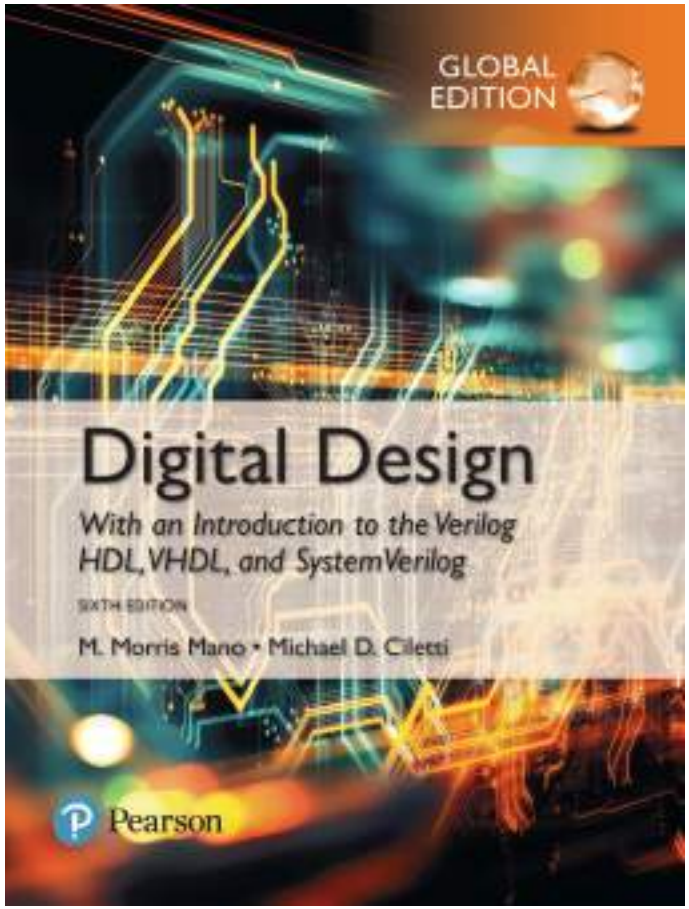
Digital Design

With an Introduction to the Verilog HDL, VHDL, and SystemVerilog

6th Edition, Global Edition



Chapter 06

Registers and Counters

레지스터와 카운터

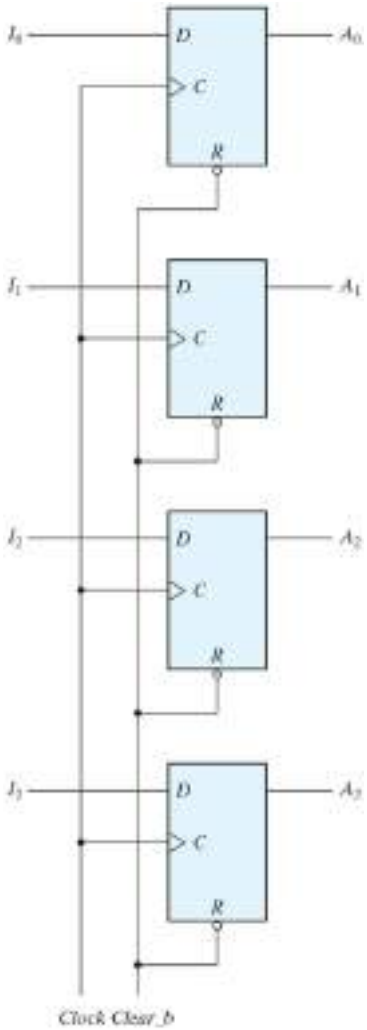
전자공학과 김동한 교수

Pearson

6.1 레지스터

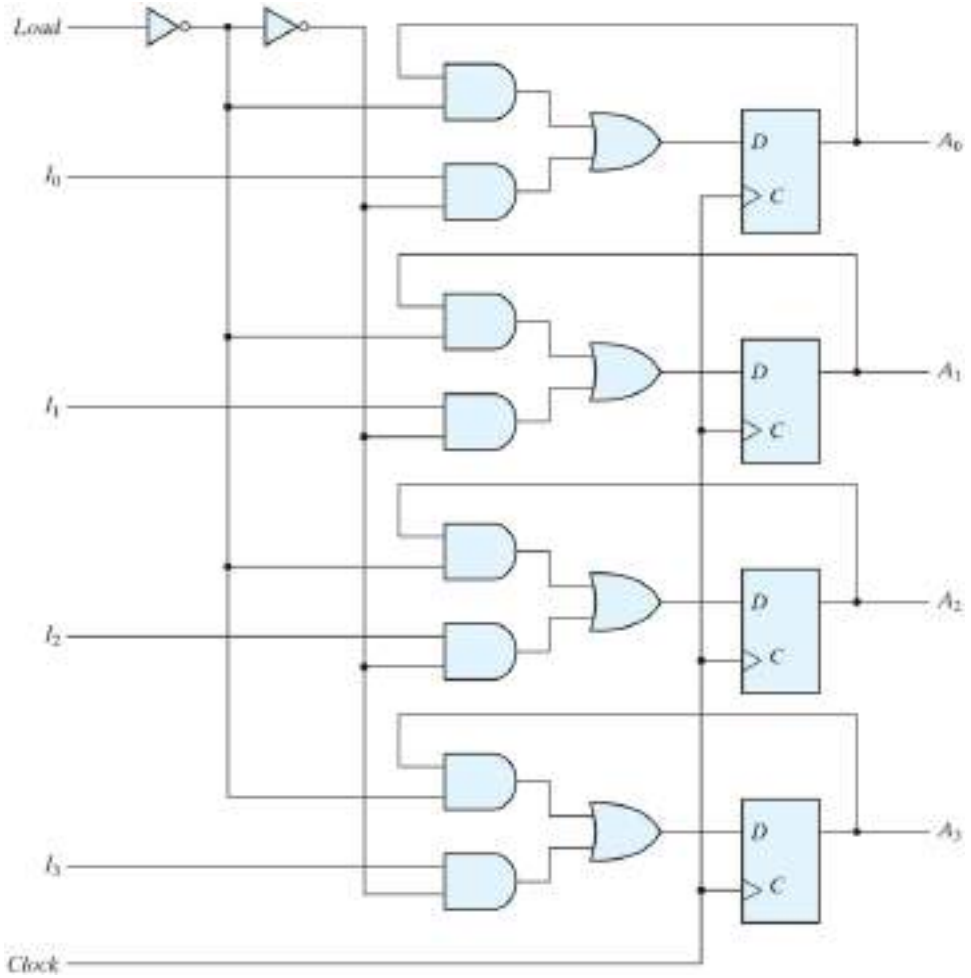
- 플립플롭을 포함하고 있는 회로는 순차회로이지만, 수행하는 기능에 의해서 레지스터와 카운터 등으로 분류함
- 레지스터: 플립플롭들로 구성되면, 각 플립플롭은 클럭을 공유하고 한 비트의 정보를 저장할 우 있음. n비트 레지스터는 n개의 플립플롭으로 이루어 짐. 추가적으로 자료 처리 작업을 수행하는 조합 게이트를 가질 수 있음
- 카운터: 입력 펄스가 가해짐에 따라 상태를 미리 정해진 순서대로 진행시키는 레지스터임. 레지스터의 특수한 형태이지만 레지스터와 구별하는 것이 보통임
- 가장 간단한 레지스터
  - 외부 게이트 없이 플립플롭만으로 구성
  - 그림 6.1은 4개의 플립플롭으로 구성된 4비트 저장 레지스터
  - 하나의 공통된 클럭 펄스 입력의 상향 에지에서 구동
  - 4개의 입력 (I)에서 출력 (A)로 전송. Clear_b 는 저활성 reset

Figure 6.1
Four-bit register.



- 병렬 로드를 가진 레지스터
  - 레지스터에서 새로운 정보를 전송하는 것을 로딩(loading) 또는 갱신(updating)이라고 하고 모든 비트가 단일 클럭 펄스에서 동시에 로드된다면, 병렬이라고 함
  - 그림 6.1에서 만약 레지스터의 내용이 변하지 않고 그대로 있어야 한다면, 입력을 유지하거나 클럭 펄스가 들어가지 않게 해야 함. 클럭 펄스를 제어하는 논리 게이트 등을 넣으면 클럭 펄스와 플립플롭 입력 간에 전파 진연이 생겨서 바람직하지 않음.
  - 따라서 그림 6.2처럼 로드 제어 입력 논리 게이트를 추가함. 로드 입력이 1일 때, 입력 데이터는 다음 클럭 펄스에서 레지스터로 전송되고, 로드 입력이 0일 때, 플립플롭의 출력은 입력으로 귀환 연결되어 상태를 유지함.

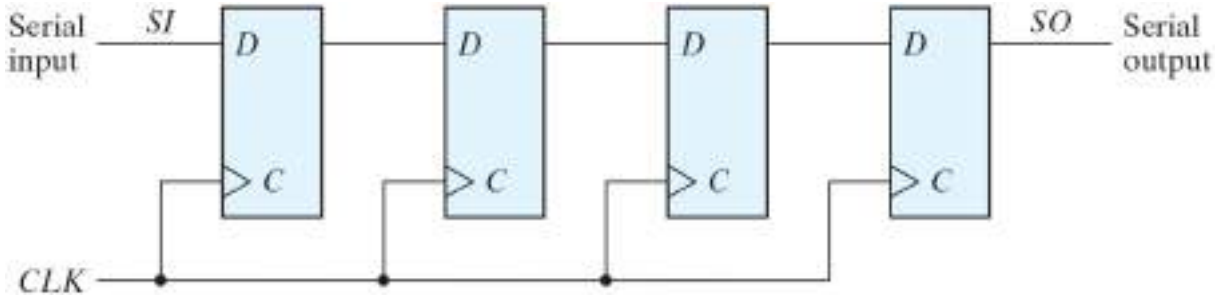
Figure 6.2
Four-bit register with parallel load.



6.2 시프트 레지스터 (Shift Register)

- 가지고 있는 2진 정보를 인접한 셀에 원하는 방향으로 자리옮김(shifting) 하는 레지스터임
- 가장 간단한 시프트 레지스터는 그림 6.3과 같이 플립플롭만을 사용한 것

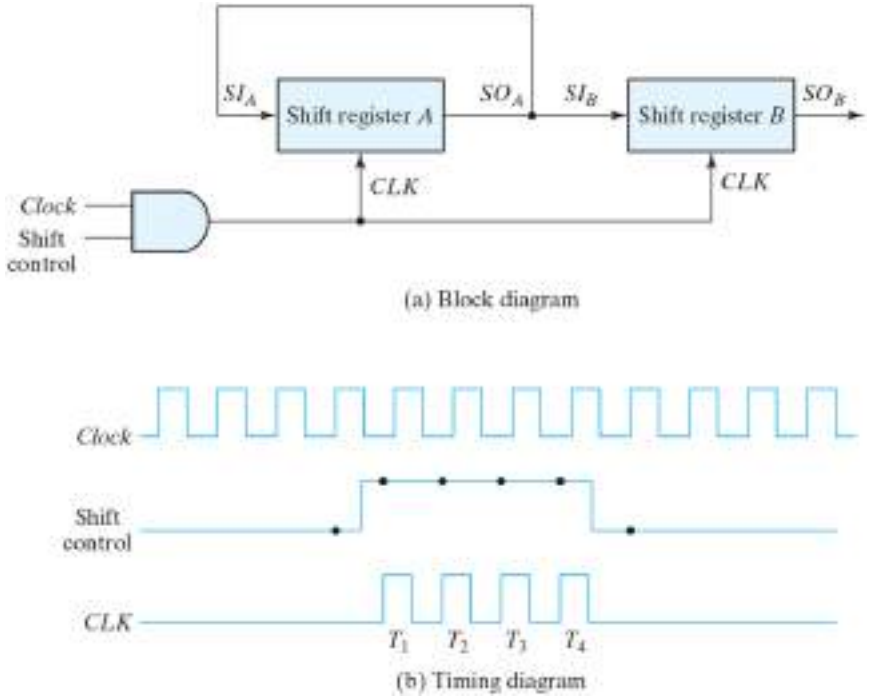
Figure 6.3
Four-bit shift register.



- 한 플립플롭의 출력이 다음 플립플롭의 입력에 종속 연결된 형태. 왼쪽에서 오른쪽 방향으로만 동작함.

- 직렬 전송
  - 송신 측 레지스터에서부터 나온 비트들이 수신 측 레지스터로 시프트되어 한 번에 한 비트씩 전송됨

Figure 6.4
Serial transfer from register *A* to register *B*.



  - 레지스터 A에서 레지스터 B로 직렬전송. 소스 레지스터에 저장된 정보의 손실을 막기 위해 레지스터 A는 출력을 자신의 입력에 연결하여 데이터가 순환하도록 만듦

– CLK는 Clock과 시프트 제어를 같이 AND 게이트에 넣어 제어함.
이것은 클럭 경로에 회로를 추가하기 때문에 문제의 여지가 있음

Table 6.1
Serial-Transfer Example.

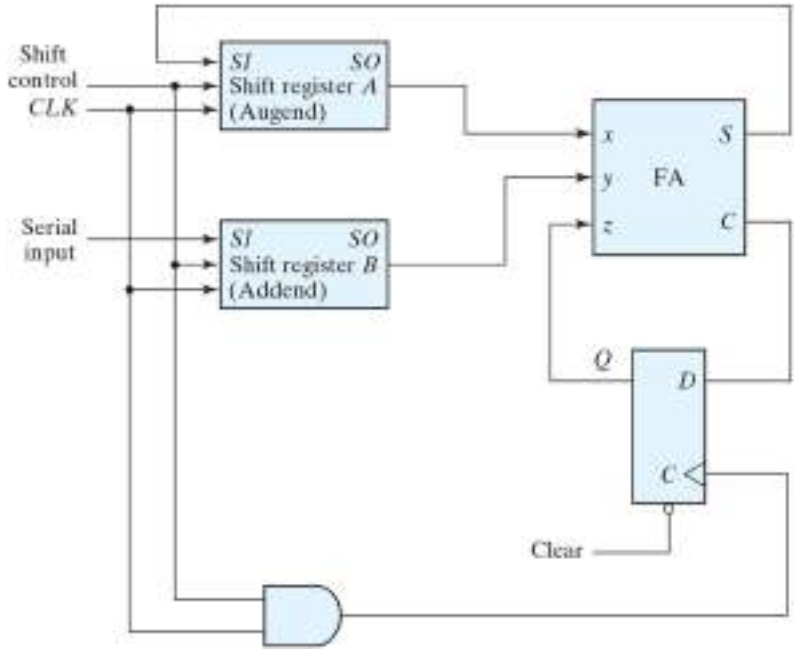
Timing Pulse	Shift Register *A*				Shift Register *B*				Serial out put of B
Initial value	1	0	1	1	0	0	1	0	
After T_1	1	1	0	1	1	0	0	1	0
After T_2	1	1	1	0	1	1	0	0	1
After T_3	0	1	1	1	0	1	1	0	0
After T_4	1	0	1	1	1	0	1	1	0

– 네 번째 시프트 뒤에 A와 B 레지스터는 모두 1011을 같음

– 직렬 방식은 1비트씩 전송하고 병렬 방식은 모든 비트가 동시에 전송

- 직렬 가산
  - 디지털 컴퓨터의 동작은 대개 병렬로 행해지는데, 더 빠른 동작 방식이기 때문이며, 직렬 동작은 여러 클럭 주기가 필요하기 때문에 느리지만 적은 수의 부품으로 간단히 구성됨
  - 2개의 2진수를 더하는 직렬 가산기를 설명하면,

Figure 6.5
Serial adder.



  - 2개의 2진수는 2개의 시프트 레지스터에 저장되고, 최하위 비트쌍들의 덧셈을 시작으로 한 번에 한 쌍씩 차례로 단일 전가산기 회로를 통해서 더해짐. 전가산기의 캐리 출력은 D 플립플롭에 전송

- A 의 비트들이 시프트되어 나가는 동안 합의 결과를 A로 시프트해 넣음으로써 하나의 레지스터를 피간산수와 합의 결과 모두를 저장하는데 쓸 수 있음
- 동작 설명
  - 처음에 레지스터 A와 레지스터 B에 숫자 저장, 캐리 플립플롭은 0으로 clear
  - A와 B의 직렬 출력(SO)은 x와 y에 전가산기에 공급되고, 캐리 플립플롭의 출력 Q는 z에 입력 캐리를 인가함
  - 시프트 제어 신호는 레지스터와 캐리 플립플롭 모두를 인에이블하고, 다음 클럭 펄스에서 양쪽 레지스터는 오른쪽으로 한 비트 시프트하고, S로부터 나온 합 비트는 A의 맨 왼쪽 레지스터로 들어감
  - 출력 캐리는 플립플롭 Q로 전송됨. 시프트 제어 신호는 레지스터의 비트 수와 같은 수만큼의 클럭 펄스 동안 레지스터를 인에이블함
  - 연속되는 매 클럭 펄스 동안 새로운 합의 결과가 A로 전송되고 새로운 캐리는 Q로 전송되며 양쪽 레지스터가 오른쪽으로 한 번 시프트됨.
  - 처음에 레지스터 A와 캐리 플립플롭은 0으로 clear되고, 첫 번째 수가 B로 더해짐. B가 전가산기를 통해 시프트되는 동안 두 번째 수가 전송되고, 이 수가 레지스터 A에 더해지면서 세 번째 수가 레지스터 B로 전송됨. 레지스터 A에 합을 누적하면서 2번째, 3번째 수가 더해질 수 있음

- 직렬 가산기와 병렬 가산기 비교
  - 직렬 가산기는 시프트 레지스터를 사용하고 병렬 가산기는 병렬 로드 능력이 있는 레지스터 사용
  - 직렬 가산기는 하나의 전가산기 회로와 하나의 캐리 플립플롭만 사용하고 병렬 가산기는 2진수의 비트 수와 같은 전가산기 사용
  - 직렬 가산기는 순차 회로이고, 병렬 가산기는 레지스터를 제외하면 조합 회로임
- 5장의 순차 회로 설계 방법으로 JK 플립플롭으로 직렬 가산기 다시 설계해보면,

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a) *JK* Flip-Flop

$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) *T* Flip-Flop

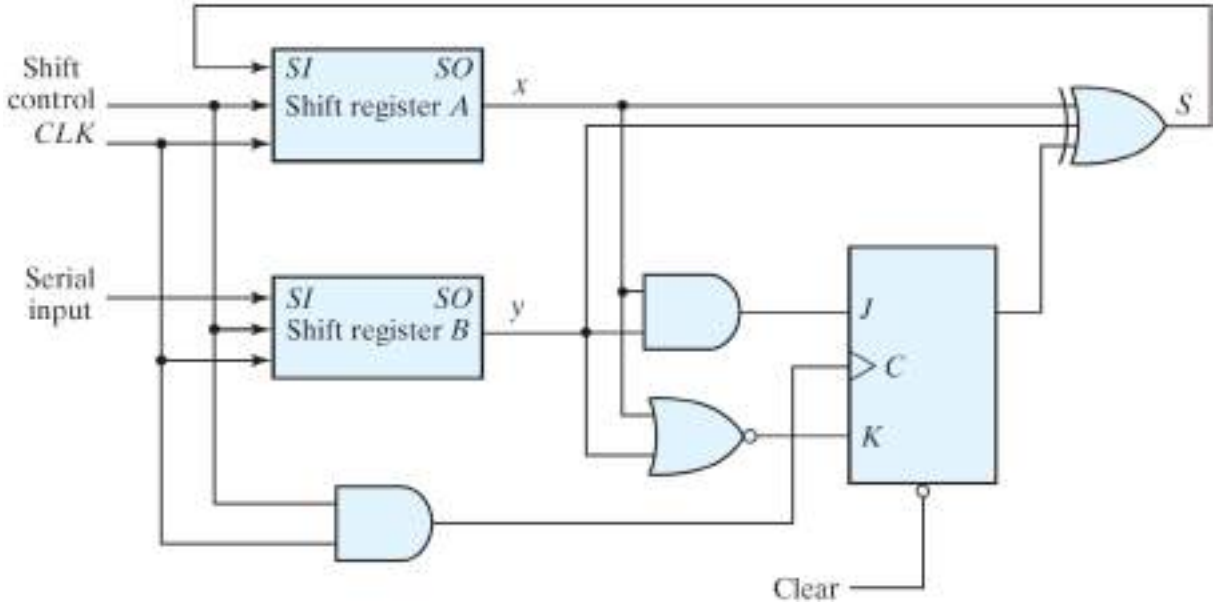
Table 6.2
State Table for Serial Adder.

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

– 표 6.2의 마지막 두 열(J_Q, K_Q)과 출력 S를 간략화하면,

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

Figure 6.6
Second form of serial adder.



– 쌍방향 시프트 레지스터 (universal shift register)
  - 한쪽 방향(오른쪽 혹은 왼쪽)으로만 시프트가 가능한 레지스터는 단방향(unidirectional) 시프트 레지스터
  - 양쪽 방향(오른쪽과 왼쪽)으로 시프트가 가능하면 양방향(bidirectional)
  - 양방향이면서 병렬 로드 기능을 가지면 쌍방향(universal) 시프트 레지스터

Figure 6.7
Four-bit universal shift register.

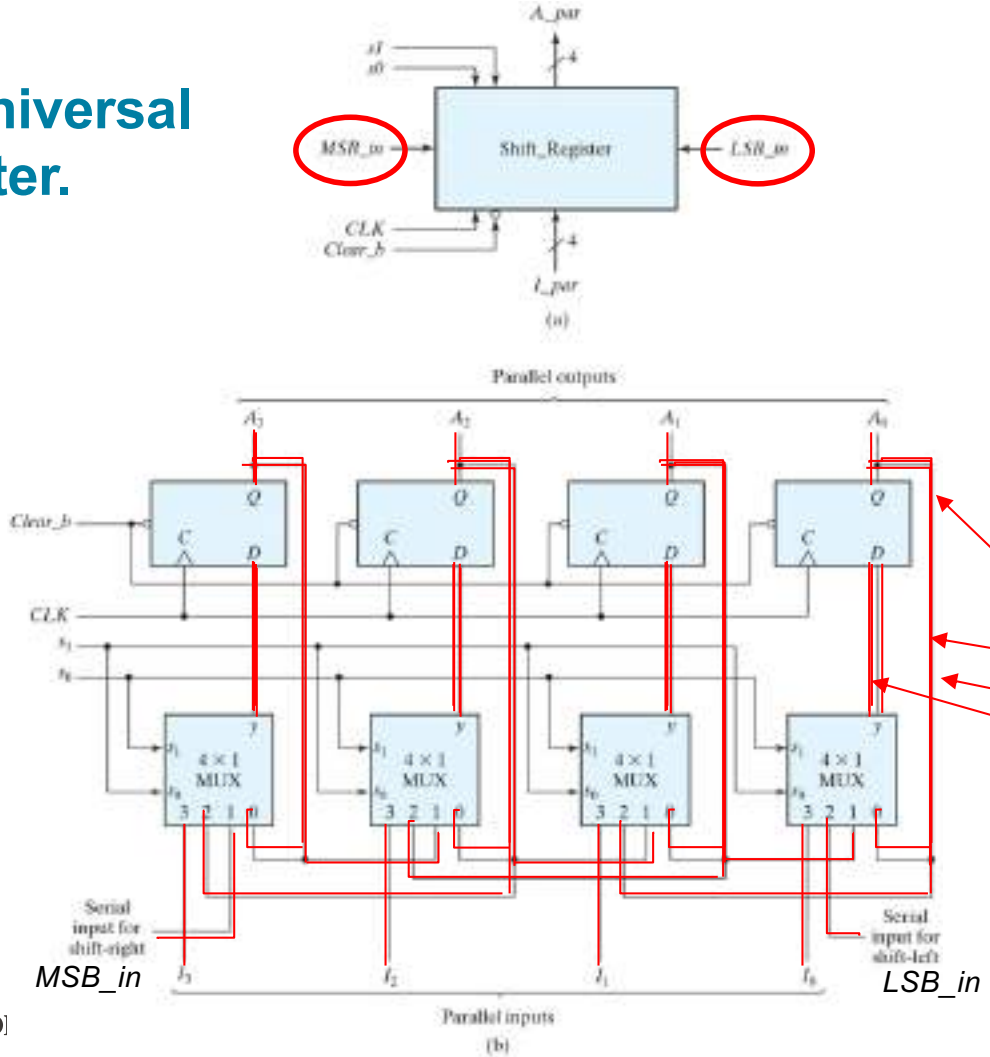


Table 6.3
Function Table for the Register of Fig. 6.7.

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

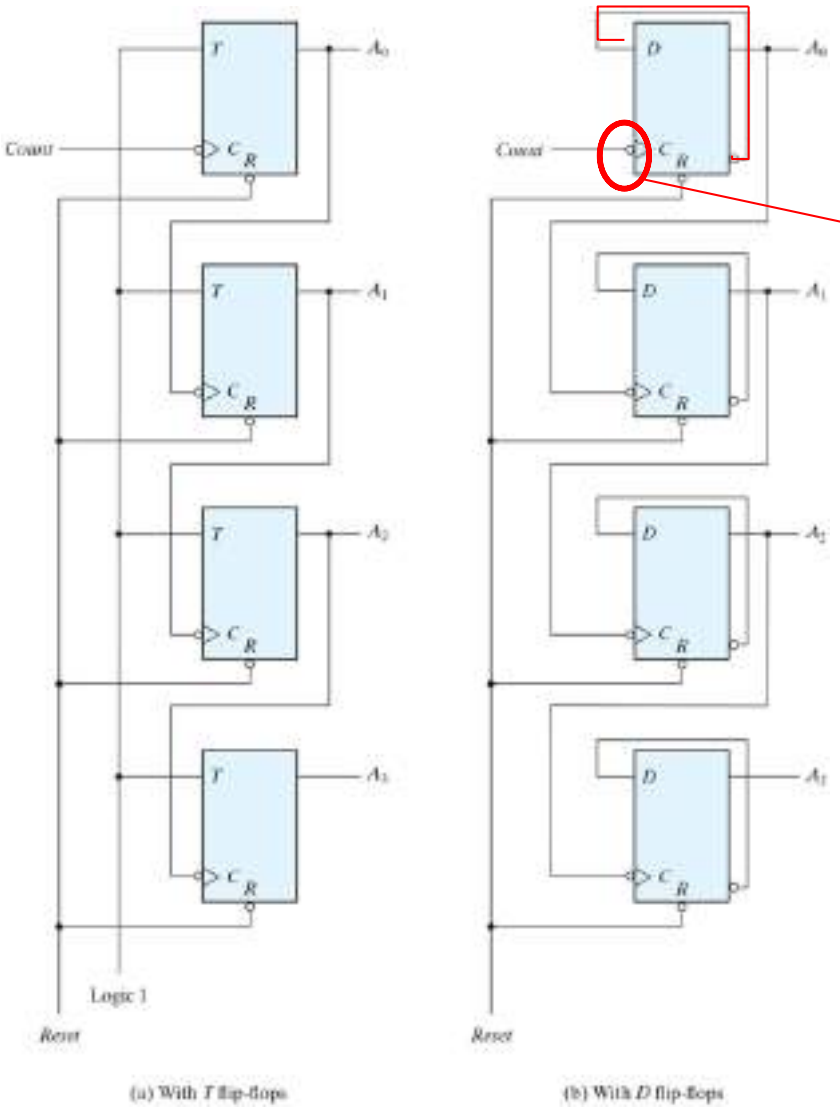
- 시프트 레지스터는 멀리 떨어져 있는 디지털 시스템들의 인터페이스(통신)로 사용됨. 예를 들어, 서로 다른 두 위치에 n비트의 양을 전송한다고 가정하면, 병렬로 n개의 전선을 길게 사용하는 방법이 비경제적일 수 있음. 따라서 하나의 라인을 사용하여 직렬 방식으로 한 비트씩 전송하는 것이 경제적임
- 마이크로프로세서 시스템 설계 시, 내부에서는 n비트 데이터를 병렬로 읽어서 하나의 전선을 통해 외부로 직렬로 데이터를 전송하고, 수신단에서는 시프트 레지스터를 통해 직렬로 들어오는 데이터를 수신함. n비트가 수신되면, 레지스터를 병렬 방식으로 출력함.

6.3 리플 카운터 (Ripple Counter)

- 카운터: 입력 펄스(clock 혹은 외부 입력)에 따라 정해진 순서대로 상태 천이가 진행되는 레지스터. 어떤 사건의 발생 횟수를 세거나 동작 순서를 제어하는 타이밍 신호를 만듦. n비트 2진 카운터는 n개의 플립플롭으로 구성되며 0에서 (2^n-1)까지 카운트함
- 카운터는 리플 카운터와 동기식 카운터 두 종류
  - 리플 카운터: 플립플롭의 출력 천이는 다른 플립플롭을 트리거 하는 소스로 동작. 플립플롭의 클럭 입력은 공통된 클럭 펄스가 아닌 다른 플립플롭의 출력에 의해 천이
  - 동기식 카운터: 플립플롭 클럭 입력은 공통된 클럭으로부터 공급됨

- 2진 리플 카운터
  - 각 플립플롭의 출력이 다음 상위 플립플롭의 C 입력에 연결되어 보수 플립플롭의 직렬 연결로 구성됨
  - 최하위 비트를 나타내는 플립플롭은 카운트 펄스를 인가받음
  - 보수 플립플롭은 J와 K입력이 묶인 JK 플립플롭이나 T 플립플롭, D 입력과 연결된 보수출력과 D 플립플롭을 사용하는 해서 얻음

Figure 6.8
Four-bit binary ripple counter.



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

Table 6.4
Binary Count Sequence.

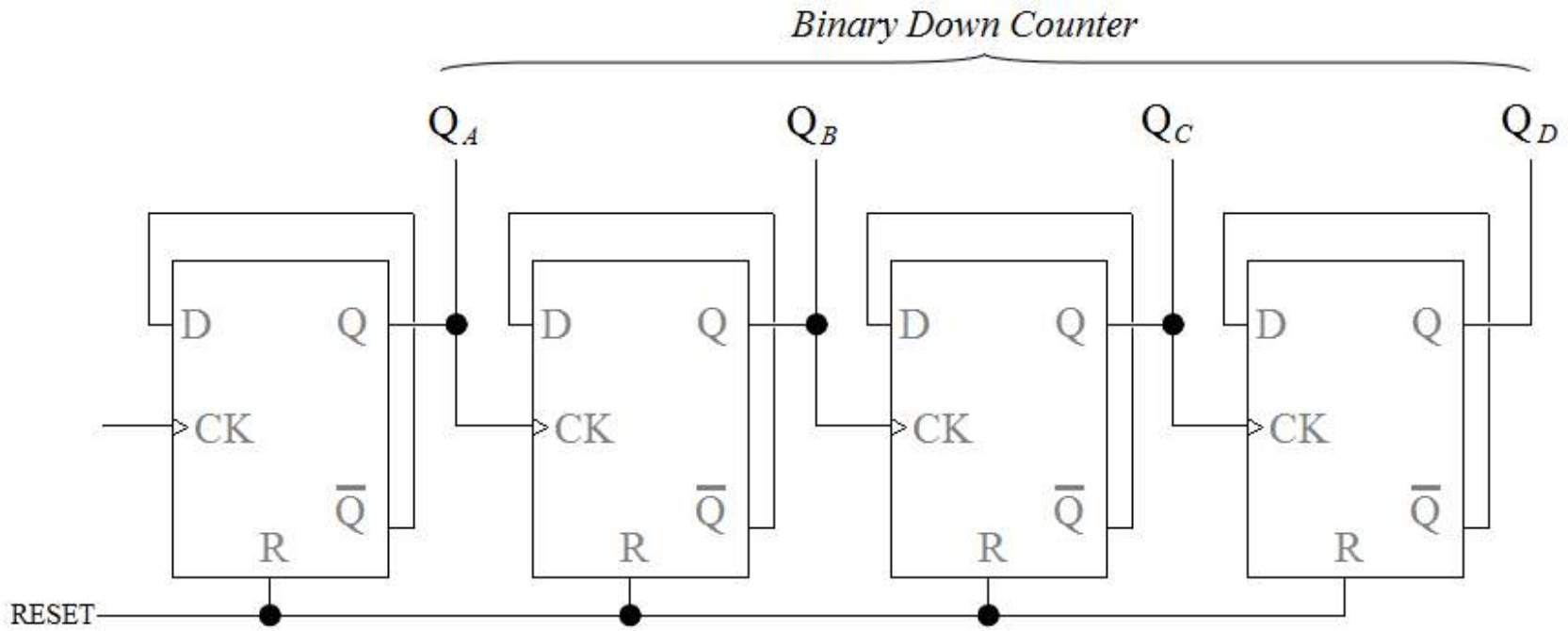
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0

falling edge

순간적으로 한 비트 씩 바뀜

0011 → 0010 → 0000 → 0100

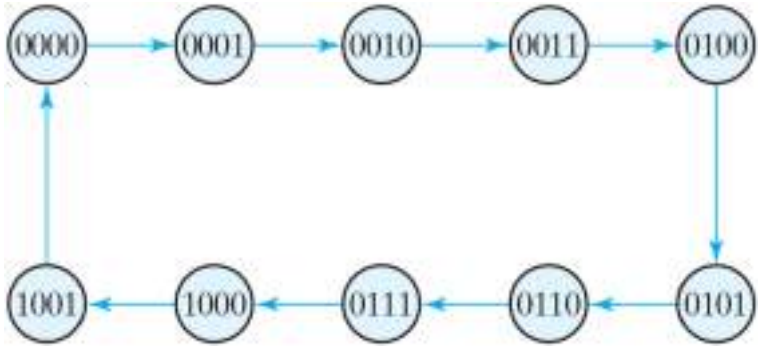
4-bit D-Type Flip-Flop Down Counter



Q_8	Q_4	Q_2	Q_1
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0

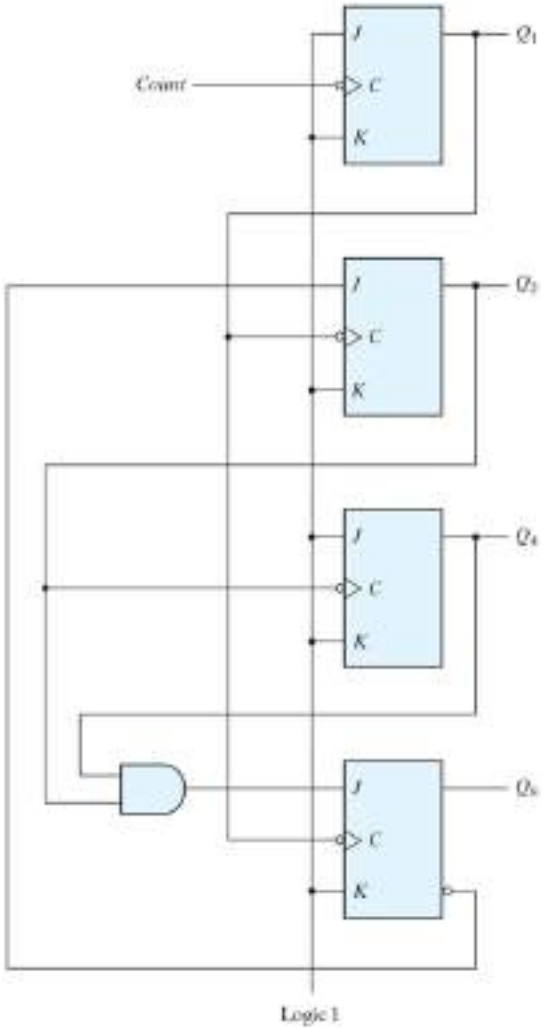
•BCD 리플 카운터

Figure 6.9
State diagram of a decimal BCD counter.



- 4개의 비트 필요 → 4개의 플립플롭 필요하고 1001 후의 상태가 0000이라는 것을 제외하고는 2진 카운터와 유사함
- 그림 6.10은 JK 플립플롭을 이용한 BCD 리플 카운터 논리도임. Count 입력이 1에서 0으로 될 때 J = 1이면 플립플롭이 세팅되고, K = 1이면 클리어(reset)되고, J = K = 1이면 보수와, J = K = 0이면 바뀌지 않음

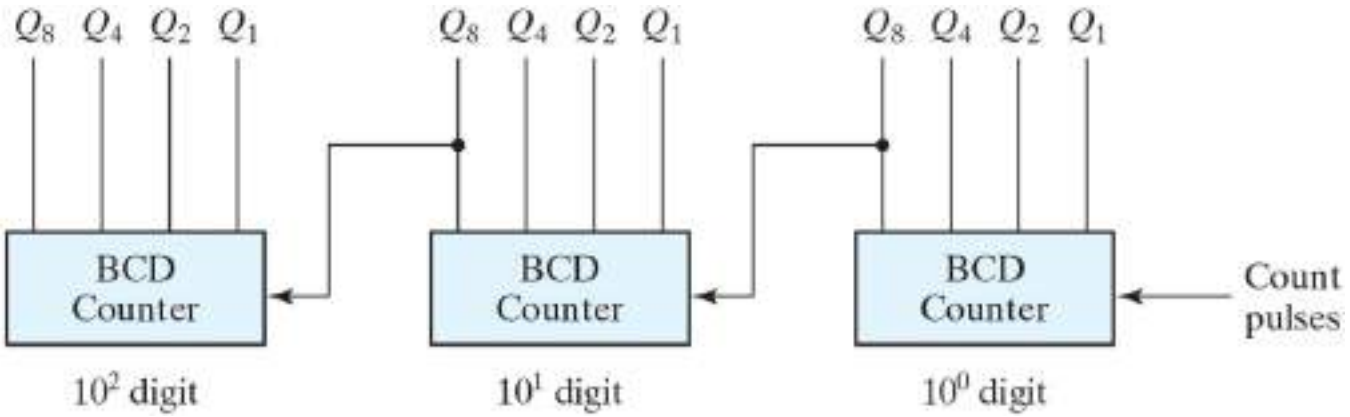
Figure 6.10
BCD ripple counter.



- Q_1은 각 클럭 펄스의 뒤에서 상태가 변하고, Q_2는 $Q_8 = 0$인 한, Q_1이 1에서 0이 될 때 반전되며, Q_8이 1이 되면 Q_2는 0으로 클리어(reset)된 상태임.
- Q_4는 Q_2가 1에서 0으로 될 때마다 반전되는데 Q_8은 Q_2 또는 Q_4가 0인 한, 계속 클리어됨. Q_2와 Q_4가 모두 1이 될 경우, Q_8은 Q_1이 1에서 0으로 될 때 반전됨
- Q_8은 Q_1의 다음 천이에서 클리어됨

Q_8	Q_4	Q_2	Q_1
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
1	0	0	1

Figure 6.11
Block diagram of a three-decade decimal BCD counter.



6.4 동기식 카운터 (Synchronous Counter)

- 동기식 카운터: 클럭 펄스가 모든 플립플롭의 C입력에 가해짐. 즉 공통된 펄스가 모든 플립플롭을 동시에 트리거시킴. 이때 T = 0 이거나 J = K = 0이면 변하지 않고, T = 1이거나 J = K = 1이면 플립플롭은 보수화됨
- 이미 5.8절에서 3비트 2진 카운터 설계 설명함
- 2진 카운터
  - 하위 비트가 모두 1과 같다면 다음 카운트(클럭) 펄스에서 플립플롭이 보수화됨 → 그림 6.12
  - 2진 하향 카운터는 2진 상태로 역순으로 1111부터 0000까지 감소했다가 다시 1111로 돌아감. 최하위 플립플롭은 매 클럭 펄스마다 보수화되며 다른 플립플롭은 보다 낮은 모든 하위의 비트가 0일 때 다음 펄스로 보수화됨 (0100 → 0011). 예를 들어, 두 번째 하위 비트는 첫 번째 비트가 0일 때 보수화되고, 세 번째 하위 비트는 첫 번째와 두 번째 하위 비트가 모두 0일 때 보수화됨

Figure 6.12
Four-bit synchronous binary counter.

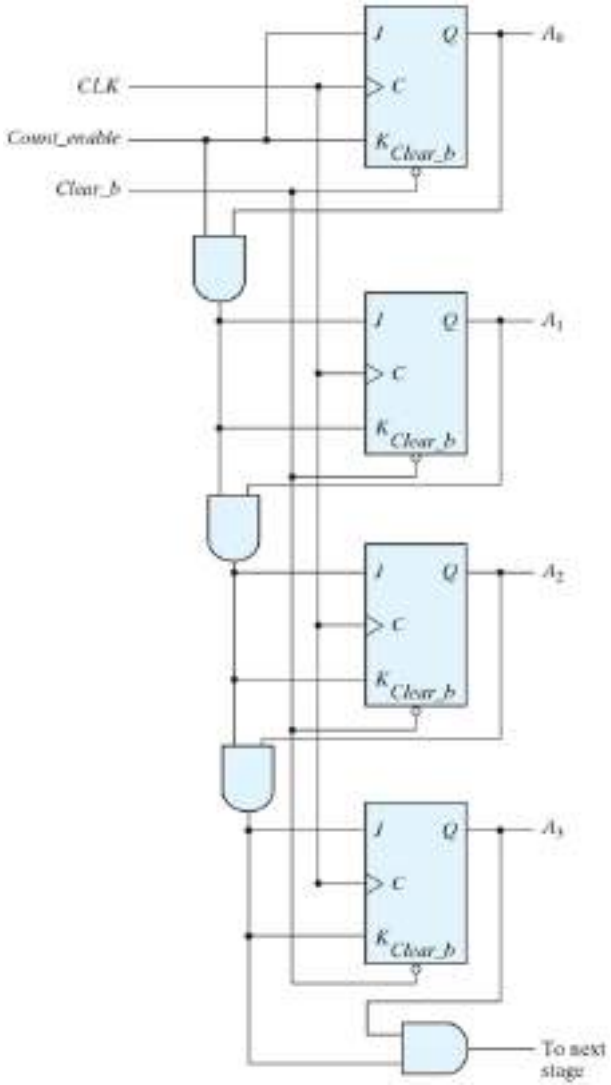
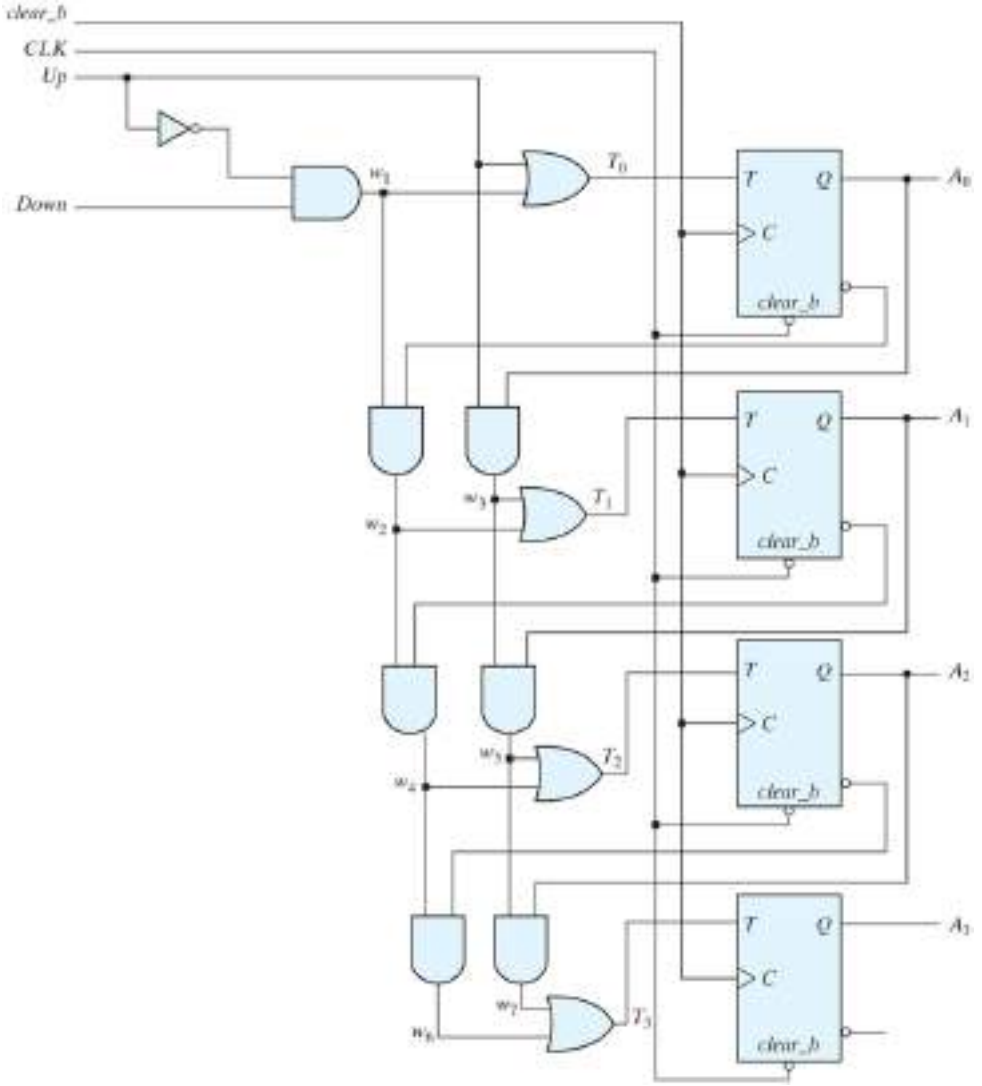


Figure 6.13
Four-bit up–down binary counter.



Pearson

- BCD 카운터
  - 2진으로 코드화된 10진수를 0000에서 1001까지 세고, 다시 0000으로 되돌아 감. 5.8절의 순차 회로 설계 과정을 따름

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	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

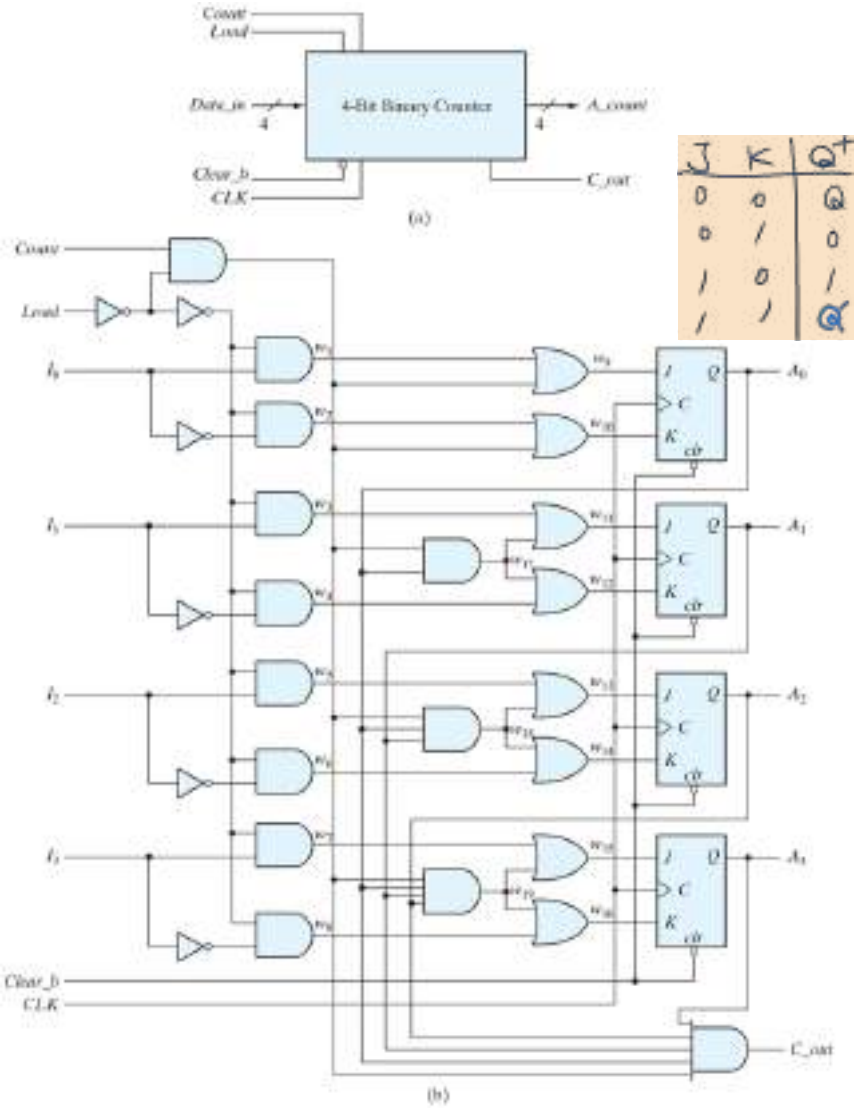
  - 카노맵으로 간략화하고, 최소항 10에서 15까지는 don't care 항으로 취급함

$$T_{Q1} = 1,\ T_{Q2} = Q_8'Q_1,\ T_{Q4} = Q_2Q_1,\ T_{Q8} = Q_8Q_1 + Q_4Q_2Q_1$$

$$y = Q_8Q_1$$

- 병렬 로드를 가진 2진 카운터
  - 카운터 동작에 앞서 초기값(2진수)을 카운터에 전송하기 위해 병렬 로드 능력을 사용함

Figure 6.14
Four-bit binary counter with parallel load.

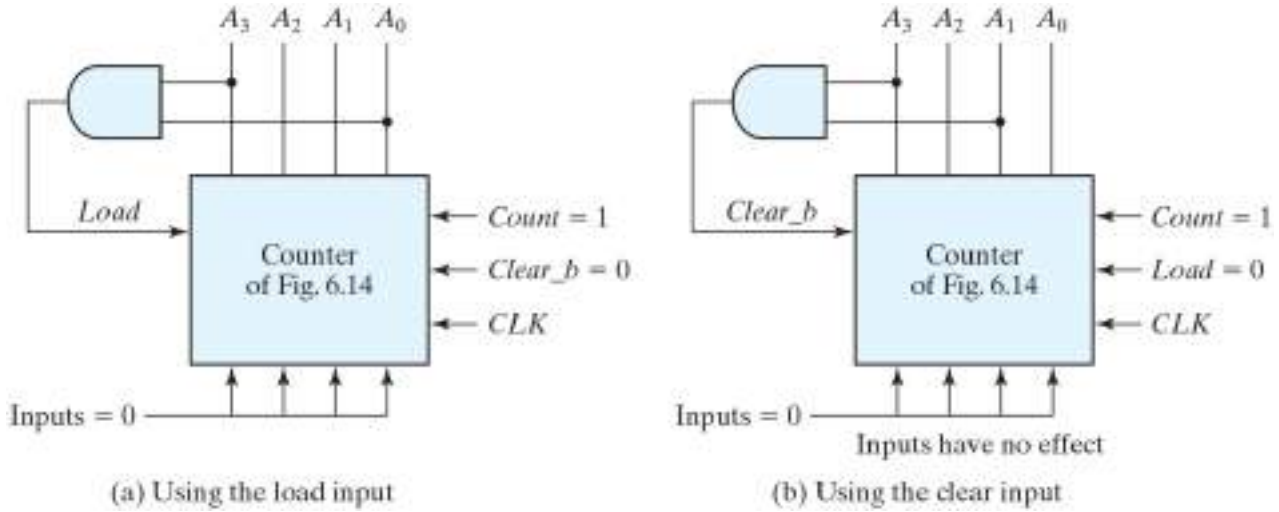


  - Load가 1일 때, 카운트 동작 멈추고 4개의 데이터 입력 (I_0~I_3)가 플립플롭에 전달됨 (A_0~A_3)
  - 캐리 출력 단자(C_out)는 카운트 입력이 enable 상태에서 모든 플립플롭이 1이면 1이 되고, 다음 높은 상위 비트를 유지하고 있는 플립플롭을 보수화하여 4비트 이상으로 확장할 때 유용

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

- 병렬 로드를 가진 2진 카운터를 이용해서 임의의 원하는 카운트 순서를 만들기 위해 사용됨 (아래 예제는 BCD 카운터)

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



  - 왼쪽 그림에서는 Clear_b와 Count가 1로 세팅되어 모든 시간에서 카운터가 동작하며, AND 게이트의 출력이 0인 동안에 상승 클럭이 카운트를 하나씩 증가시킴. 출력이 1001이 되면 AND 게이트가 1이 되어 입력 0000을 로딩함
  - 오른쪽 그림에서는 1010이 발생하는 순간 클리어 되어 0000이 됨 (스파이크 발생)

6.5 다른 종류의 카운터

- 카운터는 임의의 상태 변화 순서가 가능하도록 설계할 수 있음
- 모듈로-N 카운터(modulo-N counter)로 잘 알려진 N 분주 카운터(divide-by-N counter)는 N가지 상태의 순서를 반복하는 카운터임

- 사용되지 않는 상태를 가진 카운터
  - n개의 플립플롭을 가진 회로는 2^n개의 2진 상태를 가지지만, 모든 상태가 필요하지 않을 수 있음. 사용되지 않는 상태는 상태표에 나타내지 않고 don't care 조건이 되거나 특정한 다음 상태에 인가됨
  - 사용되지 않는 상태를 don't care 조건으로 간주하고 순차 회로를 설계한다면, 반드시 사용되지 않는 상태의 다음 상태를 분석해야 함

- 아래 표의 카운터는 6개 상태의 순서를 반복함. 연속된 2진수가 아니고 011과 111의 두 상태는 포함되지 않음.

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

  - 3개의 JK 플립플롭을 이용해서 설계하면,

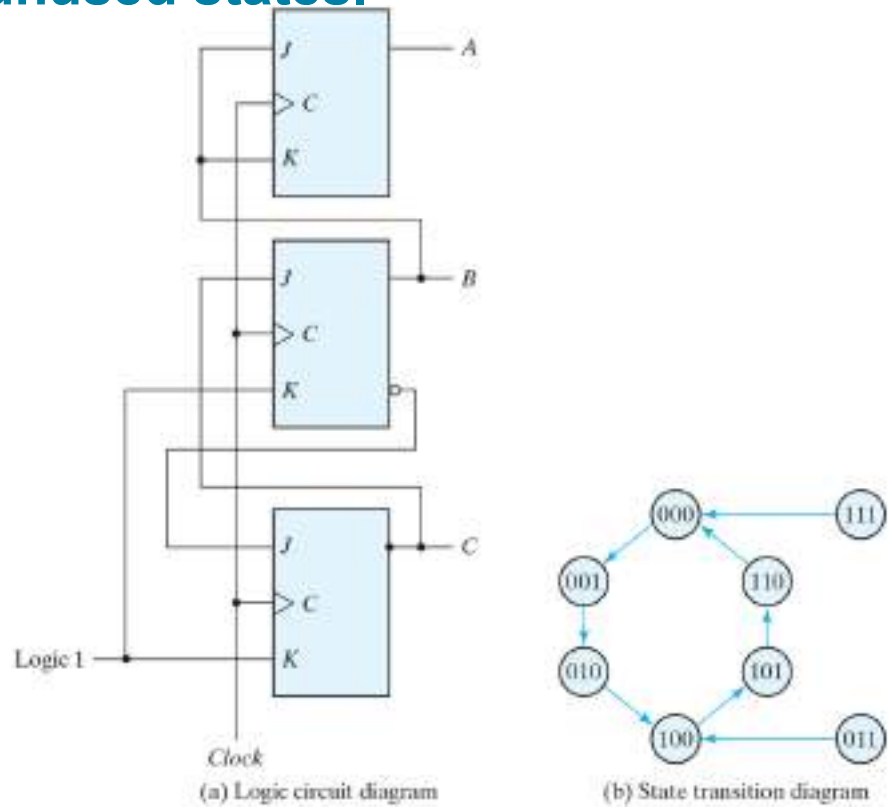
$J_A = B$ $\quad K_A = B$

$J_B = C$ $\quad K_B = 1$

$J_C = B'$ $\quad K_B = 1$

  - 결과적으로 그림 6.16 (a)처럼 논리도가 완성됨
  - 이 경우, 정의되지 않은 111과 011의 다음 상태에 대해서 분석 필요

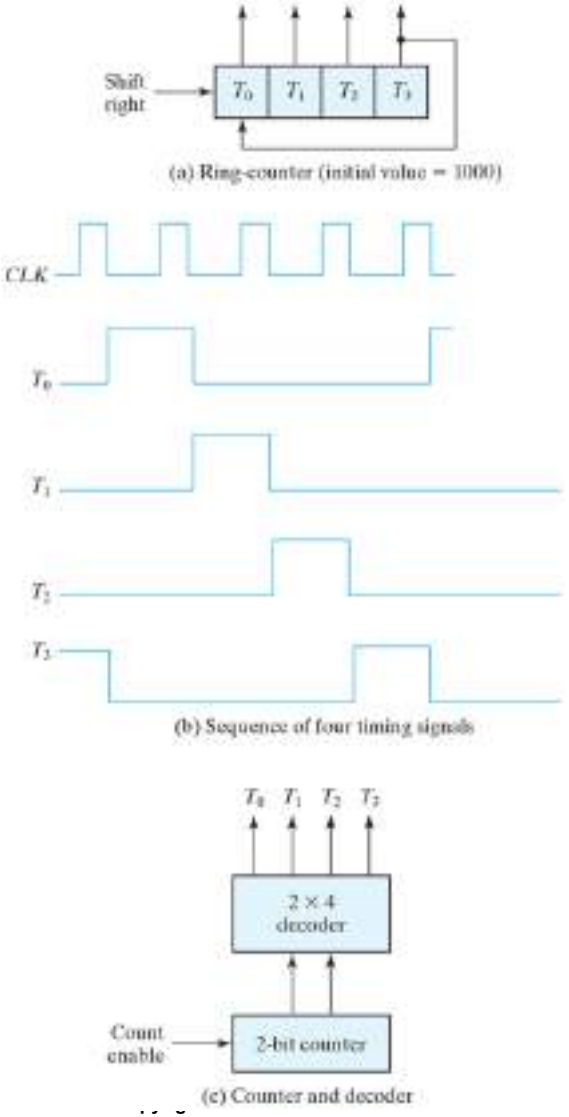
Figure 6.16
Counter with unused states.



- 회로의 상태가 011이 된다면 B = 1이 되어 다음 클럭 에지에서 A는 보수화되고, C는 0으로 클리어됨 → 100
- 회로의 상태가 111이 된다면 B = 1이 되어 다음 클럭 에지에서 A는 보수화되고, C는 0으로 클리어 되고, B는 보수화됨 → 000

- 링 카운터
  - 임의의 시간에 하나의 플립플롭만 세팅되고, 나머지 플립플롭은 모두 클리어된 상태로 동작하는 순환식 시프트 레지스터

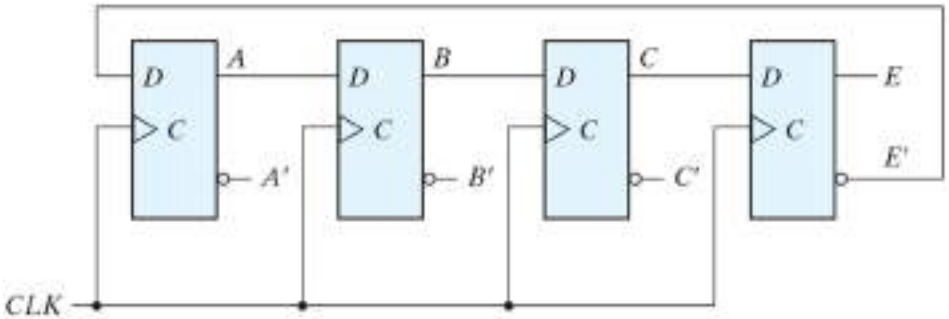
Figure 6.17
Generation of timing signals.



  - 오른쪽 그림 (c)처럼 디코더와 카운터를 조합해서 링 카운터를 설계 가능
  - 2^n개의 타이밍 신호를 발생시키려면, 2^n개의 플립플롭으로 구성된 시프트 레지스터나 n-to-2^n 라인 디코더와 함께 n비트의 레지스터가 필요
  - 예를 들어 16개의 타이밍 신호는 링 카운터로서 연결된 16비트의 시프트 레지스터로 혹은 하나의 4비트 카운터와 하나의 4-to-16 라인 디코더를 사용하여 발생 가능

- 존슨 카운터
  - K 비트 링 카운터는 K개의 다른 상태를 공급하기 위해 단일 비트가 플립플롭 사이를 순환함. switch-tail 링 카운터로 연결하면 2K개의 상태로 동작함

Figure 6.18
Construction of a Johnson counter.



(a) Four-stage switch-tail ring 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

- 존슨 카운터
  - 2K개의 상태에 상응하는 타이밍 신호는 카운터가 모두 0이거나 1이면 양쪽 끝 2개의 플립플롭 출력을 디코딩하고 그 외의 상태들은 서로 인접한 1, 0 또는 0, 1의 형태로 디코딩함 (그림 6.18 (c)참조)
  - 임의의 개수의 타이밍 시퀀스를 구성할 때, 플립플롭의 수는 타이밍 신호 갯수의 절반이 되고, 디코딩 게이트의 수는 타이밍 신호의 갯수와 같고, 2-입력 게이트만을 필요로 함