แก้ไขครั้งล่าสุดเมื่อวันที่ 19 กันยายน 2567

Chapter 6

Registers and Counters

จะโหลดอินพุต
ไปเก็บใน register
ทุกครั้งที่ได้ clock

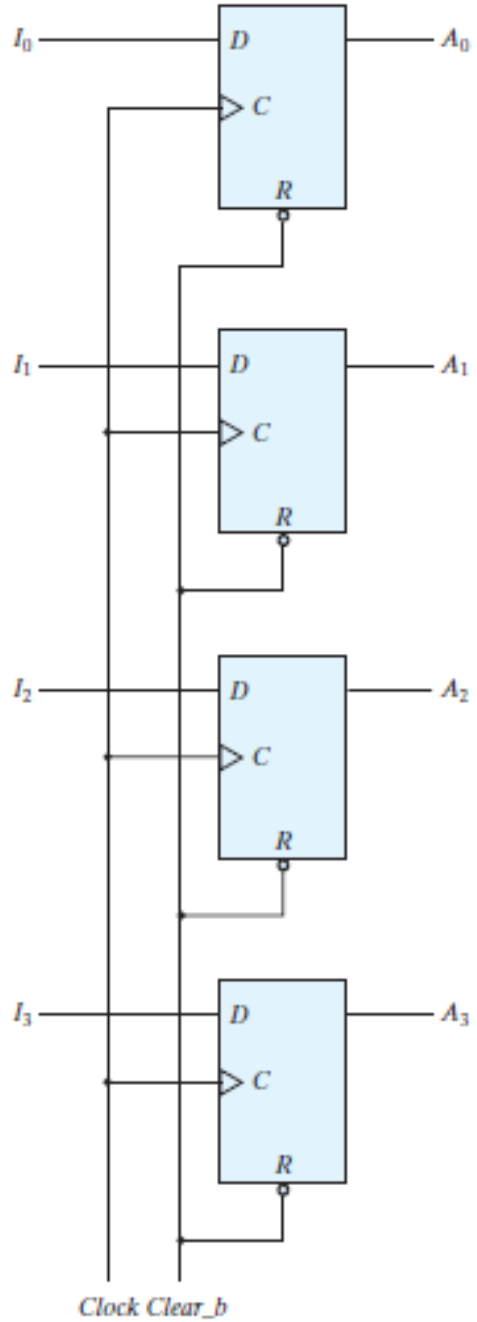


FIGURE 6.1
Four-bit register

เพื่อไม่ให้โหลดทุก clock
จึงเพิ่มขา load มาควบคุม

load = 0 ไม่โหลดอินพุต

load = 1 โหลดอินพุต

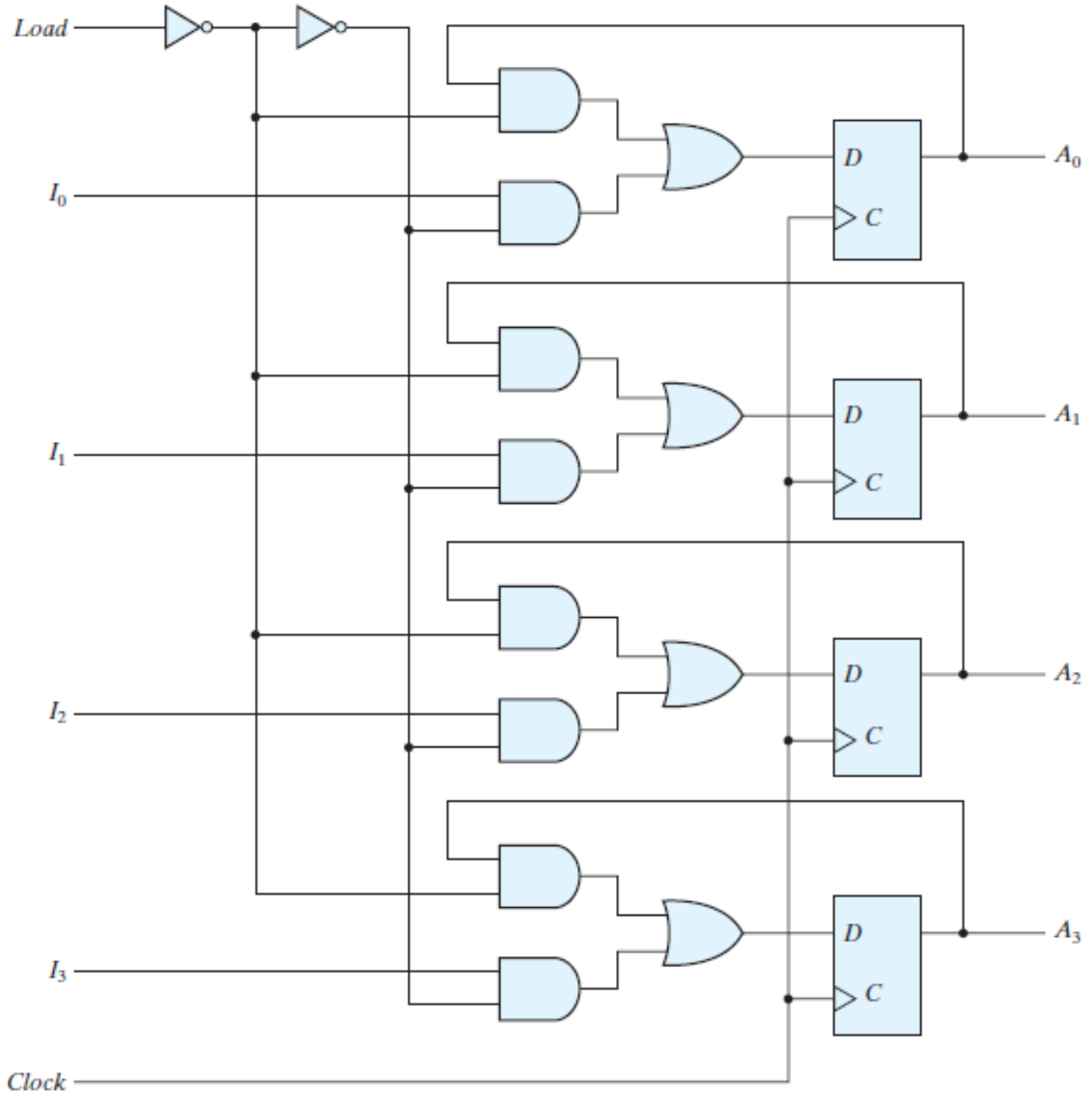


FIGURE 6.2
Four-bit register with parallel load

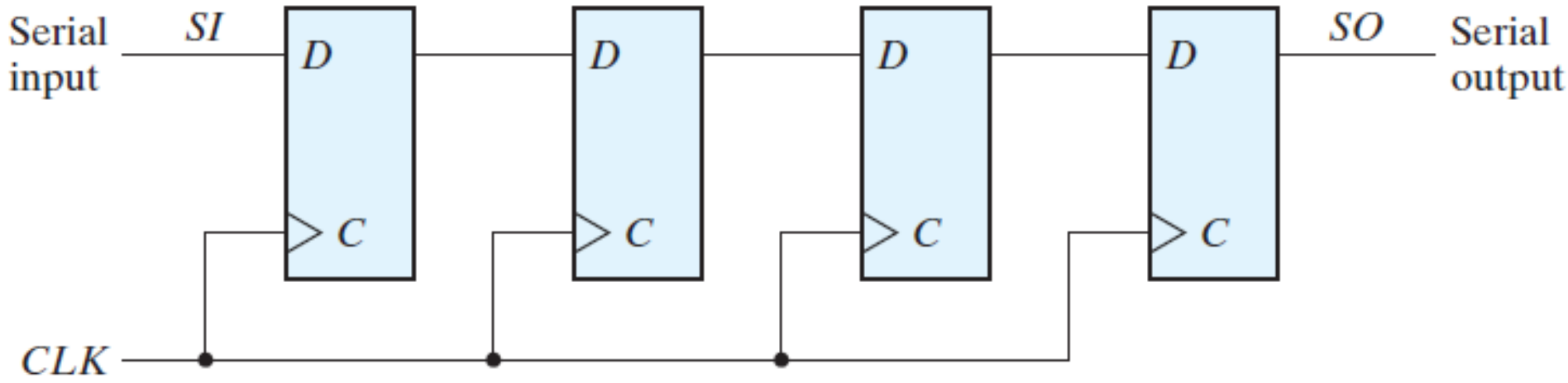
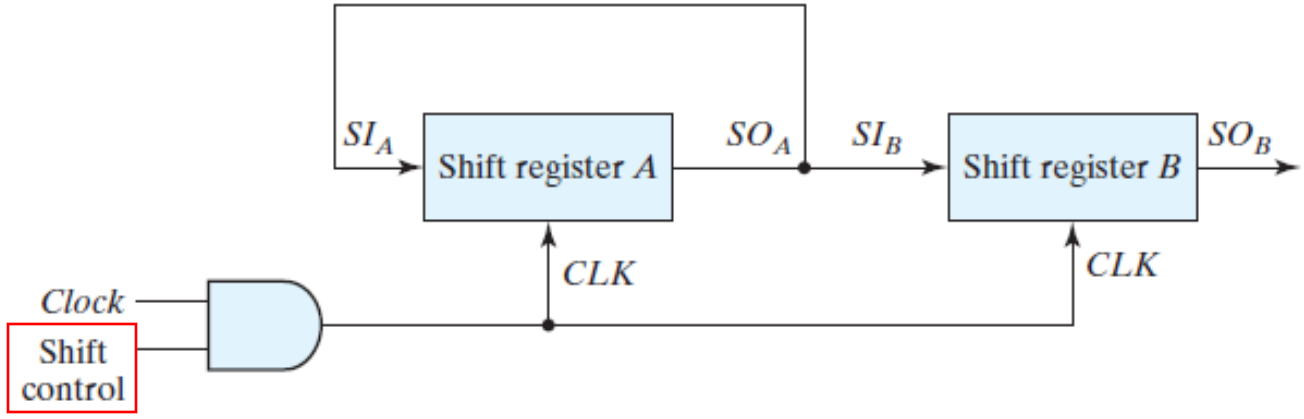
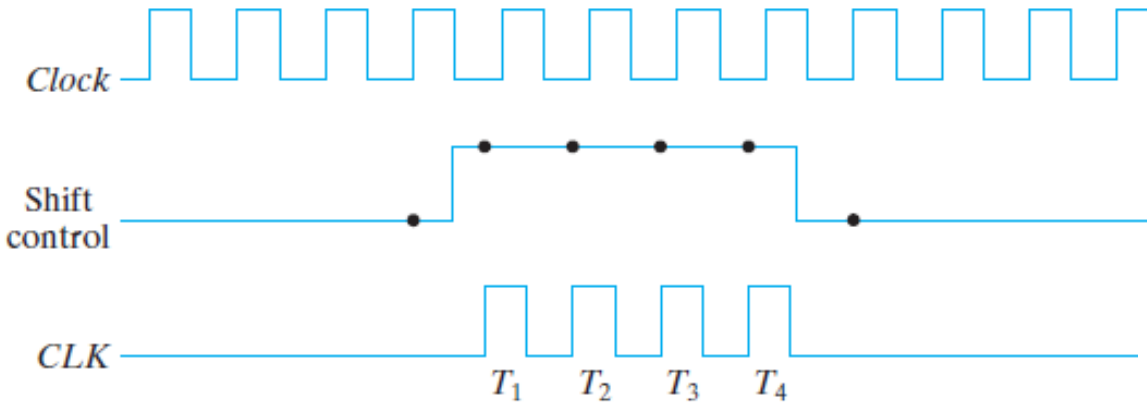


FIGURE 6.3
Four-bit shift register ต้องให้ clock 4 ครั้ง เพื่อโหลดอินพุตให้ครบ 4 บิต

ต้องเอาค่าที่ shift ออกไปวนกลับมาคืน เพื่อรักษาค่าใน register A ไว้



(a) Block diagram



(b) Timing diagram

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

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

A = A + B

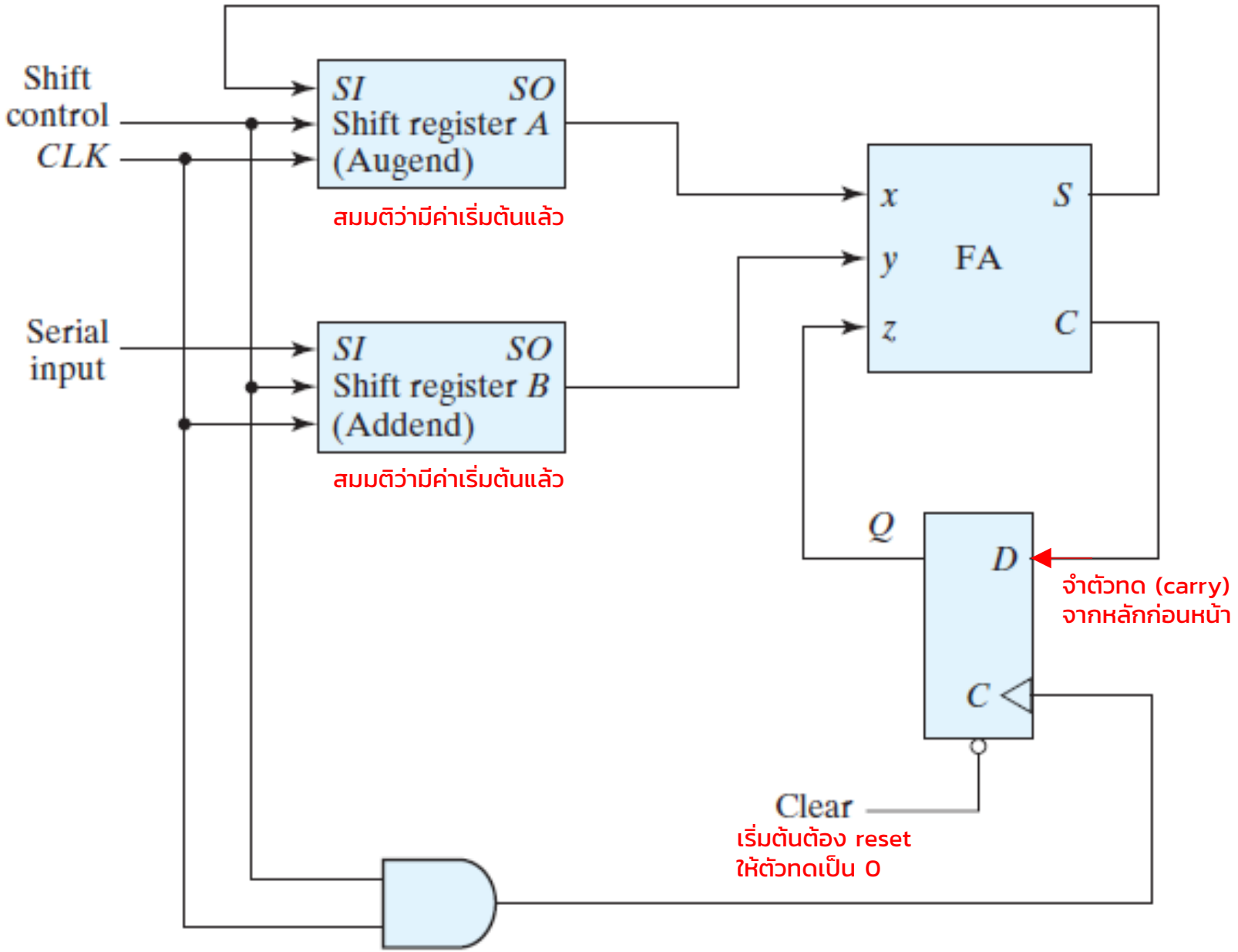


FIGURE 6.5
Serial adder

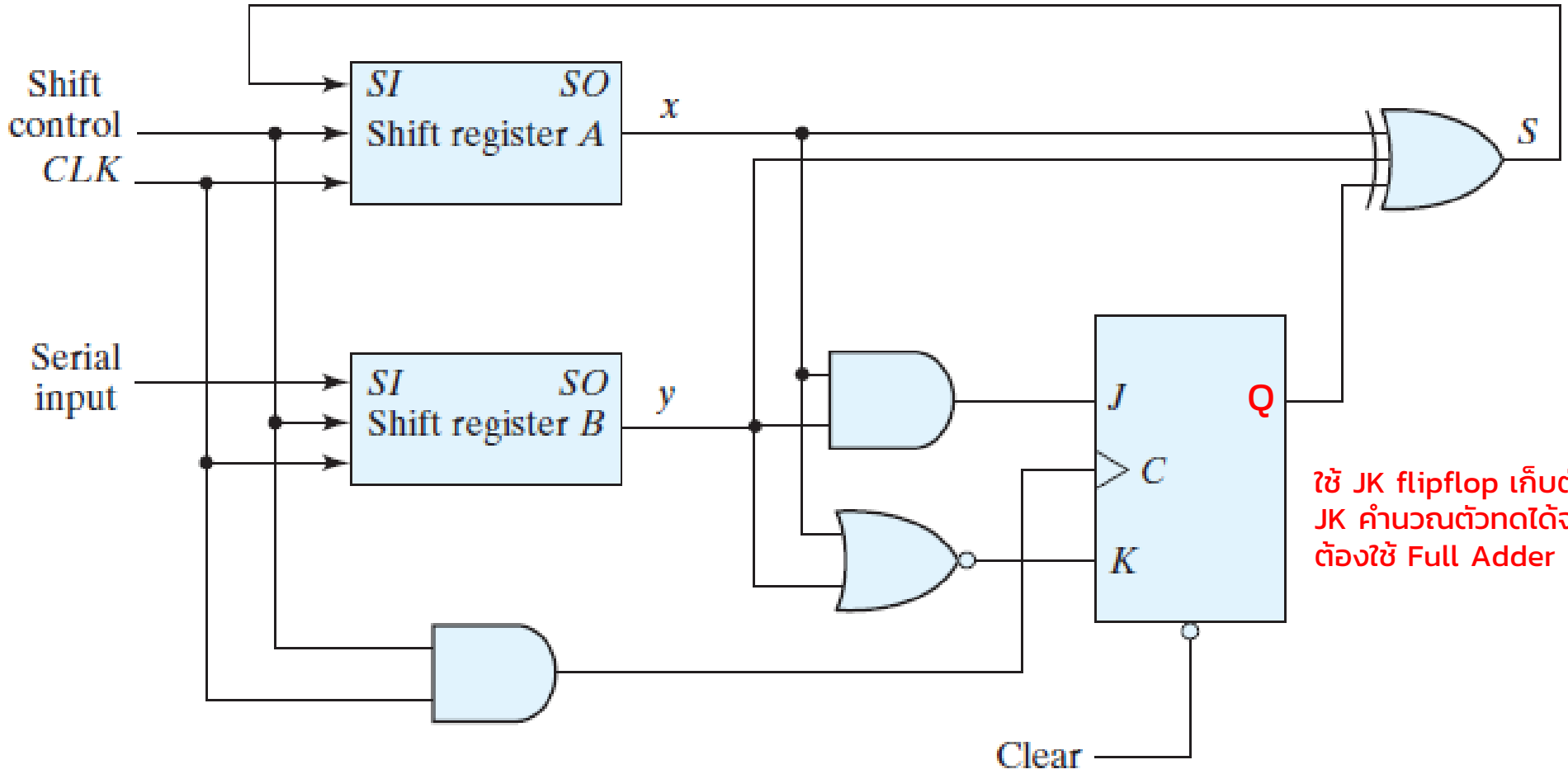
ใช้ JK แทน D และไม่ใช้ Full Adder

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

Table 6.2
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

A = A + B

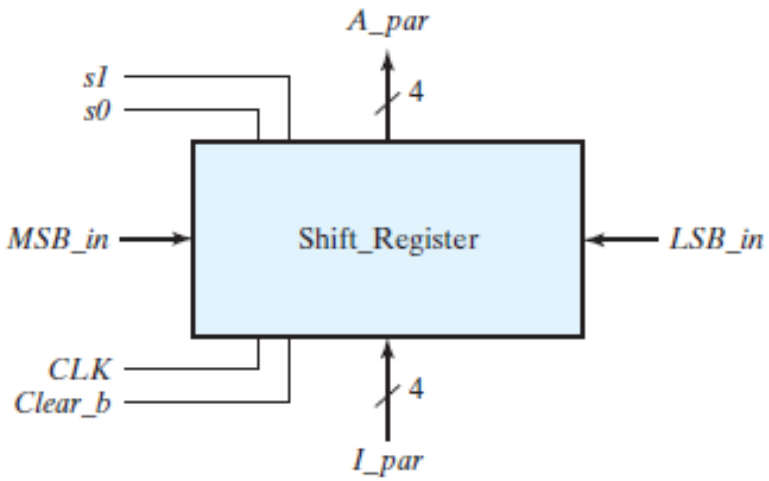


ใช้ JK flipflop เก็บตัวทด แทน D flipflop JK คำนวณตัวทดได้จาก x, y และ Q จึงไม่ต้องใช้ Full Adder

FIGURE 6.6
Second form of serial adder

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



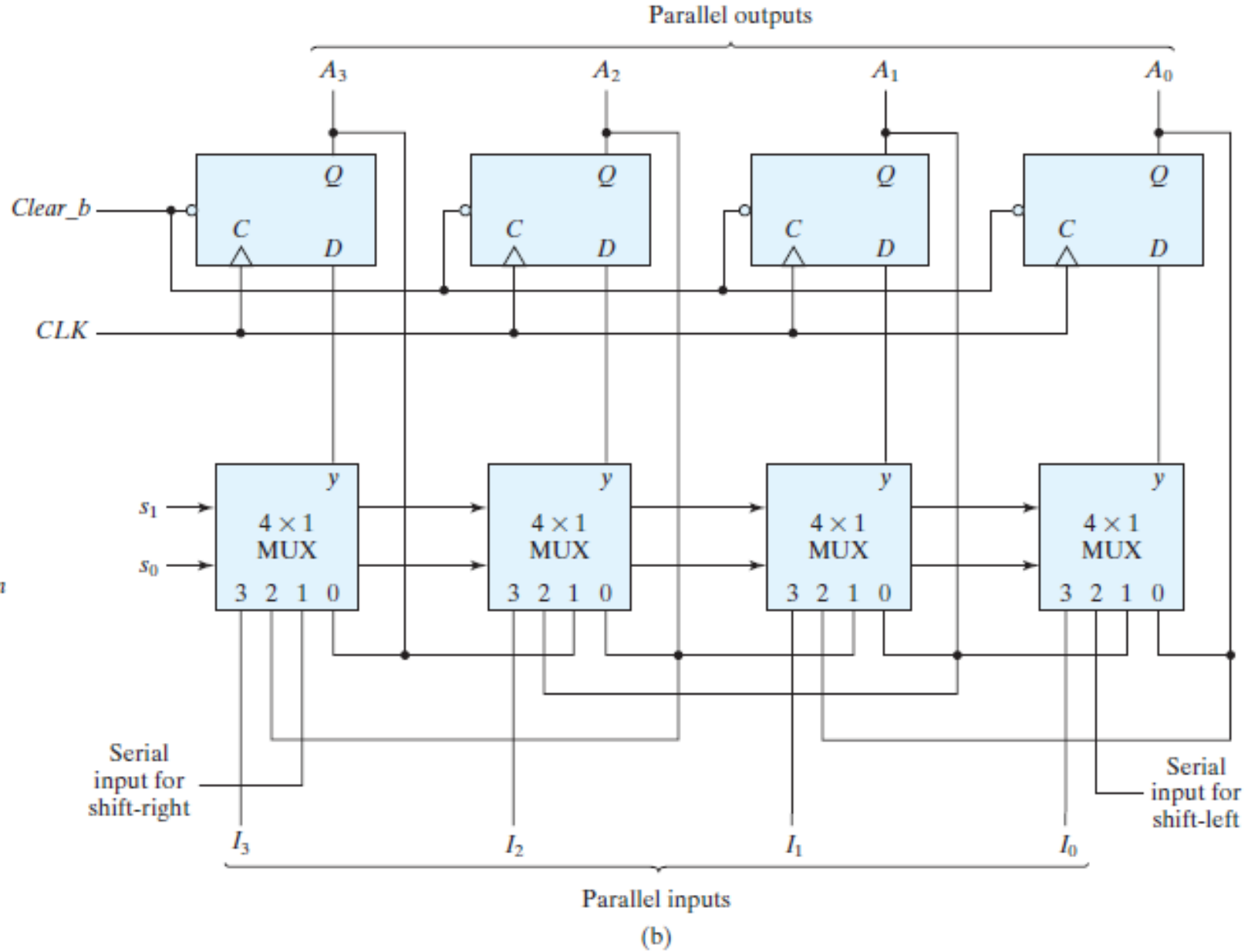


FIGURE 6.7
Four-bit universal shift register

reset ให้เป็น 0000 ก่อน
LSB เปลี่ยนทุก clock
กระเพื่อม (ripple)
จาก LSB ไป MSB

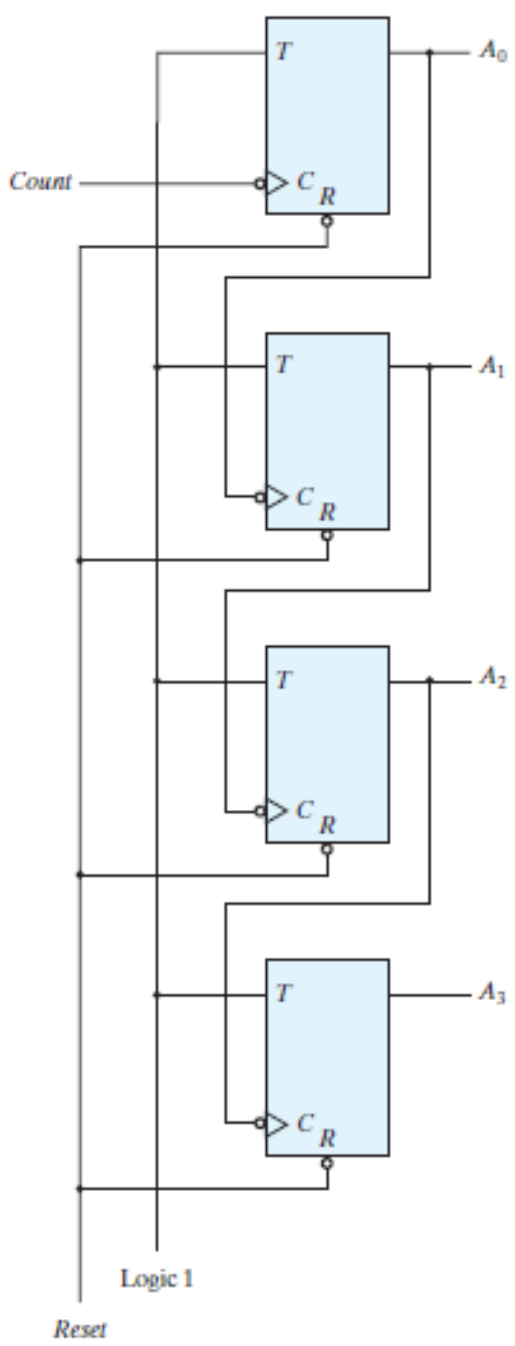
Table 6.4
Binary Count Sequence

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

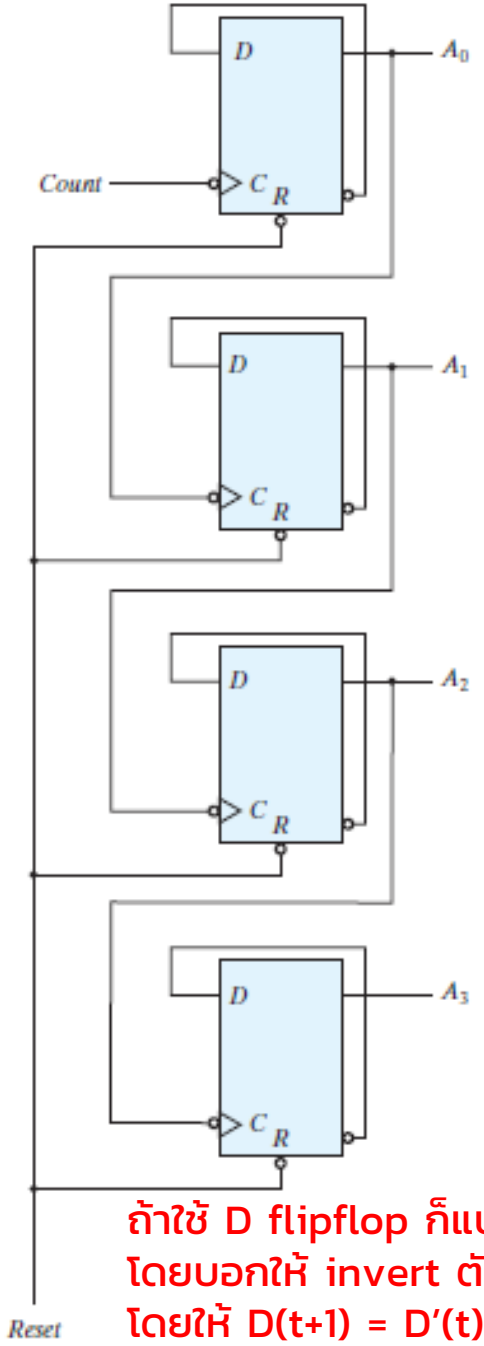
ใช้ neg. edge ไป flip บิตซ้าย

FIGURE 6.8
Four-bit binary ripple counter

แบบ ripple จะตรงข้ามกับ synchronous
ที่ flipflop ทุกตัวใช้ clock ร่วมกัน



(a) With *T* flip-flops



(b) With *D* flip-flops

ถ้าใช้ D flipflop ก็แปลงให้เป็น T
โดยบอกให้ invert ตัวมันเอง
โดยให้ D(t+1) = D'(t)

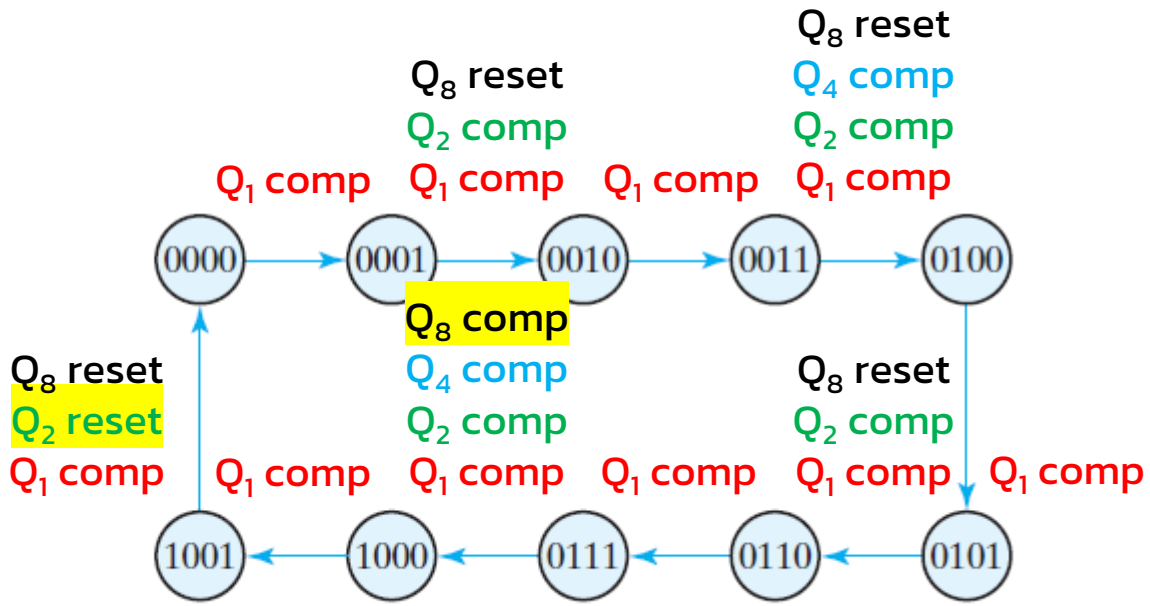


FIGURE 6.9
State diagram of a decimal BCD counter

ทำแบบ ripple ออกแบบยาก ต้องกะเอาเอง
ใช้ Karnaugh map ไม่ได้

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

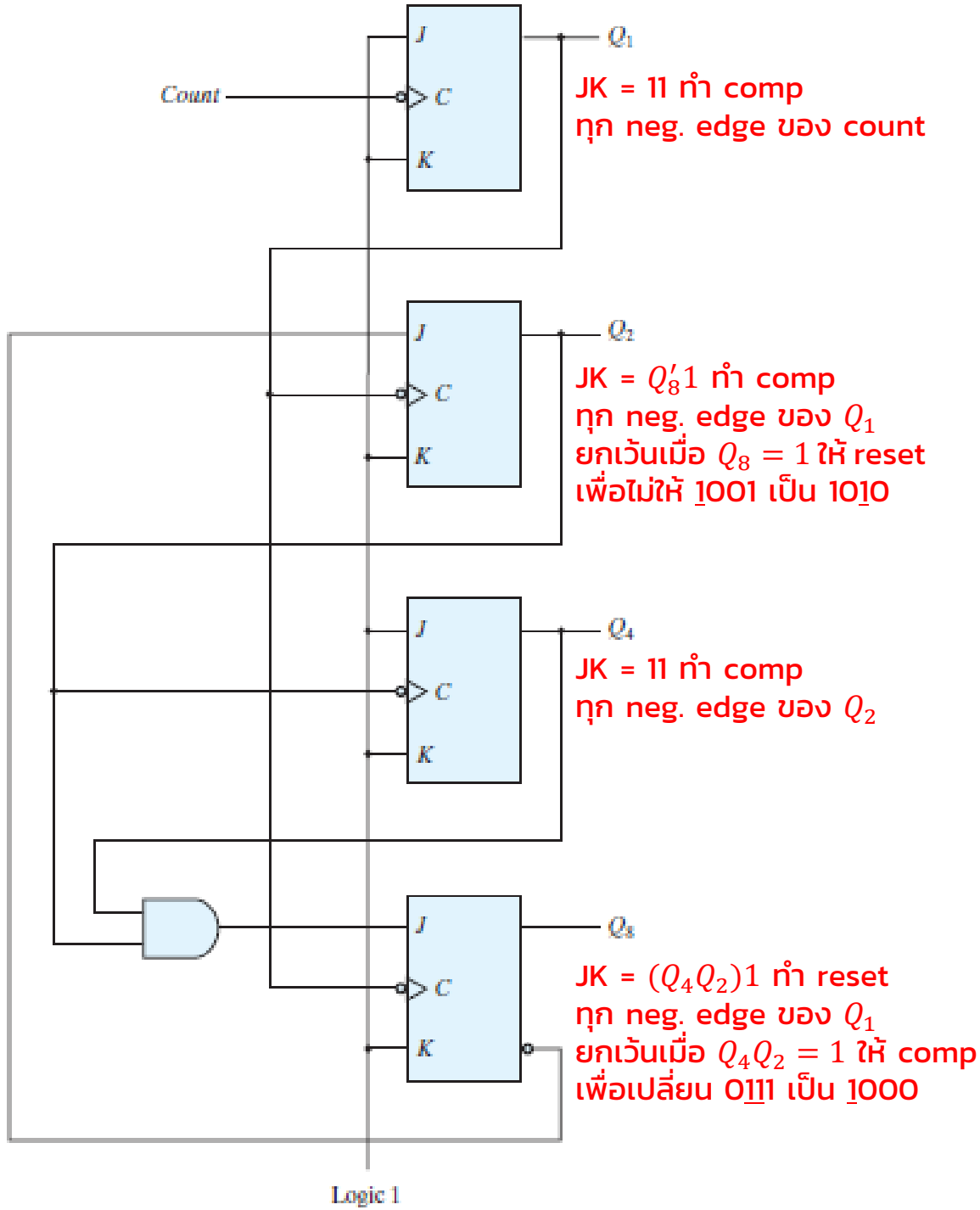


FIGURE 6.10
BCD ripple counter

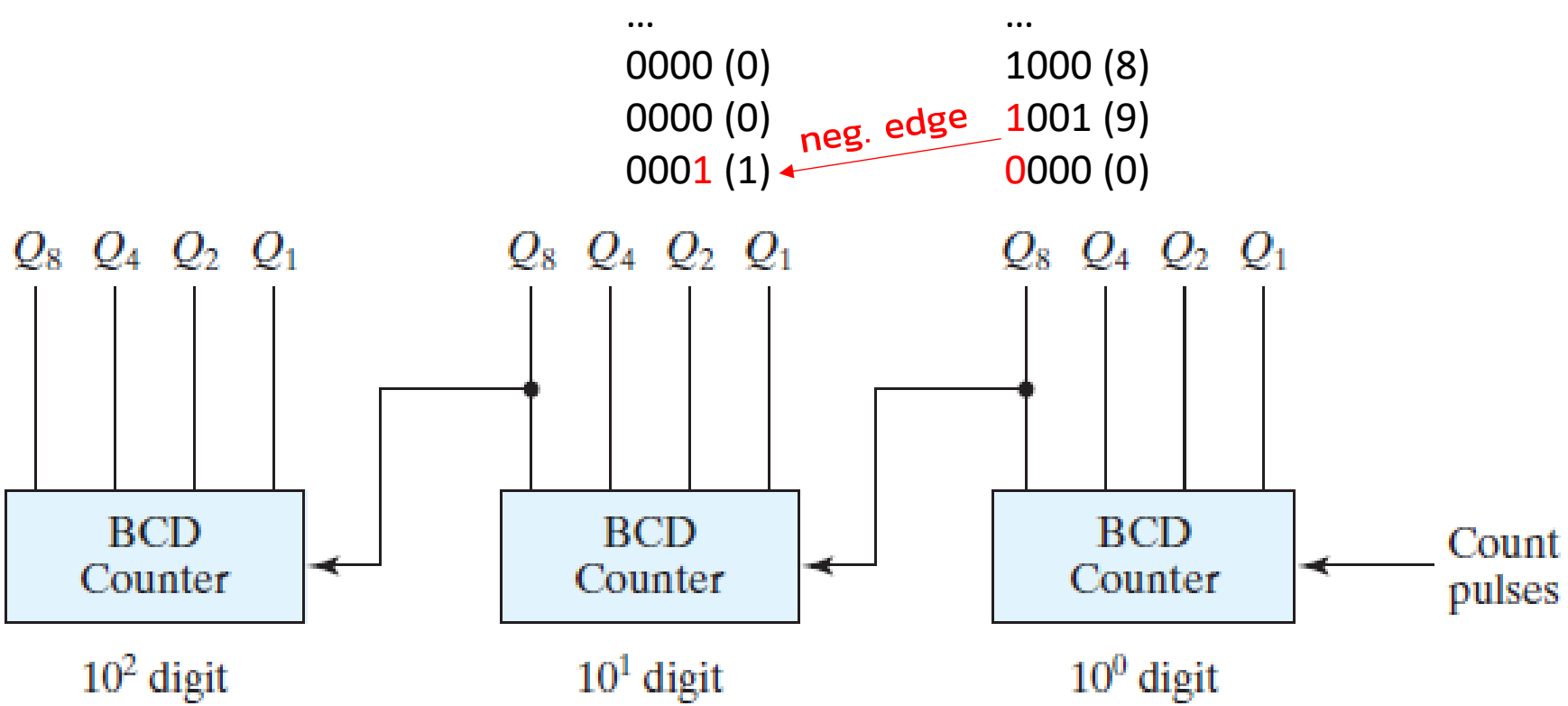


FIGURE 6.11
Block diagram of a three-decade decimal BCD counter

009 จะนับต่อเป็น 010
099 จะนับต่อเป็น 100

ใช้ T flipflop

แบบ synchronous จะเขียน truth table และทำ Karnaugh map ได้

Table 6.5
State Table for BCD Counter

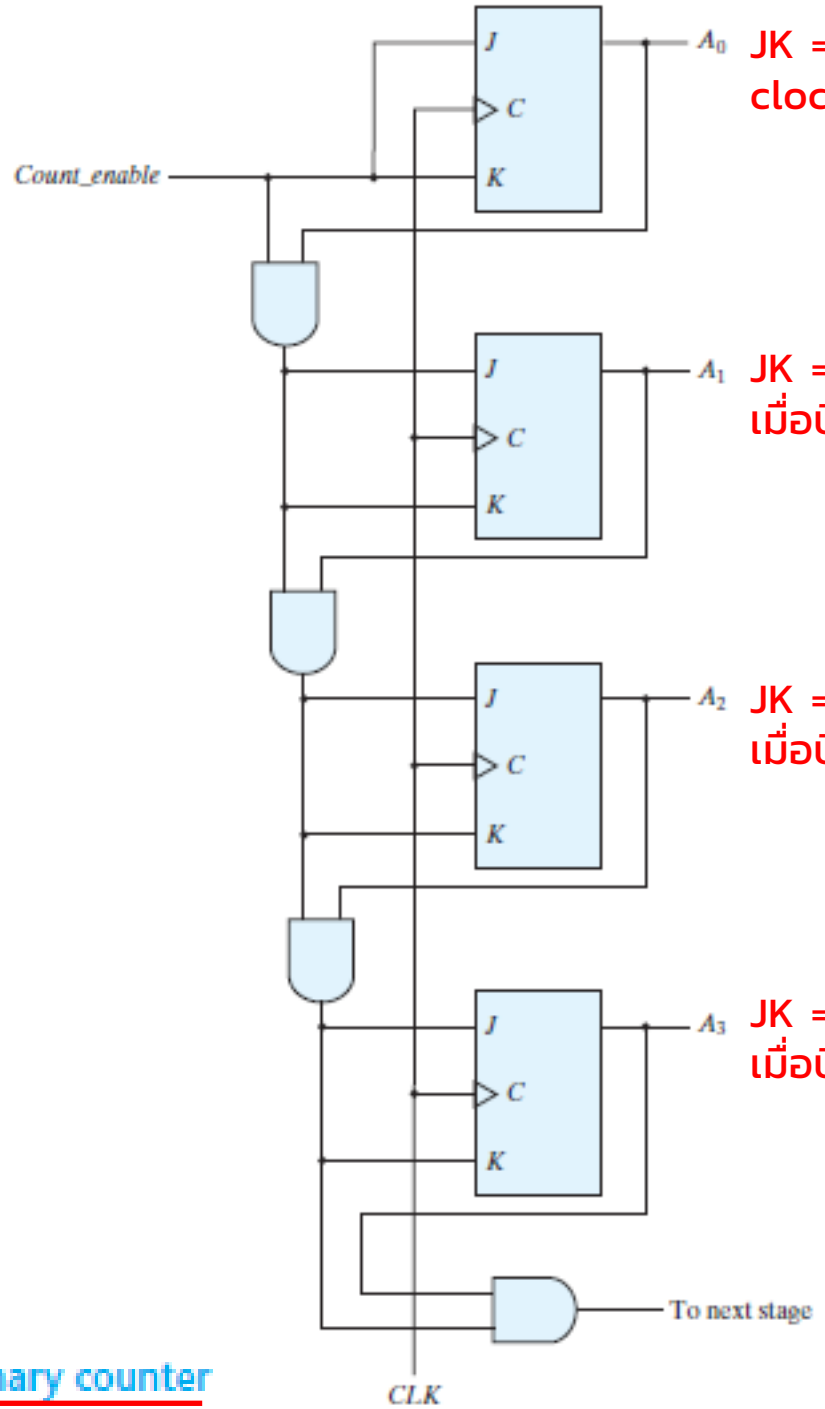
Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1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

ตัวทดไปหลักถัดไป
(carry out)

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

ยังไม่ได้ทำรูป (ใช้ Logisim วาดก็ได้)

1 ให้นับ
0 หยุดนับ



JK = 11 ทำ comp ทุก clock

JK = 11 ทำ comp เมื่อบิตทางขวาเป็น 1 หมด

JK = 11 ทำ comp เมื่อบิตทางขวาเป็น 1 หมด

JK = 11 ทำ comp เมื่อบิตทางขวาเป็น 1 หมด

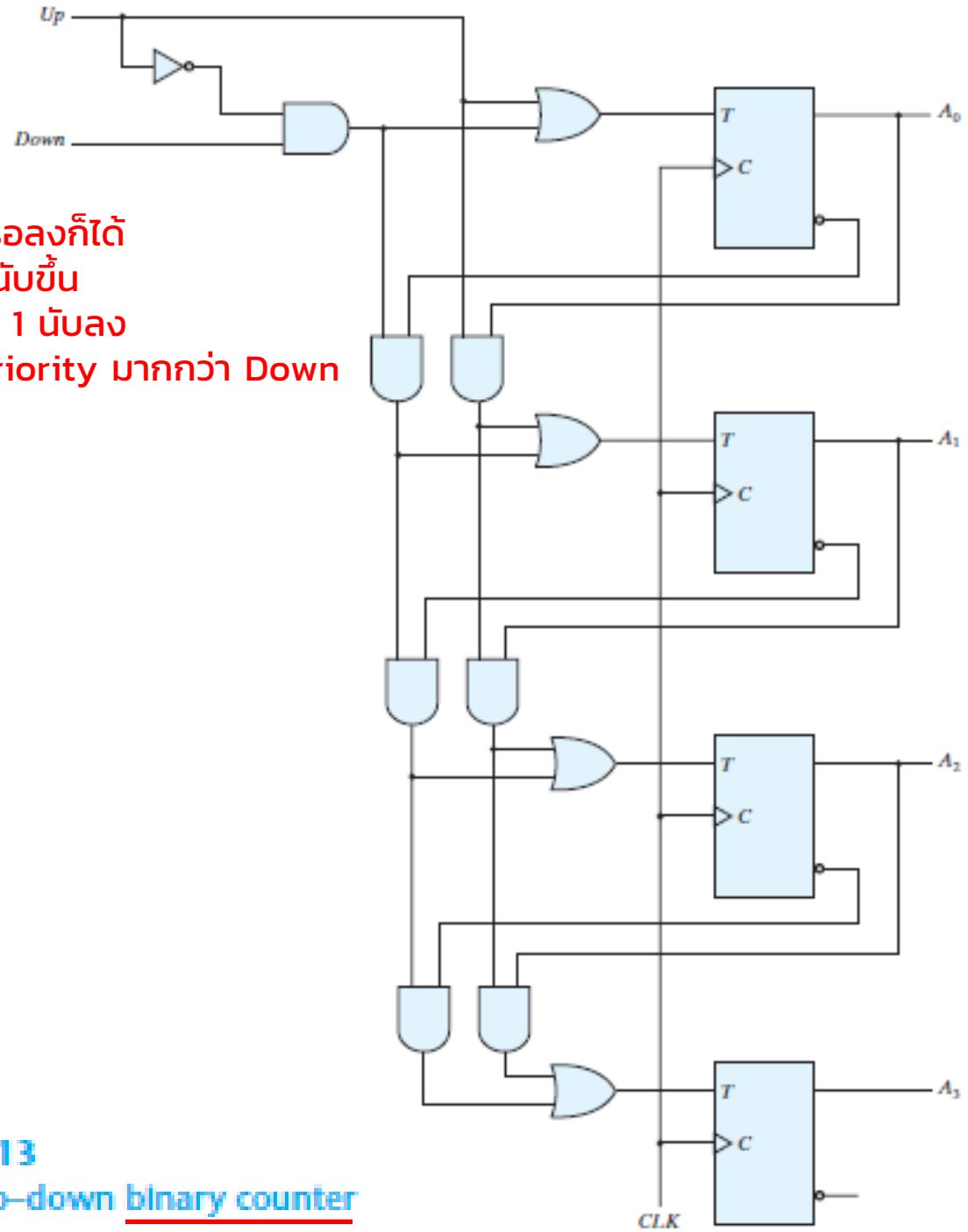
JK Flip-Flop

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

FIGURE 6.12
Four-bit synchronous binary counter

แบบ synchronous ออกแบบง่ายกว่า ripple

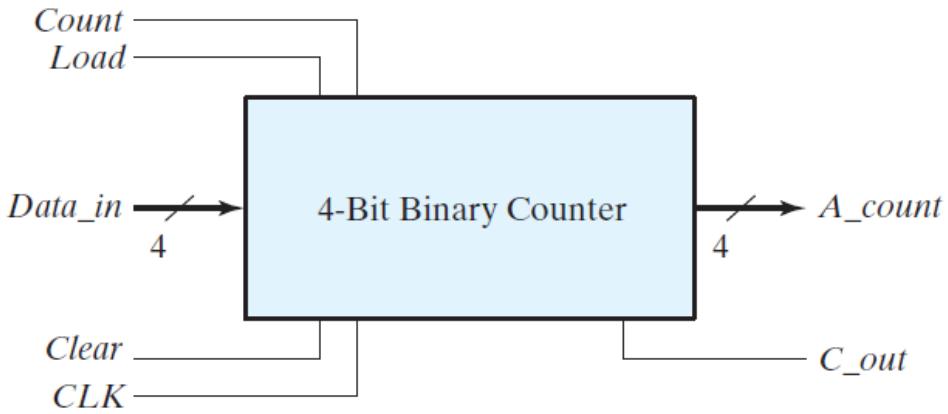
นับขึ้นหรือลงก็ได้
Up = 1 นับขึ้น
Down = 1 นับลง
Up มี priority มากกว่า Down



นับขึ้นคือ ถ้าบิตทางขวาเป็น 1 หมด ให้ comp เช่น
0111 → 1000

นับลงคือ ถ้าบิตทางขวาเป็น 0 หมด ให้ comp เช่น
1000 → 0111

FIGURE 6.13
Four-bit up–down binary counter

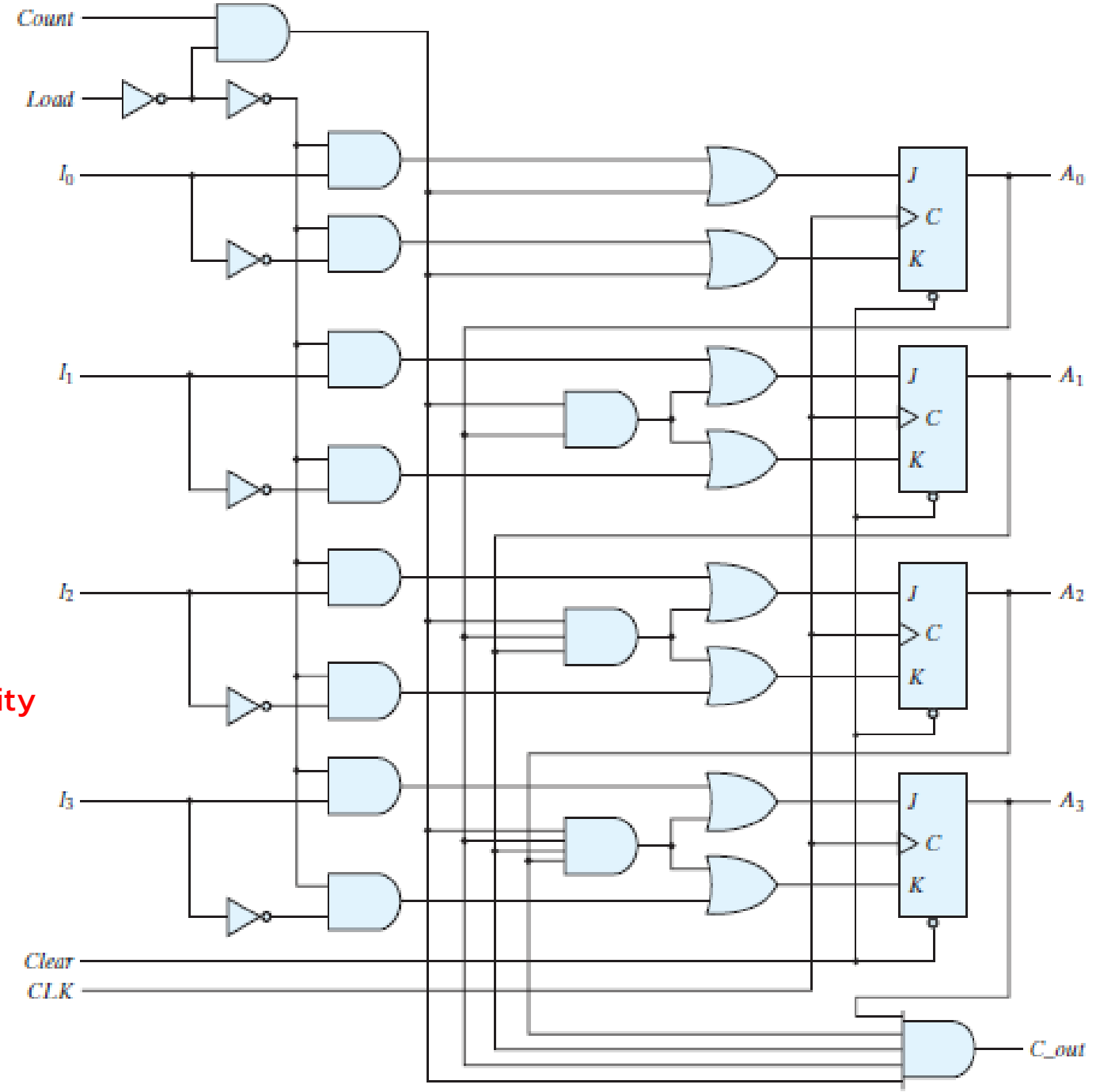


(a)

Table 6.6
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

เรียงตาม priority



(b)

FIGURE 6.14
Four-bit binary counter with parallel load

นับถึง 1001 (9) แล้วให้โหลด 0000 (0)

นับถึง 1010 (10) แล้วให้เคลียร์ 0000 (0)

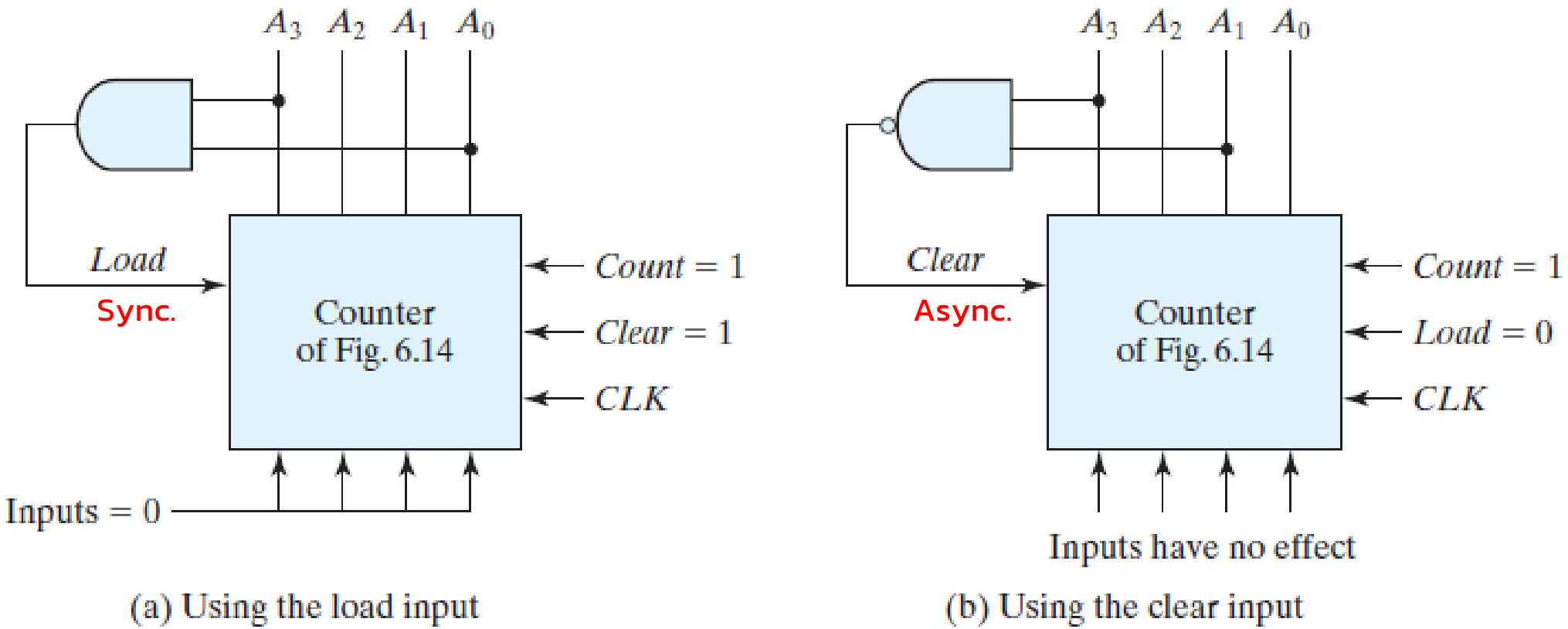
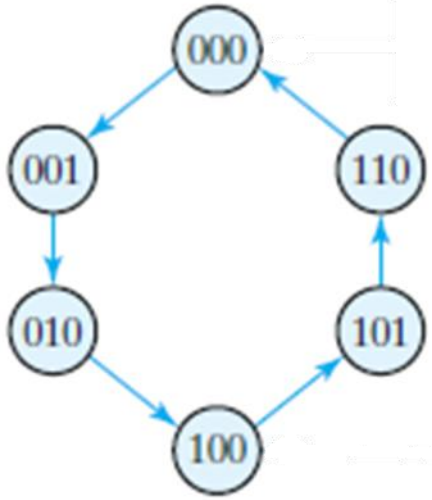


FIGURE 6.15

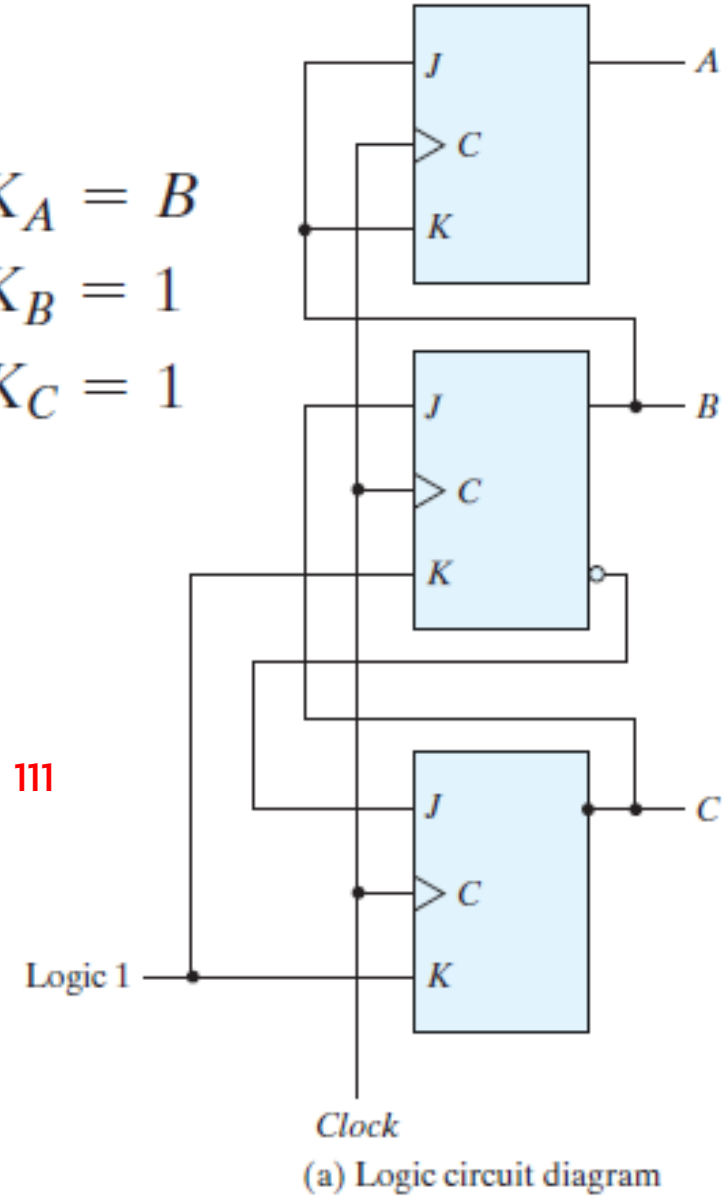
Two ways to achieve a BCD counter using a counter with parallel load

แบบ (a) ดีกว่า เพราะไม่สร้างเอาพุต (1010) ที่ผิดออกมา แบบ (b) จะสร้าง 1010 ออกมา เป็นเวลาสั้น ๆ (ns) เท่ากับ propagation delay ที่ใช้ reset flipflop



(b) State transition diagram

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



(a) Logic circuit diagram

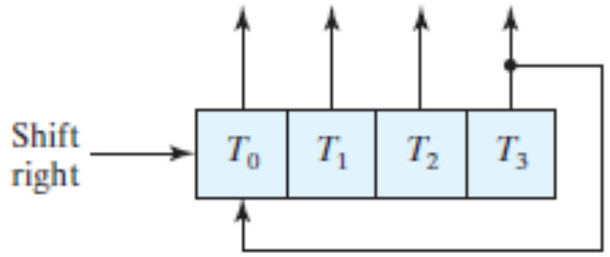
FIGURE 6.16
Counter with unused states

Table 6.7
State Table for Counter

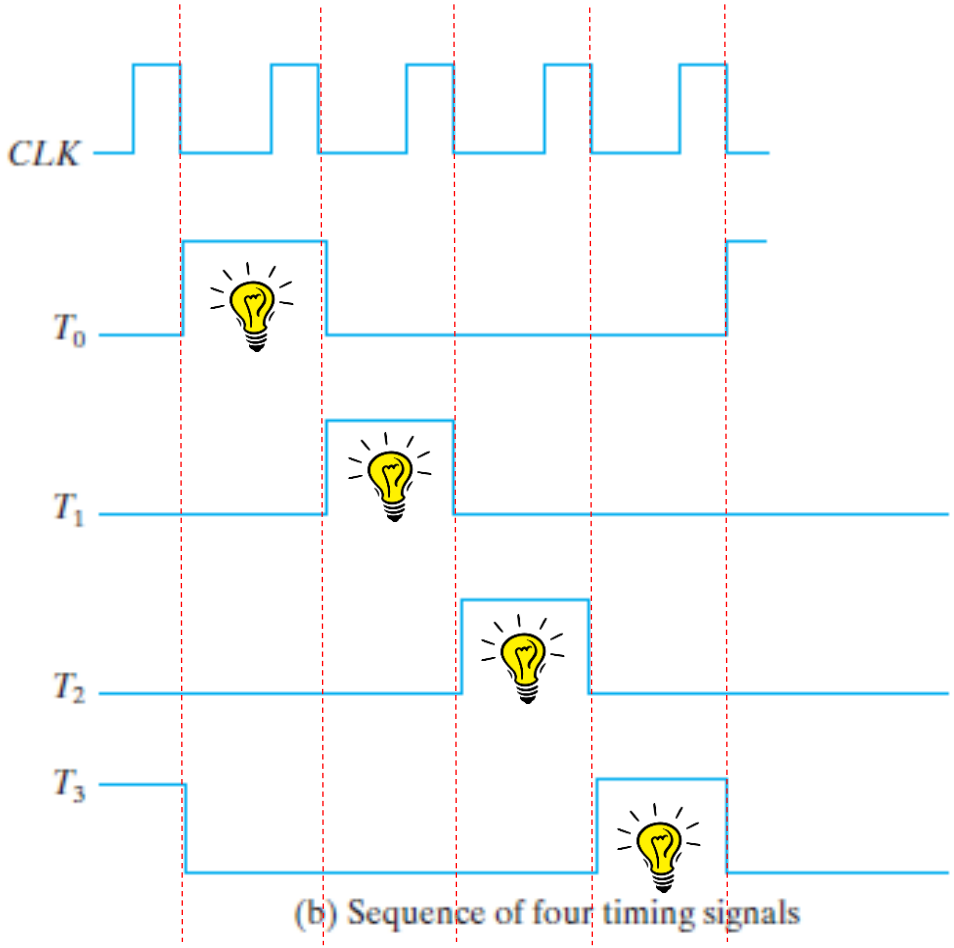
ออกแบบสำหรับกรณี state เริ่มต้นเป็น 000 ก็พอ ไม่ต้องสนใจ 011 กับ 111

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

ทำงานที่
neg. edge

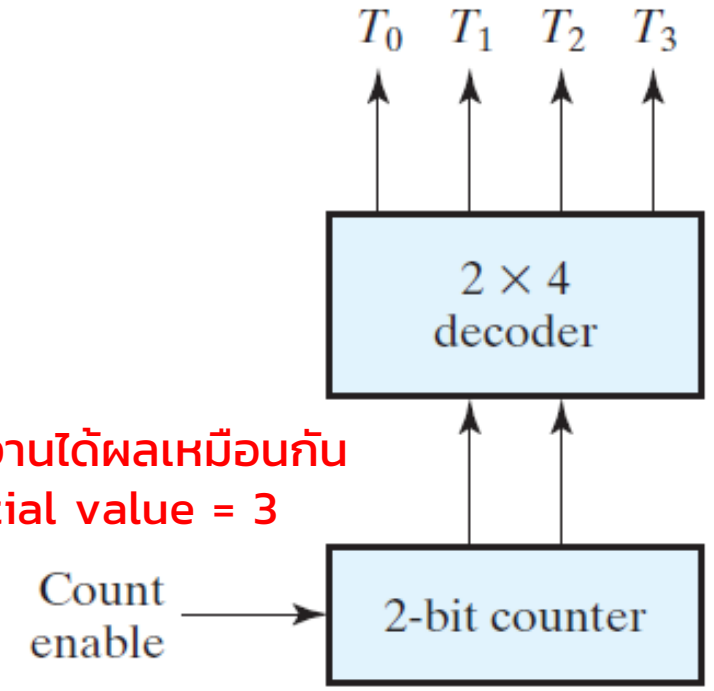


(a) Ring-counter (initial value = 1000)



(b) Sequence of four timing signals

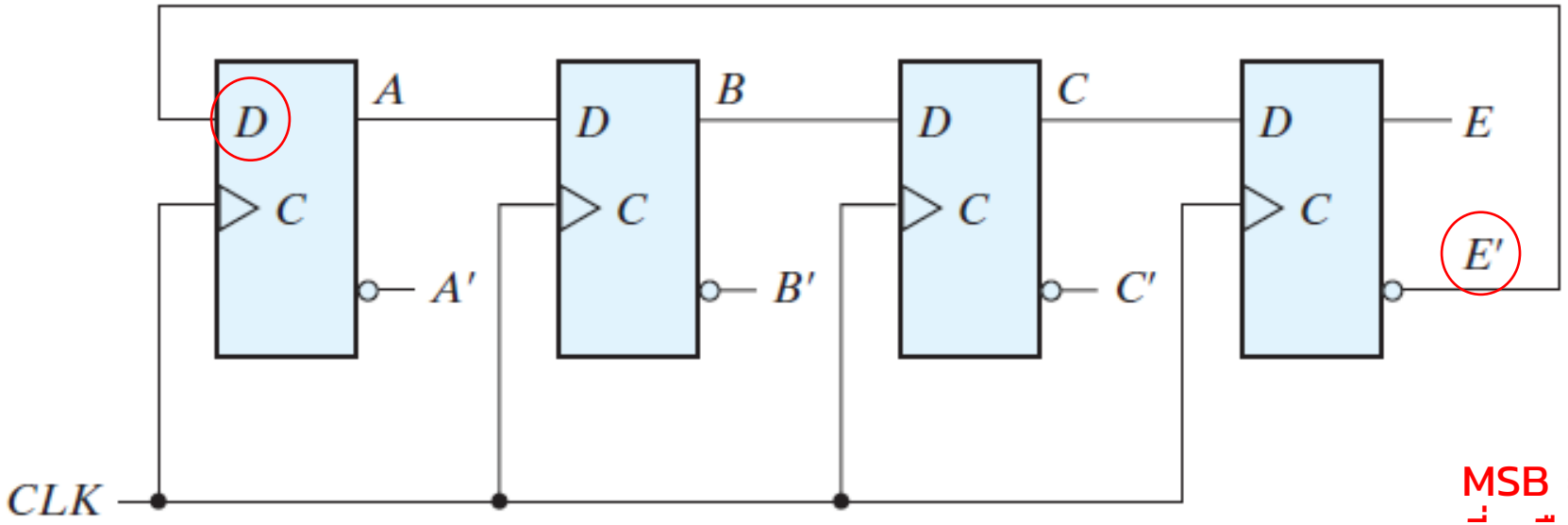
จุดไฟ 4 ดวงให้ติดตามลำดับ ทีละดวง
มี 1 ช่องเดียวที่ T_3 วนไปเรื่อย ๆ



(c) Counter and decoder

ทำงานได้ผลเหมือนกัน
initial value = 3

FIGURE 6.17
Generation of timing signals



(a) Four-stage switch-tail ring counter

input				output							
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0
1	1	1	1	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	1

ข้อเสียของวิธีก่อนหน้านี้คือ ใช้ flipflop มากถึง n ตัว หรือต้องใช้ decoder ขนาดใหญ่ แต่ว Johnson counter ใช้ flipflop แค่ n/2 ตัว

MSB = LSB′
ที่เหลือ shift right ตามปกติ

ทำ k-map 8 ครั้ง

Sequence number	Flip-flop outputs A	B	C	E	AND gate required for output (เป็นค่าที่ต่างกันทั้งหมด)	
1	0	0	0	0	0	$A'E'$
2	1	0	0	0	8	AB'
3	1	1	0	0	12	BC'
4	1	1	1	0	14	CE'
5	1	1	1	1	15	AE
6	0	1	1	1	7	$A'B$
7	0	0	1	1	3	$B'C$
8	0	0	0	1	1	$C'E$

(b) Count sequence and required decoding

เอา 1 shift เข้ามา n/2 ครั้ง จนถึง LSB เอา 0 shift เข้ามาอีก n/2 ครั้ง จะได้ค่าใน counter ที่ต่างกัน 8 แบบ ทำ Karnaugh map อีก n ครั้ง

FIGURE 6.18
Construction of a Johnson counter