

Microprocessor (W3)

- Logic Circuit Review -

Dong Min Kim
Department of IoT
Soonchunhyang University
dmk@sch.ac.kr

Contents

03 조합 논리 회로

04 순서 논리 회로

03 조합 논리 회로

1 데이터 형태에 따른 분류

□ 조합 논리 회로의 개요

- 조합 논리 회로(combational logic circuit)는 현재 입력 값으로 출력이 결정되는 회로

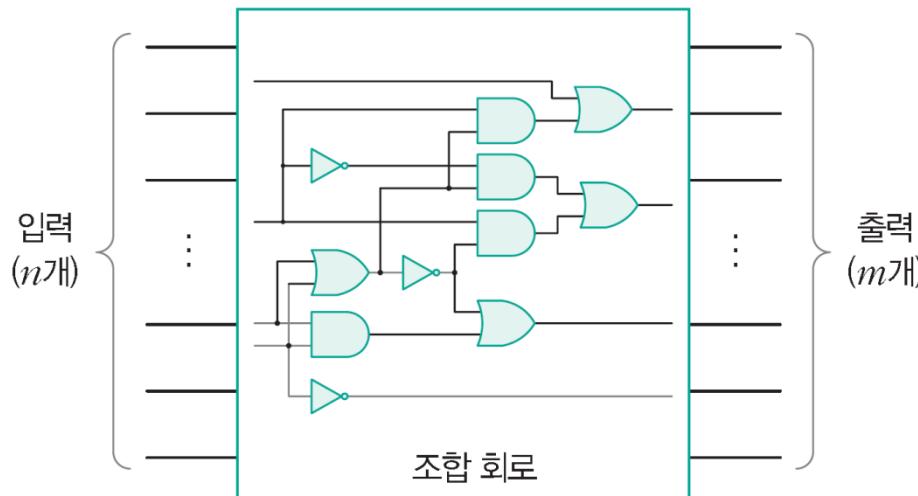


그림 3-32 조합 논리 회로의 개념도

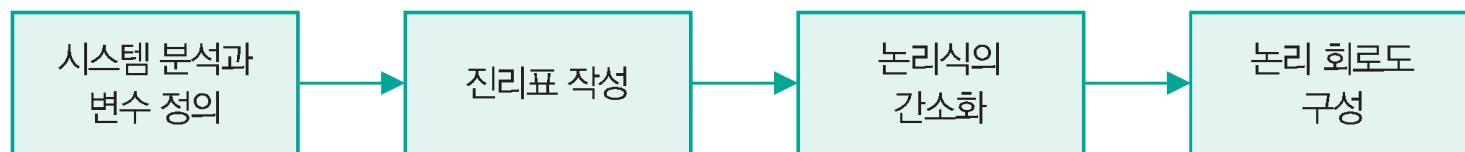


그림 3-33 조합 논리 회로의 설계 과정

03 조합 논리 회로

2 조합 논리 회로의 종류

□ 반가산기

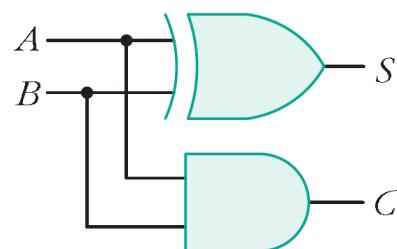
- 반가산기(Half-Adder, HA)는 1자리 2진수 2개를 입력하여 합(S)과 캐리(Carry, C)를 출력하는 조합 논리 회로

$$\begin{array}{r} A & 0 & 0 & 1 & 1 \\ + B & + 0 & + 1 & + 0 & + 1 \\ \hline C & S & 0 & 1 & 0 \\ & & 0 & 1 & 1 \end{array}$$

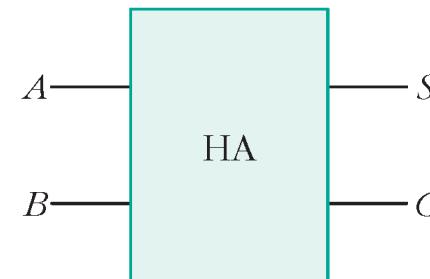
입력		출력	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{A}B + A\overline{B} = A \oplus B$$
$$C = AB$$

(a) 진리표와 논리식



(b) 논리 회로



(c) 논리 기호

그림 3-34 반가산기

03 조합 논리 회로

□ 전가산기

- 전가산기(Full-Adder, FA)는 2진수 입력 A, B 와 아랫자리에서 올라온 캐리 C_i 를 포함하여 1자리 2진수 3개를 더하는 조합 논리 회로

입력			출력	
A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a) 진리표

$$\begin{aligned}S &= \overline{ABC}_i + \overline{ABC}_i + A\overline{BC}_i + ABC_i \\&= \overline{A}(\overline{BC}_i + BC_i) + A(\overline{BC}_i + BC_i) \\&= \overline{A}(B \oplus C_i) + A(\overline{B} \oplus \overline{C}_i) \\&= A \oplus (B \oplus C_i) = (A \oplus B) \oplus C_i\end{aligned}$$

$$\begin{aligned}C_o &= \overline{ABC}_i + A\overline{BC}_i + A\overline{BC}_i + ABC_i \\&= C_i(\overline{AB} + A\overline{B}) + AB(\overline{C}_i + C_i) \\&= C_i(A \oplus B) + AB\end{aligned}$$

(b) 논리식의 간소화

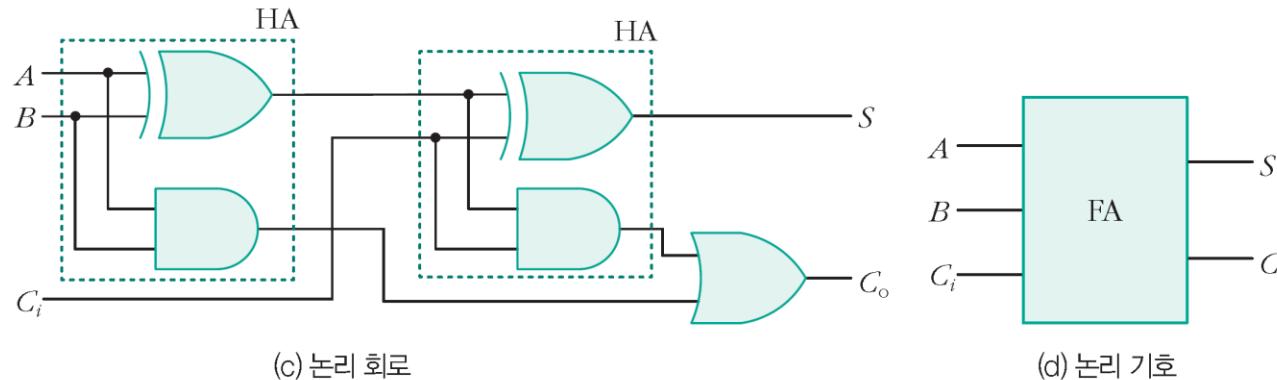


그림 3-35 전가산기

03 조합 논리 회로

□ 병렬 가감산기

- **병렬 가산기**(parallel-adder) : 전가산기 여러 개를 병렬로 연결하여 만든 2비트 이상의 가산기

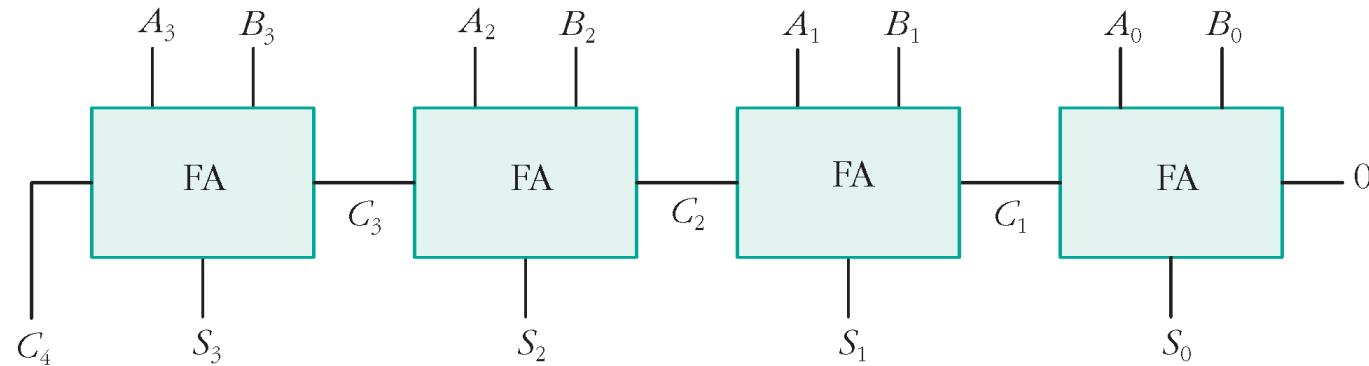


그림 3-38 전가산기를 이용한 병렬 가산기

캐리 예측 가산기(carry lookahead adder)

병렬 가산기는 속도가 매우 느리다. 그 원인은 아랫 단에서 윗단으로 전달되는 캐리 때문이다. 비트가 늘어날수록 지연이 더욱 심해진다. 이를 해결하기 위해 **캐리 예측 가산기**를 사용한다.

03 조합 논리 회로

- **병렬 가감산기**(parallel-adder/subtracter) : 병렬 가산기의 B 입력을 부호 $S(\text{sign})$ 와 XOR하여 전가산기의 입력으로 사용함으로써 덧셈과 뺄셈이 모두 가능한 회로

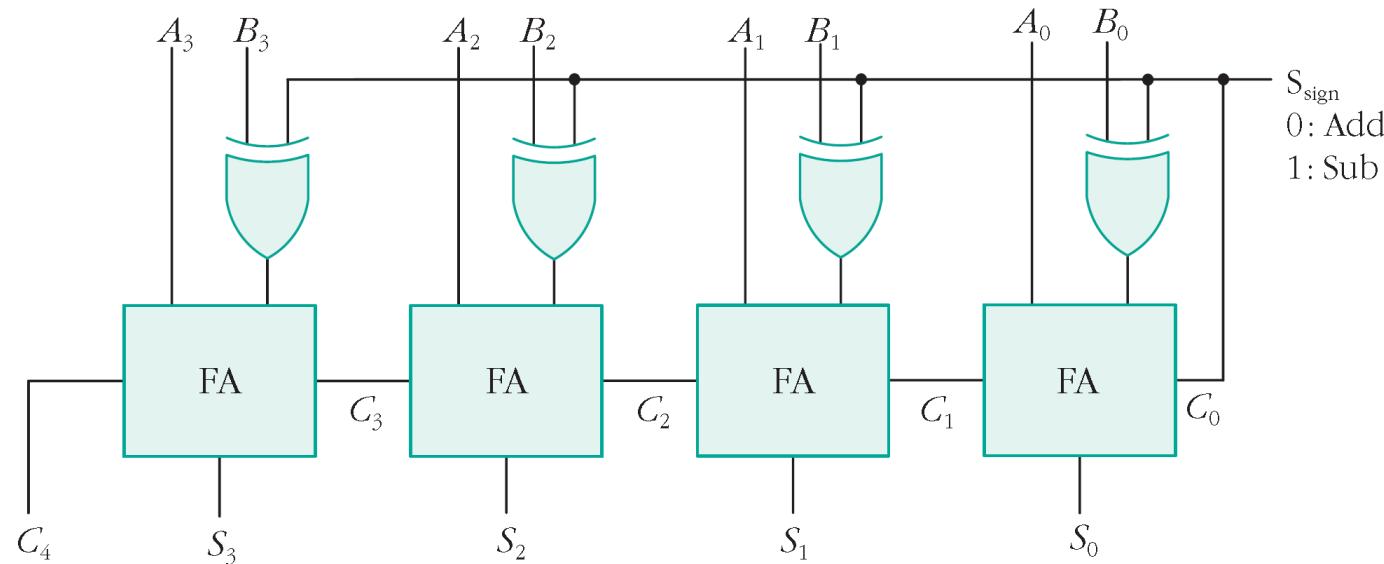


그림 3-39 병렬 가감산기

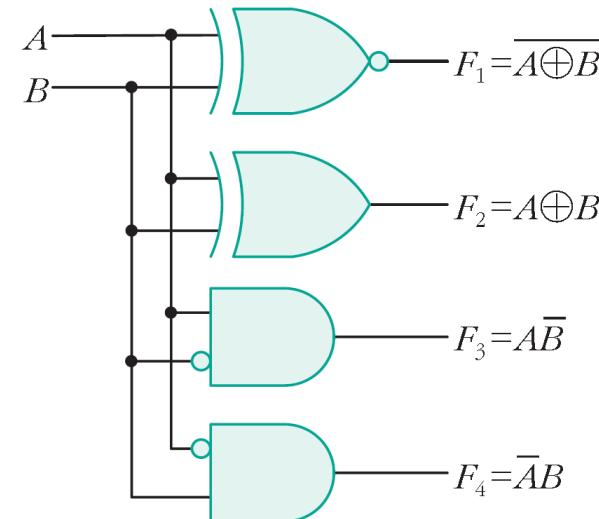
03 조합 논리 회로

□ 비교기

- 2진 비교기(comparator)는 두 2진수 값의 크기를 비교하는 회로다.

입력		출력			
A	B	$A=B$ F_1	$A \neq B$ F_2	$A > B$ F_3	$A < B$ F_4
0	0	1	0	0	0
0	1	0	1	0	1
1	0	0	1	1	0
1	1	1	0	0	0

(a) 진리표



(b) 논리 회로

그림 3-40 1비트 비교기

03 조합 논리 회로

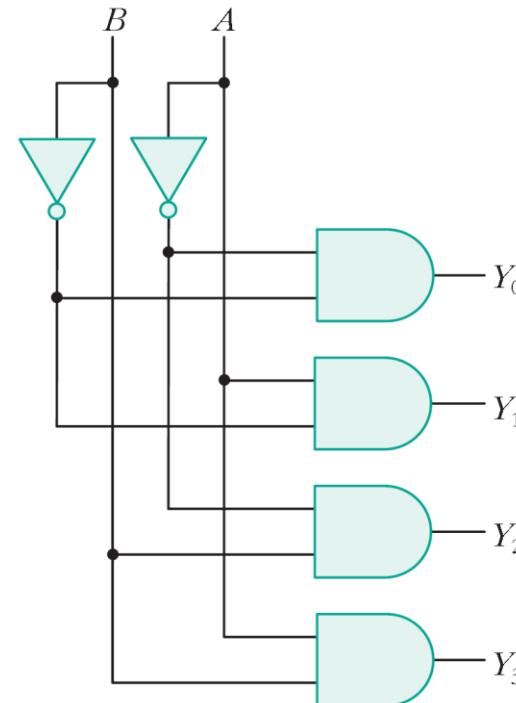
□ 디코더

- **디코더**(decoder) : 입력선에 나타나는 n 비트의 2진 코드를 최대 2^n 개의 서로 다른 정보로 바꿔주는 조합논리회로

입력		출력			
B	A	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$$Y_0 = \overline{BA}, Y_1 = \overline{B}A, Y_2 = B\overline{A}, Y_3 = BA$$

(a) 진리표와 논리식



(b) 논리 회로

그림 3-41 2×4 디코더

03 조합 논리 회로

❖ 실제 디코더 회로

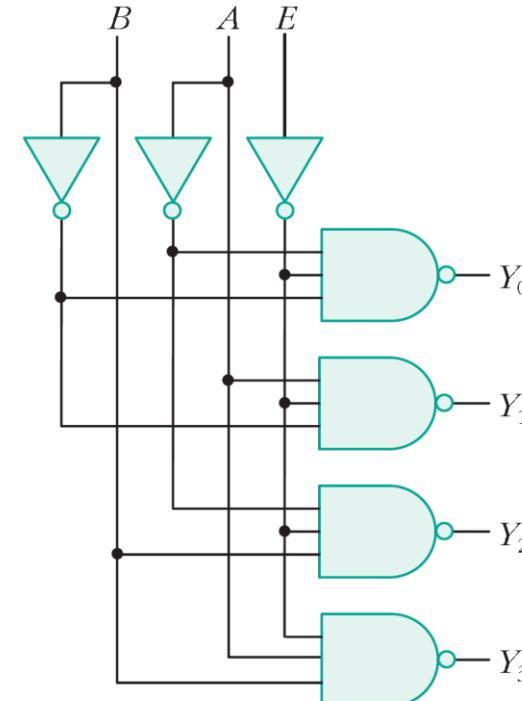
- 실제 IC들은 AND 게이트가 아닌 NAND 게이트로 구성되어 출력은 그림과 같이 반대로 된다.
- 또 대부분의 디코더 IC는 인에이블(enable) 입력이 있어 회로를 제어한다.

입력			출력			
\bar{E}	B	A	Y_3	Y_2	Y_1	Y_0
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

$$Y_0 = \overline{\bar{E}\bar{B}\bar{A}}, Y_1 = \overline{\bar{E}\bar{B}A}, Y_2 = \overline{\bar{E}BA}, Y_3 = \overline{\bar{E}BA}$$

(a) 진리표와 논리식

그림 3-42 인에이블이 있는 2×4 NAND 디코더



(b) 논리 회로

03 조합 논리 회로

❖ 3×8 디코더

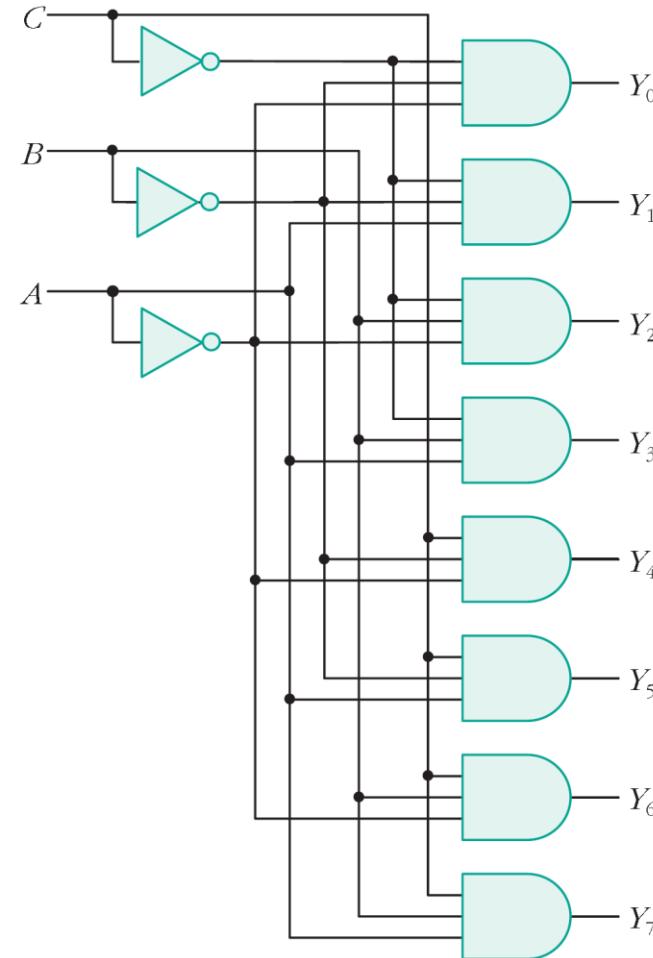
- 3개의 입력에 따라서 8개의 출력 중 하나가 선택

입력			출력							
C	B	A	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$$Y_0 = \overline{CBA}, Y_1 = \overline{C}\overline{B}A, Y_2 = \overline{C}\overline{B}\overline{A}, Y_3 = \overline{C}BA$$

$$Y_4 = C\overline{B}\overline{A}, Y_5 = C\overline{B}A, Y_6 = C\overline{B}\overline{A}, Y_7 = CBA$$

(a) 진리표와 논리식



(b) 논리 회로

그림 3-43 3×8 디코더 회로

03 조합 논리 회로

❖ 2개의 3×8 디코더로 4×16 디코더를 구성

D=0	상위 디코더만 enable되어 출력은 $Y_0 \sim Y_7$ 중의 하나가 1로 되고, 하위 디코더 출력들은 모두 0이 된다.
D=1	하위 디코더만 enable 되어 출력은 $Y_8 \sim Y_{15}$ 중의 하나가 1로 되고, 상위 디코더 출력들은 모두 0이 된다.

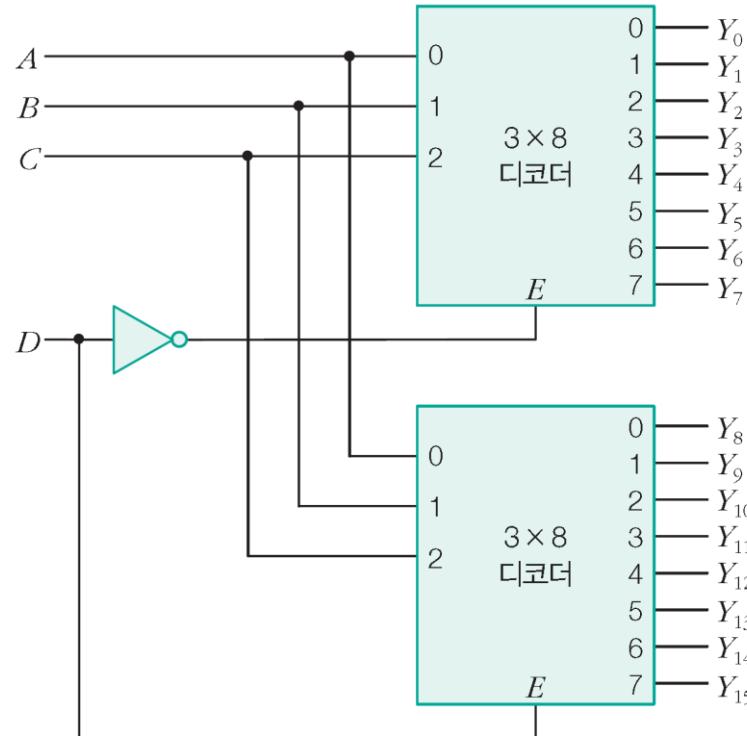


그림 3-44 3×8 디코더 2개를 이용한 4×16 디코더

03 조합 논리 회로

□ 인코더

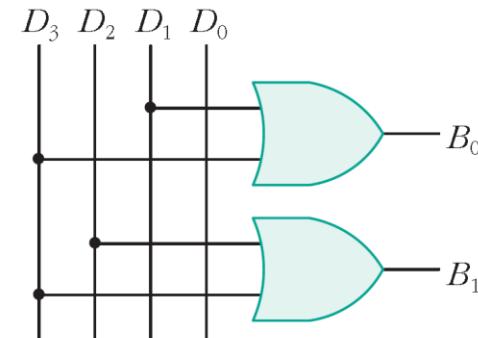
- 부호기라고도 하는 인코더(encoder)는 디코더의 반대 기능을 수행하는 조합 논리 회로로, 2^n 개를 입력받아 n 개를 출력한다.
- 인코더는 2^n 개 중 활성화된 1비트 입력 신호를 받아 그 숫자에 해당하는 n 비트 2진 정보를 출력한다.

입력				출력	
D_3	D_2	D_1	D_0	B_1	B_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$$B_1 = D_2 + D_3, \quad B_0 = D_1 + D_3$$

(a) 진리표와 논리식

그림 3-45 4×2 인코더



(b) 논리 회로

03 조합 논리 회로

❖ 8×3 인코더

- 8($=2^3$)개의 입력과 3개의 출력을 가지며, 입력의 신호에 따라 3개의 2진 조합으로 출력한다.

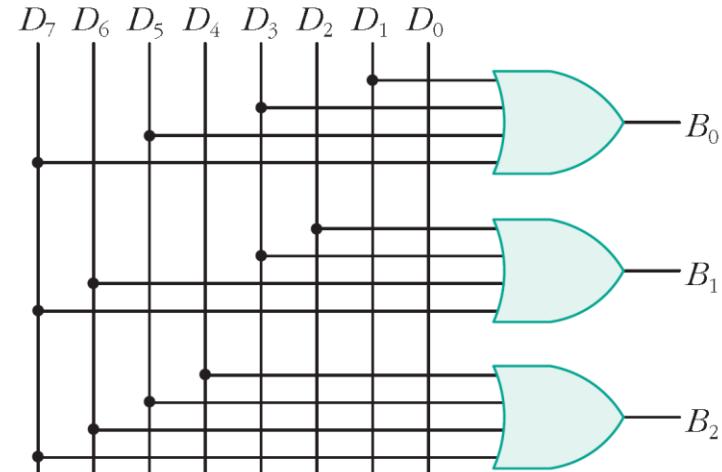
입력								출력		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	B_2	B_1	B_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1	1	1

$$B_2 = D_4 + D_5 + D_6 + D_7, \quad B_1 = D_2 + D_3 + D_6 + D_7$$

$$B_0 = D_1 + D_3 + D_5 + D_7$$

(a) 진리표와 논리식

그림 3-46 8×3 인코더



(b) 논리 회로

03 조합 논리 회로

□ 멀티플렉서

- **멀티플렉서**(multiplexer, MUX)는 여러 개의 입력선들 중에서 하나를 선택하여 출력선에 연결하는 조합논리회로이다. 선택선들의 값에 따라서 특별한 입력선이 선택된다.
- 멀티플렉서는 많은 입력들 중 하나를 선택하여 선택된 입력선의 2진 정보를 출력선에 넘겨주기 때문에 **데이터 선택기**(data selector)라 부르기도 한다.
- **디멀티플렉서**(demultiplexer, DEMUX)는 정보를 한 선으로 받아 2^n 개의 가능한 출력 선들 중 하나를 선택하여, 받은 정보를 전송하는 회로다.

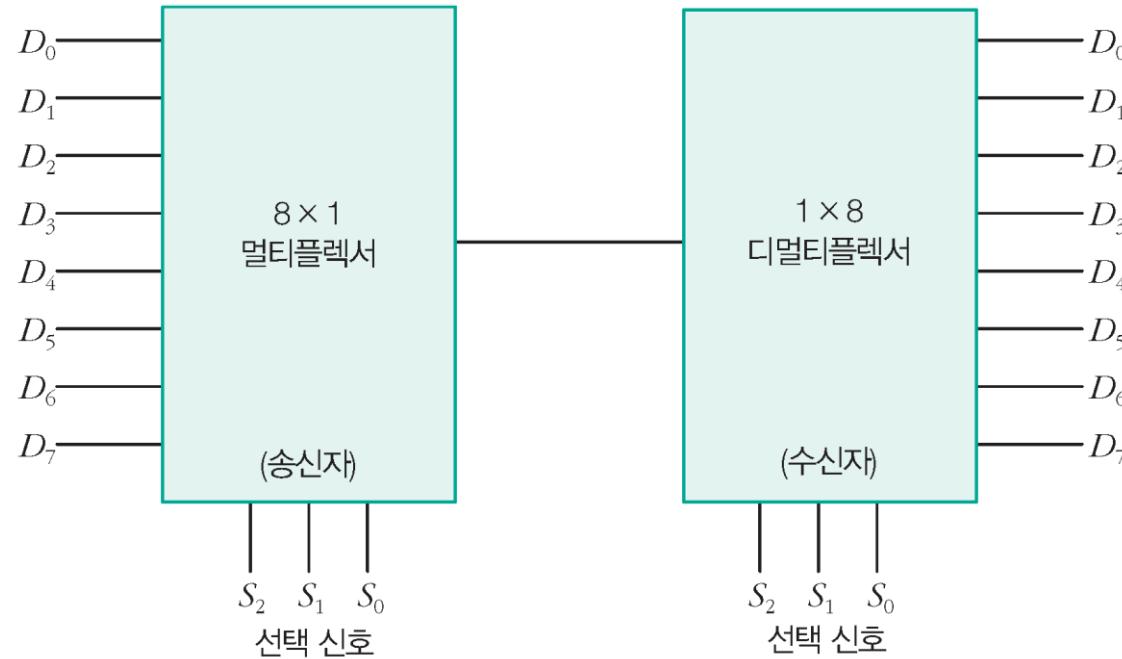


그림 3-47 멀티플렉서와 디멀티플렉서의 역할

03 조합 논리 회로

❖ 2×1 멀티플렉서

- 2($=2^1$)개의 입력중의 하나를 선택선 S 에 입력된 값에 따라서 출력으로 보내주는 조합회로

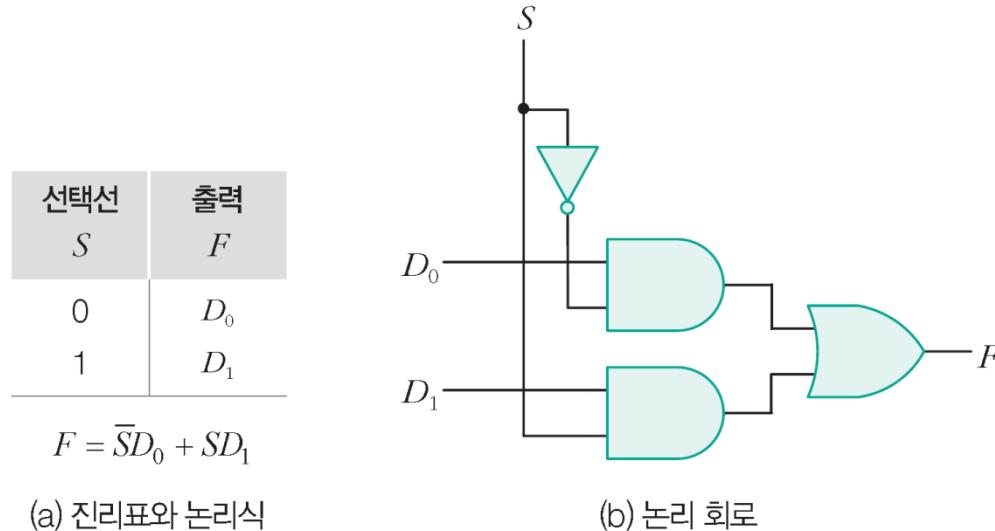


그림 3-48 2×1 멀티플렉서

03 조합 논리 회로

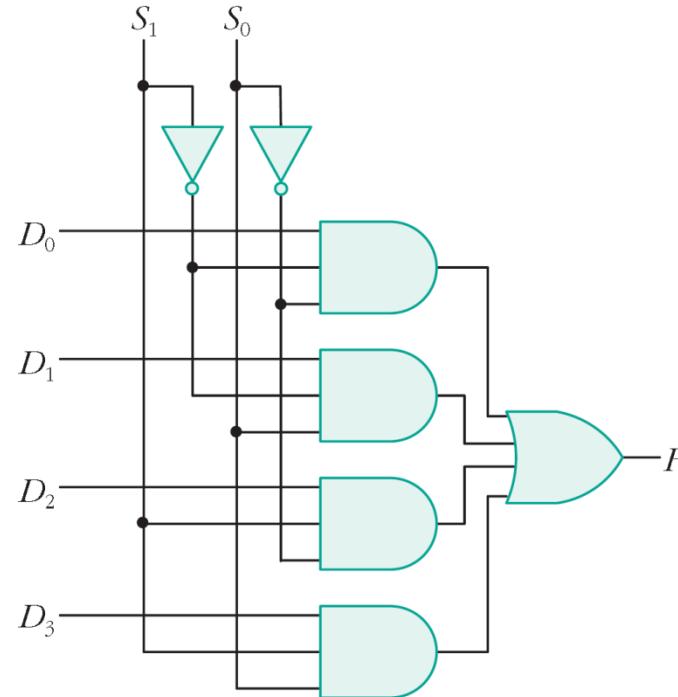
❖ 4×1 멀티플렉서

- 4($=2^2$)개의 입력중의 하나를 선택선 S_1 과 S_0 에 입력된 값에 따라서 출력으로 보내주는 조합회로

선택선		출력
S_1	S_0	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$F = \overline{S}_1 \overline{S}_0 D_0 + \overline{S}_1 S_0 D_1 + S_1 \overline{S}_0 D_2 + S_1 S_0 D_3$$

(a) 진리표와 논리식



(b) 논리 회로

그림 3-49 4×1 멀티플렉서

03 조합 논리 회로

❖ 4×1 멀티플렉서 응용

- 4×1 멀티플렉서를 이용하여 두 입력 A, B 에 대해 AND, OR, XOR, NOT 논리 연산을 수행하는 하드웨어 모듈

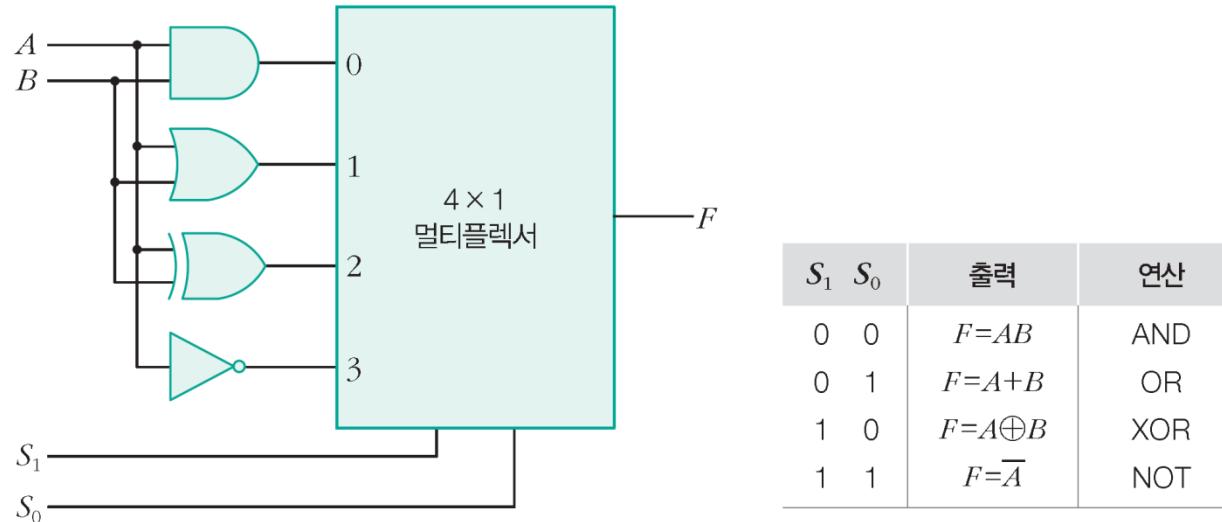


그림 3-50 4×1 멀티플렉서를 이용한 논리 연산 하드웨어 모듈

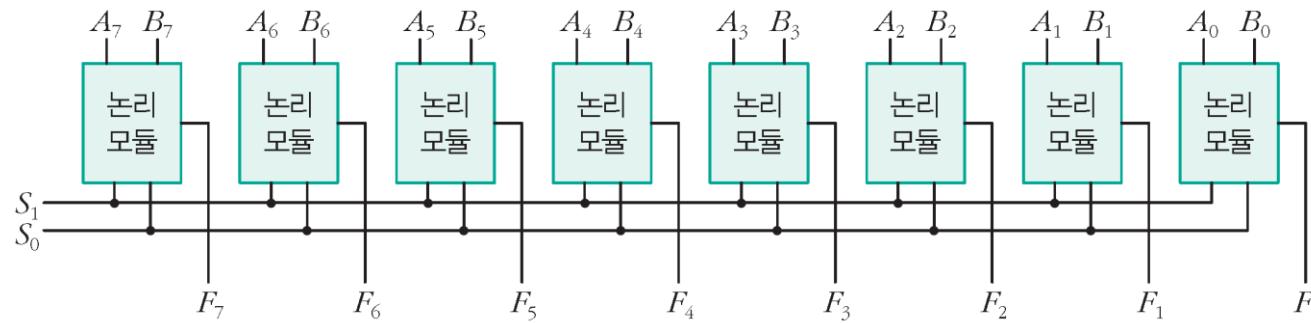


그림 3-51 8비트 논리 연산 장치 구성 예

03 조합 논리 회로

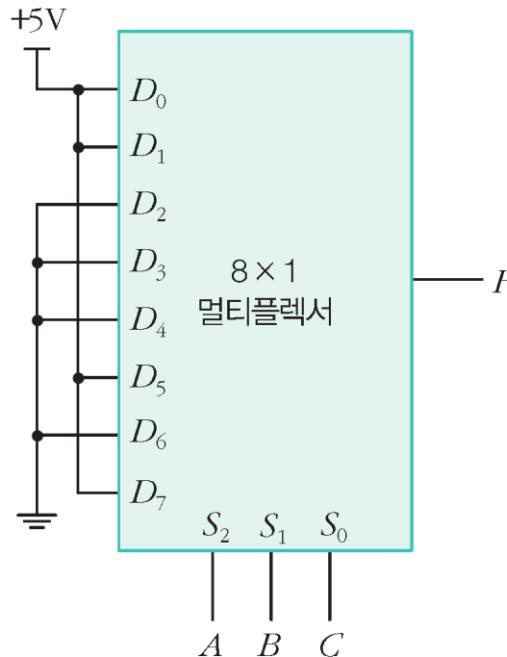
❖ 멀티플렉서를 이용한 조합회로 구현

- $F(A, B, C) = \sum m(0, 1, 5, 7)$ 를 8×1 멀티플렉서로 구현하는 경우
☞ 3개의 선택선을 입력 A, B, C 로 사용

입력			출력
A	B	C	F
0	0	0	$1(D_0)$
0	0	1	$1(D_1)$
0	1	0	$0(D_2)$
0	1	1	$0(D_3)$
1	0	0	$0(D_4)$
1	0	1	$1(D_5)$
1	1	0	$0(D_6)$
1	1	1	$1(D_7)$

$$F(A, B, C) = \sum m(0, 1, 5, 7)$$

(a) 진리표와 논리식



(b) 회로도

그림 3-52 8×1 멀티플렉서를 이용한 회로

03 조합 논리 회로

❖ 멀티플렉서를 이용한 조합회로 구현(계속)

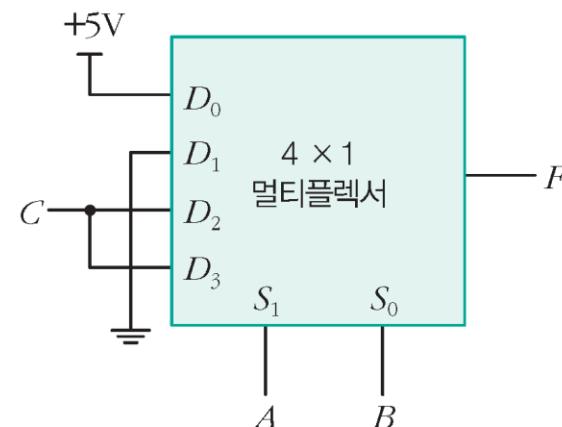
- $F(A, B, C) = \sum m(0, 1, 5, 7)$ 를 4×1 멀티플렉서로 구현하는 경우

☞ A, B는 선택선으로 C는 D_0, D_1, D_2, D_3 을 조합하여 사용

입력		출력	
A	B	C	F
0	0	0	$D_0 = 1$
		1	1
0	1	0	$D_1 = 0$
		1	0
1	0	0	$D_2 = C$
		1	1
1	1	0	$D_3 = C$
		1	1

$$F(A, B, C) = \sum m(0, 1, 5, 7)$$

(a) 진리표와 논리식



(b) 회로도

그림 3-53 4×1 멀티플렉서를 이용한 회로

03 조합 논리 회로

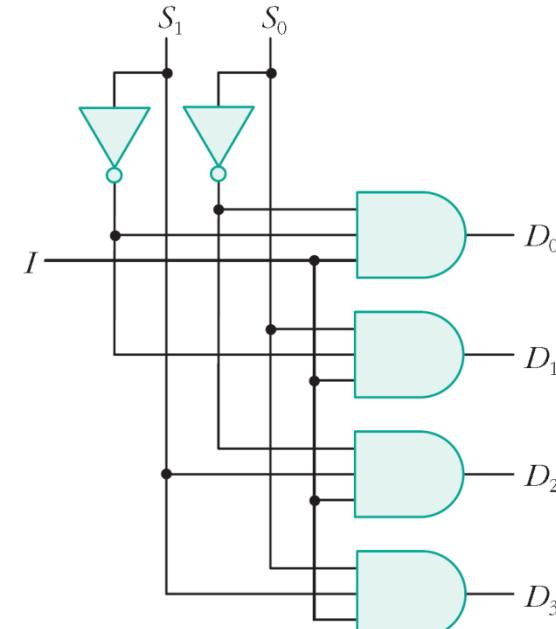
□ 디멀티플렉서

- **디멀티플렉서**(demultiplexer)는 하나의 입력선에 데이터를 입력하면 선택선 n 개로 2^n 개 중 하나를 출력하는 조합 논리 회로다. **데이터 분배기**라고도 한다.
- 1×4 디멀티플렉서는 선택선(S_1, S_0) 2개로 출력(D_3, D_2, D_1, D_0) 4개 중 하나를 선택해 입력(I)을 연결한다.

선택선		출력			
S_1	S_0	D_3	D_2	D_1	D_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

$$D_0 = \overline{S}_1 \overline{S}_0 I, D_1 = \overline{S}_1 S_0 I, D_2 = S_1 \overline{S}_0 I, D_3 = S_1 S_0 I$$

(a) 진리표와 논리식



(b) 논리 회로

그림 3-54 1×4 디멀티플렉서

03 조합 논리 회로

□ 코드 변환기 (2진 코드 → 그레이 코드 변환)

2진 코드(입력) $B_3\ B_2\ B_1\ B_0$	그레이 코드(출력) $G_3\ G_2\ G_1\ G_0$
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 0
0 1 0 1	0 1 1 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 0 0
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1
1 0 1 0	1 1 1 1
1 0 1 1	1 1 1 0
1 1 0 0	1 0 1 0
1 1 0 1	1 0 1 1
1 1 1 0	1 0 0 1
1 1 1 1	1 0 0 0

(a) 진리표

B_1B_0	00	01	11	10
B_3B_2	00			
00				
01				
11	1	1	1	1
10	1	1	1	1

$$G_3 = B_3$$

B_1B_0	00	01	11	10
B_3B_2	00			
00			1	1
01	1	1		
11	1	1		
10			1	1

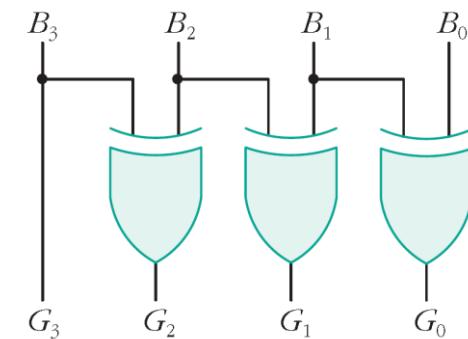
$$G_1 = \bar{B}_2B_1 + B_2\bar{B}_1 \\ = B_2 \oplus B_1$$

B_1B_0	00	01	11	10
B_3B_2	00			
00				
01	1			
11		1		
10			1	1

$$G_2 = \bar{B}_3B_2 + B_3\bar{B}_2 \\ = B_3 \oplus B_2$$

B_1B_0	00	01	11	10
B_3B_2	00			
00	1			
01		1		
11			1	
10				1

$$G_0 = \bar{B}_1B_0 + B_1\bar{B}_0 \\ = B_1 \oplus B_0$$



(c) 논리 회로

그림 3-55 2진 코드를 그레이 코드로 변환하는 회로

03 조합 논리 회로

□ 코드 변환기 (그레이 코드 → 2진 코드 변환)

그레이 코드(입력) $G_3\ G_2\ G_1\ G_0$	2진 코드(출력) $B_3\ B_2\ B_1\ B_0$
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 0
0 1 0 0	0 1 1 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 0 0
0 1 1 1	0 1 0 1
1 0 0 0	1 1 1 1
1 0 0 1	1 1 1 0
1 0 1 0	1 1 0 0
1 0 1 1	1 1 0 1
1 1 0 0	1 0 0 0
1 1 0 1	1 0 0 1
1 1 1 0	1 0 1 1
1 1 1 1	1 0 1 0

G_3G_2	00	01	11	10
G_3G_0	00			
G_3G_2	01			
G_3G_0	11	1 1 1 1		
G_3G_0	10	1 1 1 1		

$$B_3 = G_3$$

G_3G_2	00	01	11	10
G_3G_0	00			
G_3G_2	01	1 1		
G_3G_0	11		1 1	
G_3G_0	10	1 1		

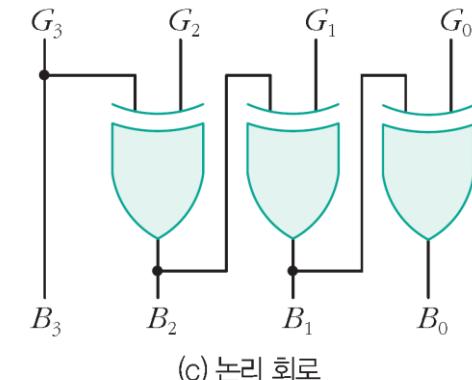
$$\begin{aligned} B_1 &= G_3 \oplus G_2 \oplus G_1 \\ &= B_2 \oplus G_1 \end{aligned}$$

G_3G_2	00	01	11	10
G_3G_0	00			
G_3G_2	01	1 1 1 1		
G_3G_0	11		1 1 1 1	
G_3G_0	10	1 1 1 1		

$$\begin{aligned} B_2 &= \bar{G}_3G_2 + G_3\bar{G}_2 \\ &= G_3 \oplus G_2 \end{aligned}$$

G_3G_2	00	01	11	10
G_3G_0	00			
G_3G_2	01	1	1	1
G_3G_0	11	1	1	1
G_3G_0	10	1	1	

$$\begin{aligned} B_0 &= G_3 \oplus G_2 \oplus G_1 \oplus G_0 \\ &= B_1 \oplus G_0 \end{aligned}$$



(c) 논리 회로

그림 3-56 그레이 코드를 2진 코드로 변환하는 회로

(a) 진리표

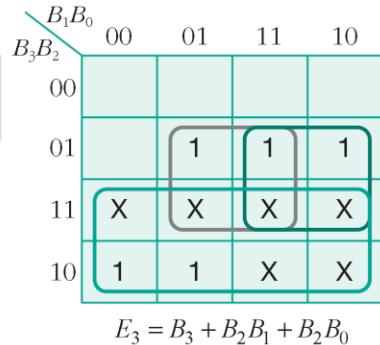
(b) 논리식의 간소화

03 조합 논리 회로

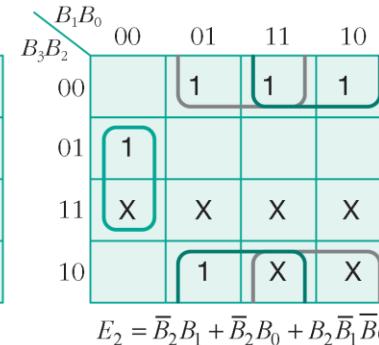
□ 코드 변환기 (BCD 코드 → 3초과 코드 변환)

BCD 코드(입력) $B_3\ B_2\ B_1\ B_0$	3초과 코드(출력) $E_3\ E_2\ E_1\ E_0$
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x

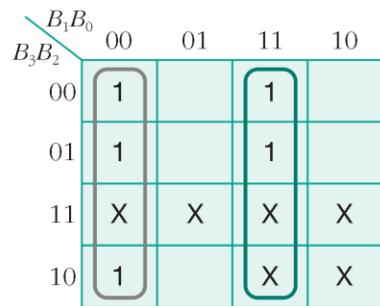
(a) 진리표



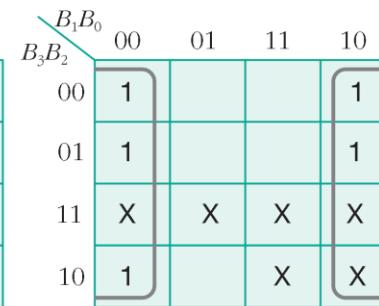
$$E_3 = B_3 + B_2B_1 + B_2B_0$$



$$E_2 = \bar{B}_2B_1 + \bar{B}_2B_0 + B_2\bar{B}_1\bar{B}_0$$

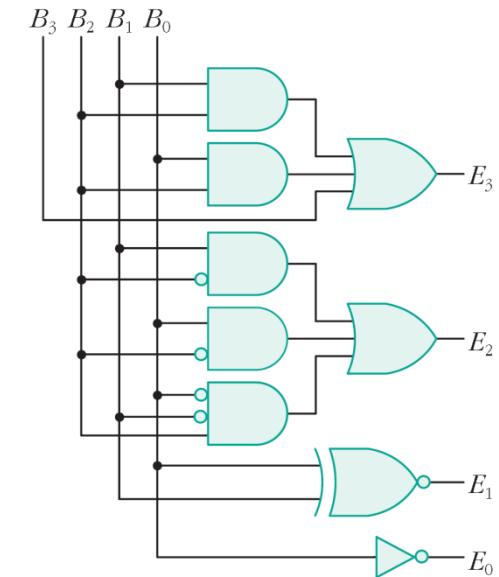


$$E_1 = \bar{B}_1\bar{B}_0 + B_1B_0 \\ = \bar{B}_1 \oplus B_0$$



$$E_0 = \bar{B}_0$$

(b) 논리식의 간소화



(c) 논리 회로

그림 3-57 BCD 코드를 3초과 코드로 변환하는 회로

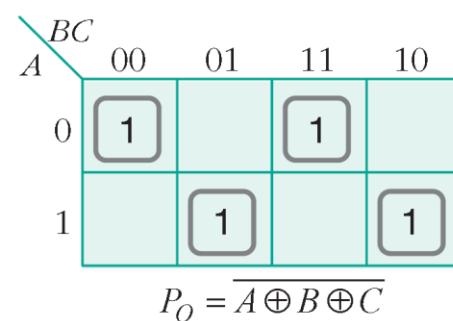
03 조합 논리 회로

□ 패리티 발생기와 검사기

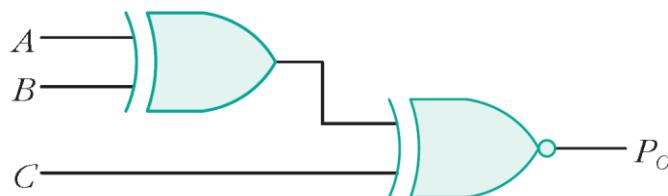
❖ 패리티 발생기

입력			출력	
A	B	C	P_O (홀수)	P_E (짝수)
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

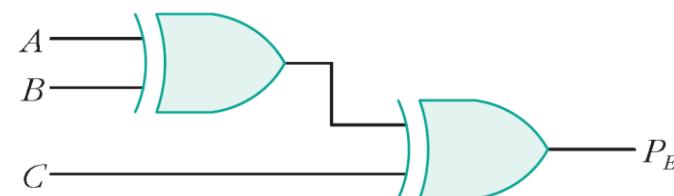
(a) 진리표



(b) 논리식의 간소화



(c) 홀수 패리티 발생기 논리 회로



(d) 짝수 패리티 발생기 논리 회로

그림 3-58 패리티 발생기

03 조합 논리 회로

❖ 패리티 검사기

입력				출력	입력				출력
A	B	C	P_O (홀수)	Y_O (홀수)	A	B	C	P_E (짝수)	Y_E (짝수)
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	0	1	0	1
0	0	1	1	1	0	0	1	1	0
0	1	0	0	0	0	1	0	0	1
0	1	0	1	1	0	1	0	1	0
0	1	1	0	1	0	1	1	0	0
0	1	1	1	0	0	1	1	1	1
1	0	0	0	0	1	0	0	0	1
1	0	0	1	1	1	0	0	1	0
1	0	1	0	1	1	0	1	0	0
1	0	1	1	0	1	0	1	1	1
1	1	0	0	1	1	1	0	0	0
1	1	0	1	0	1	1	0	1	1
1	1	1	0	0	1	1	1	0	1
1	1	1	1	1	1	1	1	1	0

(a) 진리표

03 조합 논리 회로

❖ 패리티 검사기(계속)

- 패리티 검사기 출력이 $Y = 0$ 이면 에러가 발생하지 않았다고 판단하고, $Y = 1$ 이면 에러가 발생했다고 판단한다.

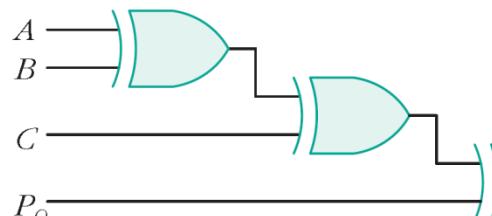
		CP_O			
		00	01	11	10
AB	00	1		1	
	01		1		1
AB	11	1		1	
	10		1		1

$$Y_O = \overline{A \oplus B \oplus C \oplus P_O}$$

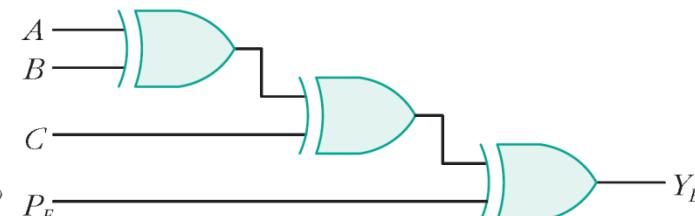
		CP_E			
		00	01	11	10
AB	00		1		1
	01	1		1	
AB	11		1		1
	10	1		1	

$$Y_E = A \oplus B \oplus C \oplus P_E$$

(b) 논리식의 간소화



(c) 홀수 패리티 검사기 논리 회로



(d) 짝수 패리티 검사기 논리 회로

그림 3-59 패리티 검사기

03 조합 논리 회로

❖ 데이터 전송 시스템

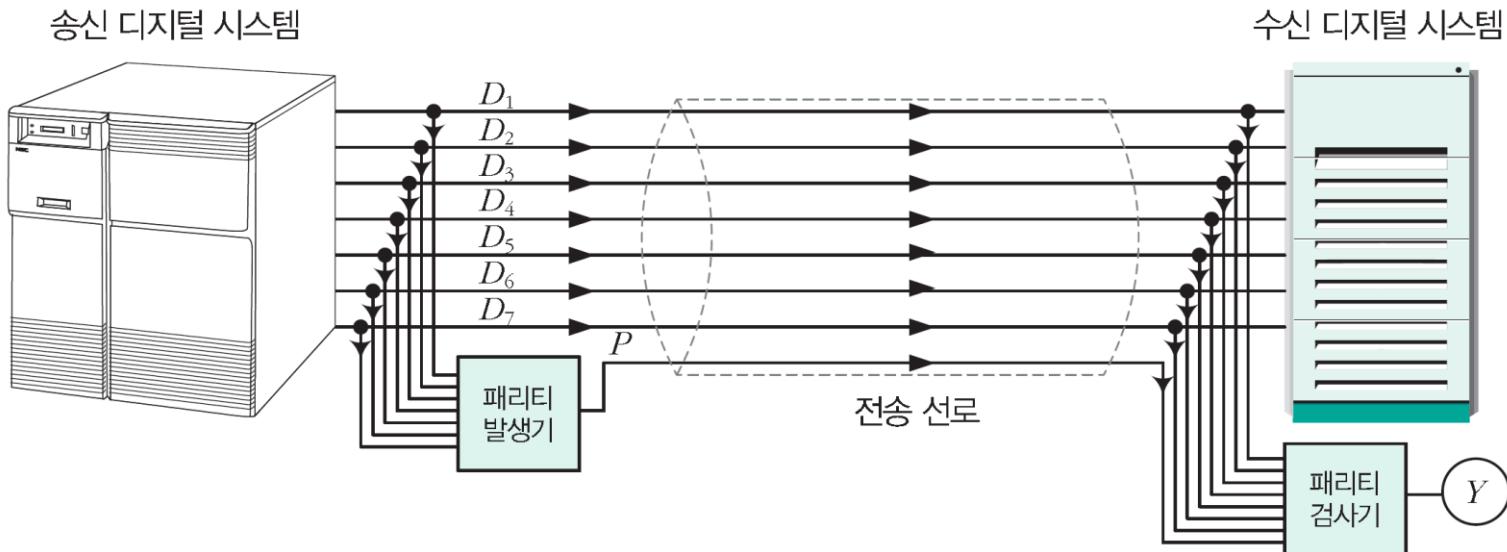


그림 3-60 데이터 전송 시스템에서 패리티 비트를 이용한 에러 검출

04 순서 논리 회로

1 순서 논리 회로의 개요

- 조합 논리 회로(combational logic circuit) : 이전 입력 값에 관계없이 현재 입력 값에 따라 출력이 결정
- 순서 논리 회로(sequential logic circuit) : 현재의 입력 값과 이전 출력 상태에 따라 출력 값이 결정

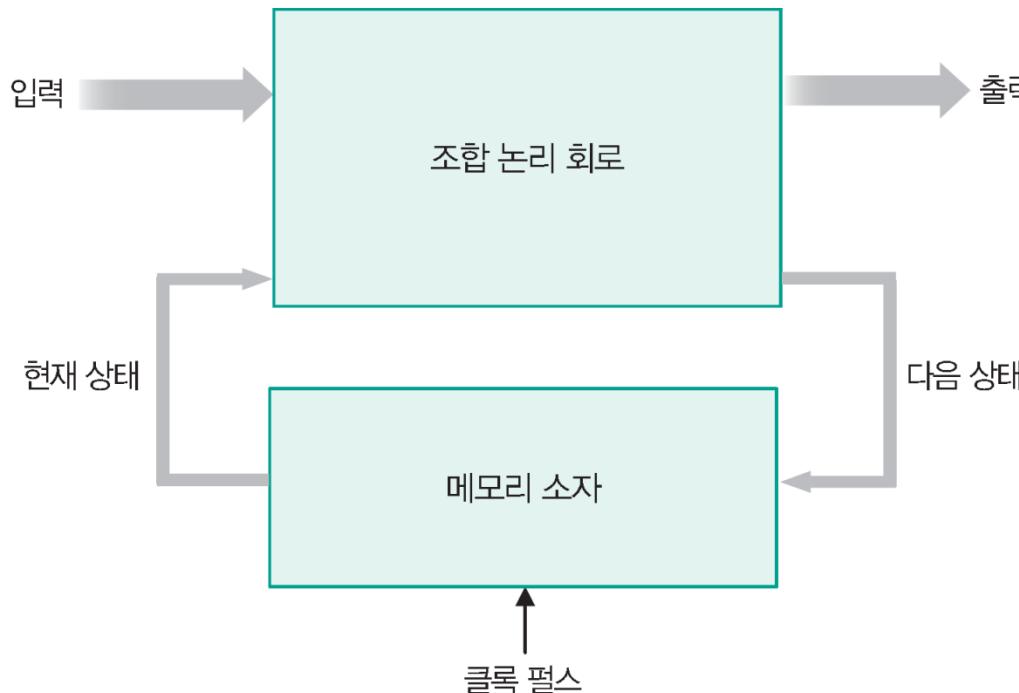


그림 3-62 순서 논리 회로의 구성

04 순서 논리 회로

❖ 순서 논리 회로의 특징

- 순서 논리 회로의 출력은 외부에서 들어온 입력과 이전 출력 상태에 따라 결정된다. 이러한 동작은 클록 펄스가 들어올 때마다 반복해서 일어난다.
- 순서 논리 회로는 기억 기능(플립플롭)이 있다.
- 대표적인 순서 논리 회로에는 플립플롭, 카운터, 레지스터 등이 있다.

□ 클록 펄스

- **플립플롭**(flip-flop)은 **클록 펄스**(Clock Pulse, CP)라는 제어 입력을 가지며, 출력은 클록 펄스에 동기되어 변하고 이러한 변화를 트리거(trigger)되었다고 한다.

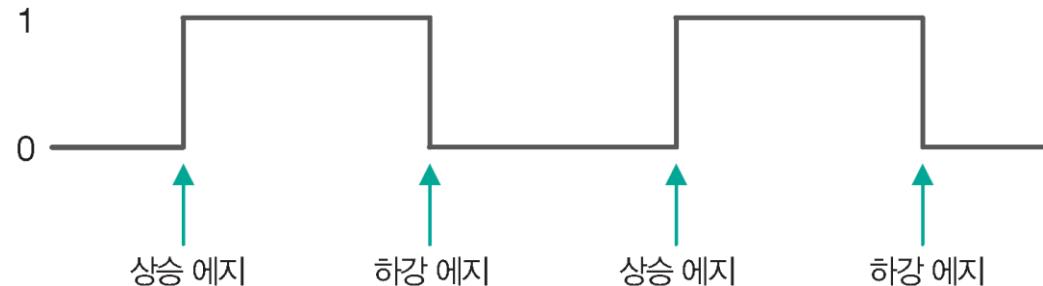
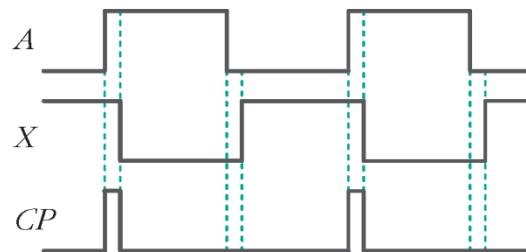
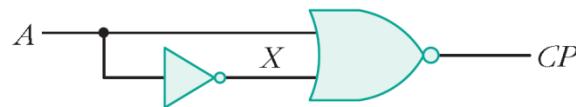
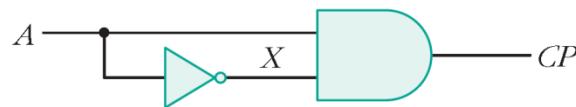


그림 3-63 클록 펄스의 형태

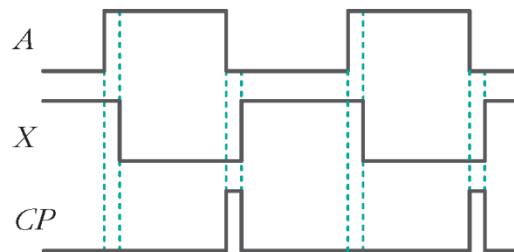
04 순서 논리 회로

❖ 펄스 전이 검출기

- 실제 회로에서 클록 펄스는 상승 에지나 하강 에지에서 순간적으로 1이 되는 날카로운 파형을 만들어 플립플롭을 동작시키는데, 이 파형을 에지 트리거(edge trigger)라고 한다.
- 클록 펄스(구형파)로 에지 트리거를 만들려면 **펄스 전이 검출기**가 필요하다



(a) 상승 에지 트리거



(b) 하강 에지 트리거

그림 3-64 펄스 전이 검출기 구조

04 순서 논리 회로

□ 플립플롭의 종류

❖ 플립플롭의 특징

- 플립플롭은 1비트의 정보를 기억할 수 있는 기억 소자다.
- 플립플롭은 제어 입력인 클록 펄스가 있으며 다음 클록 펄스가 들어올 때까지 현재 상태를 유지 한다.
- 플립플롭은 Q 와 \bar{Q} 로 표시된 출력이 2개 있으며 Q 와 \bar{Q} 의 상태는 서로 보수가 되어야 정상 상태 가 된다.
- 플립플롭은 RAM의 구성 요소로도 사용된다.
- 플립플롭에는 SR 플립플롭, JK 플립플롭, D 플립플롭, T 플립플롭이 있다.

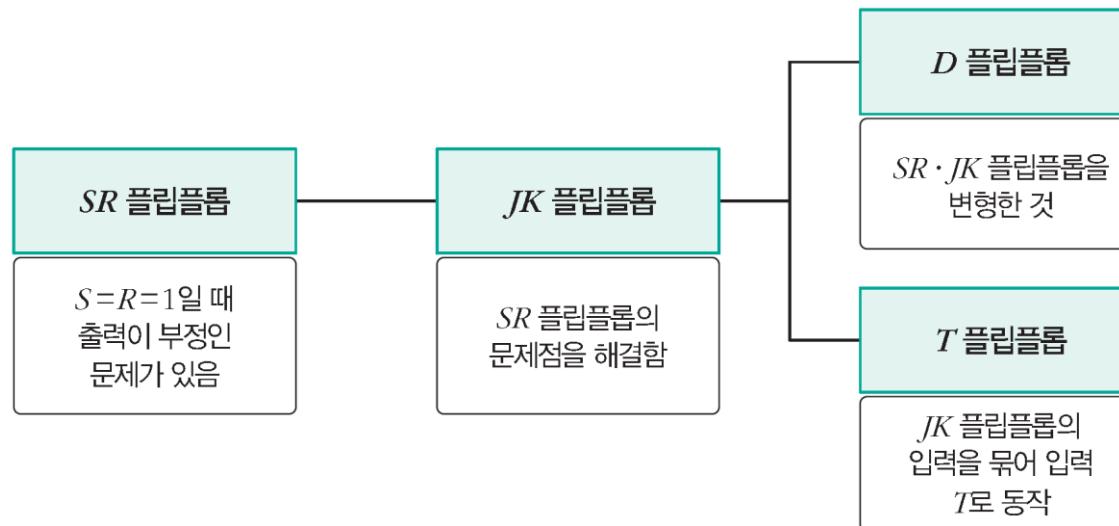
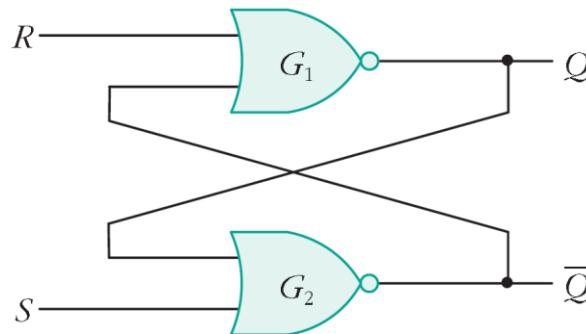


그림 3-65 플립플롭의 종류

04 순서 논리 회로

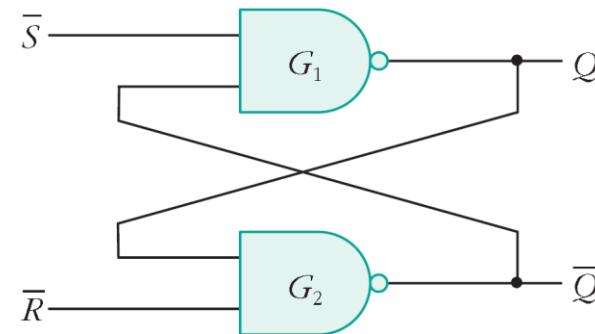
2 플립플롭

□ SR 래치



S	R	$Q(t+1)$
0	0	$Q(t)$ (불변)
0	1	0
1	0	1
1	1	부정

(a) NOR 게이트로 구성



\bar{S}	\bar{R}	$Q(t+1)$
0	0	부정
0	1	1
1	0	0
1	1	$Q(t)$ (불변)

(b) NAND 게이트로 구성

그림 3-66 SR 래치의 논리 회로와 진리표

04 순서 논리 회로

□ SR 플립플롭

- SR 플립플롭 : 클록 펄스가 있을 때만 동작하는 SR 래치를 의미
- 클록 펄스는 **상승 에지 트리거 신호**가 입력된 경우다.

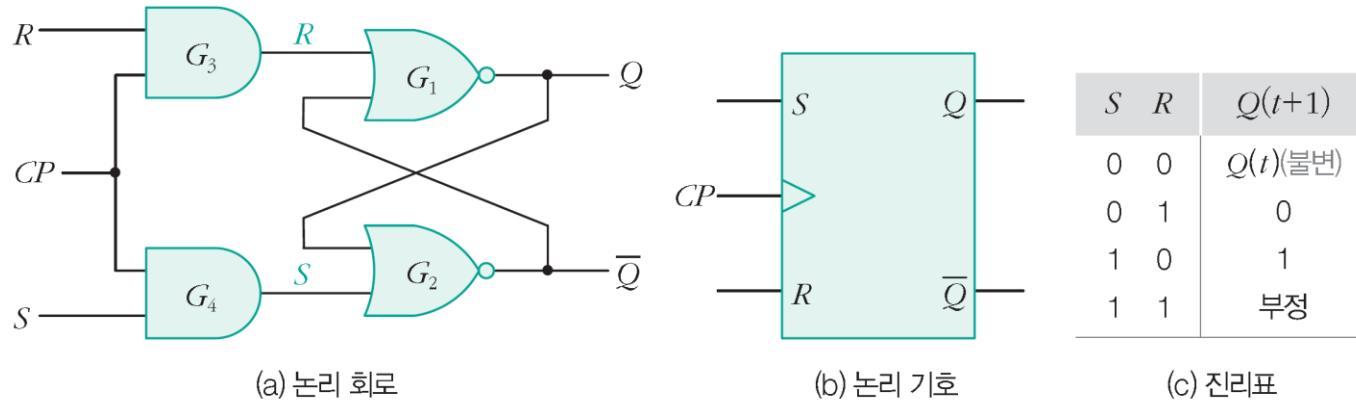


그림 3-67 SR 플립플롭의 구조

$CP=0$ 인 경우	S 와 R 의 입력에 관계없이 앞단의 AND 게이트 G_3 과 G_4 의 출력이 항상 0이므로 플립플롭의 출력은 불변
$CP=1$ 인 경우	S 와 R 의 입력이 회로 후단의 NOR 게이트 G_1 과 G_2 의 입력으로 전달되어 SR 래치와 같은 동작을 수행

04 순서 논리 회로

❖ SR 플립플롭의 특성표 및 특성 방정식

S	R	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	부정
1	1	1	부정

(a) 특성표

		$RQ(t)$			
		00	01	11	10
S		0	1		
0					
1		1	1	X	X

$$Q(t+1) = S + \bar{R}Q(t), SR = 0$$

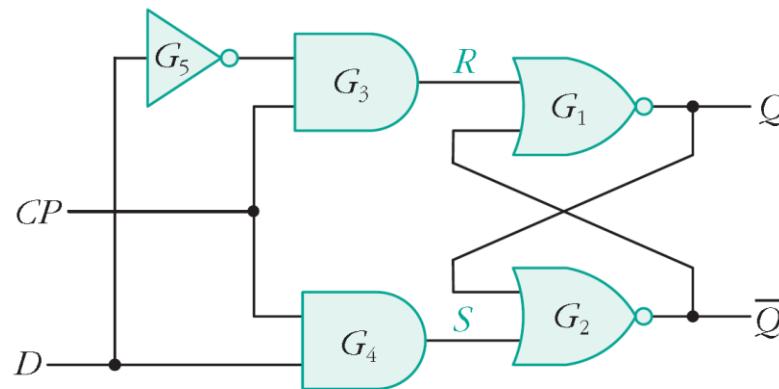
(b) 특성 방정식

그림 3-68 SR 플립플롭의 특성표와 특성 방정식

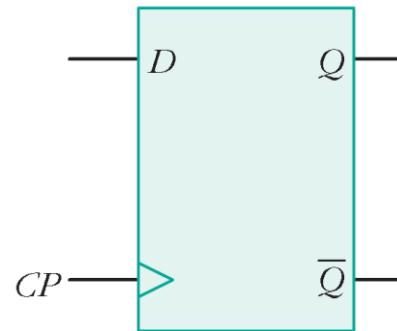
04 순서 논리 회로

□ D 플립플롭

- SR 플립플롭에서 원하지 않는 상태($S=R=1$)를 제거하는 한 가지 방법
- SR 플립플롭을 변형한 것
- 입력신호 D 가 CP 에 동기되어 그대로 출력에 전달되는 특성을 가지고 있음
- D 플립플롭이라는 이름은 데이터(Data)를 전달하는 것과 지연(Delay)하는 역할에서 유래



(a) 논리 회로



(b) 논리 기호

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

(c) 진리표

그림 3-69 D 플립플롭의 구조

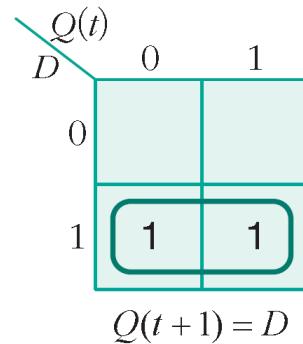
$CP=1, D=1$	G_3 의 출력은 0, G_4 의 출력은 1이 된다. 따라서 SR 래치의 입력은 $S=0, R=1$ 이 되므로 결과적으로 $Q=1$ 을 얻는다.
$CP=1, D=0$	G_3 의 출력은 1, G_4 의 출력은 0이 된다. 따라서 SR 래치의 입력은 $S=1, R=0$ 이 되므로 결과적으로 $Q=0$ 을 얻는다.

04 순서 논리 회로

❖ D 플립플롭의 특성표 및 특성 방정식

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

(a) 특성표



(b) 특성 방정식

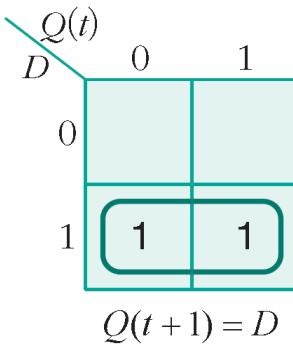


그림 3-70 D 플립플롭의 특성표와 특성 방정식

04 순서 논리 회로

□ JK 플립플롭

- JK 플립플롭은 SR 플립플롭에서 $S=1, R=1$ 인 경우 출력이 불안정한 상태가 되는 문제점을 개선하여 $S=1, R=1$ 에서도 동작하도록 개선한 회로
- JK 플립플롭의 J 는 S (set)에, K 는 R (reset)에 대응하는 입력
- $J=1, K=1$ 인 경우 JK 플립플롭의 출력은 이전 출력의 보수 상태로 변화
- JK 플립플롭은 플립플롭 중에서 가장 많이 사용되는 플립플롭이다.

04 순서 논리 회로

JK 플립플롭

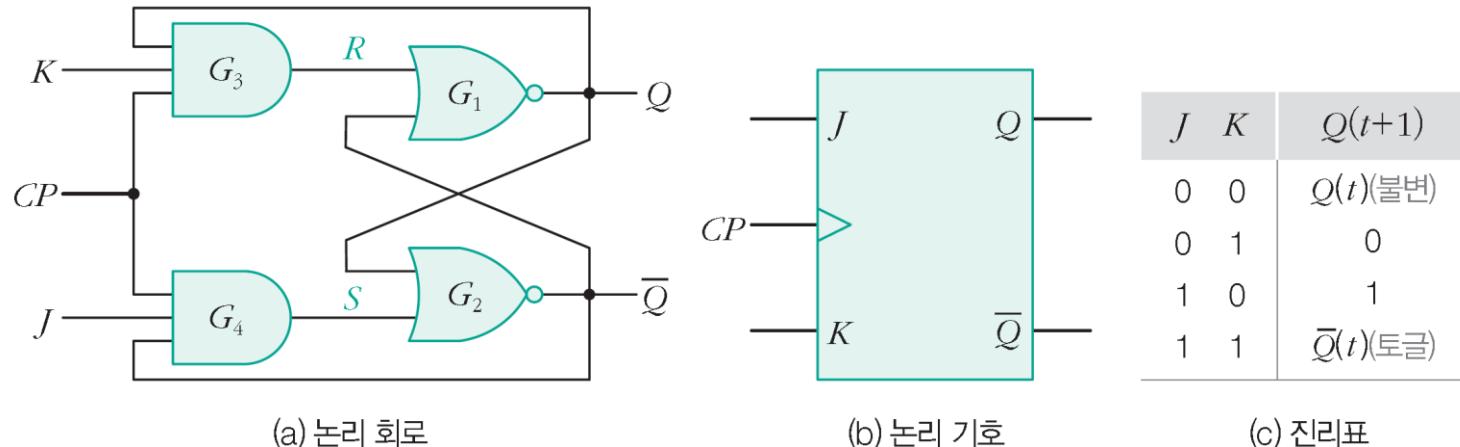


그림 3-71 JK 플립플롭의 구조

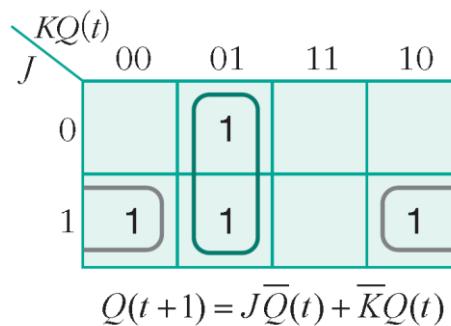
$J=0, K=0$	G_3 과 G_4 의 출력이 모두 0이므로 G_1 과 G_2 로 구성된 SR 래치는 출력이 변하지 않는다.
$J=0, K=1$	G_4 의 출력은 0이 되고 G_3 의 출력은 $Q(t) \cdot K \cdot CP$ 인데 $K=1, CP=1$ 이므로 $Q(t)$ 가 된다.
$J=1, K=0$	G_3 의 출력은 0이 되고 G_4 의 출력은 $\bar{Q}(t) \cdot J \cdot CP$ 인데 $J=1, CP=1$ 이므로 $\bar{Q}(t)$ 가 된다.
$J=1, K=1$	G_3 의 출력은 $Q(t) \cdot K \cdot CP$ 인데 $K=1, CP=1$ 이므로 $Q(t)$ 가 된다. 또한 G_4 의 출력은 $\bar{Q}(t) \cdot J \cdot CP$ 인데 $J=1, CP=1$ 이므로 $\bar{Q}(t)$ 가 된다. $Q(t)=0$ 인 경우 SR 래치의 $S=1, R=0$ 인 경우와 같으므로 출력은 $Q(t+1)=1$ 이 된다. 마찬가지로 $Q(t)=1$ 인 경우 SR 래치의 $S=0, R=1$ 인 경우와 같으므로 출력은 $Q(t+1)=0$ 이 된다. 따라서 출력은 보수가 된다.

04 순서 논리 회로

❖ JK플립플롭의 특성표 및 특성 방정식

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

(a) 특성표



(b) 특성 방정식

$$Q(t+1) = J\bar{Q}(t) + \bar{K}Q(t)$$

그림 3-72 JK 플립플롭의 특성표와 특성 방정식

04 순서 논리 회로

□ T플립플롭

- JK 플립플롭의 J 와 K 입력을 묶어서 하나의 입력신호 T 로 동작시키는 플립플롭
- T 플립플롭의 입력 $T=0$ 이면, T 플립플롭은 $J=0, K=0$ 인 JK 플립플롭과 같이 동작하므로 출력은 변하지 않는다. $T=1$ 이면, $J=1, K=1$ 인 JK 플립플롭과 같이 동작하므로 출력은 보수가 된다.

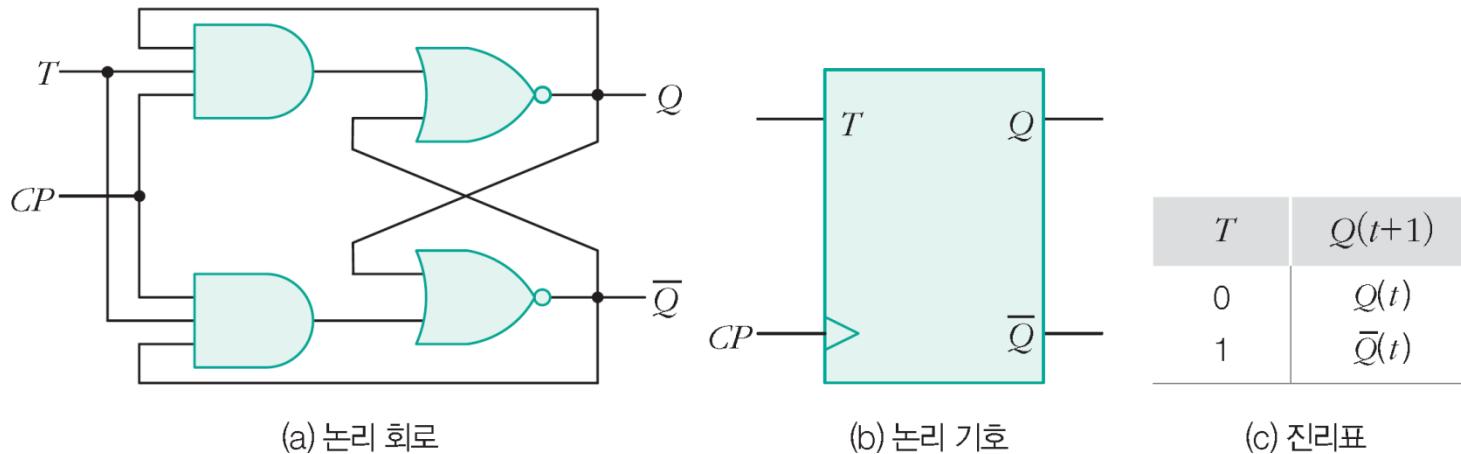
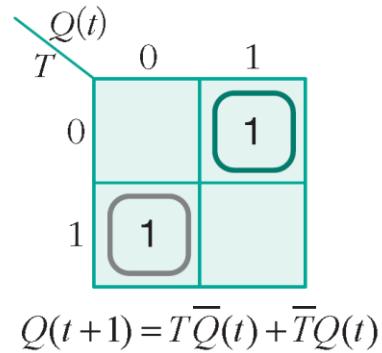


그림 3-73 T 플립플롭

04 순서 논리 회로

❖ T플립플롭의 특성표 및 특성 방정식

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



(a) 특성표

(b) 특성 방정식

그림 3-74 T 플립플롭의 특성표와 특성 방정식

04 순서 논리 회로

□ 주종형 JK 플립플롭

- 레이스 현상 문제(racing problem) : JK 플립플롭은 $J = K = 1$ 인 경우 클록 펄스가 길어지면 출력이 계속 반전되는 현상
- 이를 해결하는 방법은 에지 트리거를 이용하거나 주종형 JK 플립플롭(master-slave JK flip-flop)을 사용하는 것이다.

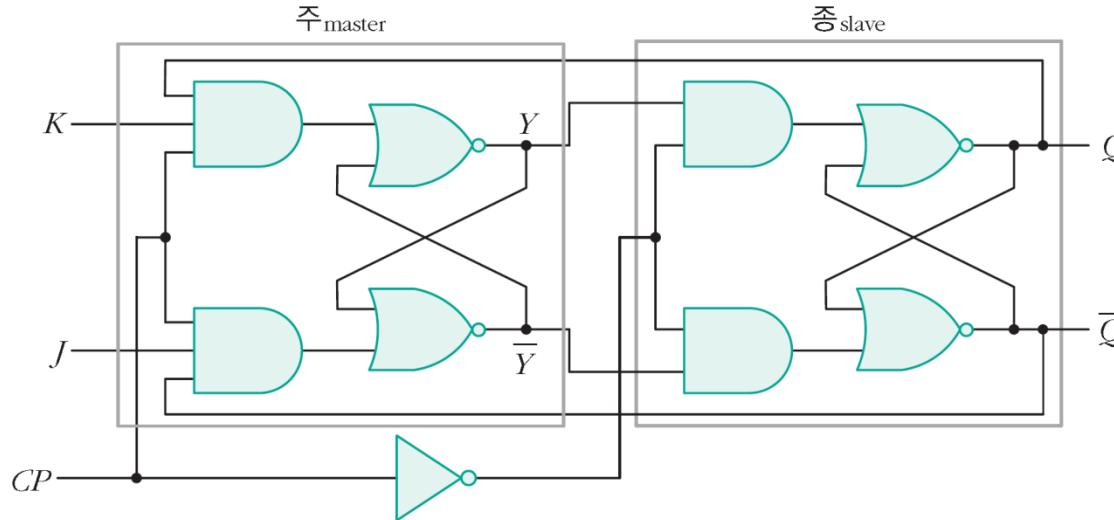


그림 3-75 주종형 JK 플립플롭

$CP=1$	외부의 J와 K의 입력이 Master 플립플롭에 전달 Slave 플립플롭은 $CP=0$ 이므로 동작하지 않음
$CP=0$	Slave 플립플롭이 동작하여 $Q = Y, \bar{Q} = \bar{Y}$ Master 플립플롭은 $CP=0$ 이므로 동작하지 않음

04 순서 논리 회로

□ 비동기 입력

- 대부분의 플립플롭은 클록펄스에 의해서 플립플롭의 상태를 변화시킬 수 있는 동기입력이 있고, 클록펄스와 관계없이 비동기적으로 변화시킬 수 있는 비동기 입력인 preset(\overline{PR}) 입력과 clear(\overline{CLR}) 입력이 있다.
- 비동기 입력들은 플립플롭의 초기조건을 결정하는 등 다방면으로 유용하게 사용

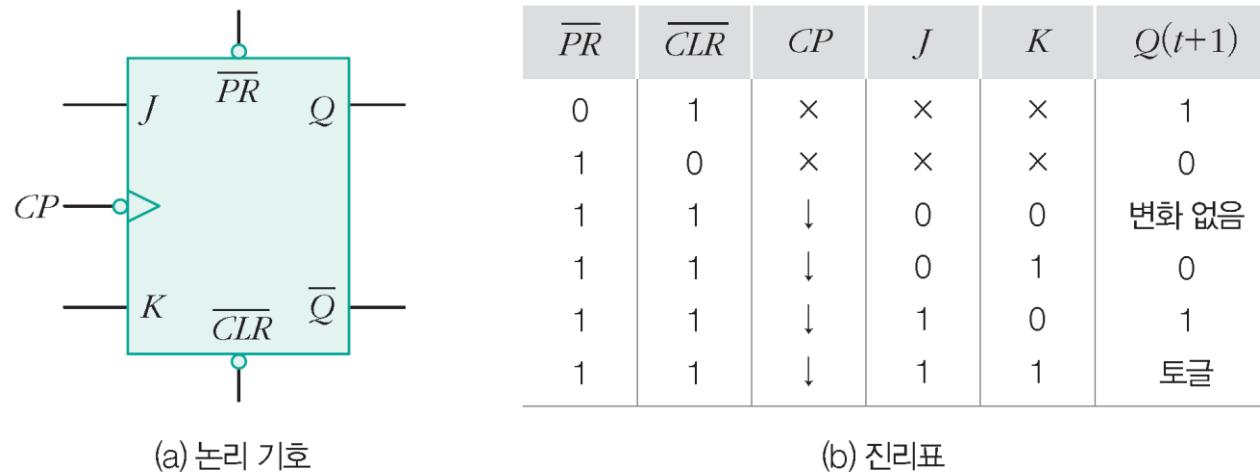


그림 3-76 비동기 입력을 가진 JK 플립플롭의 논리 기호와 진리표

3 순서 논리 회로의 설계

□ 여기표

- 플립플롭의 특성표 : 현재 상태와 입력값이 주어졌을 때, 다음 상태가 어떻게 변하는가를 나타내는 표
- 플립플롭의 여기표(excitation table) : 현재 상태에서 다음 상태로 변했을 때 플립플롭의 입력조건이 어떤 상태인가를 나타내는 표
- 플립플롭의 여기표는 순서논리회로를 설계할 때 자주 사용

04 순서 논리 회로

❖ SR 플립플롭의 예기표

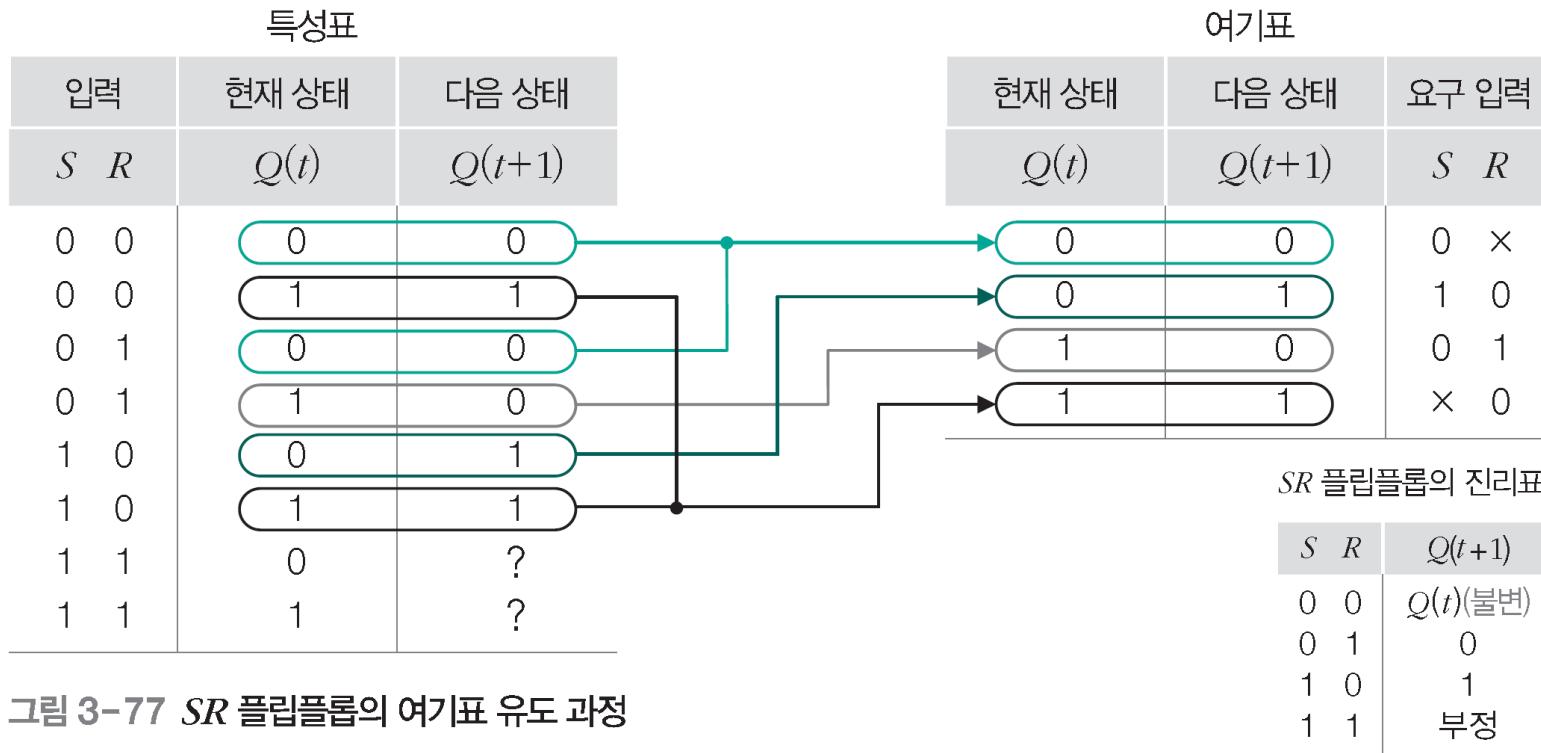


그림 3-77 SR 플립플롭의 예기표 유도 과정

04 순서 논리 회로

❖ JK플립플롭의 예기표

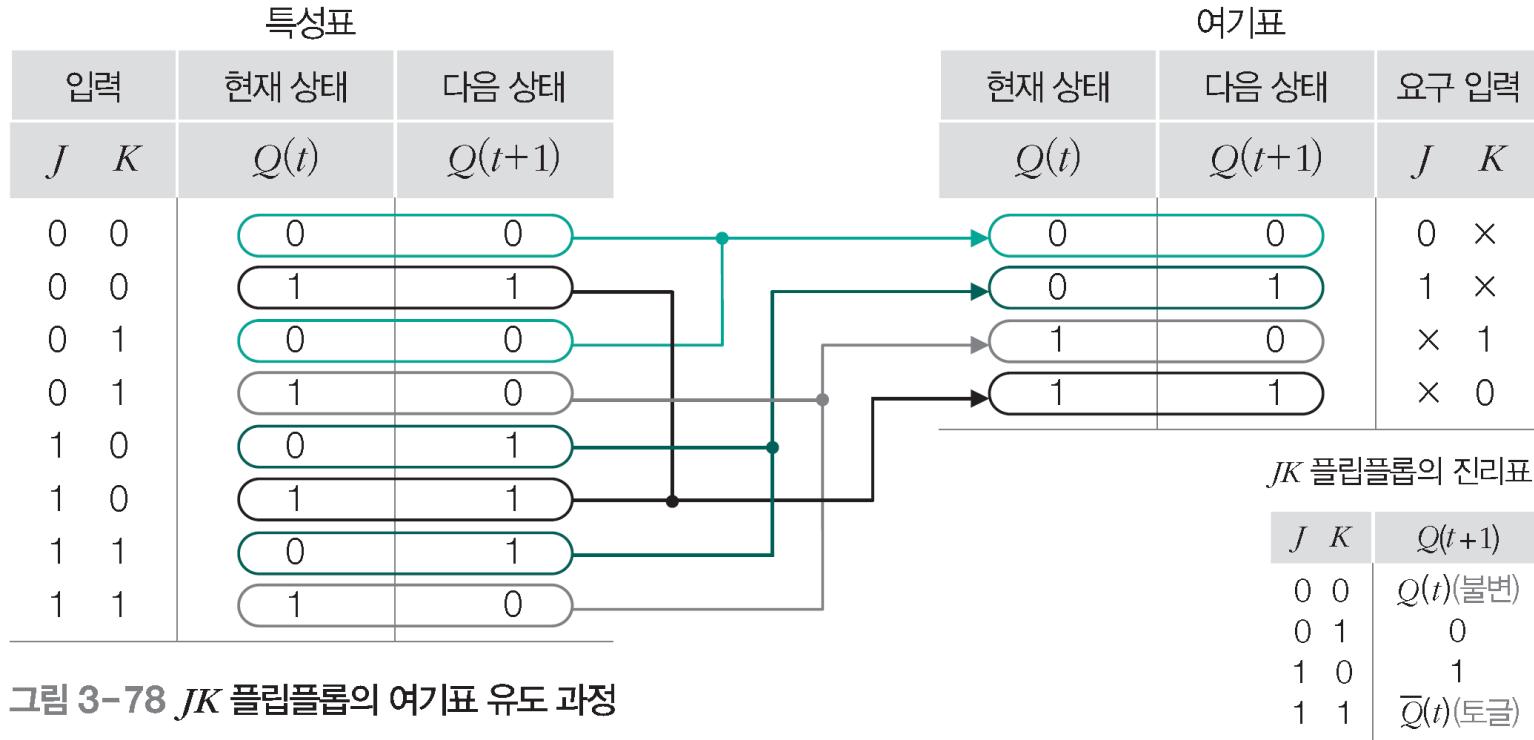


그림 3-78 JK 플립플롭의 예기표 유도 과정

04 순서 논리 회로

❖ D 플립플롭의 여기표

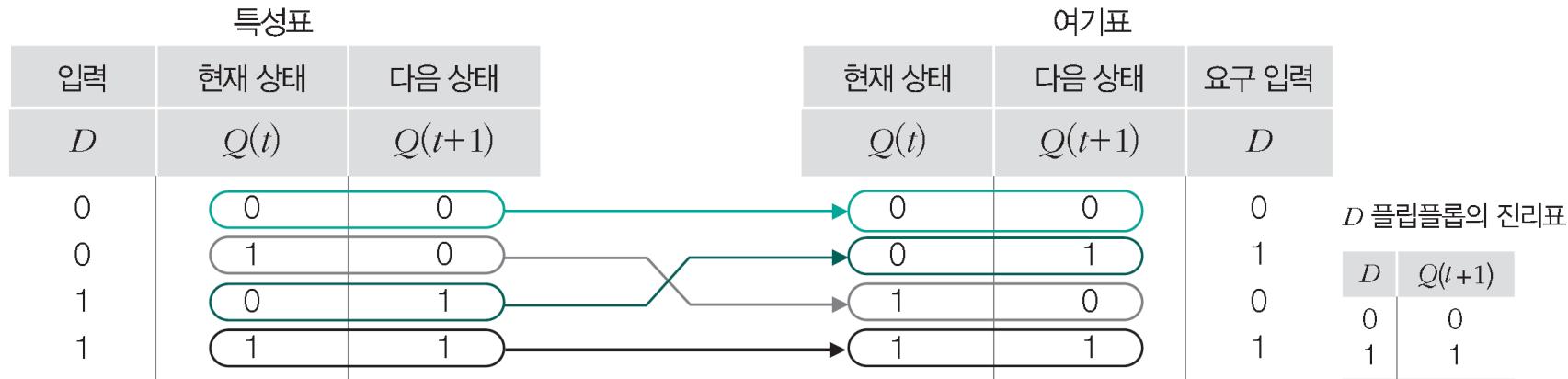


그림 3-79 D 플립플롭의 여기표 유도 과정

❖ T 플립플롭의 여기표

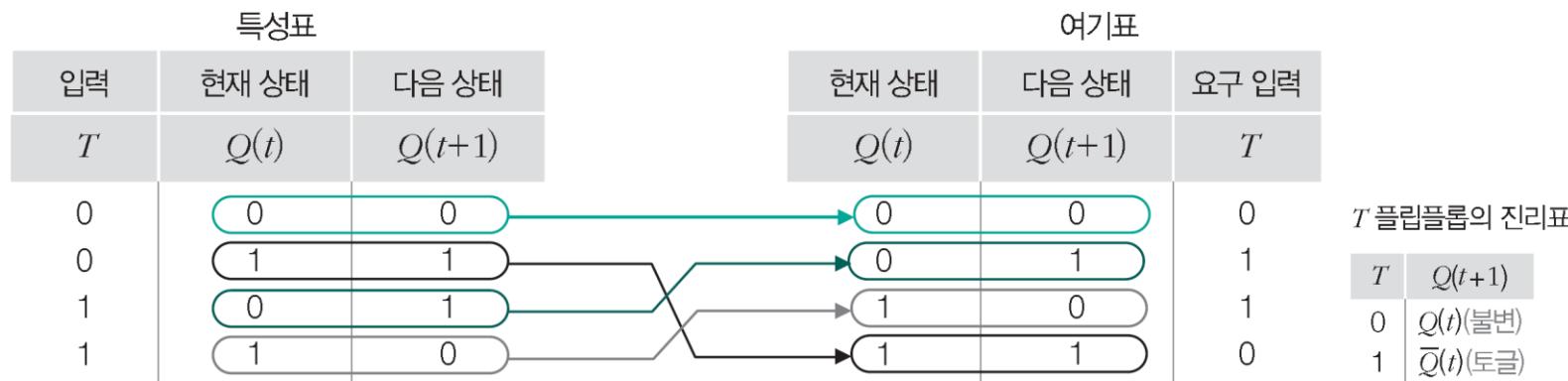


그림 3-80 T 플립플롭의 여기표 유도 과정

04 순서 논리 회로

□ 순서 논리 회로의 설계 과정

- ① 설계 사양으로부터 상태도와 상태표 작성
- ② 플립플롭의 수와 종류 결정
- ③ 플립플롭의 입력, 출력 및 각 상태에 문자 기호 부여
- ④ 상태표를 이용해 회로의 상태 여기표 작성
- ⑤ 간소화 방법을 이용해 출력 함수와 플립플롭의 입력 함수 유도
- ⑥ 순서 논리 회로도 작성

04 순서 논리 회로

① 설계 사양으로부터 상태도와 상태표 작성

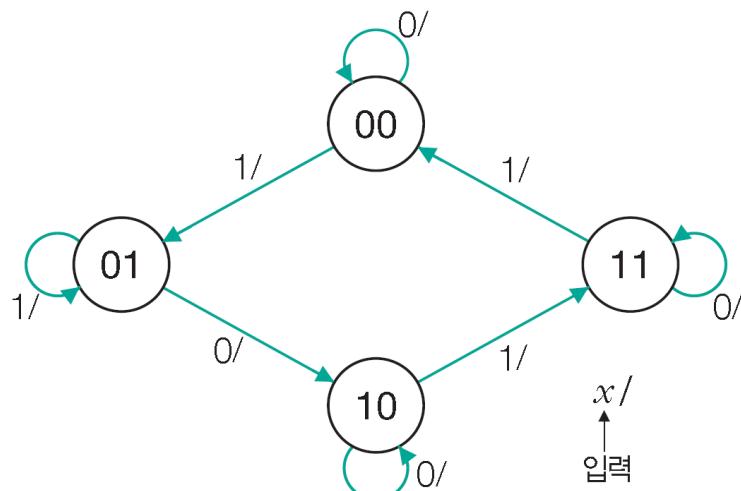


그림 3-81 순서 논리 회로에 대한 상태도

표 3-8 그림 3-81의 상태표

현재 상태		입력	다음 상태	
A	B	x	A	B
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

② 플립플롭의 수와 종류 결정하고 ③ 각 상태에 문자 기호 부여

- 네 가지 상태가 있으므로 플립플롭이 2개 필요하며, 각 플립플롭에 문자 A와 B를 할당한다.
- JK 플립플롭을 이용한다.

n 개의 서로 다른 상태를 나타내려면 플립플롭이 $\lceil \log_2 n \rceil$ 개 필요하다. 예를 들어 $n=10$ 이면 $\log_2 10 \approx 3.219$ 이므로 플립플롭이 $\lceil \log_2 10 \rceil = 4$ 개 필요하다.

04 순서 논리 회로

④ 상태표를 이용해 회로의 상태 예기표 작성

표 3-9 상태 예기표

현재 상태		입력 x	다음 상태		플립플롭 입력			
A	B		A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	1	0	1	×	×	1
0	1	1	0	1	0	×	×	0
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	0	×	0
1	1	1	0	0	×	1	×	1

JK 플립플롭의 예기표

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

04 순서 논리 회로

⑤ 간소화 방법을 이용해 출력 함수와 플립플롭의 입력 함수 유도

		00	01	11	10
		0			1
		1	X	X	X
A	Bx				

$$J_A = B\bar{x}$$

		00	01	11	10
		0	X	X	X
		1			1
A	Bx				

$$K_A = Bx$$

		00	01	11	10
		0	1	X	1
		1	X	X	X
A	Bx				

$$J_B = x$$

		00	01	11	10
		0	X	X	
		1	X	X	1
A	Bx				

$$K_B = Ax + \overline{Ax} = A \odot x$$

그림 3-82 카르노 맵을 이용한 간소화 과정

04 순서 논리 회로

⑥ 순서 논리 회로도 작성

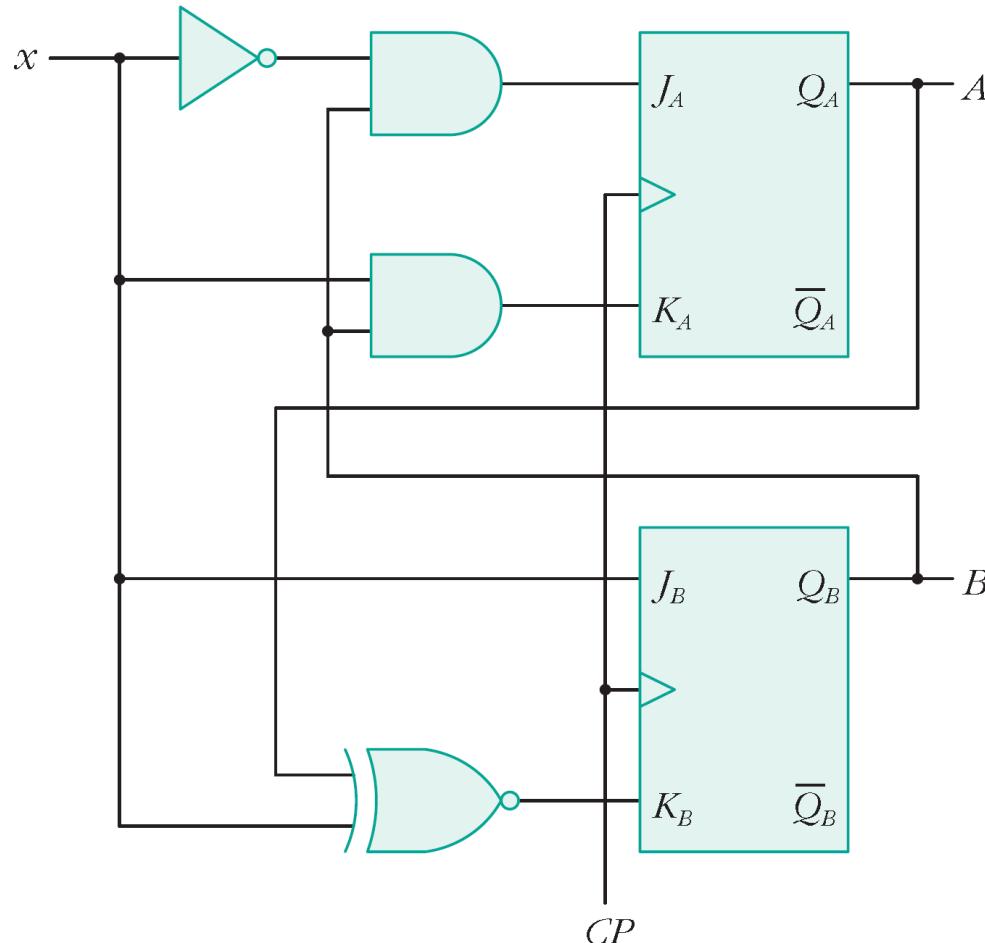


그림 3-83 순서 논리 회로의 구현

04 순서 논리 회로

4 카운터의 설계

- JK 플립플롭을 사용해 3비트 동기식 카운터를 순서 논리 회로 방식으로 설계해 보자.

상태도와 상태표 작성

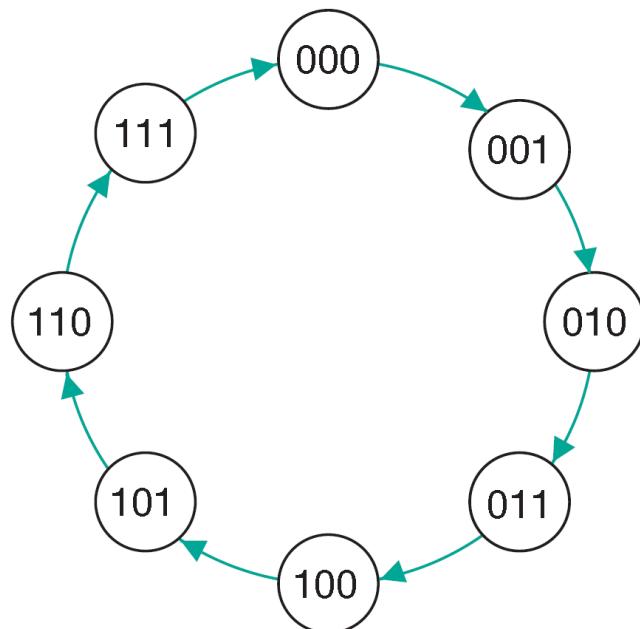


그림 3-84 3비트 동기식 2진 카운터의 상태도

상태표를 이용해 회로의 상태 여기표 작성

표 3-10 3비트 동기식 2진 카운터의 상태 여기표

현재 상태			다음 상태			플립플롭 입력					
Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	\times	0	\times	1	\times
0	0	1	0	1	0	0	\times	1	\times	\times	1
0	1	0	0	1	1	0	\times	\times	0	1	\times
0	1	1	1	0	0	1	\times	\times	1	\times	1
1	0	0	1	0	1	\times	0	0	\times	1	\times
1	0	1	1	1	0	\times	0	1	\times	\times	1
1	1	0	1	1	1	\times	0	\times	0	1	\times
1	1	1	0	0	0	\times	1	\times	1	\times	1

04 순서 논리 회로

간소화 방법을 이용해 플립플롭의 입력 함수 유도

		$Q_B Q_A$	00	01	11	10
		Q_C	0			
		0			1	
		1	X	X	X	X

$J_C = Q_B Q_A$

		$Q_B Q_A$	00	01	11	10
		Q_C	0			
		0	X	X	X	X
		1			1	

$K_C = Q_B Q_A$

		$Q_B Q_A$	00	01	11	10
		Q_C	0			
		0			1	X
		1	1	X	X	X

$J_B = Q_A$

		$Q_B Q_A$	00	01	11	10
		Q_C	0			
		0	X	X	1	
		1	X	X	1	

$K_B = Q_A$

		$Q_B Q_A$	00	01	11	10
		Q_C	0			
		0	1	X	X	1
		1	1	X	X	1

$J_A = 1$

		$Q_B Q_A$	00	01	11	10
		Q_C	0			
		0	1	1	1	X
		1	X	1	1	X

$K_A = 1$

그림 3-85 3비트 동기식 2진 카운터의 카르노 맵을 이용한 간소화 과정

04 순서 논리 회로

순서 논리 회로도 작성

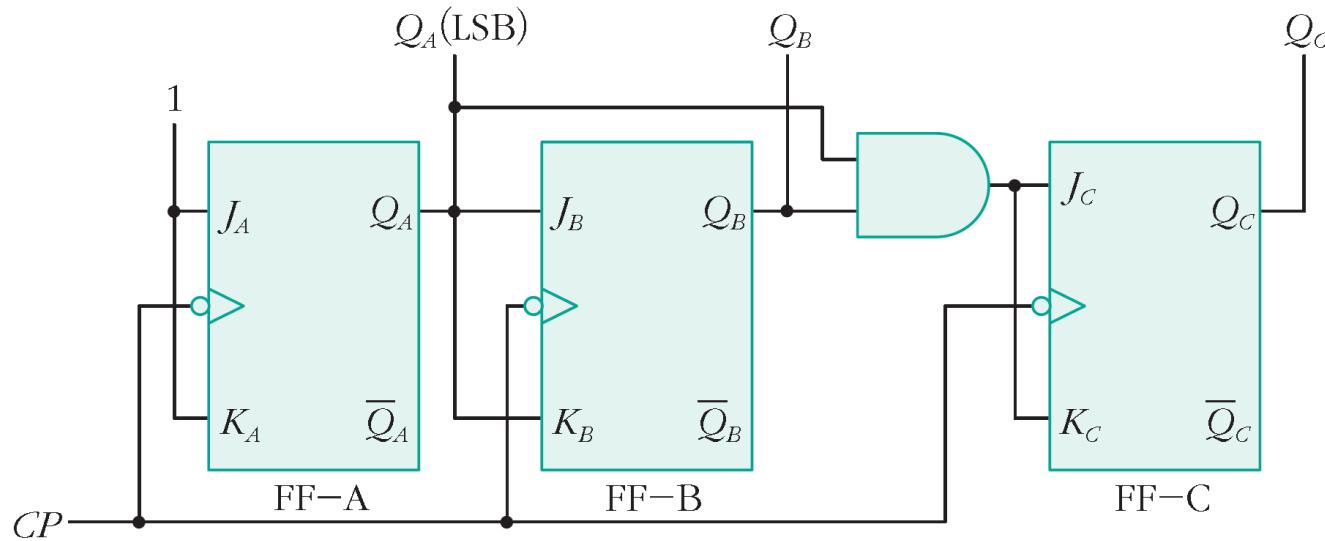


그림 3-86 3비트 동기식 2진 카운터 회로도

04 순서 논리 회로

5 레지스터

- **레지스터**(register)는 기본적으로 데이터 비트를 저장하는 소자
- 대부분의 레지스터에서는 D 플립플롭이 사용되며 각 D 플립플롭에 한 비트씩 저장
- 따라서 n 비트 레지스터는 플립플롭 n 개로 구성되며, n 비트의 2진 정보를 저장할 수 있다.

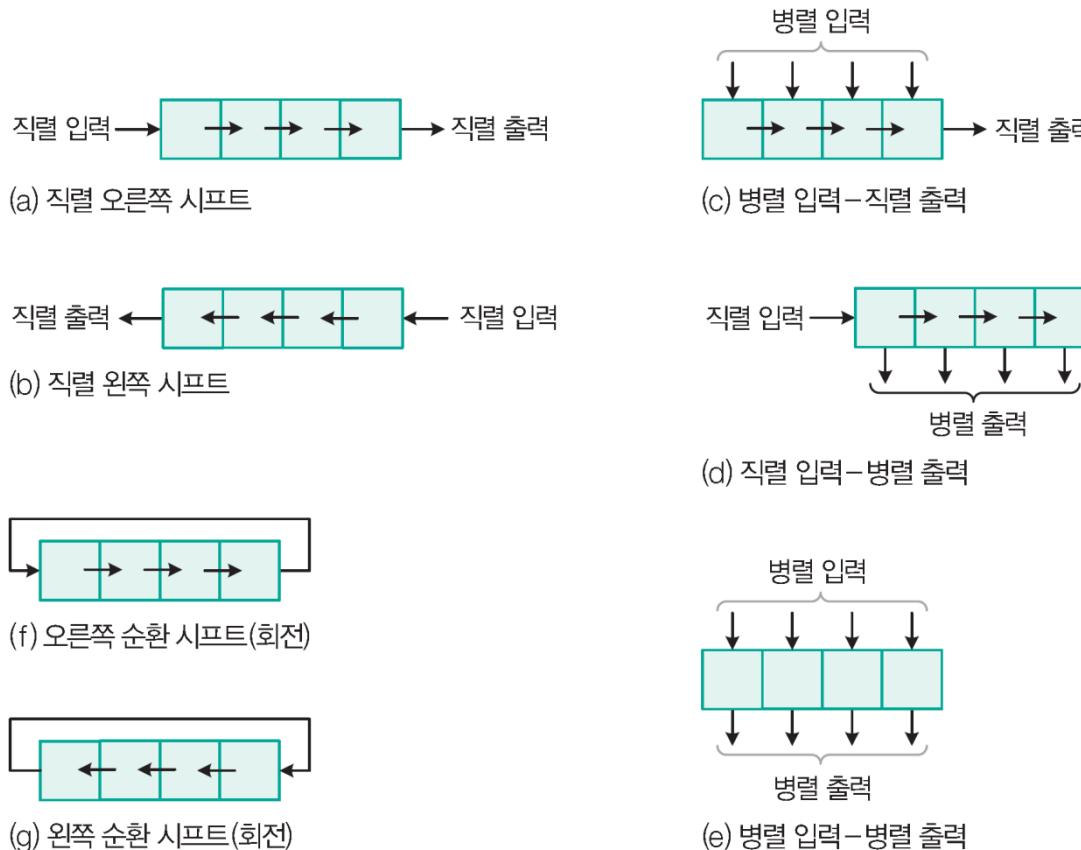


그림 3-87 레지스터 내에서 데이터 이동(4비트인 경우)

□ 4비트 레지스터

- 공통 클록 신호의 상승 에지에서 입력 데이터(I_A, I_B, I_C, I_D)가 D 플립플롭 4개에 동시에 저장되며, 출력(O_A, O_B, O_C, O_D)에서는 언제든지 저장된 데이터를 출력할 수 있다.
- $\overline{CLR} = 0$ 이면 클록에 관계없이 언제든지 모든 플립플롭의 출력을 0으로 만들 수 있다.

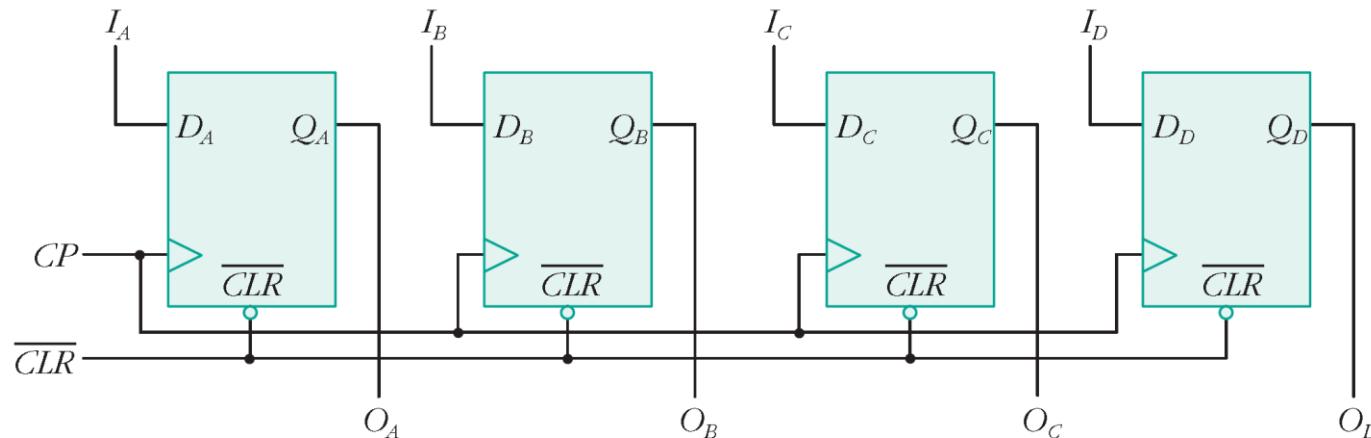


그림 3-88 4비트 레지스터

□ 시프트 레지스터

- 클록 펄스가 입력될 때마다 클록 펄스의 상승 에지에서 입력 데이터가 한 비트씩 오른쪽으로 시프트하면서 저장($I \rightarrow Q_A, Q_A \rightarrow Q_B, Q_B \rightarrow Q_C, Q_C \rightarrow Q_D$)
- 이 과정은 새로운 클록 펄스의 상승 에지마다 반복되므로 네 번째 클록 펄스의 상승 에지에서 처음에 입력된 데이터 비트가 Q_D 에 나타난다.

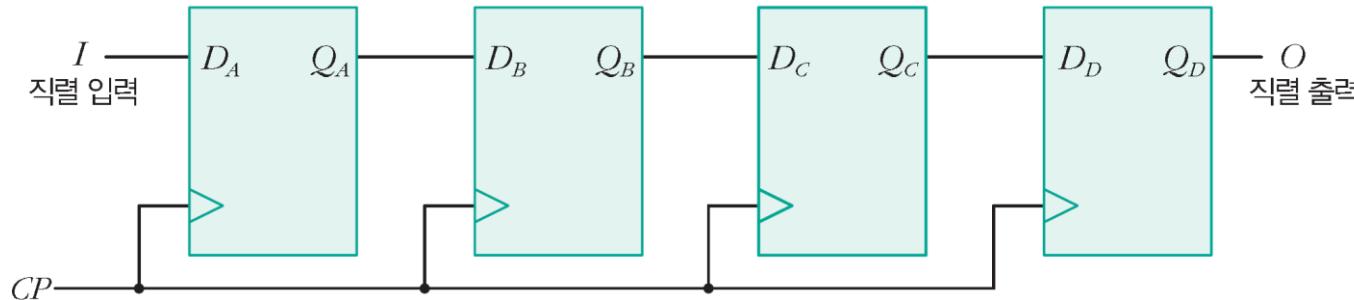


그림 3-89 4비트 시프트 레지스터(직렬 입력 – 직렬 출력)

Summary

- 여러 가지 조합 논리 회로의 동작 원리 이해하고 응용 회로 설계
- 여러 가지 순서 논리 회로의 동작 원리 이해하고 응용 회로 설계