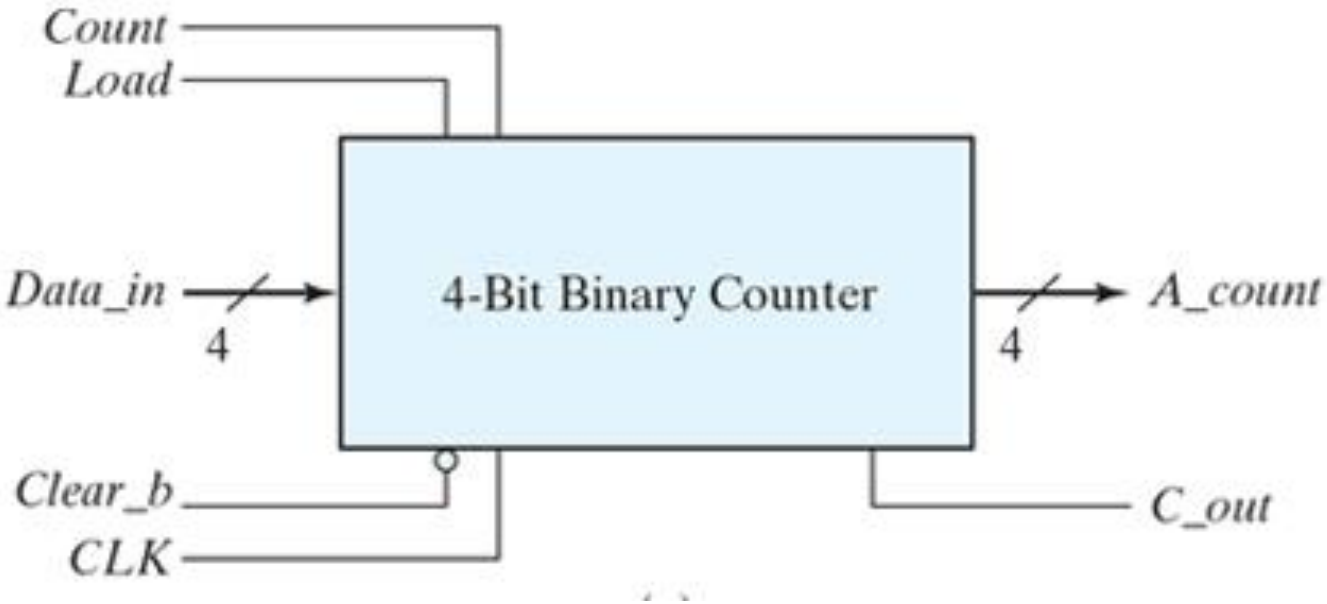
Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	T_{Q8}	T_Q4	T_{Q2}	T_{Q1}
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	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

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

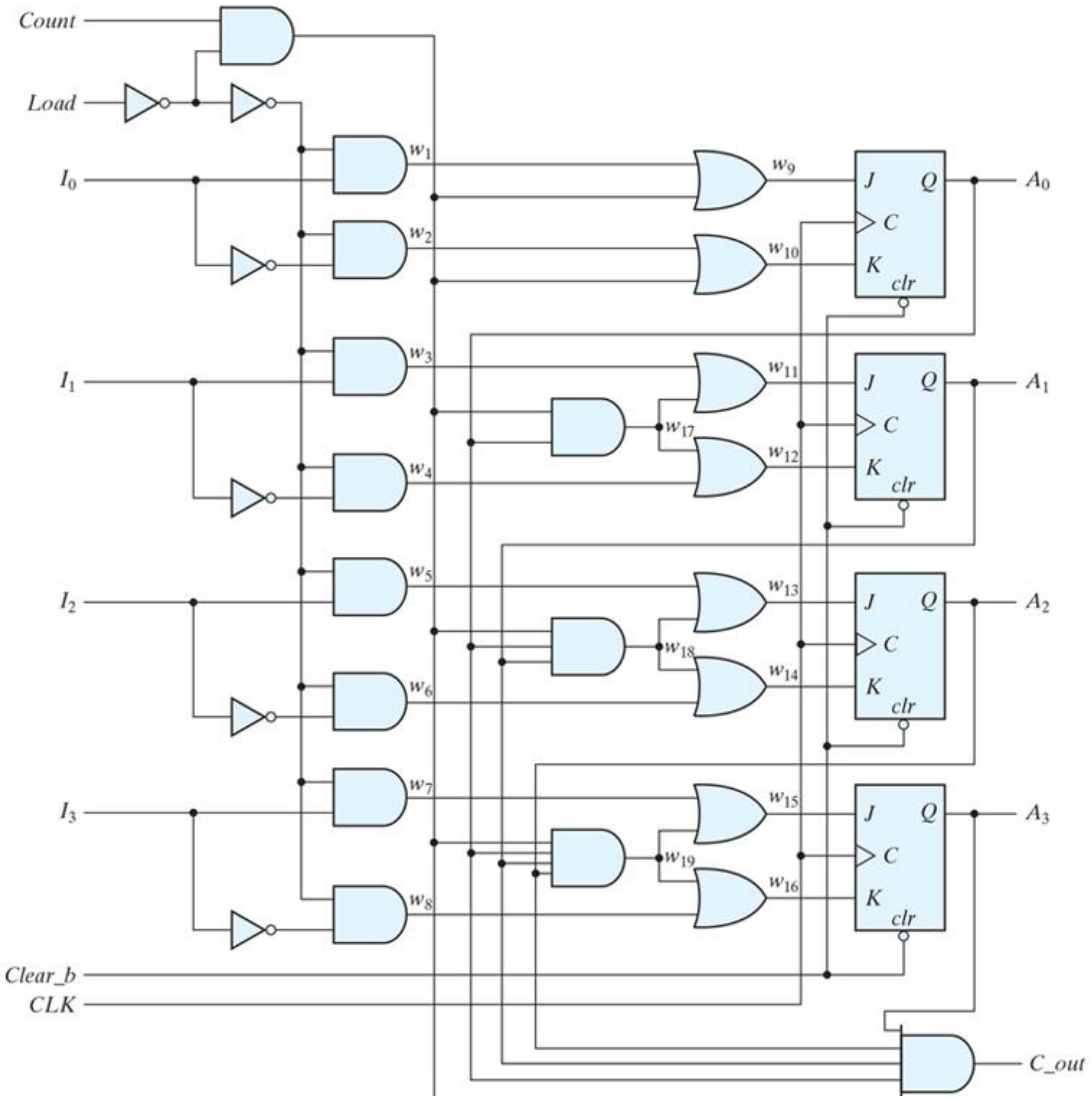
Sync. Binary Counter with Parallel Load

- Conceptual diagram



Sync. Binary Counter with Parallel Load

- Block diagram



Sync. Binary Counter with Parallel Load

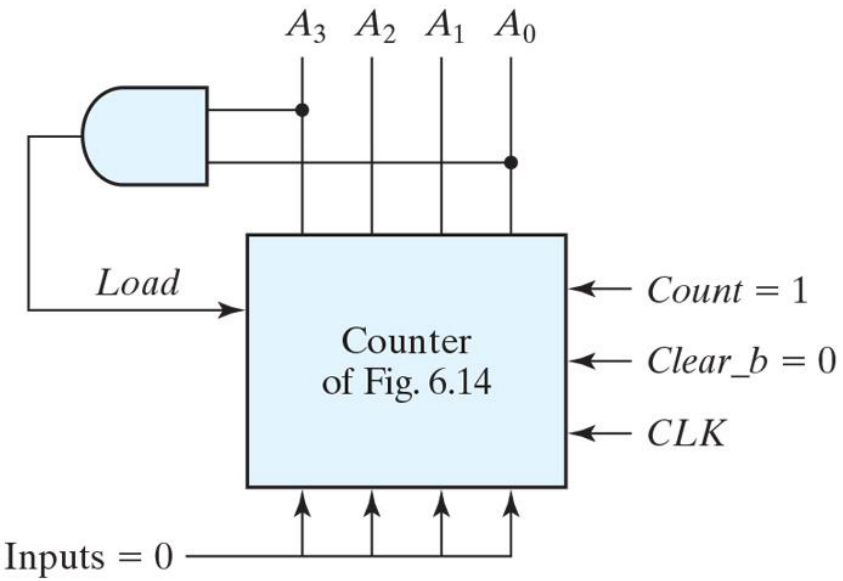
- Function table

Clear_b	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

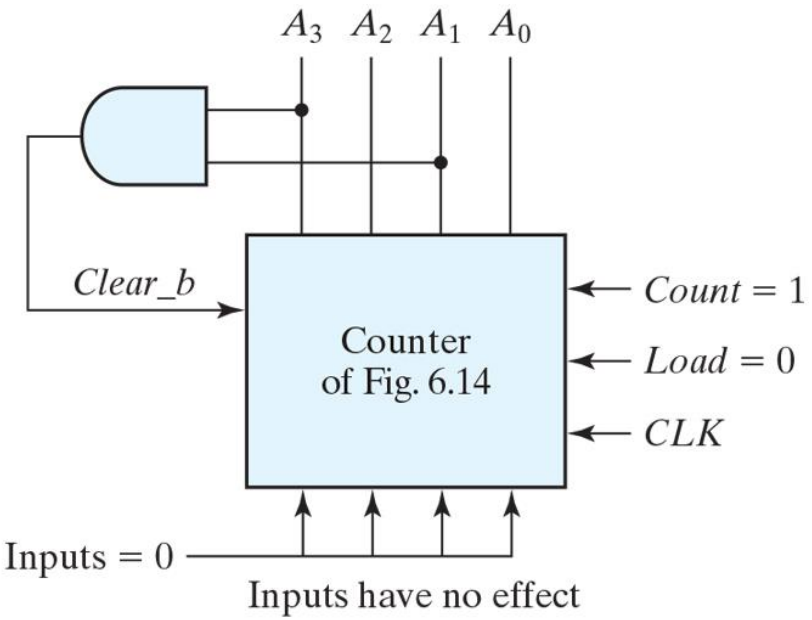
  - count input is inhibited when the load input is enabled.
  - load input priority > count input priority

Alternate Implement of Sync. BCD Cnter

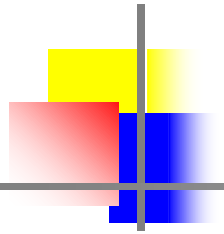
- Using a counter with parallel load



(a) Using the load input



(b) Using the clear input



Self-Correcting Counter

- Definition
  - If the counter happens to be in one of the **unused *states***, eventually reaches the ***normal count sequence*** after one or more clock pulses

- Unsed states
  - Don't care condition for simplifying the input equations
  - Redundant states for self-correcting

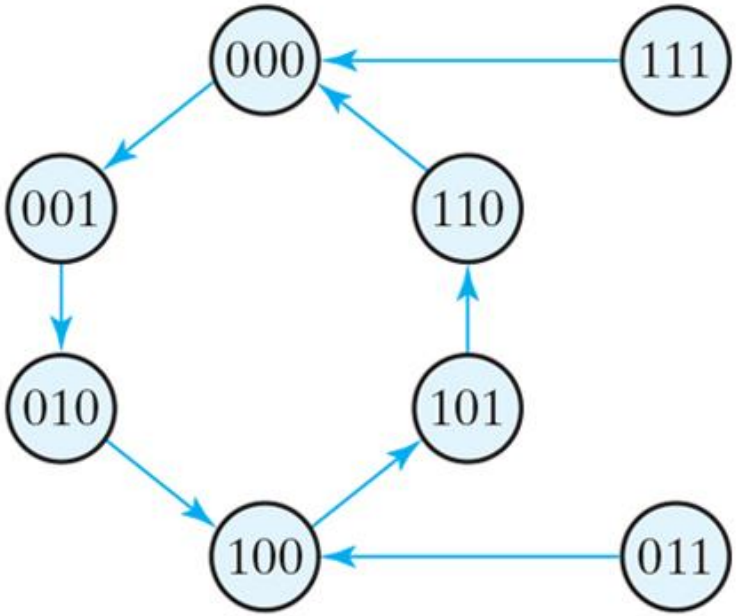
Self-correcting Counter

- State table

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

Self-correcting counter

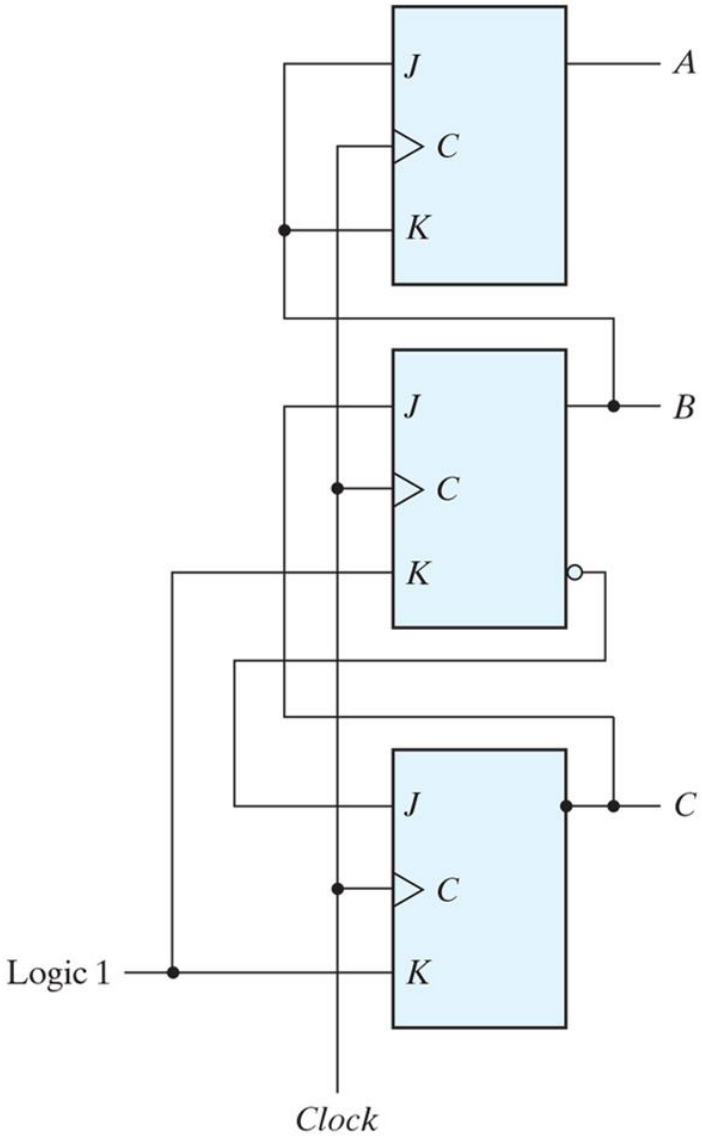
- State diagram



Self-correcting Counter

- Block diagram

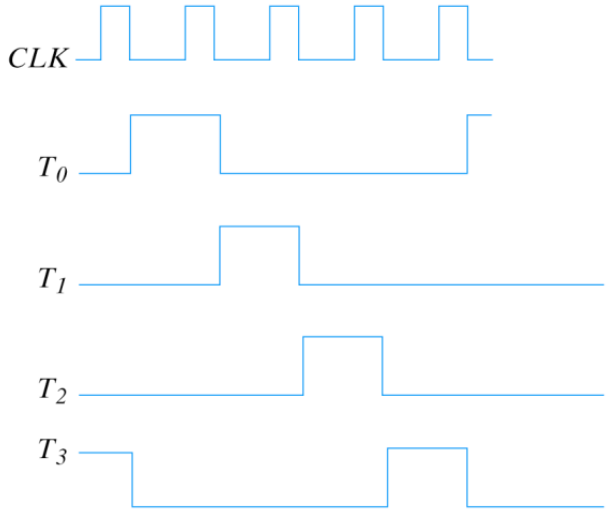
$J_A=B \quad K_A=B$
$J_B=C \quad K_B=1$
$J_C=B' \quad K_C=1$



(a) Logic circuit diagram

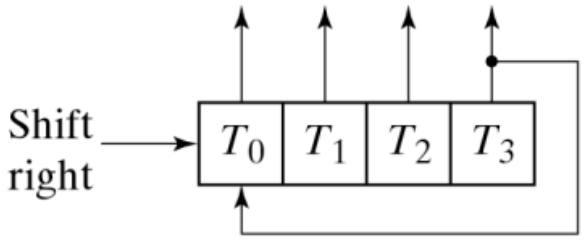
Ring Counter

- Objectives : generating multichanneled timing signal

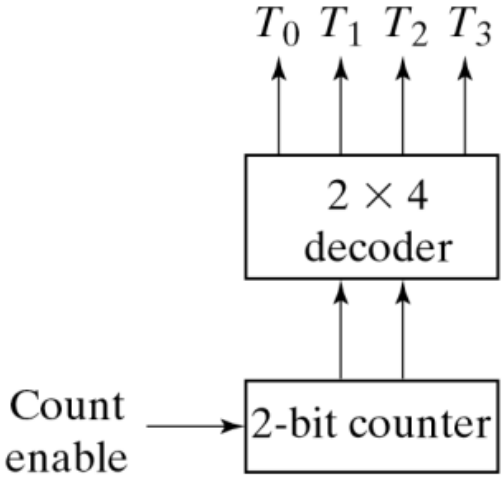


☞ Circulation of a single bit among the flip-flops to provide several distinguishable state

- Alternative implementations



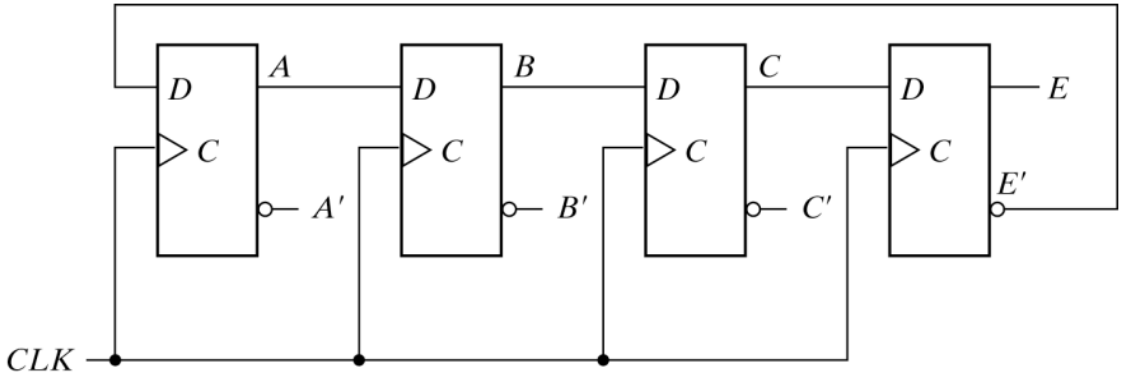
(a) Ring-counter (initial value = 1000)



(b) Counter and decoder

Johnson Counter

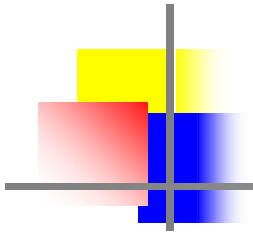
- Objective : Doubling the number of states of ring counter



☞ Core of the 8-states Johnson counter except decoding circuits

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



Discussion~~~