제7장 조합논리회로

7.1 가산기(Adder)

- 조합논리회로는 AND, OR, NOT의 세 가지 기본 논리회로의 조합으로 구성
  - ⇒ 조합논리회로는 입력신호, 논리게이트 및 출력신호로 구성
  - ⇒ 논리게이트는 2진 입력신호를 받아서 2진 출력신호를 생성
- 그림 7.1은 n개의 입력을 받아 m개의 출력을 생성하는 조합논리회로의 블록도
  - ⇒ n개의 입력에 의한 2^n개의 입력조합에 따라 각 출력신호가 결정
  - ⇒ n개의 입력신호의 조합을 진리표로 나타내고 출력함수를 유도
  - ⇒ m개의 출력을 생성하기 위해서는 m개의 논리함수가 필요

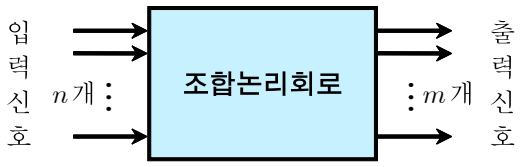


그림 7.1 조합논리회로 블록도

⊙ 반가산기(Half Adder)

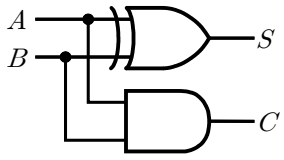
- 한 자리 2진수 2개를 입력하여 합(S; sum)과 캐리(C; carry)를 계산하는 회로
  - ⇒ 캐리 C는 입력 A와 B가 모두 1인 경우에만 1로 발생
  - ⇒ 합 S는 A와 B 둘 중 하나가 1일 때만 1로 발생

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

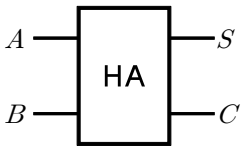
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = A \oplus B,\ \ C = AB$

(a) 진리표와 논리식



(b) 논리회로



(c) 논리기호

그림 7.2 반가산기

⊙ 전가산기(Full Adder)

- 두 개의 2진수 입력 A, B와 캐리 C_i를 포함한 3개의 2진수를 더하는 회로
  ⇒ 캐리 C_i를 고려하여 2진수 3개를 더하는 경우 고려

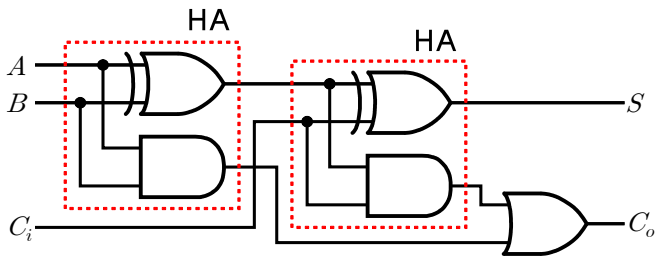
C_i	0	1	0	1	0	1	0	1
A	0	0	1	1	0	0	1	1
$+B$	$+0$	$+0$	$+0$	$+0$	$+1$	$+1$	$+1$	$+1$
$C_o S$	0 0	0 1	0 1	1 0	0 1	1 0	1 0	1 1

- 그림 7.3(a)의 진리표를 전가산기 논리식으로 나타내면 식 (7.1)과 (7.2)으로 표시
  ⇒ 그림 7.3(b)에서 전가산기 회로는 반가산기 2개와 OR 1개로 구성

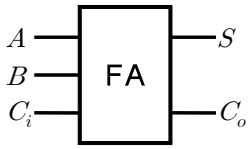
A	B	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S = (A \oplus B) \oplus C_i$
$C_o = C_i(A \oplus B) + AB$

(a) 진리표와 논리식



(b) 논리회로



(c) 논리기호

그림 7.3 전가산기

$$\begin{aligned} S &= A'B'C_i + A'BC_i' + AB'C_i' + ABC_i \\ &= A'(B'C_i + BC_i') + A(B'C_i' + BC_i) \\ &= A'(B \oplus C_i) + A(B \oplus C_i)' = A \oplus (B \oplus C_i) \\ &= (A \oplus B) \oplus C_i \end{aligned} \tag{7.1}$$

$$\begin{aligned} C_o &= A'BC_i + AB'C_i + ABC_i' + ABC_i \\ &= (A'B + AB')C_i + AB(C_i' + C_i) \\ &= (A \oplus B)C_i + AB \end{aligned} \tag{7.2}$$

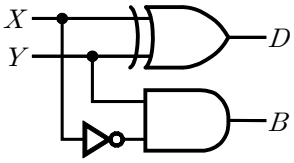
⊙ 반감산기 및 전감산기

- 한 자리 2진수 2개를 입력하여 차(D)와 빌림수(B; borrow)를 계산하는 회로
  ⇒ 빌림수 B는 입력 $X=0$와 $Y=1$인 경우에만 1로 발생
  ⇒ 차 D는 X와 Y 둘 중 하나가 1일 때만 1로 발생

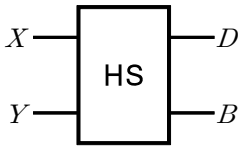
X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$D = X \oplus Y,\ B = X'Y$

(a) 진리표와 논리식



(b) 논리회로



(c) 논리기호

그림 7.2′ 반감산기

- 두 개의 2진수 입력 X, Y와 빌림수 B_i를 포함한 3개의 2진수를 빼는 회로
  ⇒ 그림 7.3′(a)의 진리표를 식 (7.3)과 (7.4)의 논리식으로 표시 가능
  ⇒ 그림 7.3′(b)에서 전감산기는 반가산기 2개와 OR 1개로 구성

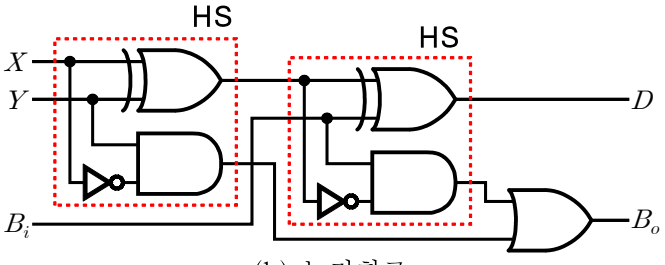
$$\begin{aligned} D &= X'Y'B_i + X'YB_i' + XY'B_i' + XYB_i \\ &= (X \oplus Y) \oplus B_i \end{aligned} \tag{7.3}$$

$$\begin{aligned} B_o &= X'Y'B_i + \underline{X'YB_i'} + \underline{X'YB_i} + XYB_i \\ &= (XY + X'Y')B_i + X'Y(B_i' + B_i) \\ &= (X \oplus Y)'B_i + X'Y \end{aligned} \tag{7.4}$$

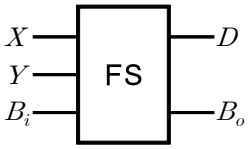
X	Y	B_i	B_o	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$D = (X \oplus Y) \oplus B_i$
$B_o = (X \oplus Y)'B_i + X'Y$

(a) 진리표와 논리식



(b) 논리회로



(c) 논리기호

그림 7.3′ 전감산기

⊙ 병렬가감산기(Parallel-Adder/Subtracter)

- 그림 7.4는 4개의 전가산기를 종속 접속하여 구성한 4비트 병렬가산기를 표시
  - ⇒ 전가산기의 출력 캐리가 상위 전가산기의 입력 캐리로 연결
  - ⇒ 최하위 입력 캐리 $C_0=0$, 최상위 캐리는 C_4, 합은 $S_3S_2S_1S_0$

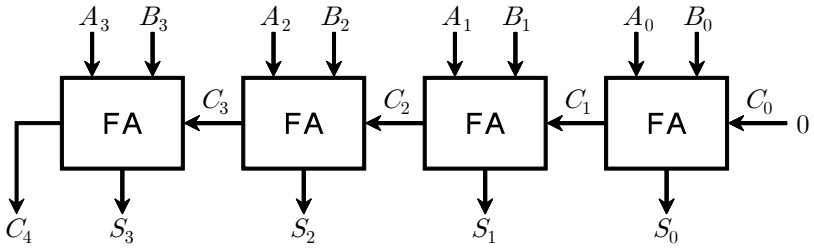


그림 7.4 전가산기를 이용한 병렬가산기

- 그림 7.5는 병렬가산기에 XOR를 추가하여 구현된 4비트 병렬가감산기를 표시
  - ⇒ 입력 B_i를 부호 S와 XOR 연산하여 전가산기의 입력으로 사용
  - ⇒ $S=0$이면 $B_i \oplus S = B_i \oplus 0 = B_i$, $C_0=0$이고 FA는 $A_i \oplus B_i$를 수행
  - ⇒ $S=1$이면 $B_i \oplus S = B_i \oplus 1 = B_i'$, $C_0=1$이고 FA는 $A_i \oplus B_i'$을 수행
  - ⇒ $S=1$이면 B_i는 반전, 즉 $B_3B_2B_1B_0$은 1의 보수가 취해져 입력
  - ⇒ $C_0=1$이 더해져 $B_3B_2B_1B_0$는 2의 보수가 취해져 $A_3A_2A_1A_0$와 합산
  - ⇒ $S=0$이면 회로는 가산기, $S=1$이면 회로는 감산기로 동작

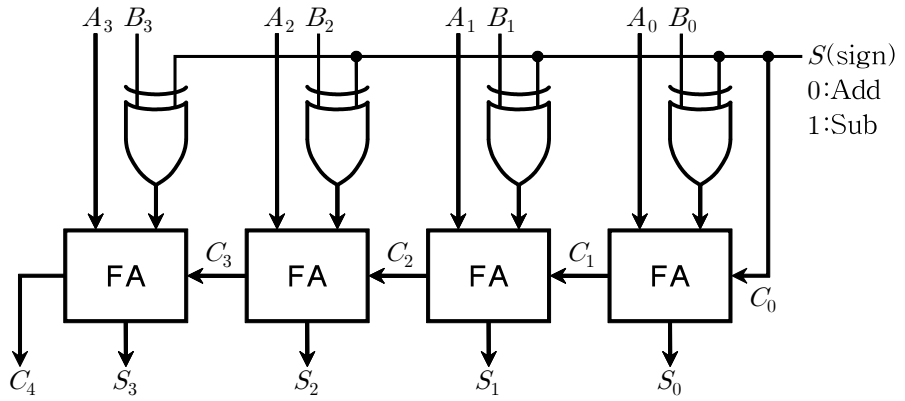
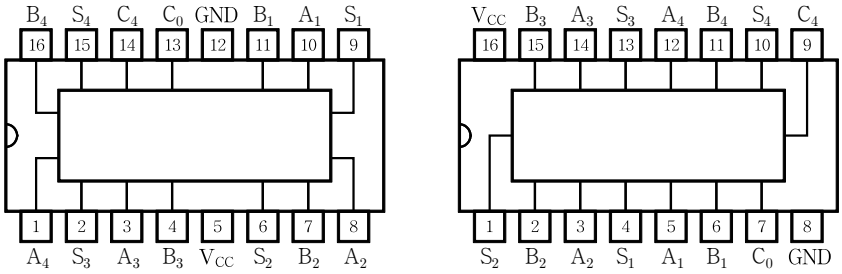


그림 7.5 병렬가감산기(Parallel-Adder/Subtracter)

- 출력 S_3와 캐리 C_4는 모든 단을 통해 캐리가 전파된 후에야 최종 출력이 발생
  - ⇒ 리플-캐리(ripple carry)에 의한 시간지연을 해결한 가산기
  - ⇒ 캐리예측가산기(CLA; Carry-Lookahead-Adder), IC 7483, 74283

- 그림 7.6은 4비트 캐리예측가산기인 IC 7483과 74283의 핀 배치도를 표시



(a) IC 7483 핀 배치도 (b) IC 74283 핀 배치도

그림 7.6 4비트 2진 덧셈기의 핀 배치도

【예제 7.1】 4비트 병렬가산기인 IC 7483 2개를 사용하여 8비트 병렬가산기를 구성하라.

- 그림 7.7과 같이 하위 가산기의 출력 캐리를 상위 가산기의 입력 캐리로 연결
  ⇒ 하위 가산기는 캐리 입력이 없으므로 13번 핀은 접지

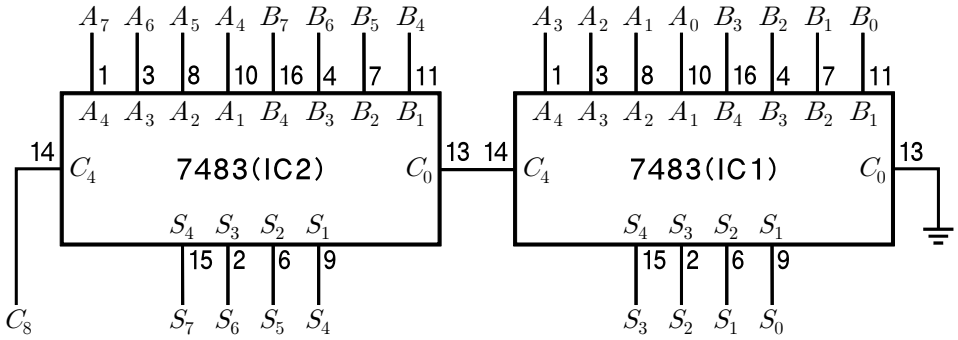


그림 7.7 두 개의 7483 IC를 연결한 8비트 병렬 2진 가산기

■

⊙ BCD 가산기

- 2진수 합이 1001(9) 이하이면 캐리가 미발생, 2진수 합과 BCD 합은 동일
  ⇒ 2진수 합이 1010(10) 이상이면 BCD에서는 캐리 발생이 필요
  ⇒ 2진수 합에 0110(6)을 더하여 캐리를 보정
- 표 7.1에서는 2진수 및 BCD 합에 의한 0~19까지의 10진수를 표시
  ⇒ K, Z_8, Z_4, Z_2, Z_1은 4비트 2진 가산기의 출력
  ⇒ 2진수 합을 BCD 합으로 코드 변환하여 출력

표 7.1 BCD 덧셈표

2진수 합					BCD 합					10진수
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

- 표 7.1에서 BCD 합의 캐리에 대한 논리식을 구하기 위해 그림 7.8의 맵을 작성

  ⇒ Z_8과 Z_4 또는 Z_8과 Z_2이 동시에 1, 또는 $K=1$인 경우에 $C=1$

  ⇒ BCD 합의 캐리 발생에 대한 논리식은 식 (7.5)로 표시

$$C = K + Z_8Z_4 + Z_8Z_2 \tag{7.5}$$

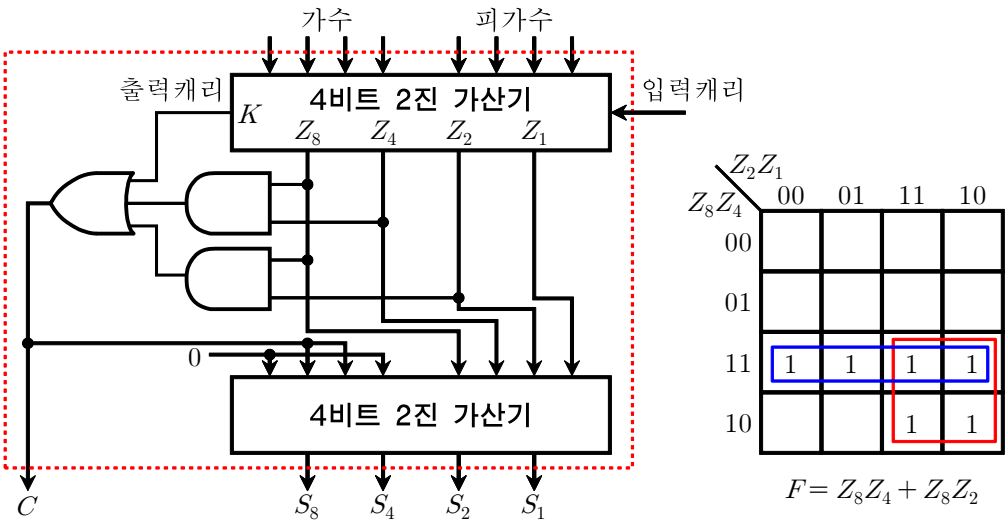


그림 7.8 4비트 BCD 가산기

7.2 비교기(Comparator)

- 2진 비교기는 두 2진수 값의 크기를 비교하는 회로이며 1비트 비교기를 고찰
  ⇒ 표 7.2는 1비트 비교기의 진리표, 그림 7.9는 논리회로를 표시
  ⇒ 1비트 비교기의 각 출력 논리식은 식 (7.6)으로 표시

$$F_1 = (A \oplus B)' = A \odot B, \quad F_2 = A \oplus B, \quad F_3 = AB', \quad F_4 = A'B \tag{7.6}$$

표 7.2 1비트 비교기의 진리표

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

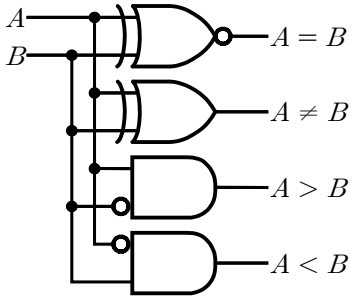


그림 7.9 1비트 비교기

- 표 7.3은 2비트 비교기의 진리표, 그림 7.10은 2비트 비교기의 논리회로를 표시
  ⇒ 진리표로부터 4변수 카르노맵을 작성하여 논리식을 유도

표 7.3 2비트 비교기의 진리표

입력		출력				입력		출력			
A A_2A_1	B B_2B_1	$A=B$ F_1	$A\neq B$ F_2	$A>B$ F_3	$A<B$ F_4	A A_2A_1	B B_2B_1	$A=B$ F_1	$A\neq B$ F_2	$A>B$ F_3	$A<B$ F_4
0 0	0 0	1	0	0	0	1 0	0 0	0	1	1	0
	0 1	0	1	0	1		0 1	0	1	1	0
	1 0	0	1	0	1		1 0	1	0	0	0
	1 1	0	1	0	1		1 1	0	1	0	1
0 1	0 0	0	1	1	0	1 1	0 0	0	1	1	0
	0 1	1	0	0	0		0 1	0	1	1	0
	1 0	0	1	0	1		1 0	0	1	1	0
	1 1	0	1	0	1		1 1	1	0	0	0

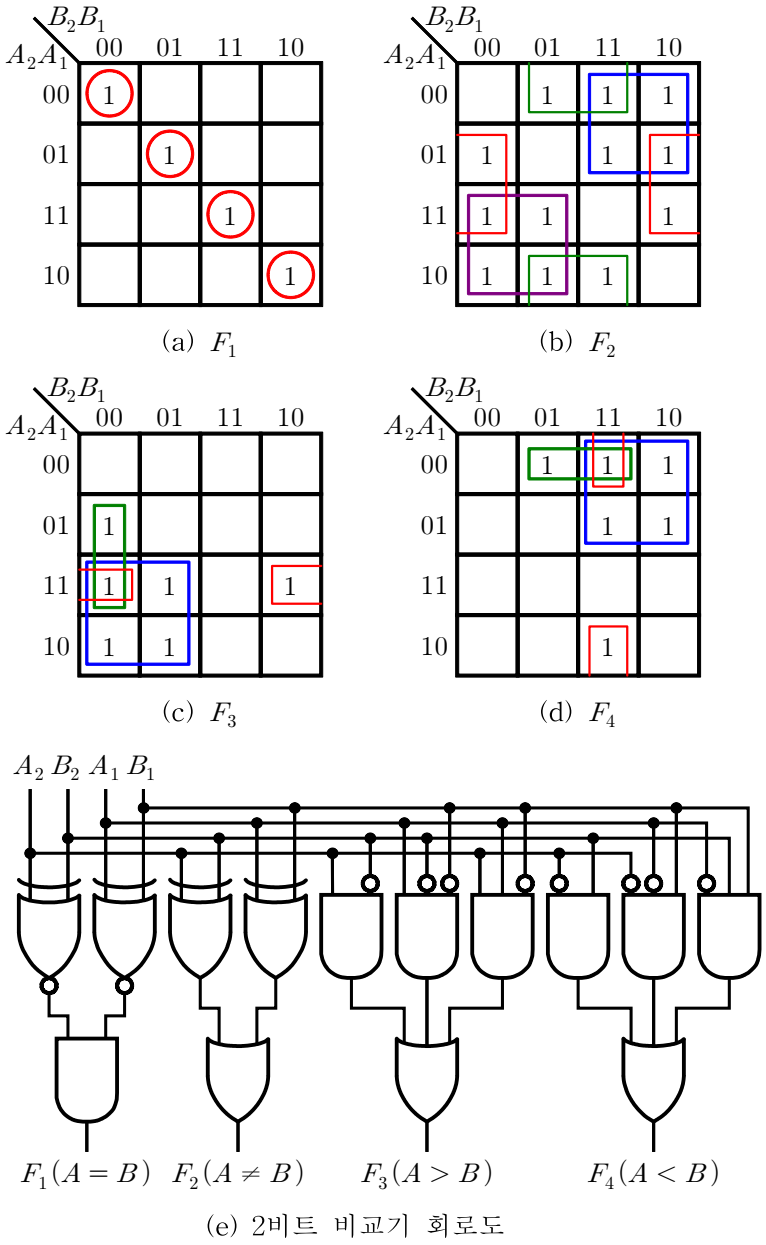


그림 7.10 2비트 비교기 회로 설계 과정과 회로도

$$\begin{aligned} F_1 &= A_2'A_1'B_2'B_1' + A_2'A_1B_2'B_1 + A_2A_1B_2B_1 + A_2A_1'B_2B_1' \\ &= A_2'B_2'(A_1'B_1' + A_1B_1) + A_2B_2(A_1B_1 + A_1'B_1') \\ &= A_2'B_2'(A_1 \oplus B_1)' + A_2B_2(A_1 \oplus B_1)' = (A_2'B_2' + A_2B_2)(A_1 \oplus B_1)' \\ &= (A_2 \oplus B_2)'(A_1 \oplus B_1)' = (A_2 \odot B_2)(A_1 \odot B_1) \end{aligned} \tag{7.7}$$

$$F_2 = A_2B_2' + A_2'B_2 + A_1B_1' + A_1'B_1 = (A_2 \oplus B_2) + (A_1 \oplus B_1) \tag{7.8}$$

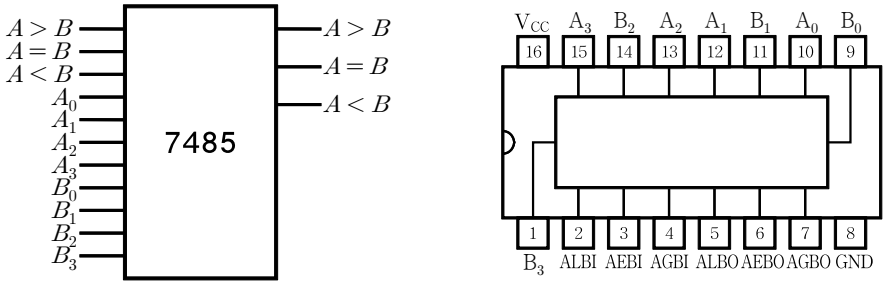
$$F_3 = A_2B_2' + A_1B_2'B_1' + A_2A_1B_1' \tag{7.9}$$

$$F_4 = A_2'B_2 + A_2'A_1'B_1 + A_1'B_2B_1 \tag{7.10}$$

- 표 7.4는 4비트 비교기인 IC 7485의 진리표, 음영부분은 비정상 입력을 표시
  ⇒ 그림 7.11에서 $AGBI$, $ALBI$, $AEBI$는 4비트 이상의 비교에 사용

표 7.4 4비트 비교기 IC 7485의 진리표

입력							출력		
A_3,B_3	A_2,B_2	A_1,B_1	A_0,B_0	$AGBI$ $A>B$	$ALBI$ $A<B$	$AEBI$ $A=B$	$AGBO$ $A>B$	$ALBO$ $A<B$	$AEBO$ $A=B$
$A_3>B_3$	×	×	×	×	×	×	1	0	0
$A_3<B_3$	×	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2>B_2$	×	×	×	×	×	1	0	0
$A_3=B_3$	$A_2<B_2$	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	×	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	×	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	×	×	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	0	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	1	0	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	1	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	1	1	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	0	1	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	1	1	0	0	1
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	1	1	0	0	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	0	1	1	0



(a) 블록도 (b) 핀 배치도

그림 7.11 IC 7485 크기 비교기

- 그림 7.12는 다단계 입력을 이용하여 16비트를 비교할 수 있도록 연결한 회로
  ⇒ A_3와 B_3가 MSB, A_0와 B_0가 LSB, $A>B$일 때 $AGBO$의 출력이 1
  ⇒ $A<B$일 때 $ALBO$의 출력이 1, $A=B$일 때 $AEBO$의 출력이 1
  ⇒ $AGBO$, $ALBO$, $AEBO$의 출력은 윗단의 $AGBI$, $ALBI$, $AEBI$에 연결
  ⇒ 맨 아랫단의 $AGBI$, $ALBI$는 0, $AEBI$는 1을 입력

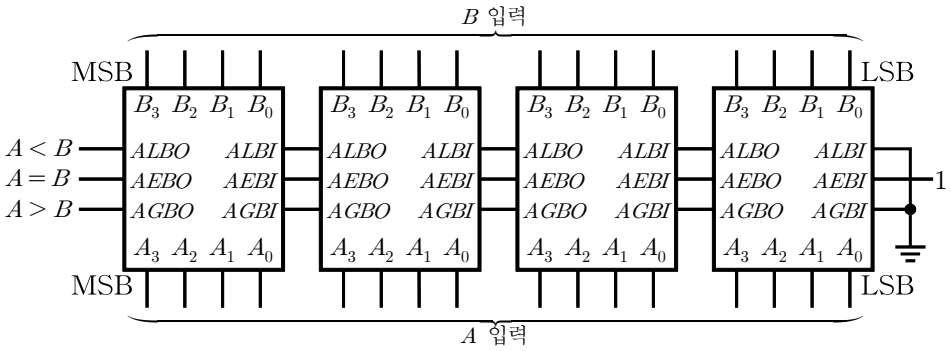


그림 7.12 IC 7485를 이용한 16비트

【예제 7.2】 4비트 비교기인 IC 7485를 사용하여 각 입력파형에 대한 출력파형을 그려라.

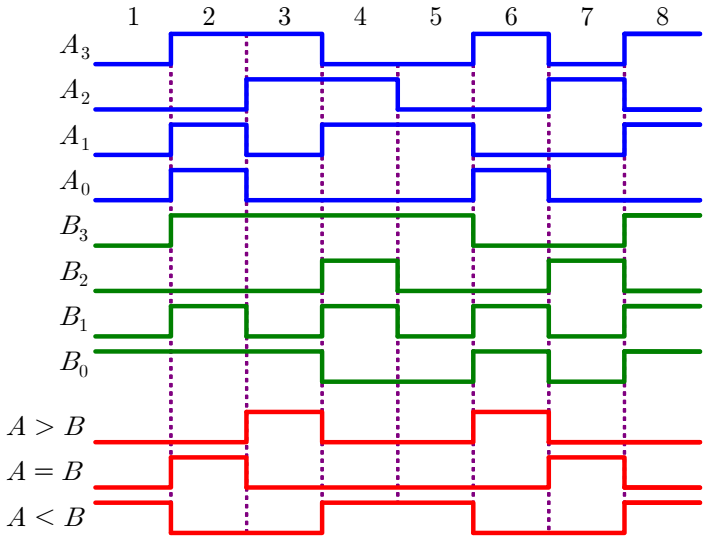


그림 7.13 4비트 비교기의 입출력 파형 예

- 구간1 : $A_3A_2A_1A_0 = 0000,\ B_3B_2B_1B_0 = 0001$이므로 $A < B$
- 구간2 : $A_3A_2A_1A_0 = 1011,\ B_3B_2B_1B_0 = 1011$이므로 $A = B$
- 구간3 : $A_3A_2A_1A_0 = 1100,\ B_3B_2B_1B_0 = 1001$이므로 $A > B$
- 구간4 : $A_3A_2A_1A_0 = 0110,\ B_3B_2B_1B_0 = 1110$이므로 $A < B$
- 구간5 : $A_3A_2A_1A_0 = 0010,\ B_3B_2B_1B_0 = 1000$이므로 $A < B$
- 구간6 : $A_3A_2A_1A_0 = 1001,\ B_3B_2B_1B_0 = 0011$이므로 $A > B$
- 구간7 : $A_3A_2A_1A_0 = 0100,\ B_3B_2B_1B_0 = 0100$이므로 $A = B$
- 구간8 : $A_3A_2A_1A_0 = 1010,\ B_3B_2B_1B_0 = 1011$이므로 $A < B$

■

7.3 디코더(Decoder)

- n비트 2진 입력 정보를 2^n개의 출력으로 변환할 수 있도록 설계된 조합회로

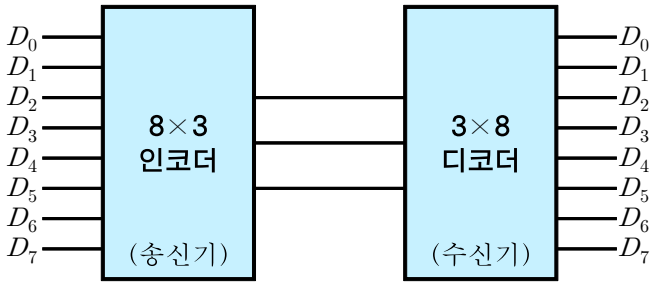


그림 7.14 디코더와 인코더의 기능

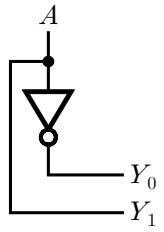
⊙ 1×2 디코더

- 1개의 입력과 2개의 출력으로 구성, 입력에 따라 2개의 출력 중 1개가 선택
  ⇒ 그림 7.15는 1×2 라인 디코더의 진리표와 논리회로를 표시
  ⇒ 그림 7.16은 Enable(E) 입력이 있는 1×2 라인 디코더를 표시

입력	출력	
A	Y_1	Y_0
0	0	1
1	1	0

$Y_0 = A'$, $Y_1 = A$

(a) 진리표와 논리식



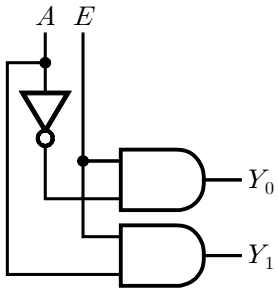
(b) 논리회로

그림 7.15 1×2 디코더

입력		출력	
E	A	Y_1	Y_0
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

$Y_0 = EA'$, $Y_1 = EA$

(a) 진리표와 논리식



(b) 논리회로

그림 7.16 Enable 입력이 있는 1×2 디코더

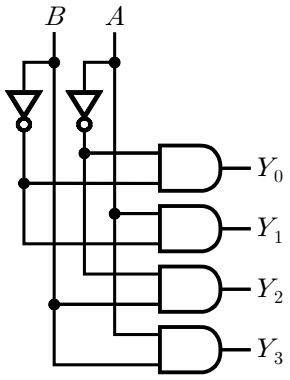
⊙ 2×4 디코더

- 2개의 입력과 4개의 출력으로 구성, 입력에 따라 4개의 출력 중 1개가 선택
  ⇒ 그림 7.17은 2×4 라인 디코더의 진리표와 논리회로를 표시

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

$Y_0 = B'A', \quad Y_1 = B'A$
$Y_2 = BA', \quad Y_3 = BA$

(a) 진리표와 논리식



(b) 논리회로

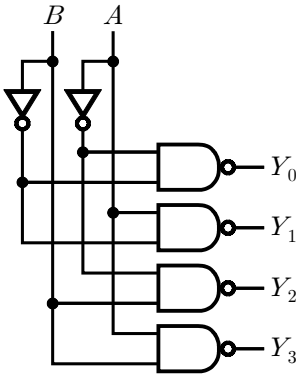
그림 7.17 2×4 디코더

- 그림 7.18은 2×4 라인 디코더를 나타내며, 실제 IC들은 NAND 게이트로 구성
  ⇒ 입력에 의해 선택된 출력선만 출력이 0, 나머지 출력은 1

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

$Y_0 = (B'A')', \quad Y_1 = (B'A)'$
$Y_2 = (BA')', \quad Y_3 = (BA)'$

(a) 진리표와 논리식



(b) 논리회로

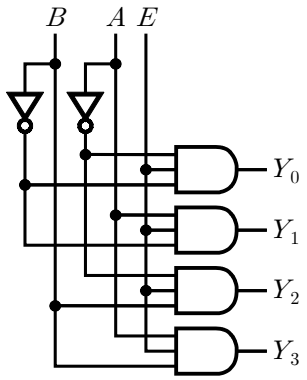
그림 7.18 2×4 NAND 디코더

- 그림 7.19는 제어 입력인 Enable 입력을 갖는 2×4 라인 디코더를 표시
  ⇒ $E=0$이면 회로가 동작하지 않고 $E=1$일 때만 동작

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

$Y_0 = EB'A', \quad Y_1 = EB'A$
$Y_2 = EBA', \quad Y_3 = EBA$

(a) 진리표와 논리식



(b) 논리회로

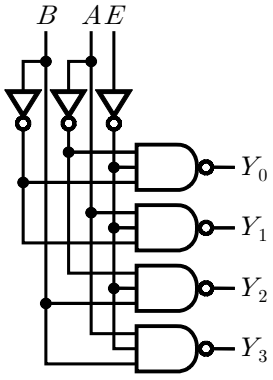
그림 7.19 Enable 입력이 있는 2×4 디코더

- 그림 7.20은 제어 입력인 Enable 입력을 갖는 NAND 2×4 디코더를 표시
  ⇒ $E=1$이면 회로가 동작하지 않고 $E=0$일 때만 동작
  ⇒ IC 74139에는 Enable 입력을 갖는 2×4 디코더가 2개 포함

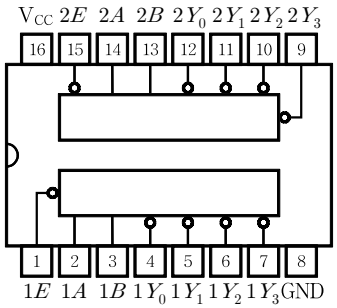
입력			출력			
E	B	A	Y_3	Y_2	Y_1	Y_0
1	×	×	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

$Y_0 = (E'B'A')', \quad Y_1 = (E'B'A)'$
$Y_2 = (E'BA')', \quad Y_3 = (E'BA)'$

(a) 진리표와 논리식



(b) 논리회로



(c) IC 74139 핀 배치도

그림 7.20 Enable 입력이 있는 NAND 2×4 디코더

⊙ 3×8 디코더

- 3개의 입력과 8개의 출력으로 구성, 입력에 따라 8개의 출력 중 1개가 선택
  ⇒ 표 7.5는 3×8 디코더의 진리표, 그림 7.21은 논리회로를 표시
  ⇒ 각 출력은 3입력 변수의 최소항 중의 하나를 표시

표 7.5 3×8 라인 디코더의 진리표

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

$$Y_0 = C'B'A', \quad Y_1 = C'B'A, \quad Y_2 = C'BA', \quad Y_3 = C'BA$$
$$Y_4 = CB'A', \quad Y_5 = CB'A, \quad Y_6 = CBA', \quad Y_7 = CBA$$

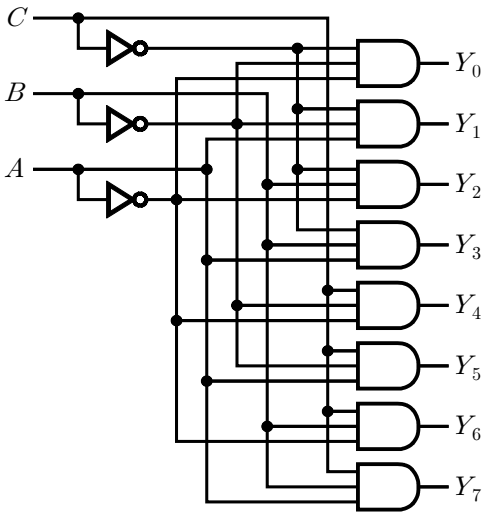
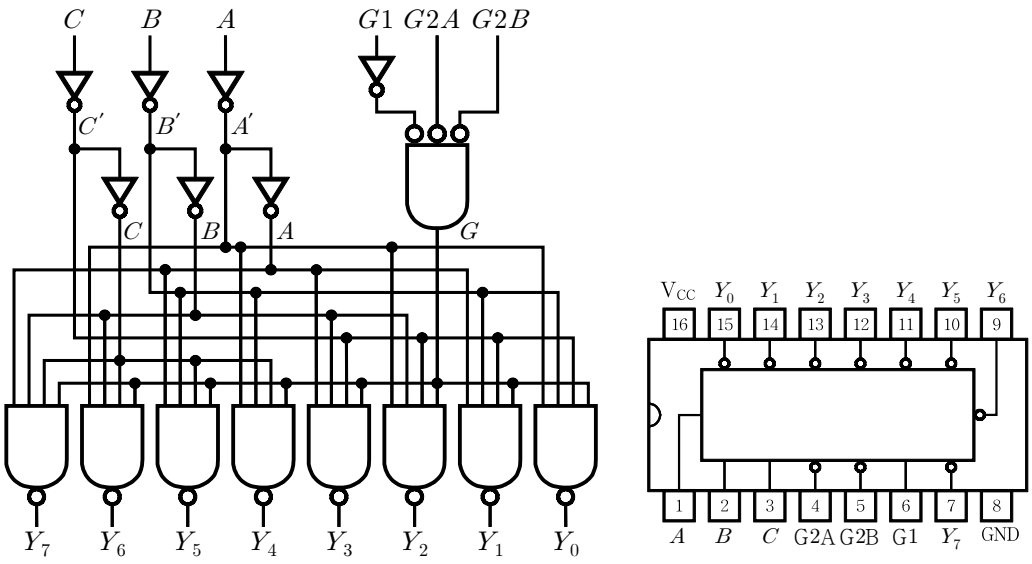


그림 7.21 3×8 디코더의 논리회로

- 표 7.6는 디코더 IC 74138의 진리표, 그림 7.22은 논리회로와 핀 배치도를 표시
  ⇒ 3개의 Enable을 가지며, $G1$은 1, $G2A$와 $G2B$는 0일 때 동작
  ⇒ 8개의 출력 $Y_7 \sim Y_0$은 활성화되었을 때 0(active-low)
- $G1=1$, $G2A=G2B=0$이면 입력 C, B, A에 의해 8개의 출력 중 1개가 선택
  ⇒ 선택된 라인의 출력은 0, 선택되지 않은 나머지 출력은 1
  ⇒ $G1=0$이거나 $G2A=1$ 또는 $G2B=1$이면 모든 출력은 1

표 7.6 Enable이 있는 3×8 디코더 IC 74138의 진리표

입력						출력							
C	B	A	$G1$	$G2A$	$G2B$	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	1	0	0	1	1	1	1	1	1	1	0
0	0	1	1	0	0	1	1	1	1	1	1	0	1
0	1	0	1	0	0	1	1	1	1	1	0	1	1
0	1	1	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	0	0	1	1	0	1	1	1	1	1
1	1	0	1	0	0	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	1	1	1	1	1	1	1
×	×	×	0	×	×	1	1	1	1	1	1	1	1
×	×	×	×	1	×	1	1	1	1	1	1	1	1
×	×	×	×	×	1	1	1	1	1	1	1	1	1



(a) 내부 회로도 (b) 핀 배치도

그림 7.22 3×8 디코더 IC 74138 내부 회로도 및 핀 배치도

⊙ 4×16 디코더

- 그림 7.24와 같이 3×8 디코더 2개를 이용하여 4×16 디코더를 구성 가능
  ⇒ $D=0$이면 위쪽 디코더, $D=1$이면 아래쪽의 디코더가 활성화
  ⇒ $D=0$이면 아래쪽 디코더, $D=1$이면 위쪽 디코더 출력이 모두 0

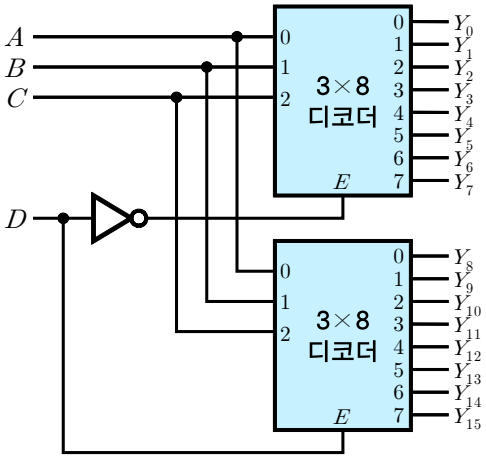


그림 7.24 4×16 디코더

- 그림 7.25의 74154는 $G_1=G_2=0$일 때 입력 D, C, B, A에 따라 반전되어 출력
  ⇒ 그림 7.26은 2×4 디코더 5개를 이용하여 구성한 4×16 디코더

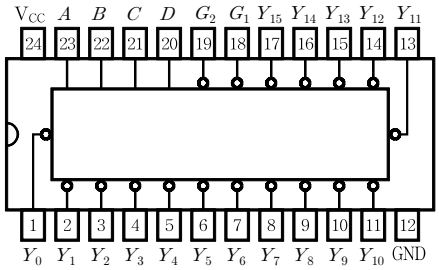


그림 7.25 IC 74154 핀 배치도

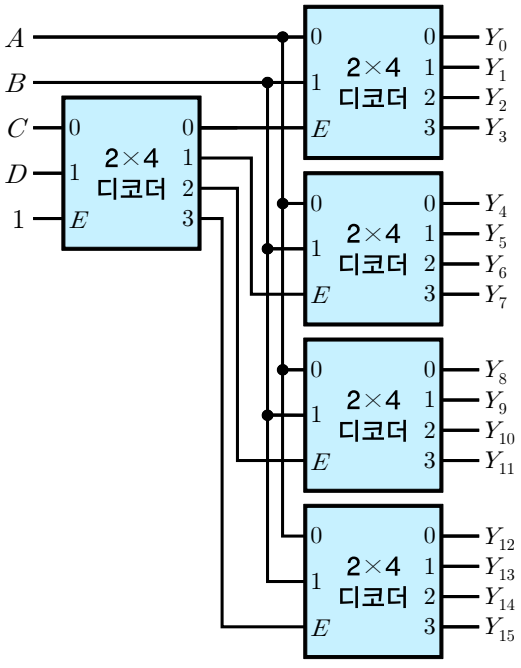
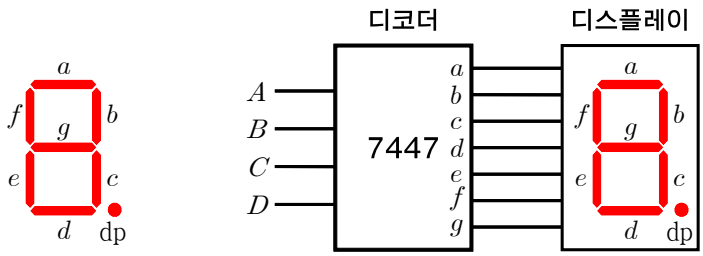


그림 7.26 4×16 디코더

⊙ BCD 7-세그먼트 디코더

- 발광 다이오드를 이용한 응용회로 소자로서 각 세그먼트는 1개의 LED로 구성
  ⇒ 그림 7.27과 같이 각 세그먼트는 a~g를 나타내고 dp는 소수점 표시
  ⇒ 표 7.7은 해당 숫자를 표시하기 위한 7-세그먼트 디코더의 진리표
  ⇒ 디코더는 0일 때 동작하는 active-low



(a) 7-세그먼트 구성 (b) 7-세그먼트와 디코더의 연결

그림 7.27 7-세그먼트의 구성 및 디코더와의 연결

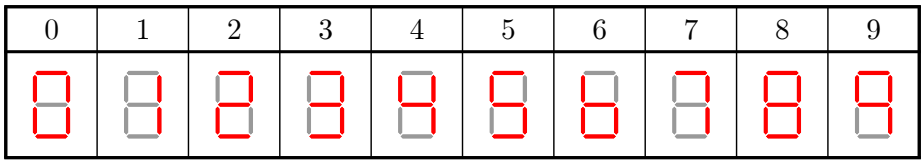


그림 7.28 7-세그먼트의 숫자 표시

표 7.7 7-세그먼트 디코더의 진리표

입력				출력						
D	C	B	A	a'	b'	c'	d'	e'	f'	g'
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	0	×	×	×	×	×	×	×
1	0	1	1	×	×	×	×	×	×	×
1	1	0	0	×	×	×	×	×	×	×
1	1	0	1	×	×	×	×	×	×	×
1	1	1	0	×	×	×	×	×	×	×
1	1	1	1	×	×	×	×	×	×	×

- 그림 7.29에서 각 출력에 대해 카르노맵을 작성하고 간소화하여 논리식을 유도

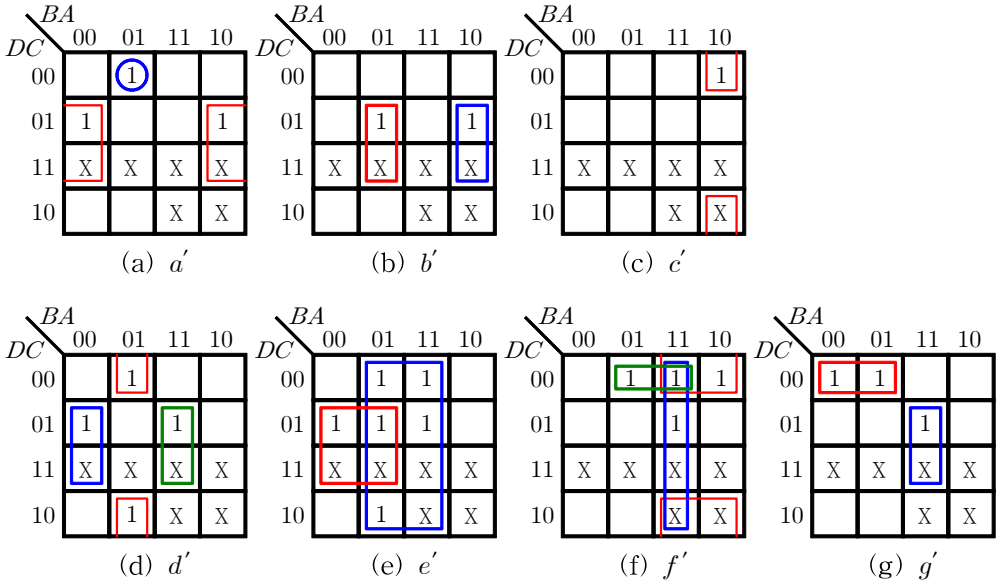


그림 7.29 7-세그먼트 디코더의 논리식 유도 과정

$$a' = D'C'B'A + CA', \quad b' = CB'A + CBA' = C(B \oplus A)$$
$$c' = C'BA', \quad d' = C'B'A + CB'A' + CBA, \quad e' = A + CB'$$
$$f' = BA + C'B + D'C'A, \quad g' = D'C'B' + CBA$$

- 그림 7.30은 논리식을 이용하여 구성한 7-세그먼트 디코더의 논리회로를 표시

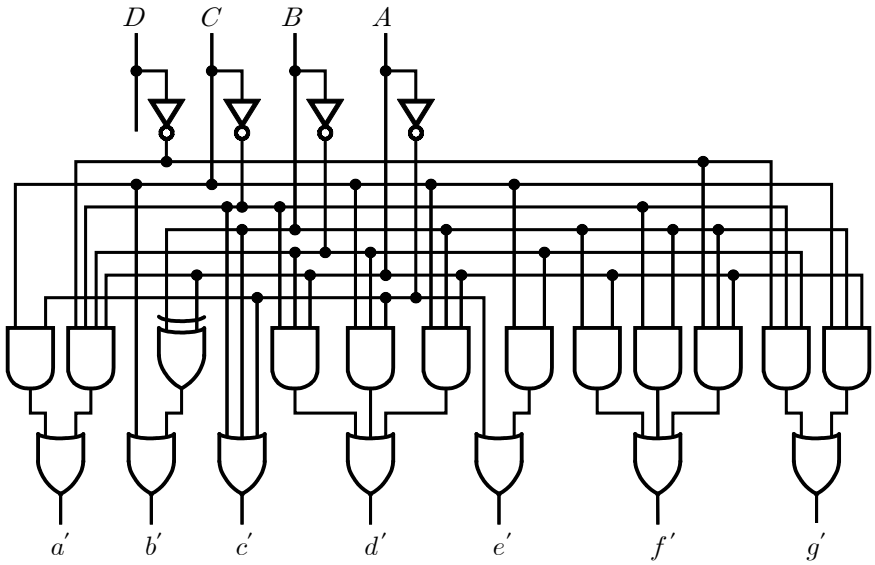


그림 7.30 7-세그먼트 디코더 회로

- 표 7.8은 7-세그먼트 디코더로 주로 사용되는 IC 7447의 진리표를 표시
  ⇒ IC 7447 디코더는 출력이 0일 때 활성화되는 active-low

표 7.8 7-세그먼트 디코더(IC 7447)의 진리표

10진값 기능	입력							출력						
	$\overline{LT}$	$\overline{RBI}$	D	C	B	A	$\overline{BI}/\overline{RBO}$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
0	1	1	0	0	0	0	1	0	0	0	0	0	0	1
1	1	×	0	0	0	1	1	1	0	0	1	1	1	1
2	1	×	0	0	1	0	1	0	0	1	0	0	1	0
3	1	×	0	0	1	1	1	0	0	0	0	1	1	0
4	1	×	0	1	0	0	1	1	0	0	1	1	0	0
5	1	×	0	1	0	1	1	0	1	0	0	1	0	0
6	1	×	0	1	1	0	1	1	1	0	0	0	0	0
7	1	×	0	1	1	1	1	0	0	0	1	1	1	1
8	1	×	1	0	0	0	1	0	0	0	0	0	0	0
9	1	×	1	0	0	1	1	0	0	0	1	1	0	0
10	1	×	1	0	1	0	1	1	1	1	0	0	1	0
11	1	×	1	0	1	1	1	1	1	0	0	1	1	0
12	1	×	1	1	0	0	1	1	0	1	1	1	0	0
13	1	×	1	1	0	1	1	0	1	1	0	1	0	0
14	1	×	1	1	1	0	1	1	1	1	0	0	0	0
15	1	×	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{BI}$	×	×	×	×	×	×	0	1	1	1	1	1	1	1
$\overline{RBI}$	1	0	0	0	0	0	0	1	1	1	1	1	1	1
$\overline{LT}$	0	×	×	×	×	×	1	0	0	0	0	0	0	0

- 램프 테스트(Lamp Test) $\overline{LT}$=0이면 입력에 관계없이 모든 세그먼트가 점등
  ⇒ Default=1(5V)
- Ripple Blanking Input $\overline{RBI}$=1로 하거나 개방 상태로 두면 LED에 0이 표시
  ⇒ $\overline{RBI}$=0일 때 모든 입력이 0이면 모든 세그먼트 소등, Default=1
- $\overline{BI}/\overline{RBO}$은 7-세그먼트를 종속 접속할 때 윗자리에 0이 표시되는 것을 방지
  ⇒ Blanking Input / Ripple Blanking Output, Default=1(5V)
  ⇒ $\overline{BI}/\overline{RBO}$는 모든 입력이 0일 때 모든 세그먼트 소등
- 그림 7.31에서 7447의 입력 *DCBA*가 10~15(1010~1111)이면 그림과 같이 표시

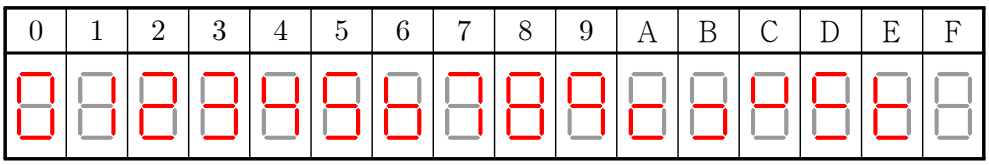
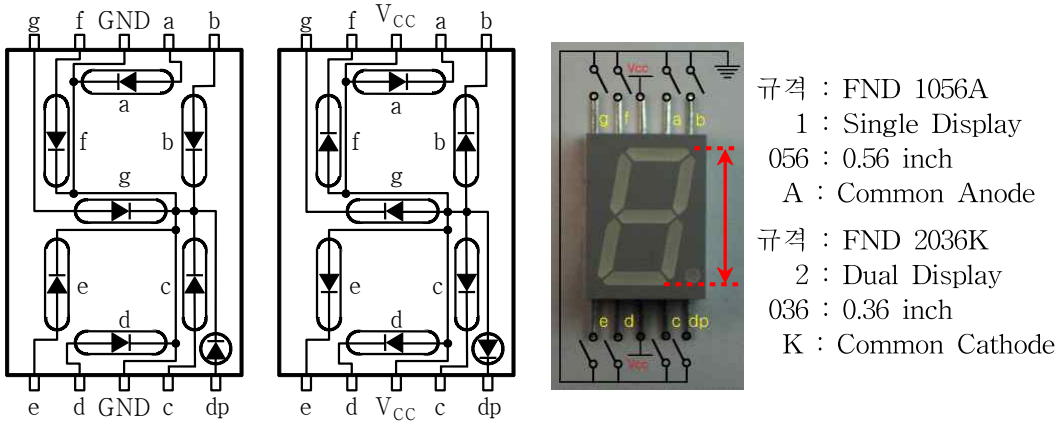


그림 7.31 7447의 LED 점등 패턴도

- 7-세그먼트는 공통 양극 및 공통 음극 방식으로 구분, 전류 제어용 저항 필요
  ⇒ 일반적으로 공통 음극 방식보다 공통 양극 방식이 많이 사용
- 그림 7.32(a)와 같이 공통 음극 방식은 LED의 모든 음극을 공통 단자로 접속
  ⇒ 음극 단자를 0V(접지, GND)에 연결, 양극 단자에 5V의 전압 인가
  ⇒ 공통 음극 LED 표시기를 구동할 때는 IC 7448과 7449를 사용
- 그림 7.32(B)와 같이 공통 양극 방식은 LED의 모든 양극을 공통 단자로 접속
  ⇒ 양극 단자를 5V를 연결, 음극 단자에 0V(접지, GND)의 전압 인가
  ⇒ 공통 양극 LED 표시기를 구동할 때는 IC 7446과 7447을 사용



(a) 공통 음극(CC) (b) 공통 양극(CA) (c) 실물사진

그림 7.32 7-세그먼트 공통 회로

- LED는 5~25mA에서 발광하며 최대 정격전류는 30mA, 전류 제한 저항 필요
  ⇒ 그림 7.33은 전류 제한 저항을 사용한 7-세그먼트 구동회로

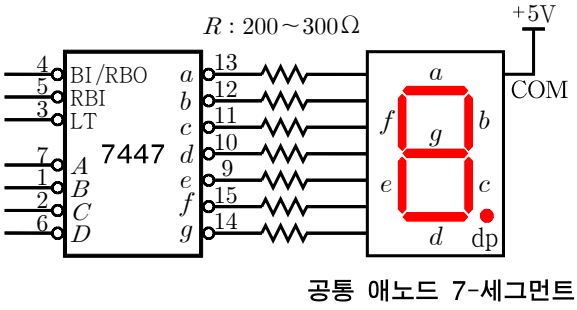


그림 7.33 전류 제한 저항을 사용한 회로

7.4 인코더(Encoder)

- 인코더는 2^n개의 입력 중 활성화된 1비트에 대응하는 n비트의 2진 정보를 출력
  ⇒ 2^n개의 신호를 입력받아서 n비트의 최소항의 번호를 출력
- 10진수를 2진수로 변환하거나 정보 전송을 일정한 규칙에 따라 암호로 변환
  ⇒ 컴퓨터 모니터의 RGB 정보를 TV의 NTSC 방식으로 변환 등

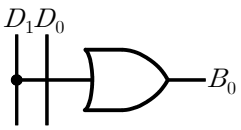
⊙ 2×1 인코더

- 2개의 입력과 1개의 출력으로 구성되며, 입력 신호에 따라 0 또는 1을 출력
  ⇒ 그림 7.34는 2×1 인코더의 진리표와 논리회로를 표시

입력		출력
D_1	D_0	B_0
0	1	0
1	0	1

$B_0 = D_1$

(a) 진리표와 논리식



(b) 논리회로

그림 7.34 2×1 인코더

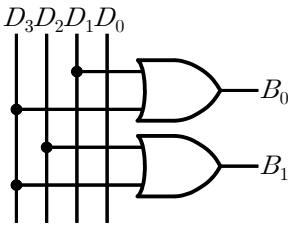
⊙ 4×2 인코더

- 4개의 입력과 2개의 출력으로 구성, 입력에 따라 2개의 2진 조합으로 출력
  ⇒ 그림 7.35는 4×2 인코더의 진리표와 논리회로를 표시

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

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

(a) 진리표와 논리식



(b) 논리회로

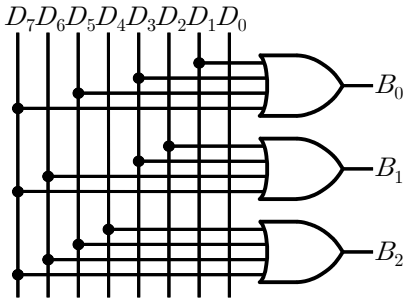
그림 7.35 4×2 인코더

⊙ 8×3 인코더

- 8개의 입력과 3개의 출력으로 구성, 입력에 따라 3개의 2진 조합으로 출력
  ⇒ 진리표에 대한 출력식은 식 (7.11)로 표시

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	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

(a) 진리표



(b) 논리회로

그림 7.36 8×3 인코더

$$\begin{aligned} B_2 &= D_4 + D_5 + D_6 + D_7 \\ B_1 &= D_2 + D_3 + D_6 + D_7 \\ B_0 &= D_1 + D_3 + D_5 + D_7 \end{aligned} \tag{7.11}$$

- $D_3 = D_6 = 1$이면 출력은 $B_2B_1B_0 = 111$, $D_7 = 1$일 때도 출력은 $B_2B_1B_0 = 111$
  ⇒ 인코더의 1개의 입력만이 1이 되도록 입력의 우선순위 결정 필요
  ⇒ 높은 아래첨자의 수를 갖는 입력에 대해 높은 우선순위 부여
  ⇒ $D_3 = D_6 = 1$일 때 D_6에 높은 우선순위 부여하여 $B_2B_1B_0 = 110$
- $D_0 \sim D_7$의 입력이 0이면 출력은 $B_2B_1B_0 = 000$, $D_0 = 1$일 때도 $B_2B_1B_0 = 000$
  ⇒ 적어도 1개의 입력이 1임을 나타내기 위한 1개의 출력을 추가

⊙ 4×2 우선순위 인코더

- 2개 이상의 입력이 1일 때는 가장 큰 아래첨자의 입력에 우선순위를 부여
  ⇒ 2개의 출력 B_1, B_0에 3번째 출력 V를 추가
- 추가된 3번째 출력 V는 1개 이상의 입력이 1일 때 출력이 1이 되도록 설계
  ⇒ 유효출력 지시기(valid-output indicator)
- $V=0$일 때 다른 2개의 출력 B_1, B_0는 조사하지 않고 무관항으로 처리
  ⇒ 출력단의 ×는 무관항, 입력단의 ×는 축약된 형태

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

- $2^4 = 16$개의 입력을 모두 열거하지 않고 진리표에 0과 1 대신에 ×를 사용

  $\Rightarrow$ 입력 D_3가 가장 높은 우선순위를 가지므로 $D_3 = 1$이면 $B_1B_0 = 11$

  $\Rightarrow$ $D_3 = 0$일 때 $D_2 = 1$이면 D_1과 D_0에 관계없이 $B_1B_0 = 10$

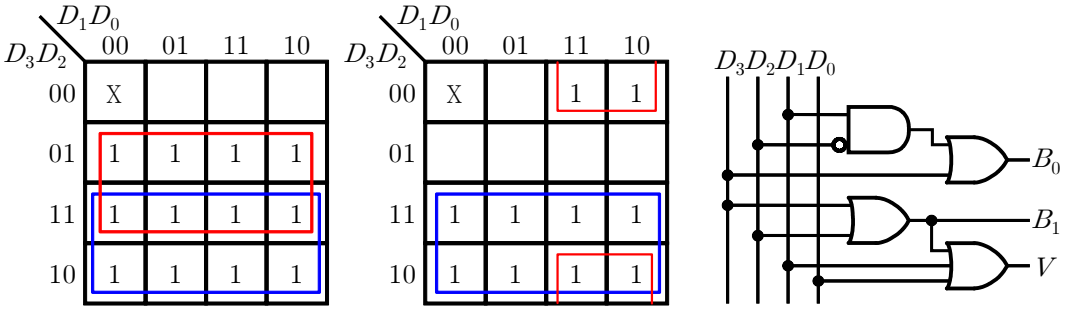
  $\Rightarrow$ $D_3 = 0$, $D_2 = 0$일 때 $D_1 = 1$이면 D_0에 관계없이 $B_1B_0 = 01$

  $\Rightarrow$ $D_3 = 0$, $D_2 = 0$, $D_1 = 0$일 때 $D_0 = 1$이면 $B_1B_0 = 00$

  $\Rightarrow$ $D_3 = D_2 = D_1 = D_0 = 0$이면 출력은 무관항

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

- 우선순위 인코더의 진리표에 대한 카르노맵을 작성하여 간소화된 출력식 유도



$$B_1 = D_3 + D_2, \quad B_0 = D_3 + D_2'D_1, \quad V = D_3 + D_2 + D_1 + D_0 \tag{7.12}$$

⊙ 8×3 우선순위 인코더

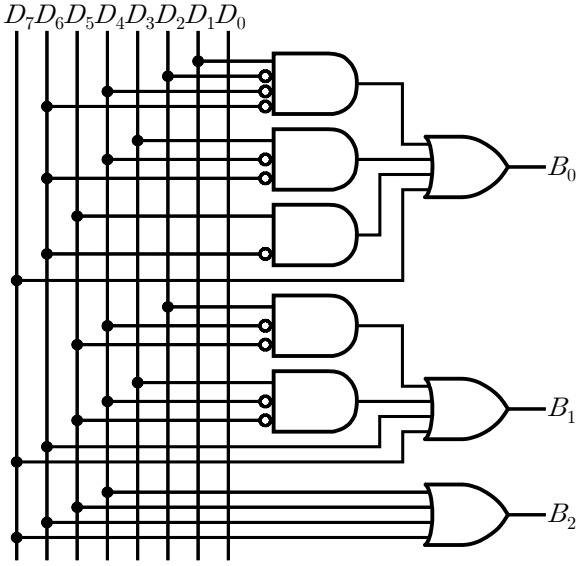
- 그림 7.37은 8×3 우선순위 인코더의 진리표와 논리회로를 표시
  ⇒ 그림 7.37(a)의 진리표로부터 식 (7.13)의 논리식을 유도 가능
  ⇒ $D_7 + D_7'D_6 = (D_7 + D_7')(D_7 + D_6) = D_7 + D_6$에 의해 식 (7.14)로 표시

$$\begin{aligned} B_2 &= D_7 + D_7'D_6 + D_7'D_6'D_5 + D_7'D_6'D_5'D_4 \\ B_1 &= D_7 + D_7'D_6 + D_7'D_6'D_5'D_4'D_3 + D_7'D_6'D_5'D_4'D_3'D_2 \\ B_0 &= D_7 + D_7'D_6'D_5 + D_7'D_6'D_5'D_4'D_3 + D_7'D_6'D_5'D_4'D_3'D_2'D_1 \end{aligned} \tag{7.13}$$

$$\begin{aligned} B_2 &= D_7 + D_6 + D_5 + D_4 \\ B_1 &= D_7 + D_6 + D_5'D_4'D_3 + D_5'D_4'D_2 \\ B_0 &= D_7 + D_6'D_5 + D_6'D_4'D_3 + D_6'D_4'D_2'D_1 \end{aligned} \tag{7.14}$$

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	1
0	0	0	0	0	1	×	×	0	1	0
0	0	0	0	1	×	×	×	0	1	1
0	0	0	1	×	×	×	×	1	0	0
0	0	1	×	×	×	×	×	1	0	1
0	1	×	×	×	×	×	×	1	1	0
1	×	×	×	×	×	×	×	1	1	1

(a) 진리표



(b) 논리회로

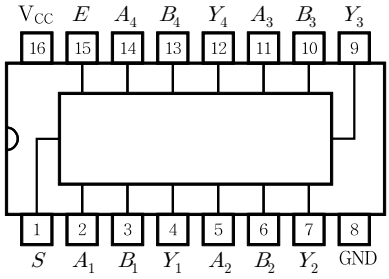
그림 7.37 8×3 우선순위 인코더

⊙ 인코더 IC

- IC 74158은 4개의 2×1 인코더 논리반전 출력, Enable이 1일 때 모든 출력 1
  ⇒ Enable $E=0$일 때 $S=0$이면 $Y=A'$이고, $S=1$이면 $Y=B'$

입력		출력
S	E	Y
×	1	1
0	0	A'
1	0	B'

(a) 진리표



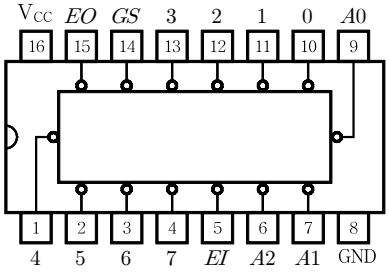
(b) 논리회로

그림 7.38 IC 74158 인코더

- 그림 7.39는 IC 74148은 8×3 우선순위 인코더의 진리표와 핀 배치도를 표시
  ⇒ 데이터 입력중 하나가 0이고 Enable 입력 $EI=0$일 때 $GS=0$
  ⇒ $EI=0$이면 출력은 가장 우선순위가 높은 데이터 입력, $EO=1$
  ⇒ $EI=0$이고 모든 데이터 입력이 1이면 출력 $EO=0$

입력									출력				
EI	7	6	5	4	3	2	1	0	$A2$	$A1$	$A0$	GS	EO
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	×	×	×	×	×	×	×	0	0	0	0	1
0	1	0	×	×	×	×	×	×	0	0	1	0	1
0	1	1	0	×	×	×	×	×	0	1	0	0	1
0	1	1	1	0	×	×	×	×	0	1	1	0	1
0	1	1	1	1	0	×	×	×	1	0	0	0	1
0	1	1	1	1	1	0	×	×	1	0	1	0	1
0	1	1	1	1	1	1	0	×	1	1	0	0	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1

(a) 진리표



(b) 핀 배치도

그림 7.39 IC 74148의 진리표 및 핀 배치도

【예제 7.4】 IC 74147은 10진-BCD 우선순위 인코더이며, 입력과 출력이 모두 active-low로 동작한다. IC 74147을 이용하여 키보드에 있는 10개의 10진 숫자들을 BCD로 변환하는 회로를 구성하여라.

표 7.17 10진-BCD 우선순위 인코더(74147)의 진리표

입력									출력			
I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	Y_3'	Y_2'	Y_1'	Y_0'
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	×	1	1	0	1
1	1	1	1	1	1	0	×	×	1	1	0	0
1	1	1	1	1	0	×	×	×	1	0	1	1
1	1	1	1	0	×	×	×	×	1	0	1	0
1	1	1	0	×	×	×	×	×	1	0	0	1
1	1	0	×	×	×	×	×	×	1	0	0	0
1	0	×	×	×	×	×	×	×	0	1	1	1
0	×	×	×	×	×	×	×	×	0	1	1	0
1	1	1	1	1	1	1	1	0	0	0	0	0

- 풀업 저항 R을 통해 $+V_{CC}$에 연결된 10개의 스위치들은 각 키를 표시
  - ⇒ 풀업 저항에 의해 키가 눌려지지 않았을 때 High 입력
  - ⇒ 키가 눌려지면 입력이 접지되어 인코더 입력에 Low 공급
  - ⇒ 0 이외의 키들 중 아무 것도 누르지 않으면 BCD 코드 0 출력

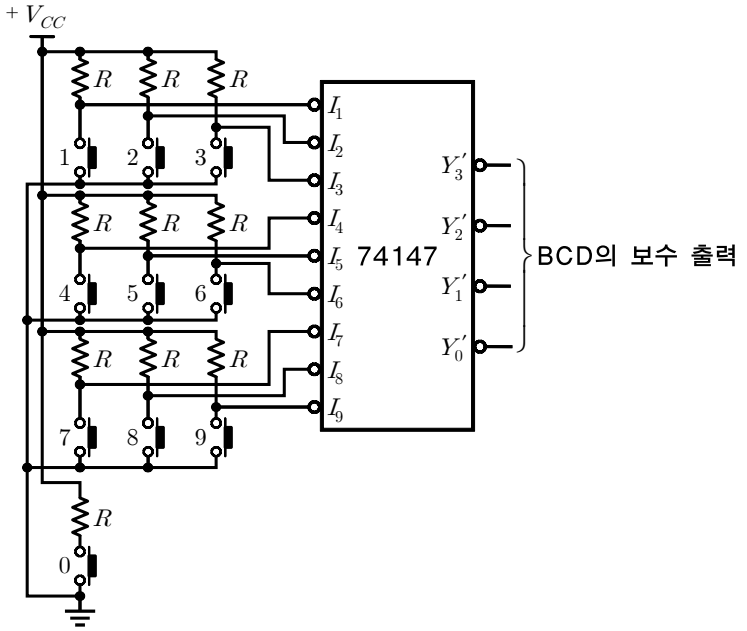


그림 7.40 IC 74147을 이용한 키보드 인코더

■

7.5 멀티플렉서(Multiplexer)

- 다수의 정보장치의 데이터를 소수의 채널이나 선을 통해 선택적으로 전송
  ⇒ 멀티플렉싱(multiplexing)
- 멀티플렉서는 여러 입력선 중 1개를 선택하여 출력선에 연결하는 조합회로
  ⇒ 2^n개의 입력선, n개의 선택선 및 1개의 출력선으로 구성
  ⇒ n개의 선택선의 2진 조합에 의해 입력 중 1개를 선택
- 멀티플렉서는 여러 입력선 중 1개를 선택하여 2진 정보를 출력선으로 전송
  ⇒ 데이터 선택기(data selector)
- 디멀티플렉서는 신호선에서 정보를 받아서 2^n개의 출력선 중의 하나로 전송
  ⇒ 특정 출력의 선택은 n개의 선택선의 비트 조합에 의해 결정

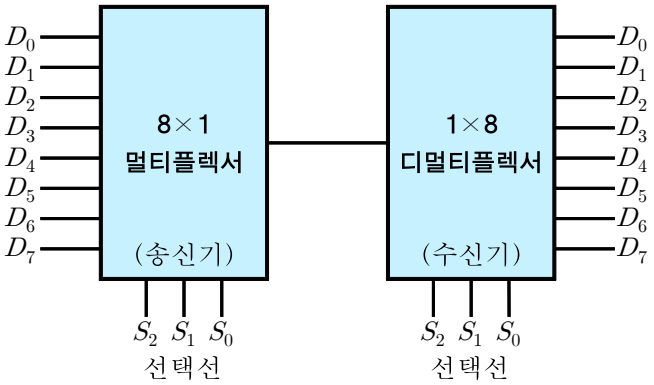


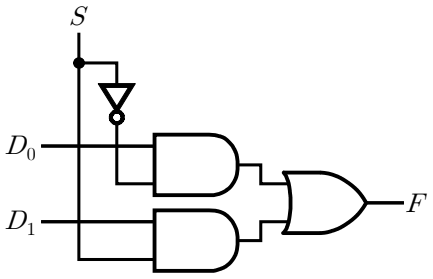
그림 7.41 멀티플렉서와 디멀티플렉서의 역할

⊙ 2×1 멀티플렉서

- 선택선 S의 입력값에 따라 2개의 입력 중 1개를 출력으로 보내주는 조합회로
  ⇒ 그림 7.42의 진리표를 이용하여 논리식을 구하면 $F = S'D_0 + SD_1$

선택선	출력
S	F
0	D_0
1	D_1

(a) 진리표



(b) 논리회로

그림 7.42 2×1 멀티플렉서

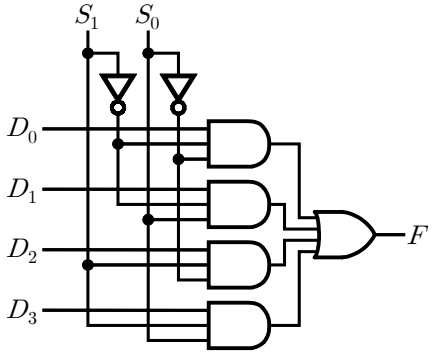
⊙ 4×1 멀티플렉서

- S_1과 S_0의 입력값에 따라 4개의 입력 중 1개를 출력으로 보내주는 조합회로

⇒ 논리식은 $F = S_1'S_0'D_0 + S_1'S_0D_1 + S_1S_0'D_2 + S_1S_0D_3$

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

(a) 진리표



(b) 논리회로

그림 7.43 4×1 멀티플렉서

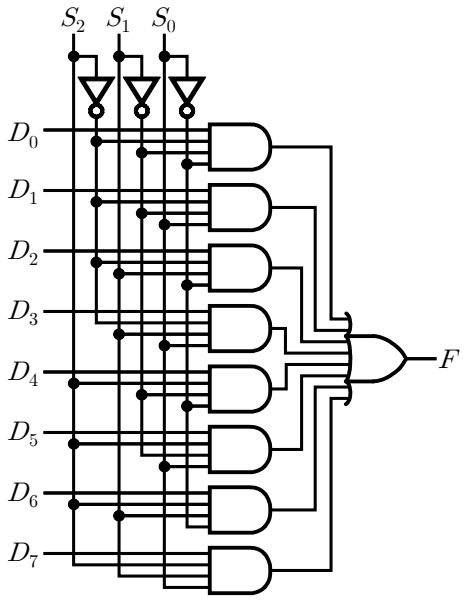
⊙ 8×1 멀티플렉서

- 3개의 선택선에 의해 8개의 입력 중 1개를 선택하여 출력하는 조합회로

⇒ 논리식은 $F = S_2'S_1'S_0'D_0 + S_2'S_1'S_0D_1 + S_2'S_1S_0'D_2 + S_2'S_1S_0D_3$
$+ S_2S_1'S_0'D_4 + S_2S_1'S_0D_5 + S_2S_1S_0'D_6 + S_2S_1S_0D_7$

선택선			출력
S_2	S_1	S_0	F
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

(a) 진리표



(b) 논리회로

그림 7.44 4×1 멀티플렉서

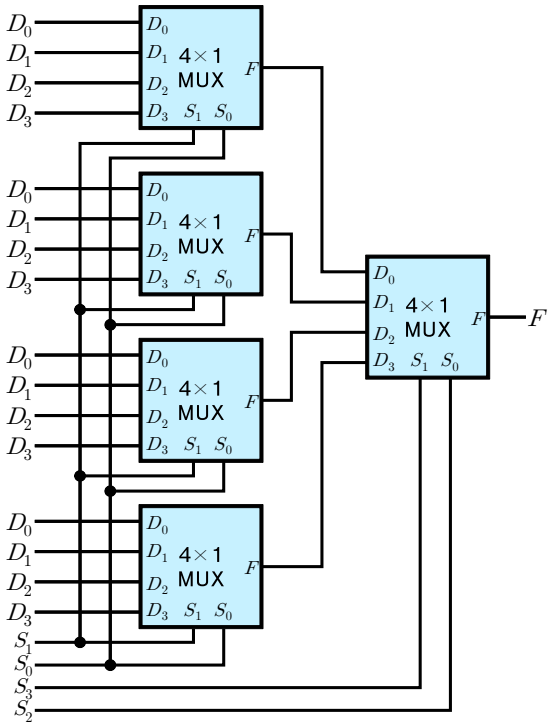


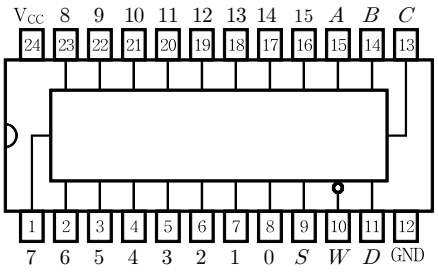
그림 7.45 16×1 멀티플렉서

⊙ 멀티플렉서 IC

- IC 74150은 $E0'$~$E15'$ 중 1개를 선택선 D, C, B, A에 따라 출력으로 전송

Select				Strobe	출력
D	C	B	A	S	W
×	×	×	×	1	1
0	0	0	0	0	$E0'$
0	0	0	1	0	$E1'$
0	0	1	0	0	$E2'$
0	0	1	1	0	$E3'$
0	1	0	0	0	$E4'$
0	1	0	1	0	$E5'$
0	1	1	0	0	$E6'$
0	1	1	1	0	$E7'$
1	0	0	0	0	$E8'$
1	0	0	1	0	$E9'$
1	0	1	0	0	$E10'$
1	0	1	1	0	$E11'$
1	1	0	0	0	$E12'$
1	1	0	1	0	$E13'$
1	1	1	0	0	$E14'$
1	1	1	1	0	$E15'$

(a) 진리표



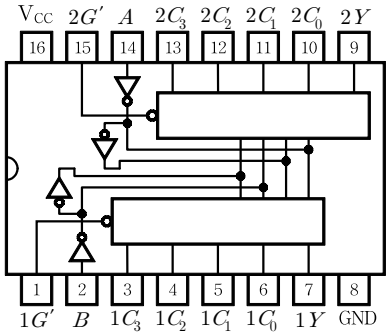
(b) 핀 배치도

그림 7.46 IC 74150의 진리표 및 핀 배치도

- IC 74153은 $C_0 \sim C_3$ 중 1개를 선택선 입력 B, A에 따라 출력으로 전송

Select		Strobe	출력
B	A	G	Y
×	×	1	0
0	0	0	C_0
0	1	0	C_1
1	0	0	C_2
1	1	0	C_3

(a) 진리표



(b) 핀 배치도

그림 7.47 IC 74153의 진리표 및 핀 배치도

【예제 7.5】 그림 7.43(b)의 멀티플렉서에 그림 7.48과 같은 데이터 입력(D_3, D_2, D_1, D_0)과 데이터 선택(S_1, S_0)파형을 입력하였다. 출력 F의 파형을 그려보아라.

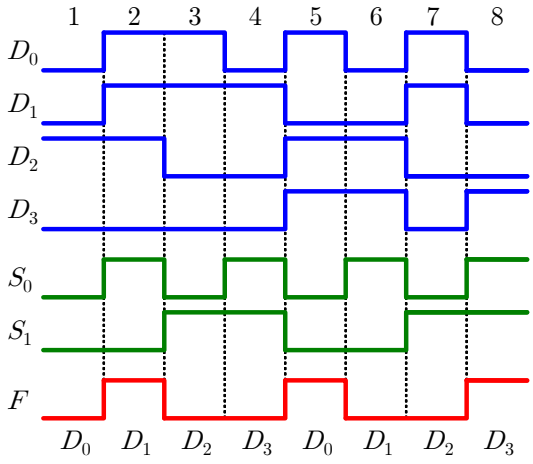


그림 7.48 4×1 멀티플렉서의 입출력 파형

- 데이터 선택 신호 S_1과 S_0에 의해서 데이터 입력 D_3, D_2, D_1, D_0 중 하나 선택

⇒ 구간 1과 5 : $S_1 = 0$, $S_0 = 0$이므로 D_0이 선택되어 $F = 0$

⇒ 구간 2과 6 : $S_1 = 0$, $S_0 = 1$이므로 D_1이 선택되어 $F = 1$

⇒ 구간 3과 7 : $S_1 = 1$, $S_0 = 0$이므로 D_2이 선택되어 $F = 0$

⇒ 구간 4과 8 : $S_1 = 1$, $S_0 = 1$이므로 D_3이 선택되어 $F = 0$

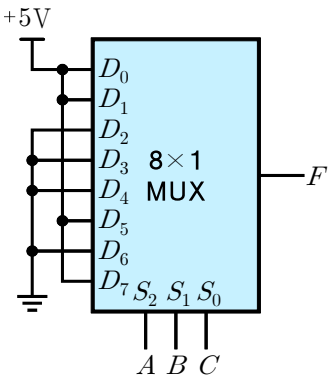
■

⊙ 멀티플렉서를 이용한 조합회로 구현

- $F(A,B,C)=\sum(0,1,5,7)$을 8×1 멀티플렉서와 4×1 멀티플렉서를 이용하여 설계
  - ⇒ 8×1 멀티플렉서는 3개의 선택선이 있으므로 A, B, C로 사용
  - ⇒ $D_0=D_1=D_5=D_7=1(+5\text{V})$, $D_2=D_3=D_4=D_6=0(0\text{V})$로 설정

선택선			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	$1(D_7)$

(a) 진리표



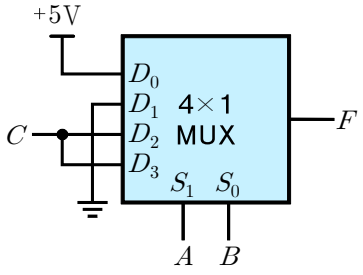
(b) 회로도

그림 7.49 8×1 멀티플렉서를 이용한 $F=\sum(0,1,5,7)$ 회로

- 4×1 멀티플렉서에서 A, B를 2개의 선택선으로 사용, C는 D_2, D_3의 값 결정
  - ⇒ 그림 7.49의 진리표에서 $AB=00$이면 $F=1$, $AB=01$이면 $F=0$
  - ⇒ $AB=10$과 $AB=11$이면 출력은 $F=C$로 그대로 출력
  - ⇒ $D_0=+5\text{V}$, $D_1=0\text{V}$, D_2와 D_3는 C를 입력으로 연결

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

(a) 진리표



(b) 회로도

그림 7.50 4×1 멀티플렉서를 이용한 $F=\sum(0,1,5,7)$ 회로

【예제 7.6】 논리식 $F(A,B,C)=A+BC'$를 8×1 및 4×1 멀티플렉서로 설계하려라.

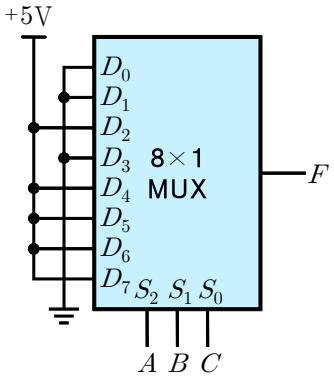
- 8×1 및 4×1 멀티플렉서로 설계하려면 제거된 항을 복구한 후 설계

$$\begin{aligned}F(A,B,C) &= A(B'+B)(C'+C)+(A'+A)BC' \\ &= AB'C'+AB'C+\underline{ABC'}+ABC+A'BC'+\underline{ABC'} \\ &= A'BC'+AB'C'+AB'C+ABC'+ABC \\ &= m_2+m_4+m_5+m_6+m_7=\sum(2,4,5,6,7)\end{aligned}$$

- 8×1 멀티플렉서에서 $D_2=D_4=D_5=D_6=D_7=+5\text{V}$, $D_0=D_1=D_3=0\text{V}$
- 4×1 멀티플렉서에서 $AB=00$이면 $F=0$, $AB=01$이면 $F=C'$로 반전 출력
  $\Rightarrow$ $AB=10$과 $AB=11$이면 C에 관계없이 출력 $F=1$
  $\Rightarrow$ $D_0=0\text{V}$, $D_2=D_3=+5\text{V}$, D_1에는 C를 반전하여 입력

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

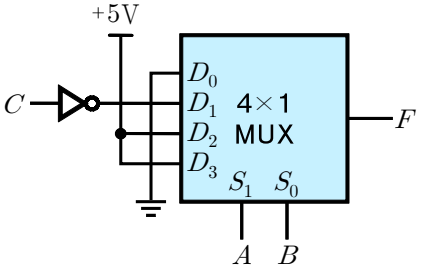
(a) 진리표(8×1)



(b) 회로도(8×1)

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

(c) 진리표(4×1)



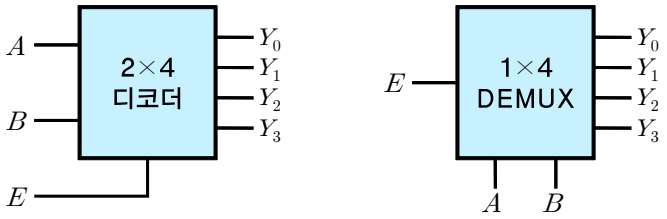
(d) 회로도(4×1)

그림 7.51 8×1 및 4×1 멀티플렉서를 이용한 $F=A+BC'$ 회로

■

7.6 디멀티플렉서(Demultiplexer)

- 1개의 Enable 입력을 갖는 디코더는 디멀티플렉서로서의 기능을 수행 가능
  ⇒ 그림 7.22의 디코더 IC 74138의 Enable($G1$)을 데이터 입력
  ⇒ A, B, C를 선택선으로 사용하면 $G1$의 부정 $G1'$이 출력



(a) 2×4 디코더 (b) 1×4 디멀티플렉서

그림 7.52 디코더와 디멀티플렉서 비교

표 7.10 IC 74138을 디멀티플렉서로 사용할 경우

입력			출력							
C	B	A	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	1	1	1	1	1	1	1	$G1'$
0	0	1	1	1	1	1	1	1	$G1'$	1
0	1	0	1	1	1	1	1	$G1'$	1	1
0	1	1	1	1	1	1	$G1'$	1	1	1
1	0	0	1	1	1	$G1'$	1	1	1	1
1	0	1	1	1	$G1'$	1	1	1	1	1
1	1	0	1	$G1'$	1	1	1	1	1	1
1	1	1	$G1'$	1	1	1	1	1	1	1

【예제 7.7】 그림 7.52(b)의 1×4 디멀티플렉서에 그림 7.53과 같은 데이터 입력파형(E)과 데이터 선택파형(S_1, S_0)을 입력하였다. 출력 Y_3, Y_2, Y_1, Y_0의 파형 도시.

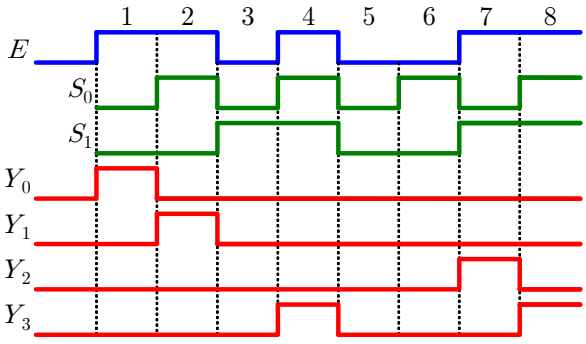


그림 7.53 1×4 디멀티플렉서의 입출력파형 ■

7.7 코드 변환기

⊙ 2진 코드-그레이 코드 변환

- 표 7.11은 2진 코드를 그레이 코드로 변환하는 진리표, 그림 7.54는 카르노맵
  ⇒ 그림 7.55는 간소화된 논리식에 의해 구현된 논리회로를 표시

표 7.11 2진 코드-그레이 코드 변환 진리표

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

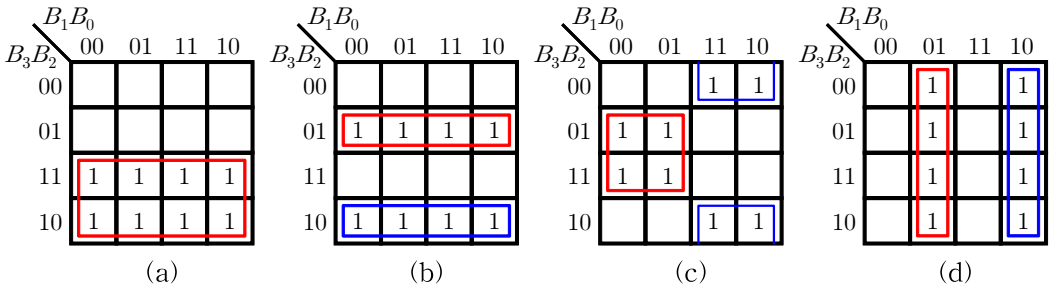


그림 7.54 2진 코드를 그레이 코드로 변환하는 카르노맵

$$G_3 = B_3, \quad G_2 = B_3'B_2 + B_3B_2' = B_3 \oplus B_2$$
$$G_1 = B_2'B_1 + B_2B_1' = B_2 \oplus B_1, \quad G_0 = B_1'B_0 + B_1B_0' = B_1 \oplus B_0$$

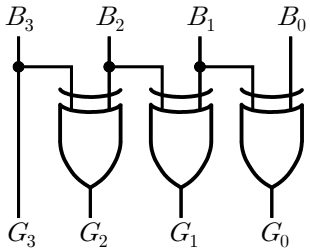


그림 7.55 2진 코드를 그레이 코드로 변환하는 회로

⊙ 그레이 코드-2진 코드 변환

- 표 7.12은 그레이 코드를 2진 코드로 변환하는 진리표, 그림 7.56은 카르노맵
  ⇒ 그림 7.57는 간소화된 논리식에 의해 구현된 논리회로를 표시

표 7.12 그레이 코드-2진 코드 변환 진리표

그레이코드(입력)				2진코드(출력)				그레이코드(입력)				2진코드(출력)			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1
0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	0
0	0	1	0	0	0	1	1	1	0	1	0	1	1	0	0
0	0	1	1	0	0	1	0	1	0	1	1	1	1	0	1
0	1	0	0	0	1	1	1	1	1	0	0	1	0	0	0
0	1	0	1	0	1	1	0	1	1	0	1	1	0	0	1
0	1	1	0	0	1	0	0	1	1	1	0	1	0	1	1
0	1	1	1	0	1	0	1	1	1	1	1	1	0	1	0

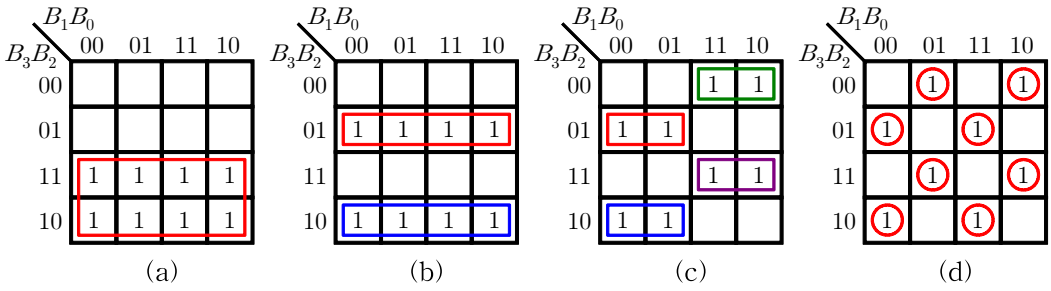


그림 7.56 그레이 코드를 2진 코드로 변환하는 카르노맵

$$B_3 = G_3, \quad B_2 = G_3'G_2 + G_3G_2' = G_3 \oplus G_2$$
$$B_1 = G_3 \oplus G_2 \oplus G_1 = B_2 \oplus G_1, \quad B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0 = B_1 \oplus G_0$$

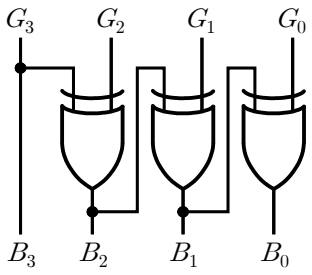


그림 7.57 그레이 코드를 2진 코드로 변환하는 회로

⊙ BCD 코드-3초과 코드 변환

- 표 7.13은 그레이 코드를 2진 코드로 변환하는 진리표, 그림 7.58은 카르노맵
  ⇒ BCD 코드에서는 1010~1111을 사용하지 않으므로 무관항 처리
  ⇒ 그림 7.59는 간소화된 논리식에 의해 구현된 논리회로를 표시

표 7.13 BCD 코드-3초과 코드 변환 진리표

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

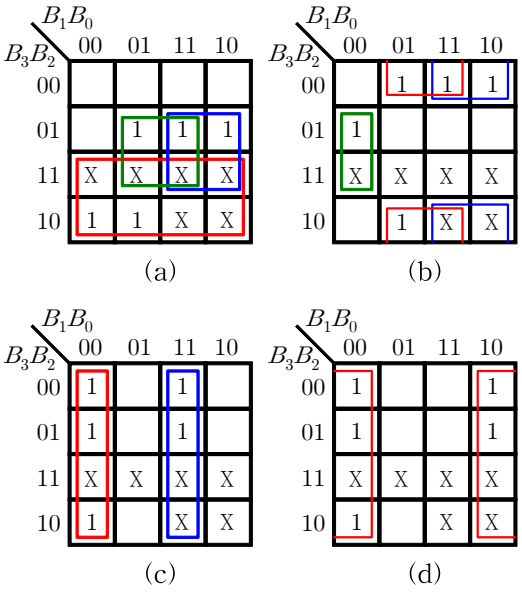


그림 7.58 카르노맵

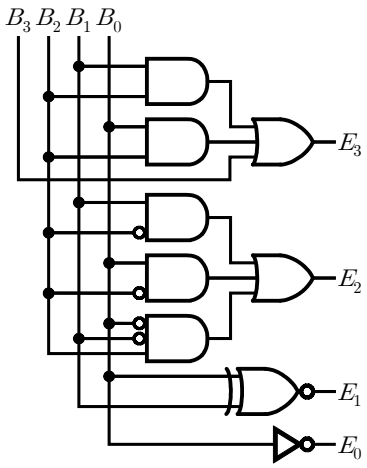


그림 7.59 논리회로

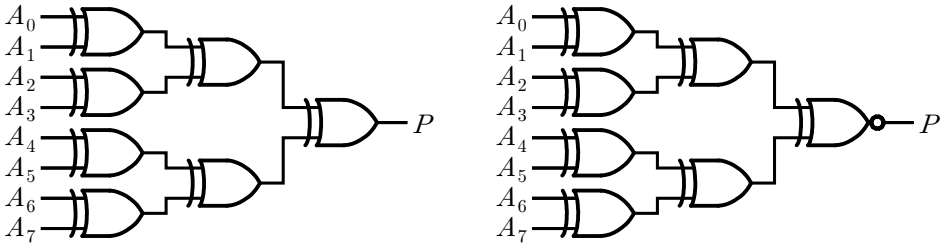
$$E_3 = B_3 + B_2B_1 + B_2B_0, \quad E_2 = B_2'B_1 + B_2'B_0 + B_2B_1'B_0'$$
$$E_1 = B_1'B_0' + B_1B_0 = (B_1 \oplus B_0)' = B_1 \odot B_0, \quad E_0 = B_0'$$

7.8 패리티 발생기/검출기

- 데이터에 1비트 패리티를 추가하여 1의 개수를 짝수 또는 홀수로 맞추는 회로
  - ⇒ 짝수 패리티 발생기는 데이터의 1의 개수를 짝수로 맞추는 회로
  - ⇒ 홀수 패리티 발생기는 데이터의 1의 개수를 홀수로 맞추는 회로
- 패리티 검출기는 데이터에서 1의 개수가 짝수인지 홀수인지를 검사하는 회로
- XOR는 입력 데이터에 포함된 1의 개수가 홀수일 때 출력 1이 발생하는 특성
  - ⇒ 1의 개수가 홀수일 때 XOR의 출력을 추가하면 1의 개수는 짝수
  - ⇒ XOR를 사용하여 짝수 패리티 발생기 구현
- XNOR는 입력 데이터에 포함된 1의 개수가 짝수일 때 출력이 1인 게이트
  - ⇒ 1의 개수가 짝수일 때 XNOR의 출력을 추가하면 1의 개수는 홀수
  - ⇒ XNOR를 이용하여 홀수 패리티 발생기 구현
- 그림 7.60은 데이터가 8비트일 때 짝수 패리티와 홀수 패리티를 발생하는 회로
  - ⇒ 식 (7.15)과 식 (7.16)은 각각 짝수 및 홀수 패리티 발생기의 논리식
  - ⇒ 짝수 패리티 발생기는 $A_0 \sim A_7$ 중에서 1의 개수가 홀수일 때 $P=1$
  - ⇒ 홀수 패리티 발생기는 $A_0 \sim A_7$ 중에서 1의 개수가 짝수일 때 $P=1$

$$P = A_0 \oplus A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7 \tag{7.15}$$

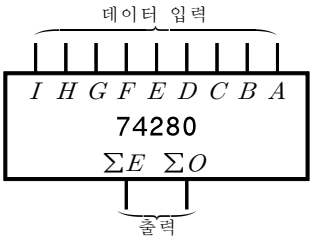
$$P = (A_0 \oplus A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7)' \tag{7.16}$$



(a) 짝수 패리티 발생회로 (b) 홀수 패리티 발생회로

그림 7.60 8비트 짝수 및 홀수 패리티 발생회로

- IC 74280은 9비트(1비트 패리티 포함)에 대해서 짝수와 홀수 모두 검사 가능
  - ⇒ 9비트($A \sim I$)까지의 2진 데이터에 대해 패리티 발생기로 사용 가능
- 그림 7.6은 9비트 패리티 발생기/검출기 IC 74280의 블록도 및 동작표를 표시
  - ⇒ $A \sim I$ 중에서 1의 개수가 짝수이면 ΣE = High, ΣO = Low
  - ⇒ $A \sim I$ 중에서 1의 개수가 홀수이면 ΣE = Low, ΣO = High



(a) 블록도

High인 입력 (A~I)이 개수	출력	
	ΣE	ΣO
짝수 개(0, 2, 4, 6, 8)	H	L
홀수 개(1, 3, 5, 7, 9)	L	H

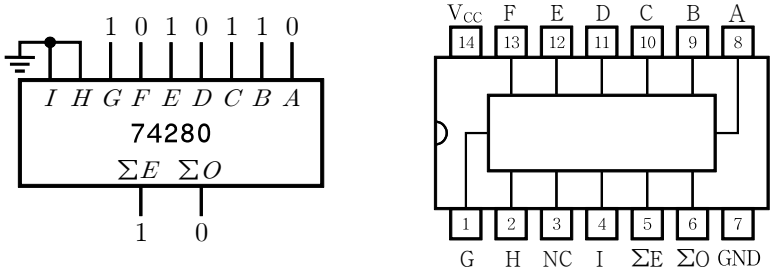
(b) 블록도

그림 7.61 9비트 패리티 발생기/검출기(IC 74280)

- 출력 ΣO는 입력 비트 중 1의 개수가 짝수이면 0, 1의 개수가 홀수이면 1
  ⇒ 74280을 짝수 패리티 발생기로 사용하면 ΣO가 패리티 비트
- 출력 ΣE는 입력 비트 중 1의 개수가 홀수이면 0, 1의 개수가 짝수이면 1
  ⇒ 74280을 홀수 패리티 발생기로 사용하면 ΣE가 패리티 비트
- 74280을 패리티 검출기로 사용하려면 A~I까지 9비트를 입력한 뒤 출력 검사
  ⇒ 입력 중에서 1의 개수가 짝수(ΣE)인지, 홀수(ΣO)인지 검사 가능
  ⇒ 입력 중 1의 개수가 짝수이면 ΣE = High, 홀수이면 ΣO = High

【예제 7.7】 74280을 8비트 홀수 패리티(7개의 데이터 비트와 1개의 패리티 비트)로 사용하려고 한다. 7비트 입력 데이터 1010110에 대하여 패리티 비트를 생성하는 구성도를 그려보아라.

- A~G에 1010110을 연결하고 H와 I는 출력에 영향을 주지 않도록 접지
  ⇒ 출력은 ΣE이며 홀수 패리티 비트
- 입력 중에서 1의 개수가 4개이므로 출력 ΣE = 1이 되어 결국 11010110



(a) 구성도 (b) 핀 배치도

그림 7.62 IC 74280을 사용한 홀수 패리티 발생

■