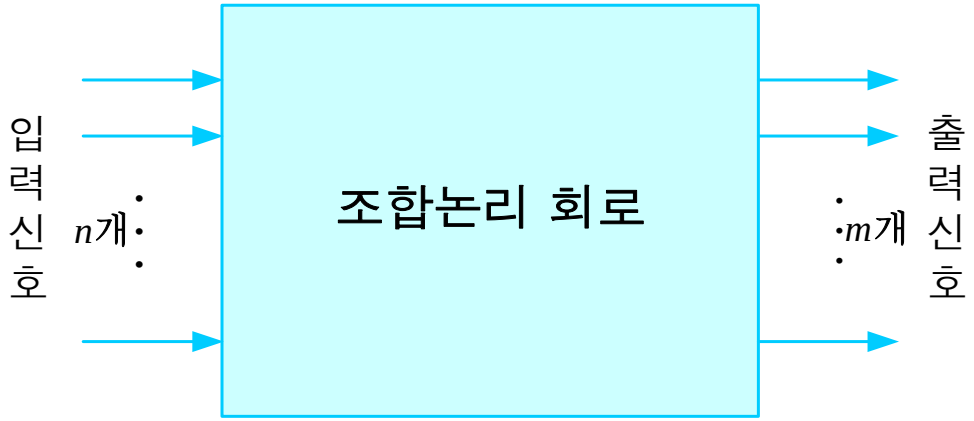
7. 조합논리회로

논리회로

부경대 컴퓨터·인공지능공학부 최필주

조합논리회로 개요

- 기본 논리 회로: AND, OR, NOT, NAND, NOR, XOR, NOR
- 조합논리회로
  - 논리곱(AND), 논리합(OR), 논리 부정(NOT)의 세 가지 기본 논리 회로를 조합하여 구성한 논리 회로
  - 구성: 입력변수, 논리 게이트, 출력변수



<조합논리회로 블록도>

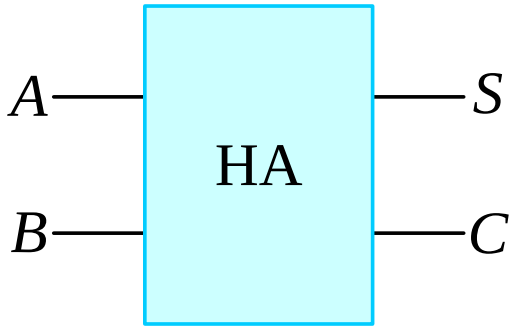
목차

- 가산기
- 비교기
- 디코더와 인코더
- 멀티플렉서와 디멀티플렉서
- 코드 변환기
- 패리티 발생기/검출기

가산기(Adder)

- 반가산기(HA, half-adder)
  - 두 비트의 덧셈 수행 → 출력: Sum, Carry

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

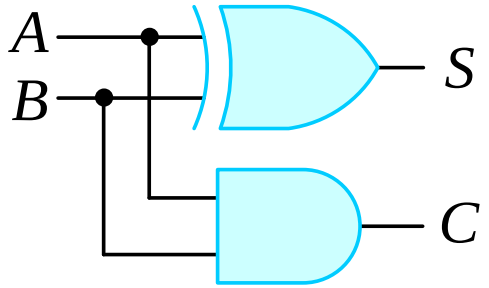


<논리기호>

  - 진리표, 논리식, 논리회로

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

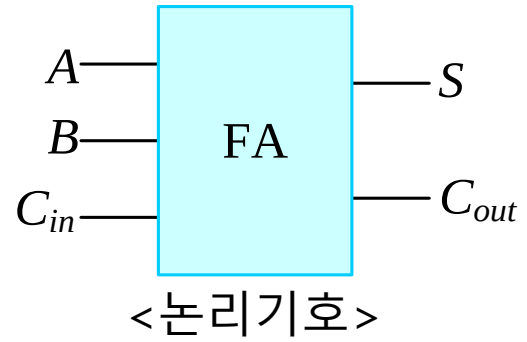
$$S = \bar{A}B + A\bar{B} = A \oplus B$$
$$C = A \cdot B$$



가산기(Adder)

- 전가산기(FA, full-adder)
  - 세 비트의 덧셈 수행 → 출력: Sum, Carry

C_{in}	0	1	0	1	0	1	0	1
A	0	0	1	1	0	0	1	1
$+ \quad B$	$+ \ 0$	$+ \ 0$	$+ \ 0$	$+ \ 0$	$+ \ 1$	$+ \ 1$	$+ \ 1$	$+ \ 1$
$C_{out} S$	0 0	0 1	0 1	1 0	0 1	1 0	1 0	1 1



<논리기호>

  - 진리표, 논리식

입력			출력	
A	B	C_{in}	S	C_{out}
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 \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = A \oplus B \oplus C_{in}$$

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

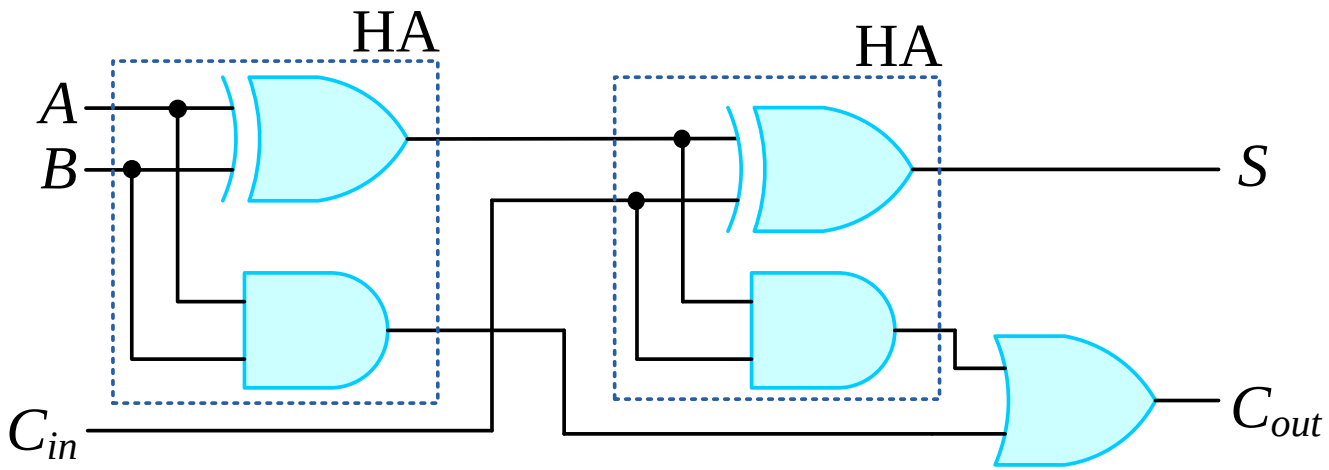
$$C_{out} = AB + BC_{in} + C_{in}A$$

가산기(Adder)

- 전가산기(FA, full-adder)
  - 논리식
    - $S = A \oplus B \oplus C_{in}$
    - $C_{out} = AB + BC_{in} + C_{in}A$
  - 논리회로 구현1: 3-input XOR, 2-input AND × 3, 3-input OR

가산기(Adder)

- 전가산기(FA, full-adder)
  - 논리식
    - $S = A \oplus B \oplus C_{in}$
    - $C_{out} = AB + BC_{in} + C_{in}A = AB + C_{in}(A + B) = AB + C_{in}(A \oplus B)$
  - 논리회로 구현 2: HA × 2, 2-input OR
    - 1st HA
      - $S' = A \oplus B$
      - $C' = AB$
    - 2nd HA
      - $S = S' \oplus C_{in} = (A \oplus B) \oplus C_{in}$
      - $C'' = S' \cdot C_{in} = (A \oplus B) \cdot C_{in}$
    - OR
      - $C' + C'' = AB + C_{in}(A \oplus B)$



가산기(Adder)

- HA vs. FA

```
   1        1
 + 0      + 1
 ---      ---
 0 1      1 0
```

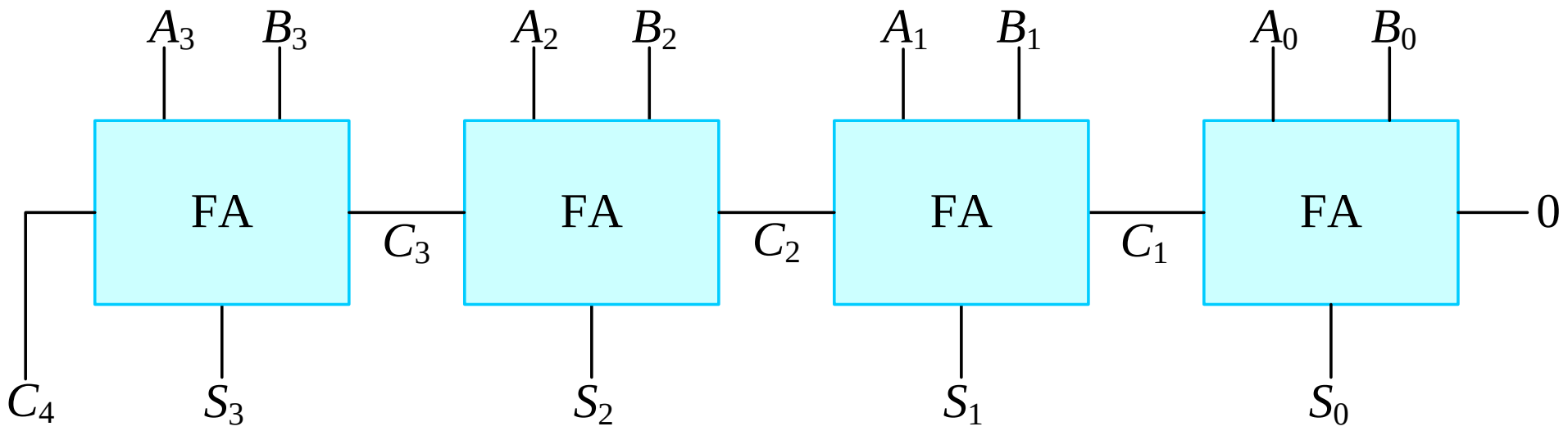
<한 자리 2진수를 더할 때>

```
     1 1 1
   0 0 1 1
 + 1 1 0 1
 ---------
 1 0 0 0 0
```

<두 자리 이상의 2진수를 더할 때>

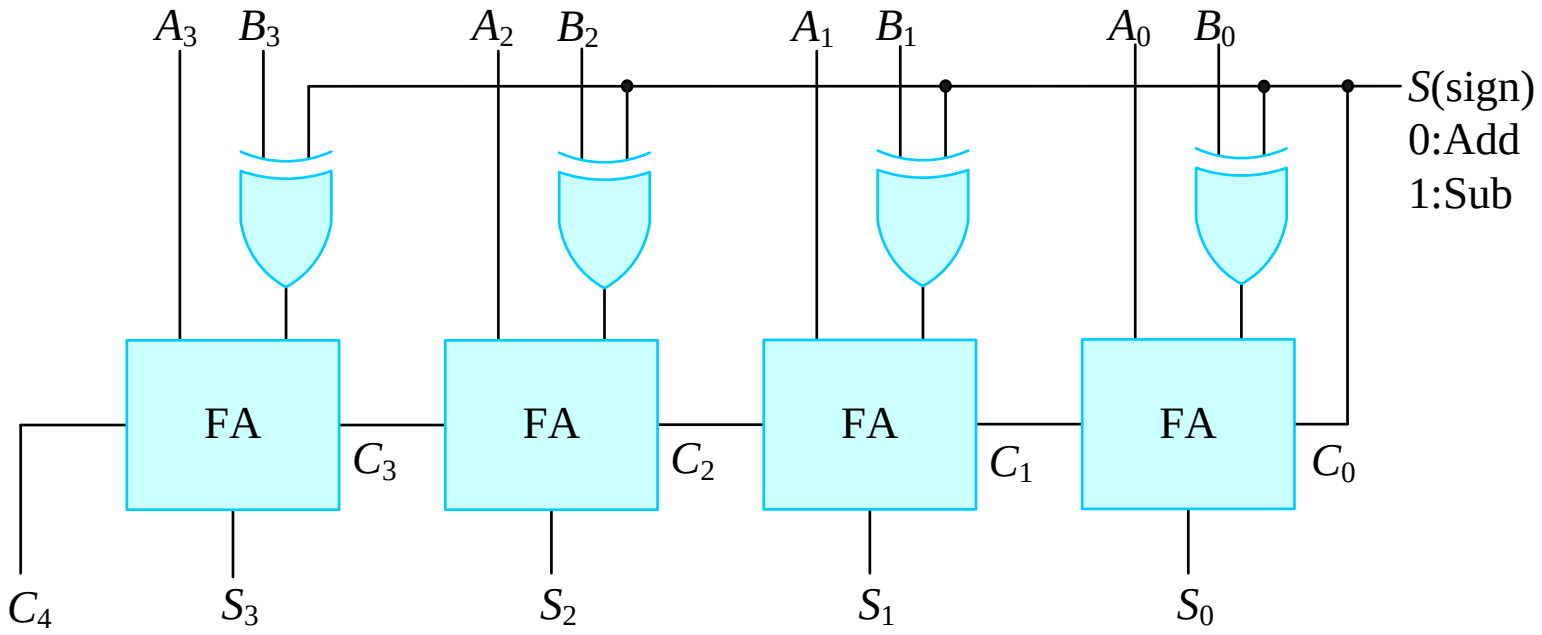
가산기(Adder)

- 병렬가감산기(Parallel-adder/subtractor)
  - 병렬가산기: FA를 여러 개 병렬로 연결
    - 2비트 이상의 수를 더할 때 사용
    - 예) 4비트 덧셈 $\{C_4 S_3 S_2 S_1 S_0\} = \{A_3 A_2 A_1 A_0\} + \{B_3 B_2 B_1 B_0\}$



가산기(Adder)

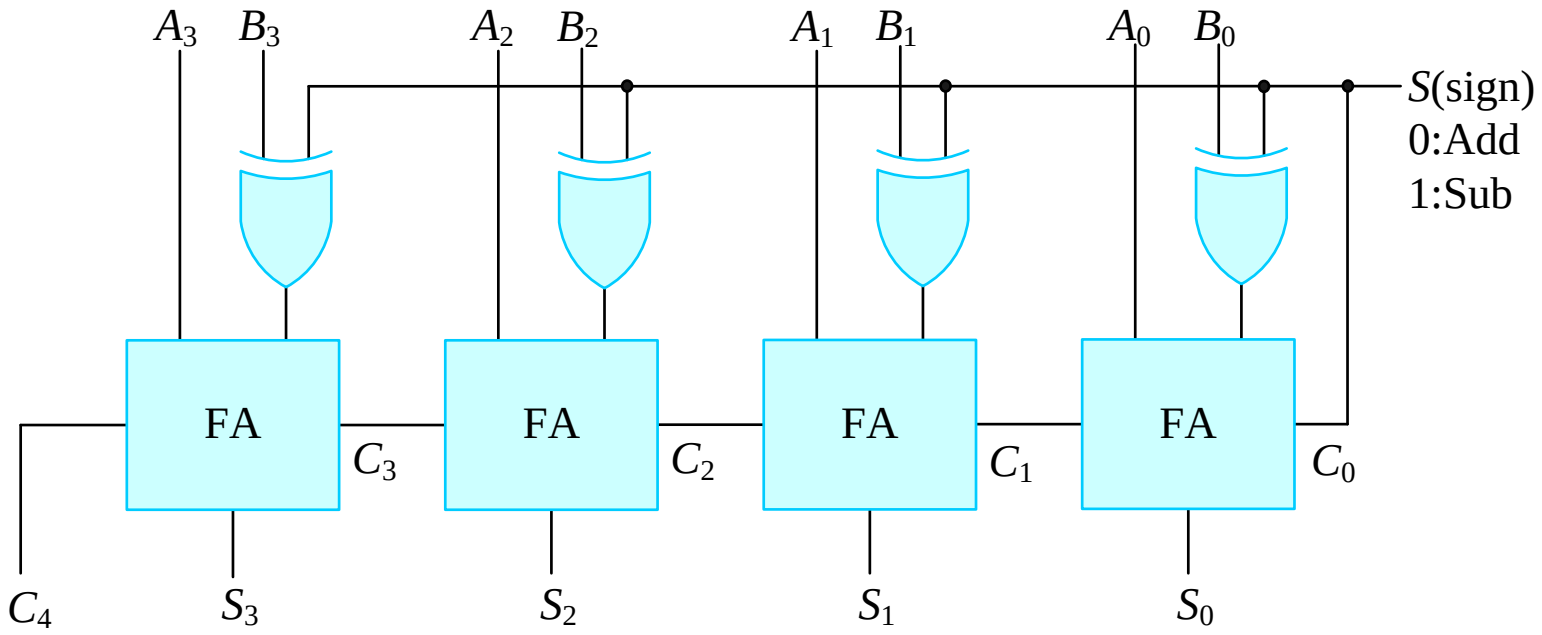
- 병렬가감산기(Parallel-adder/subtractor)
  - 병렬가감산기: XOR를 추가하여 감산(뺄셈) 기능 추가



- $S = 0$일 때:$\{C_4 S_3 S_2 S_1 S_0\} = \{A_3 A_2 A_1 A_0\} + \{B_3 B_2 B_1 B_0\} + 0$
- $S = 1$일 때: $\{C_4 S_3 S_2 S_1 S_0\} = \{A_3 A_2 A_1 A_0\} + \{\overline{B_3}\ \overline{B_2}\ \overline{B_1}\ \overline{B_0}\} + 1$
  $= \{A_3 A_2 A_1 A_0\} + (-\{B_3 B_2 B_1 B_0\})$

가산기(Adder)

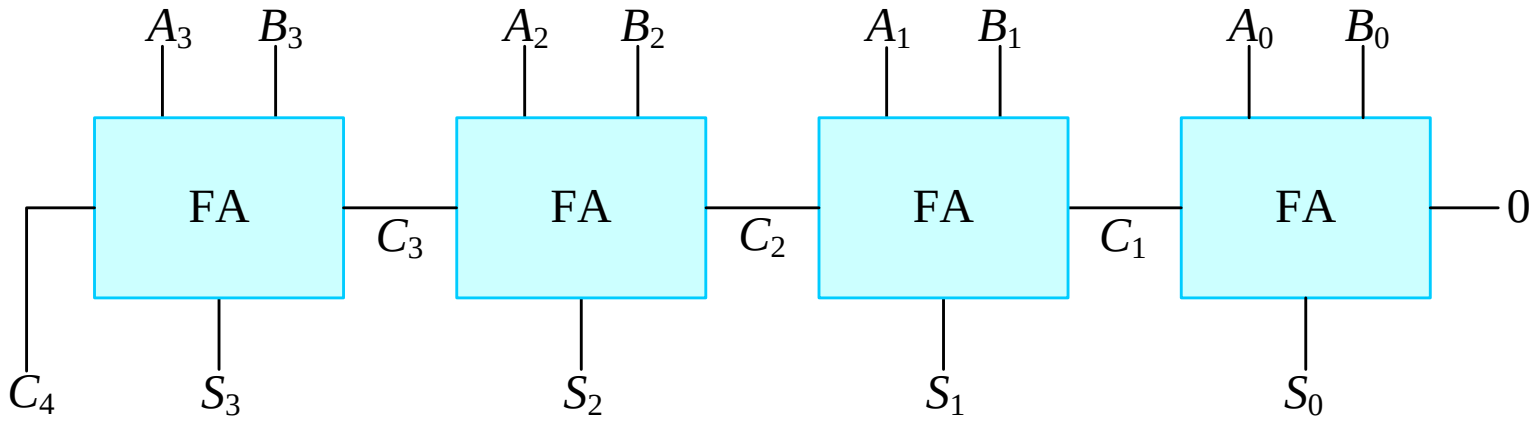
- 고속가산기(high-speed-adder)
  - 병렬가산기는 carry 이동 때문에 속도가 매우 느림



  - 대안
    - Carry-Lookahead Adder (CLA)
    - Carry-select Adder, …

가산기(Adder)

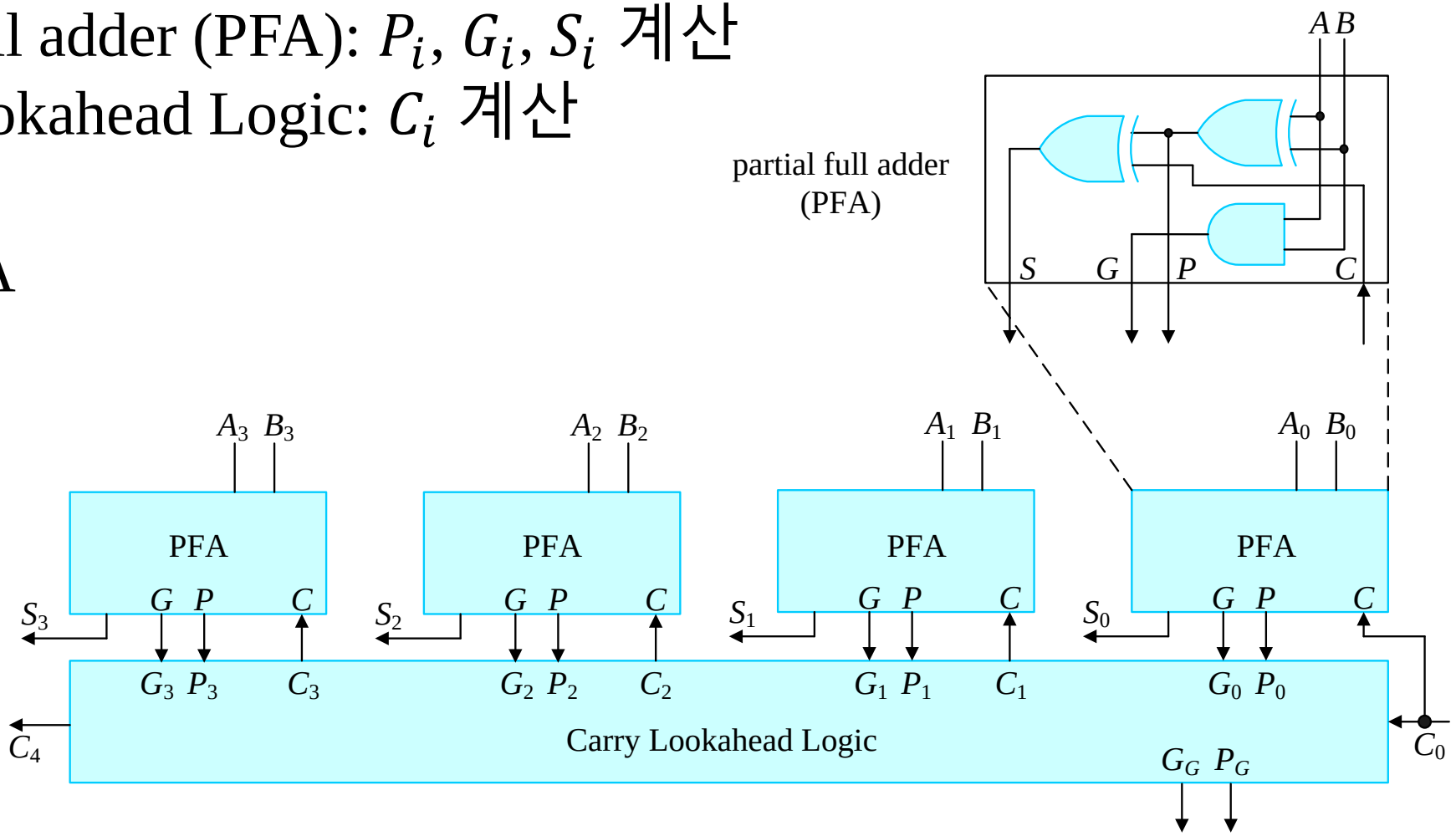
- CLA - 논리식



  - $C_{i+1} = A_iB_i + C_i(A_i \oplus B_i) = G_i + P_iC_i$ where $G_i = A_iB_i$, $P_i = A_i \oplus B_i$
  - 4비트 가산기에 적용
    - $C_1 = G_0 + P_0C_0$
    - $C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$
    - $C_3 = G_2 + P_2C_2 = G_2 + P_2(G_1 + P_1G_0 + P_1P_0C_0) = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
    - $C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0 = G_G + P_GC_0$
    - $P_G = P_3P_2P_1P_0$
    - $G_G = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$

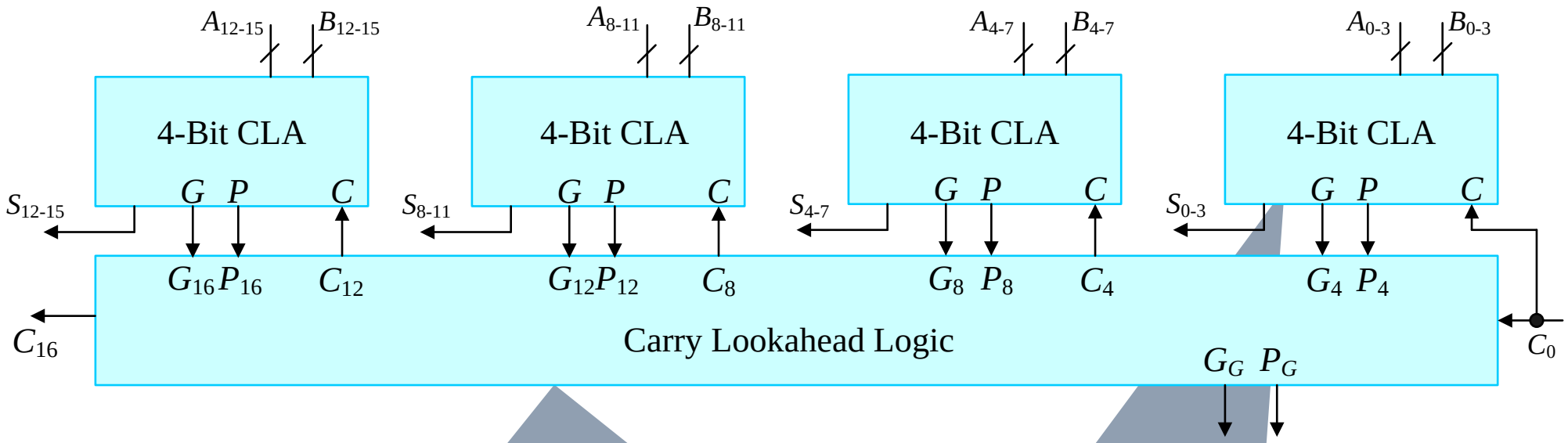
가산기(Adder)

- CLA - 구성
  - Partial full adder (PFA): P_i, G_i, S_i 계산
  - Carry Lookahead Logic: C_i 계산
  - 4-bit CLA



가산기(Adder)

- CLA - 구성
  - 16-bit CLA



$C_4 = G_4 + P_4C_0$
$C_8 = G_8 + P_8G_4 + P_8P_4C_0$
$C_{12} = G_{12} + P_{12}G_8 + P_{12}P_8G_4 + P_{12}P_8P_4C_0$
$C_{16} = G_{16} + P_{16}G_{12} + P_{16}P_{12}G_8 + P_{16}P_8P_4G_4 + P_{16}P_{12}P_8P_4C_0$

$C_1 = G_0 + P_0C_0$
$C_2 = G_1 + P_1G_0 + P_1P_0C_0$
$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

가산기(Adder)

- BCD 가산기
  - BCD 코드: 10진수 1자리가 BCD 코드 4 bits와 대응
  - 가산기 설계 시 고려사항: 덧셈 결과가 10~15인 경우 이를 보정(+6)
    - 예: 6 + 7 = 13

가산기(Adder)

- BCD 가산기
  - 논리식

$$\begin{array}{rrrrr} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ + & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} \end{array} \rightarrow \{KZ_3Z_2Z_1Z_0\}$$

$$\begin{array}{rrrrr} + & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \hline \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{array} \rightarrow \{CS_3S_2S_1S_0`\}$$

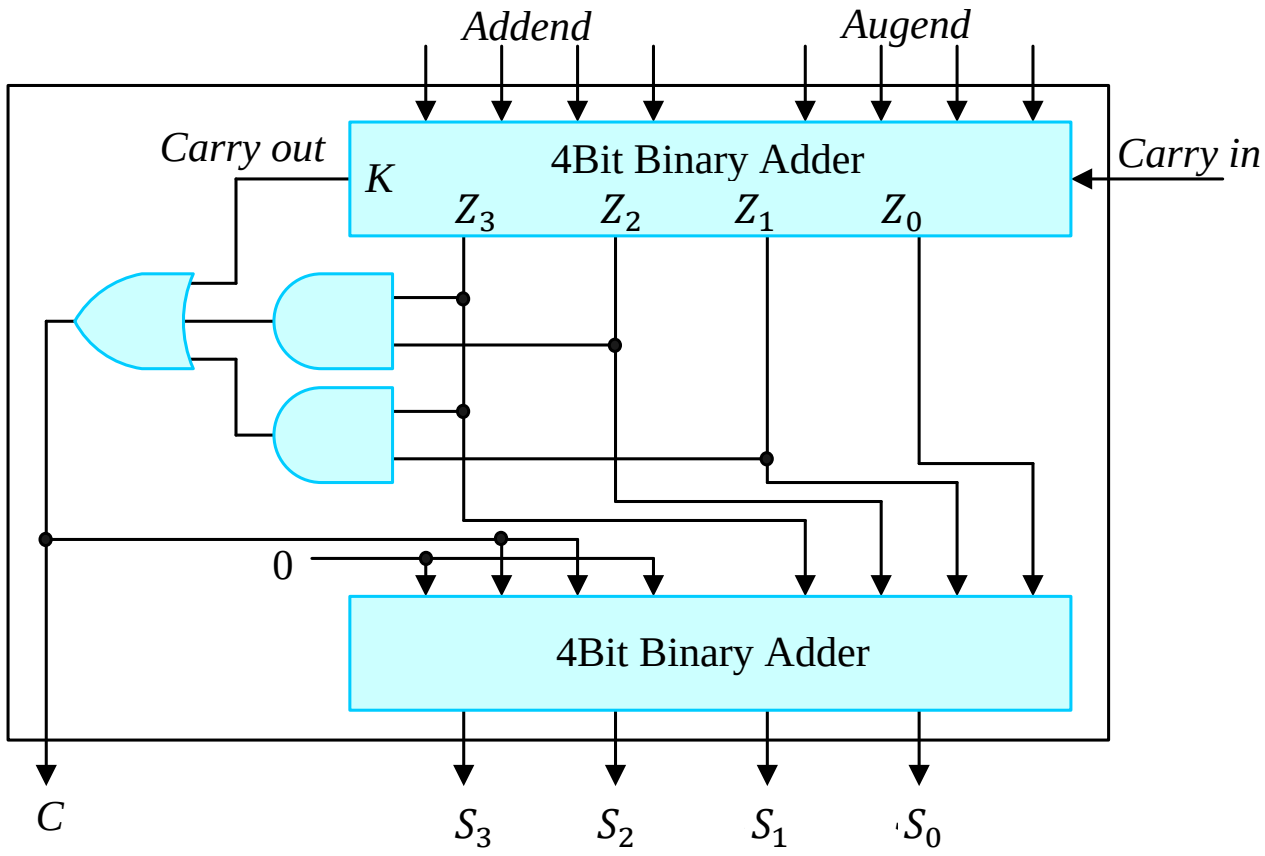
$\{Z_3Z_2\}$ \ $\{Z_1Z_0\}$	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

$\{Z_3Z_2\}$ \ $\{Z_1Z_0\}$	00	01	11	10
00				
01				
11	1	1	1	1
10			1	1

- $C = K + Z_3Z_2 + Z_3Z_1$
- $\{S_3S_2S_1S_0\} = \{Z_3Z_2Z_1Z_0\} + \{C\ C\ 0\}$

가산기(Adder)

- BCD 가산기
  - BCD합이 $\{KZ_3Z_2Z_1Z_0\}$일 때
    - $C = K + Z_3Z_2 + Z_3Z_1$
    - $\{S_3S_2S_1S_0\} = \{Z_3Z_2Z_1Z_0\} + \{C\ C\ 0\}$

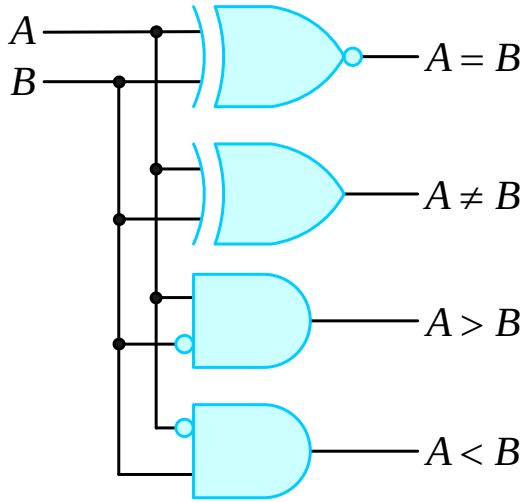


비교기(Comparator)

- 2진 비교기
  - 두 개의 2진수의 크기를 비교하는 회로

- 1비트 비교기

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



$$F_1 = \overline{A \oplus B}, \quad F_2 = A \oplus B,$$
$$F_3 = A\overline{B}, \quad F_4 = \overline{A}B$$

비교기(Comparator)

- 2비트 비교기

입력		출력			
A	B	$A=B$	$A\neq B$	$A>B$	$A<B$
A_1A_0	B_1B_0	F_1	F_2	F_3	F_4
0 0	0 0	1	0	0	0
	0 1	0	1	0	1
	1 0	0	1	0	1
	1 1	0	1	0	1
0 1	0 0	0	1	1	0
	0 1	1	0	0	0
	1 0	0	1	0	1
	1 1	0	1	0	1
1 0	0 0	0	1	1	0
	0 1	0	1	1	0
	1 0	1	0	0	0
	1 1	0	1	0	1
1 1	0 0	0	1	1	0
	0 1	0	1	1	0
	1 0	0	1	1	0
	1 1	1	0	0	0

$$F_1 = (\overline{A_1 \oplus B_1}) \cdot (\overline{A_0 \oplus B_0})$$

$$F_2 = \overline{F_1} = (A_1 \oplus B_1) + (A_0 \oplus B_0)$$

$$F_3 = A_1\overline{B_1} + A_1A_0\overline{B_0} + A_0\overline{B_1}\,\overline{B_0}$$

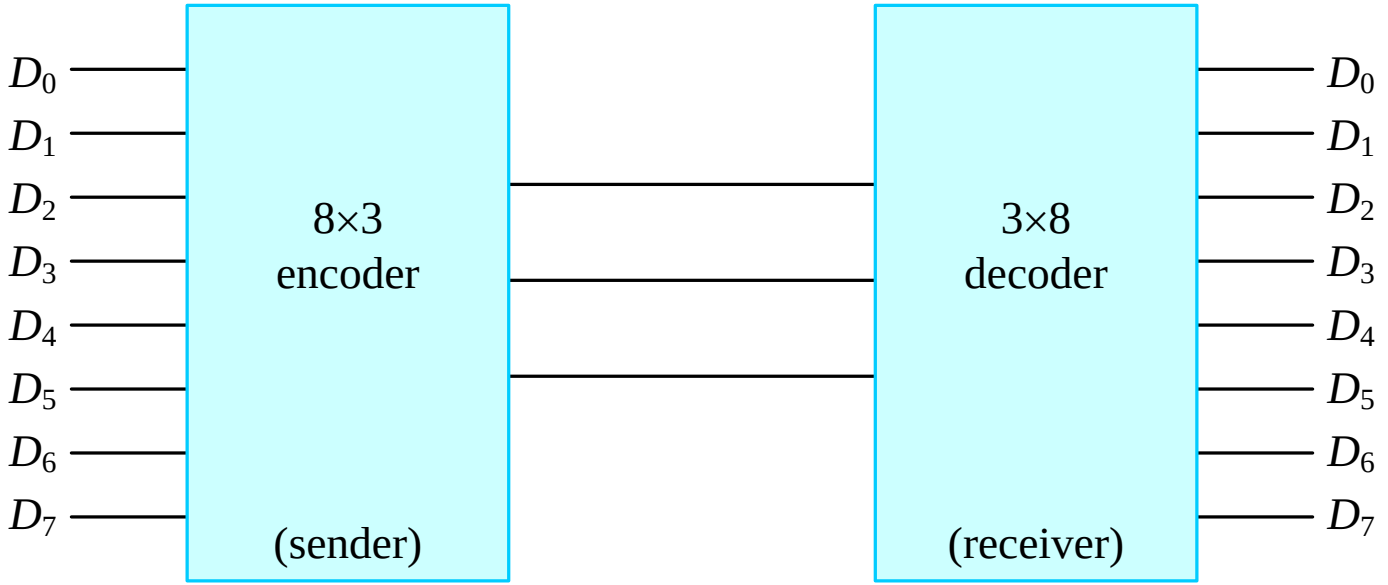
$\{A_1A_0\}$ \ $\{B_1B_0\}$	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

$$F_4 = B_1\overline{A_1} + B_1B_0\overline{A_0} + B_0\overline{A_1}\,\overline{A_0} = \overline{F_3} \cdot \overline{F_1}$$

인코더(Encoder)와 디코더(Decoder)

- n비트 2진 코드 ↔ 2^n개의 정보

- Encoder vs. Decoder

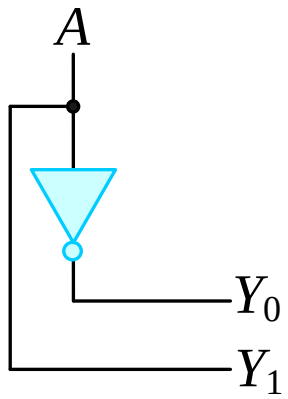


  - Encoder: D_0~D_7 중 on된 bit의 위치를 2진수로 출력
  - Decoder: D_0~D_7 중 2진수 입력이 가리키는 곳을 on

디코더(Decoder)

- 1×2 디코더
  - 논리식: $Y_0 = \bar{A}, Y_1 = A$
  - 진리표와 논리회로

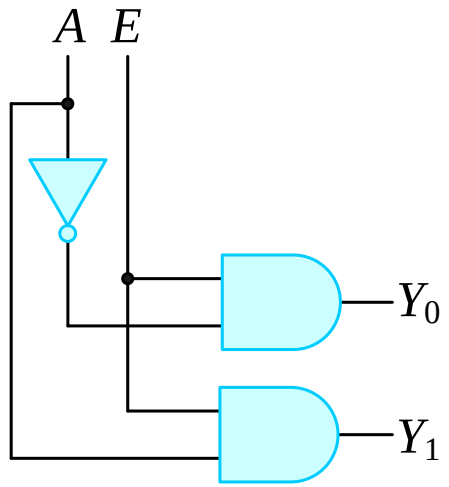
입력	출력	
A	Y_1	Y_0
0	0	1
1	1	0



디코더(Decoder)

- 1×2 디코더 - Enable 신호가 있는 경우
  - 논리식: $Y_0 = E\bar{A}, Y_1 = EA$
  - 진리표와 논리회로

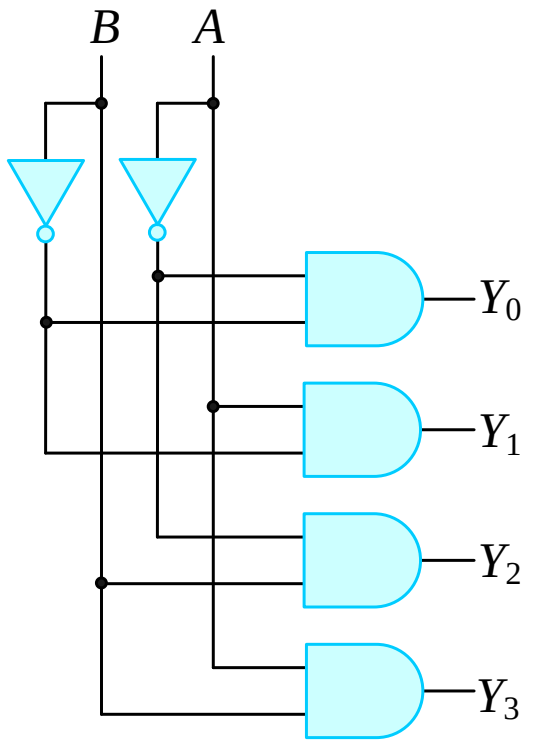
입력		출력	
E	A	Y_1	Y_0
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0



디코더(Decoder)

- 2×4 디코더
  - 논리식: $Y_0 = \bar{B}\bar{A}, Y_1 = \bar{B}A, Y_2 = B\bar{A}, Y_3 = BA$
  - 진리표와 논리회로

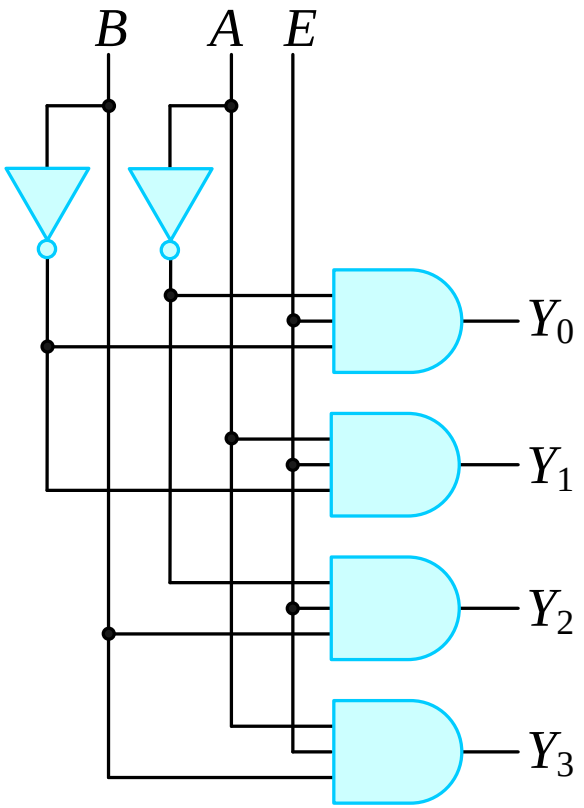
입력		출력			
B	A	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



디코더(Decoder)

- 2×4 디코더 - Enable 신호가 있는 경우
  - 논리식: $Y_0 = E\bar{B}\bar{A}, Y_1 = E\bar{B}A, Y_2 = EB\bar{A}, Y_3 = EBA$
  - 진리표와 논리회로

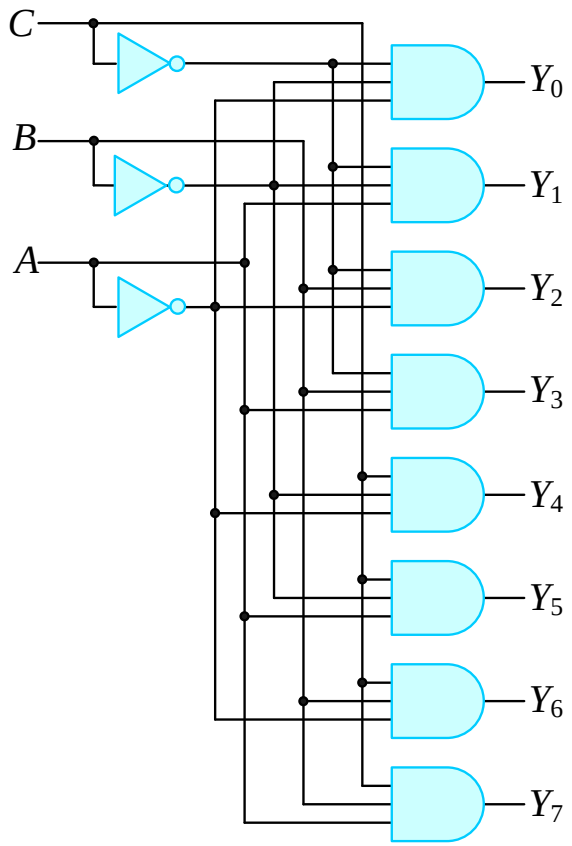
입력			출력			
E	B	A	Y_3	Y_2	Y_1	Y_0
0	×	×	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



디코더(Decoder)

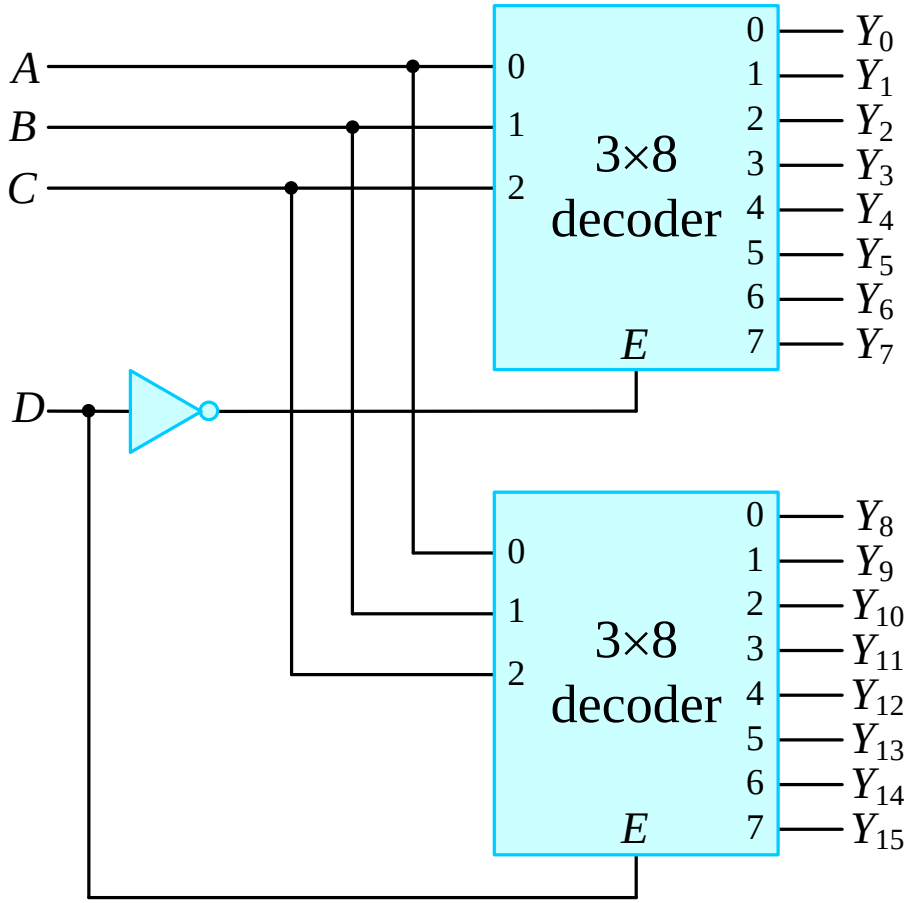
- 3×8 디코더
  - 논리식: $Y_0 = \bar{C}\bar{B}\bar{A}, Y_1 = \bar{C}\bar{B}A, Y_2 = \bar{C}B\bar{A}, Y_3 = \bar{C}BA$
    $Y_4 = C\bar{B}\bar{A}, Y_5 = C\bar{B}A, Y_6 = CB\bar{A}, Y_7 = CBA$
  - 진리표와 논리회로

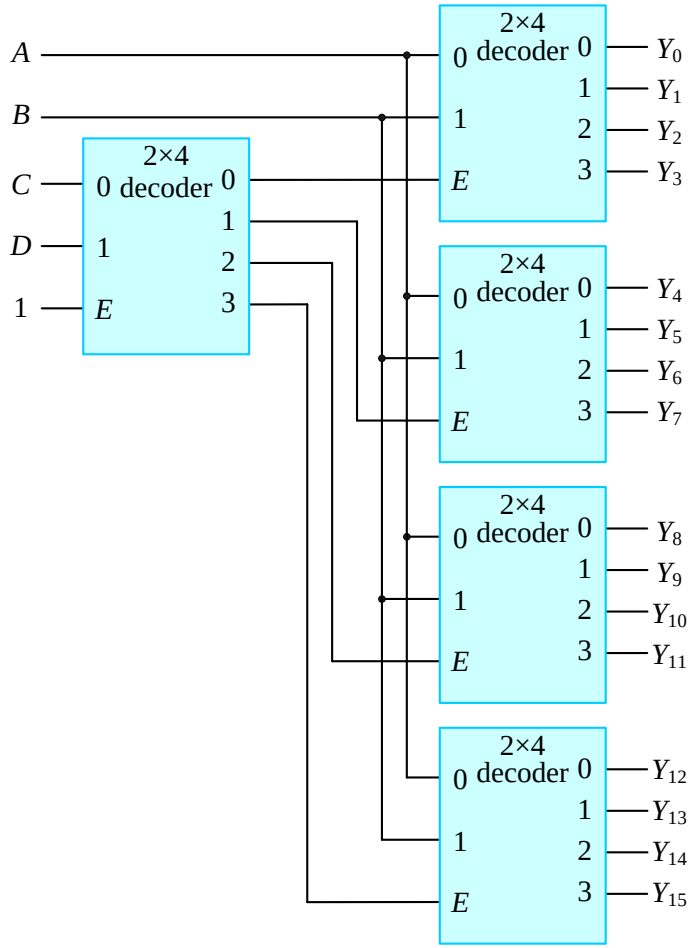
입력			출력							
C	B	A	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



디코더(Decoder)

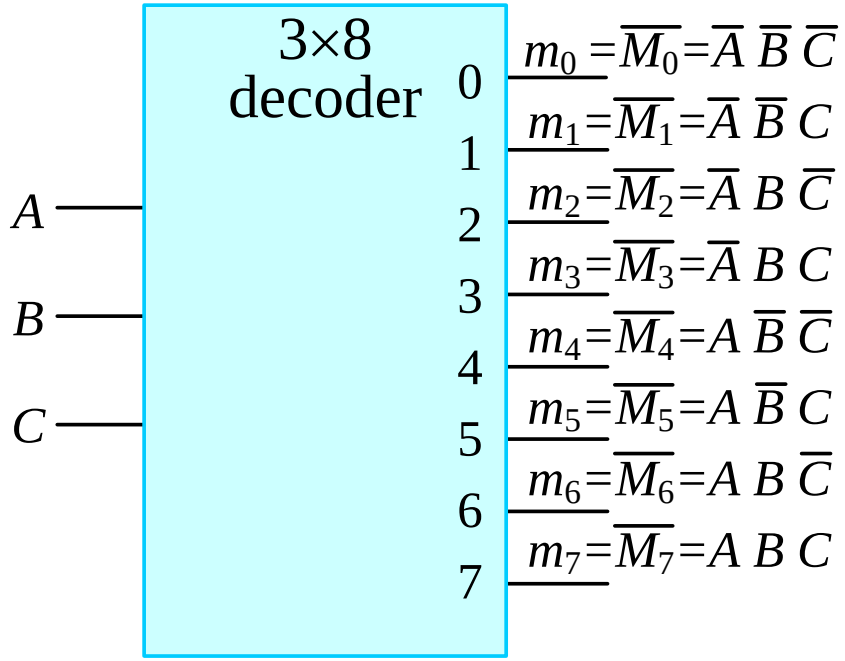
- 4×16 디코더
  - 2개의 3×8 디코더, 5개의 2×4 디코더로 구성 가능



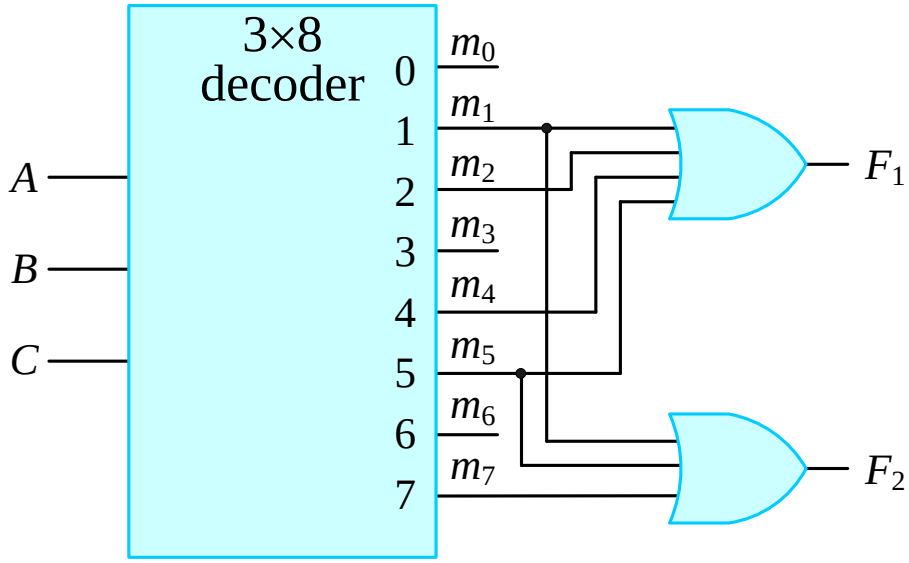


디코더(Decoder)

- 디코더를 이용한 조합논리회로
  - 디코더의 출력은 minterm을 의미



예

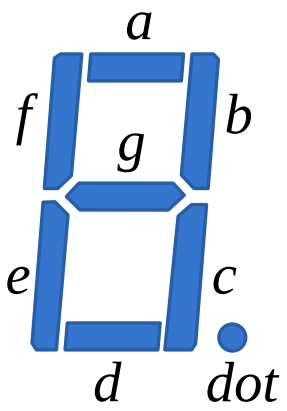


$$F_1(A,B,C) = \sum m(1, 2, 4, 5)$$

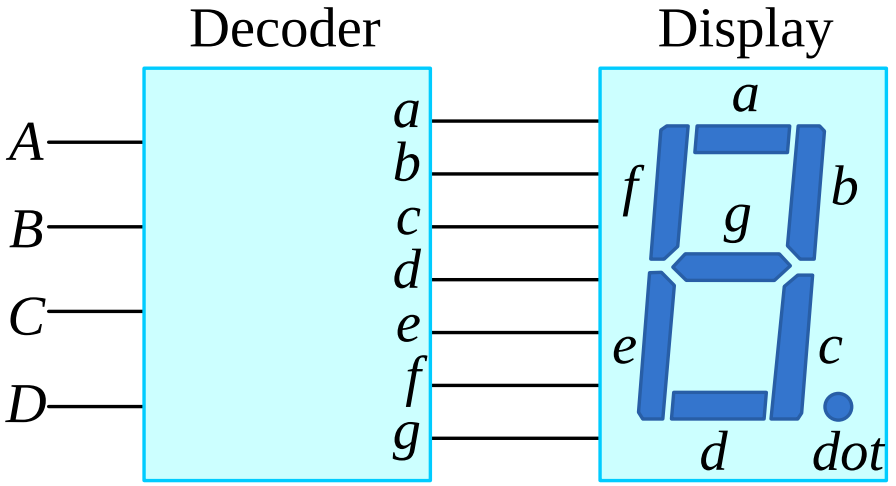
$$F_2(A,B,C) = \sum m(1, 5, 7)$$

디코더(Decoder)

- 7-Segment 디코더



<7-세그먼트 구성>



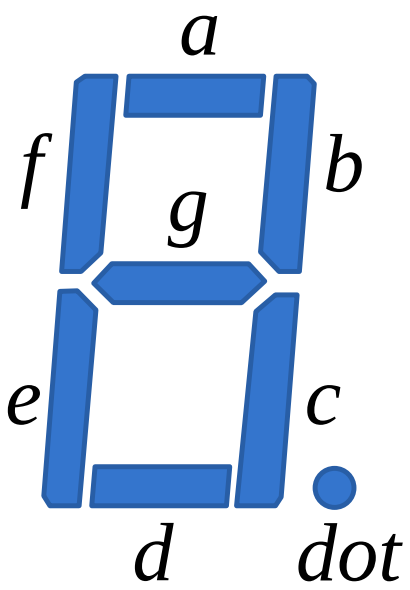
<7-세그먼트와 디코더의 연결>

0	1	2	3	4	5	6	7	8	9
8	8	8	8	8	8	8	8	8	8

디코더(Decoder)

- 7-Segment 디코더
  - 진리표

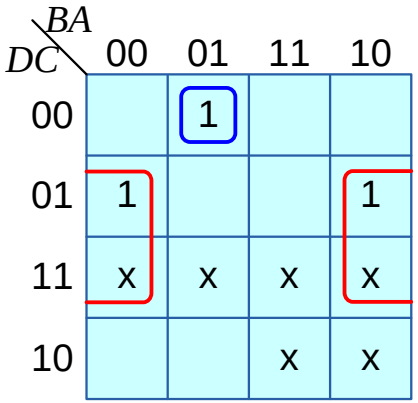
입력				출력						
D	C	B	A	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	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	×	×	×	×	×	×	×



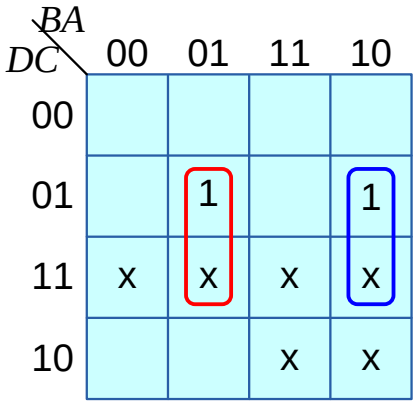
디코더(Decoder)

- 7-Segment 디코더
  - 진리표

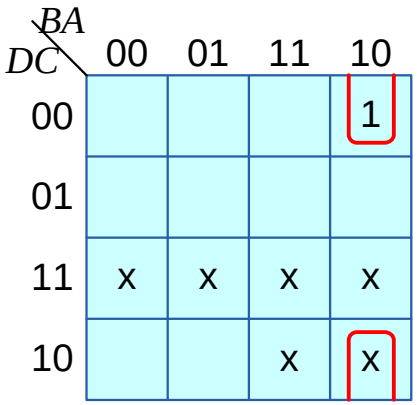
입력				출력						
D	C	B	A	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$\bar{f}$	$\bar{g}$
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	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	×	×	×	×	×	×	×



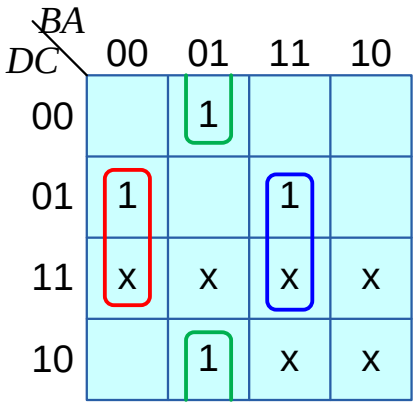
$$\bar{a} = \bar{D}\bar{C}\bar{B}A + C\bar{A}$$



$$\bar{b} = C\bar{B}A + CB\bar{A} = C(B \oplus A)$$



$$\bar{c} = \bar{C}B\bar{A}$$

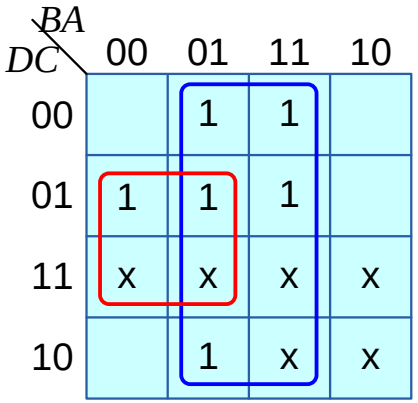


$$\bar{d} = \bar{C}\bar{B}A + C\bar{B}\bar{A} + CBA$$

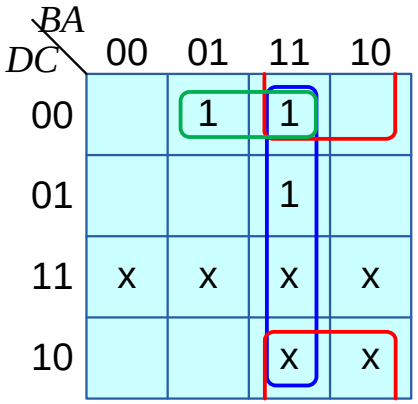
디코더(Decoder)

- 7-Segment 디코더
  - 진리표

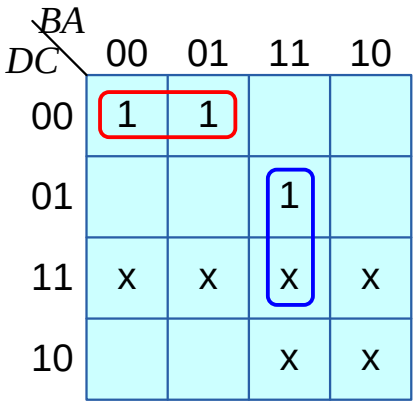
입력				출력						
D	C	B	A	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$\bar{f}$	$\bar{g}$
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	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	×	×	×	×	×	×	×



$$\bar{e} = A + C\bar{B}$$



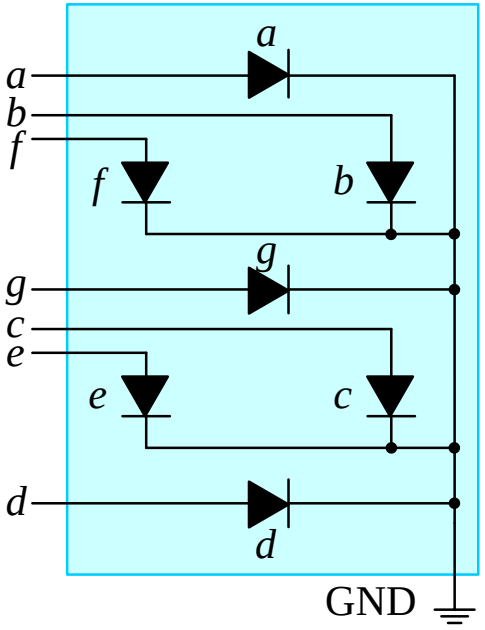
$$\bar{f} = BA + \bar{C}B + \bar{D}\bar{C}A$$



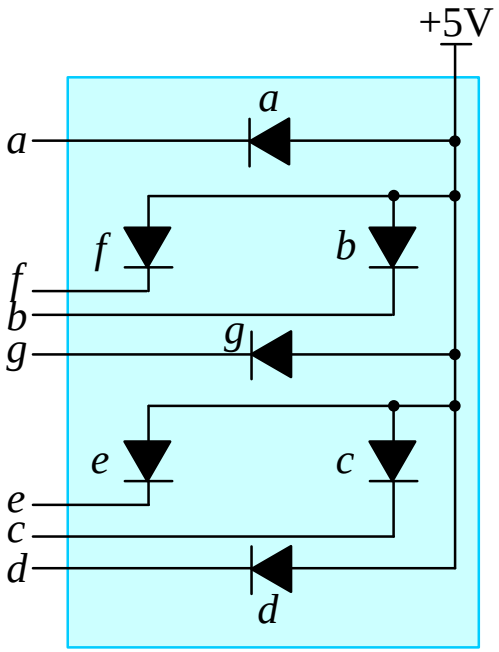
$$\bar{g} = \bar{D}\bar{C}\bar{B} + CBA$$

디코더(Decoder)

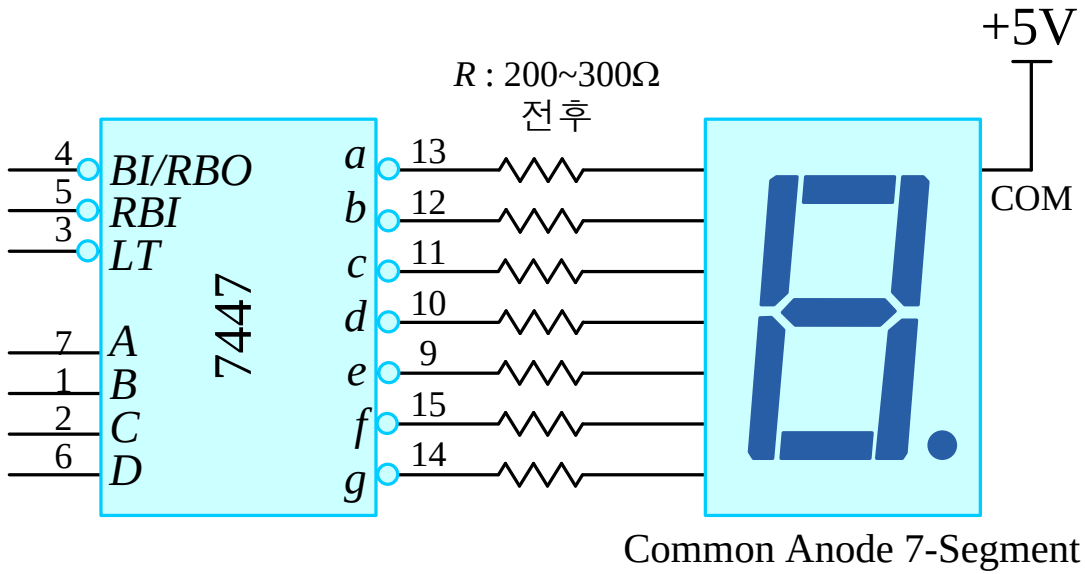
- 7-Segment 디코더
  - IC 7447: 7-segment 디코더로 많이 사용되는 IC 칩, active-low로 동작
  - 7-Segment 공통 회로와 IC 7447과의 연결



<캐소드 공통>



<애노드 공통>



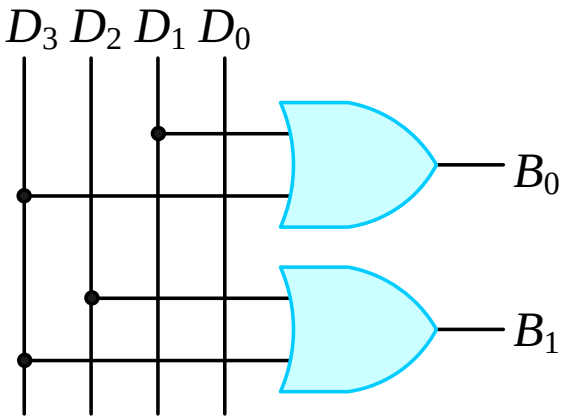
<전류 제한 저항을 사용한 7-세그먼트 회로의 예>

인코더(Encoder)

- 2^n개의 신호를 받아 n비트 2진 코드로 변경

- 4×2 인코더
  - 논리식: $B_1 = D_3 + D_2, B_0 = D_3 + D_1$
  - 진리표 및 논리회로

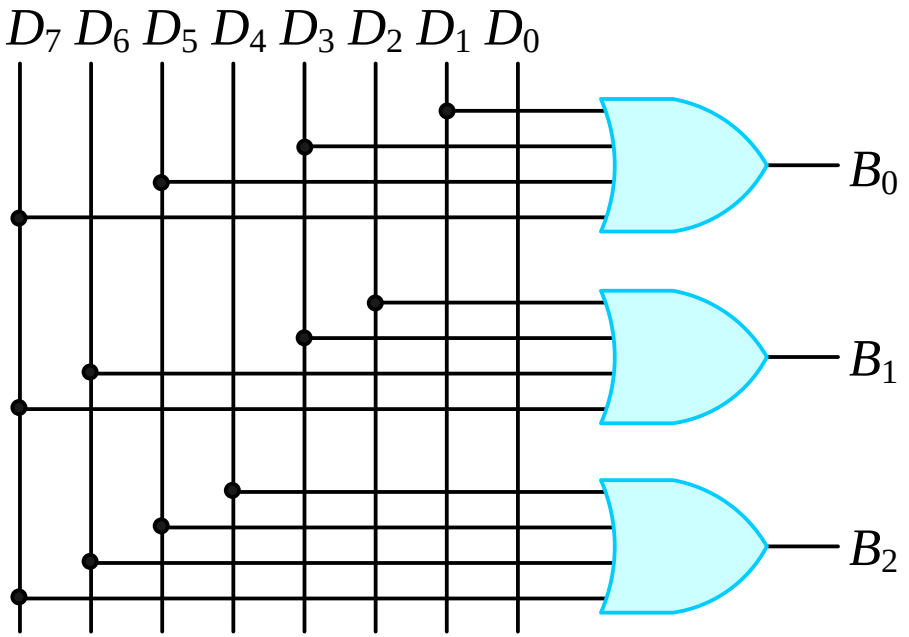
입력				출력	
D_3	D_2	D_1	D_0	B_1	B_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



인코더(Encoder)

- 8×3 우선 순위 인코더
  - 논리식: $B_2 = D_7 + D_6 + D_5 + D_4, B_1 = D_7 + D_6 + D_3 + D_2, B_0 = D_7 + D_5 + D_3 + D_1$
  - 진리표 및 논리회로

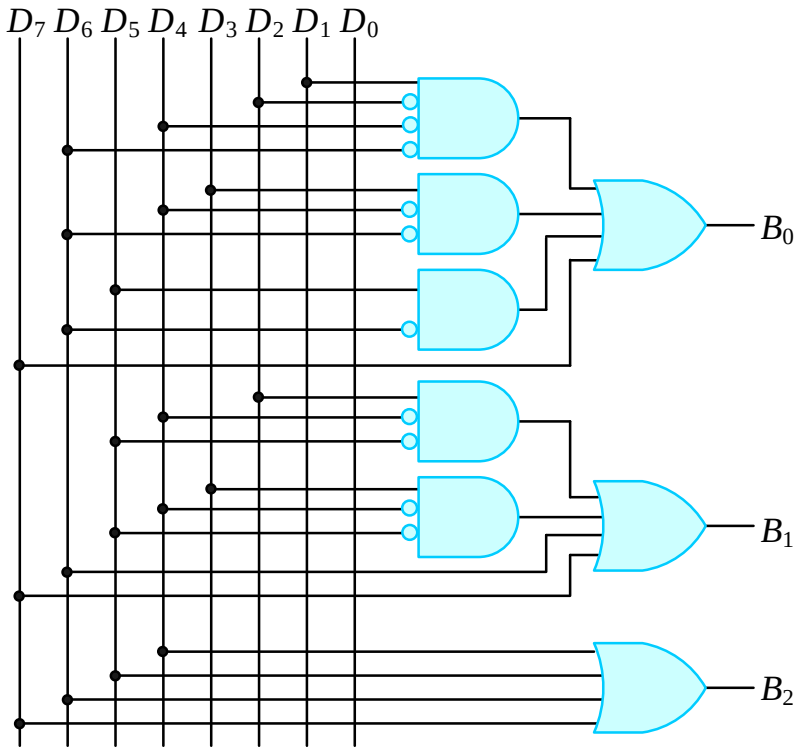
입력								출력		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	B_2	B_1	B_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1



인코더(Encoder)

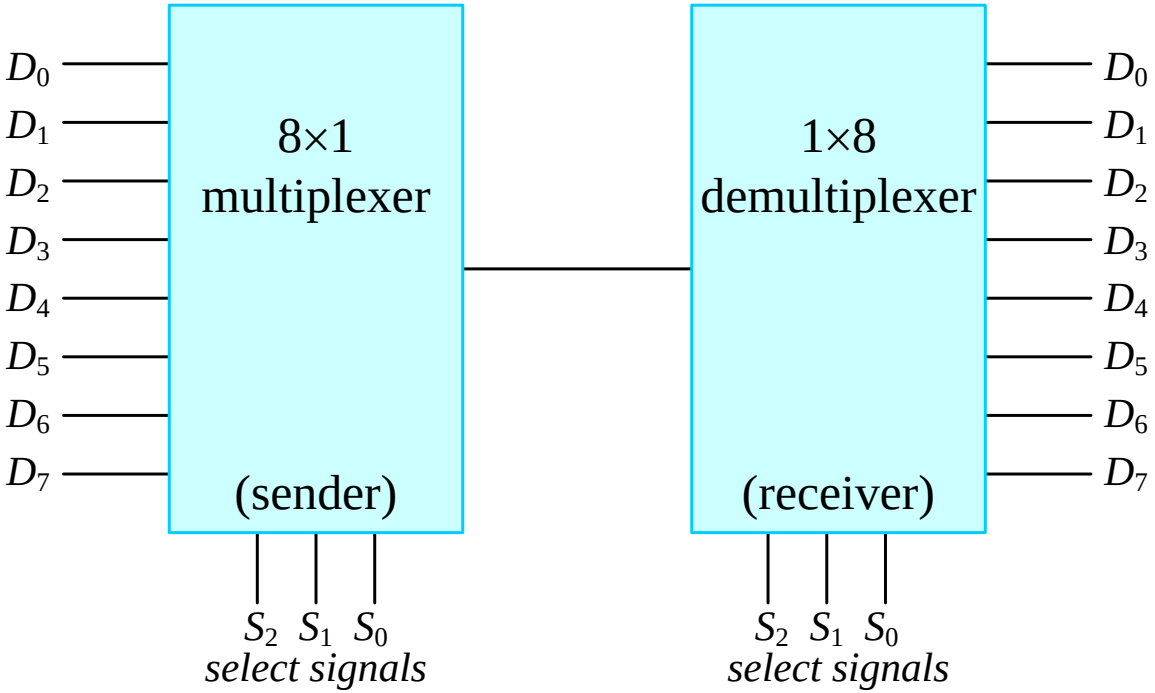
- 8×3 우선 순위 인코더
  - 논리식: $B_2 = D_7 + D_6 + D_5 + D_4$, $B_1 = D_7 + D_6 + \overline{D_5}\,\overline{D_4}(D_3 + D_2)$
    $B_0 = D_7 + \overline{D_6}D_5 + \overline{D_6}\,\overline{D_4}D_3 + \overline{D_6}\,\overline{D_4}\,\overline{D_2}D_1$
  - 진리표 및 논리회로

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



멀티플렉서와 디멀티플렉서

- Multiplexer vs. Demultiplexer

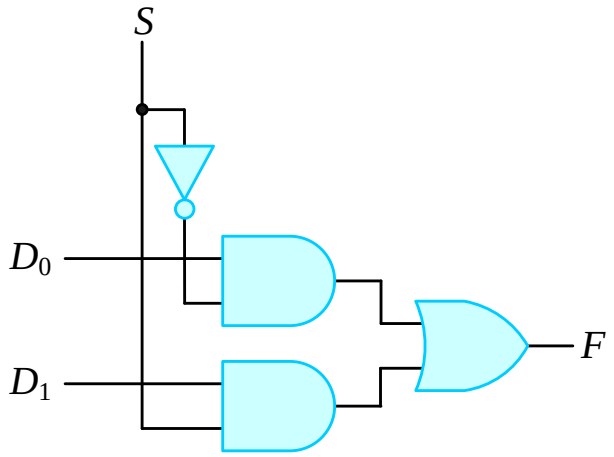


  - Multiplexer: D_0~D_7 중 $\{S_2S_1S_0\}$이 가리키는 것을 출력
  - Demultiplexer: D_0~D_7 중 $\{S_2S_1S_0\}$이 가리키는 곳에 출력

멀티플렉서(Multiplexer, MUX)

- 2×1 MUX
  - 논리식: $F = \bar{S}D_0 + SD_1$
  - 진리표 및 논리회로

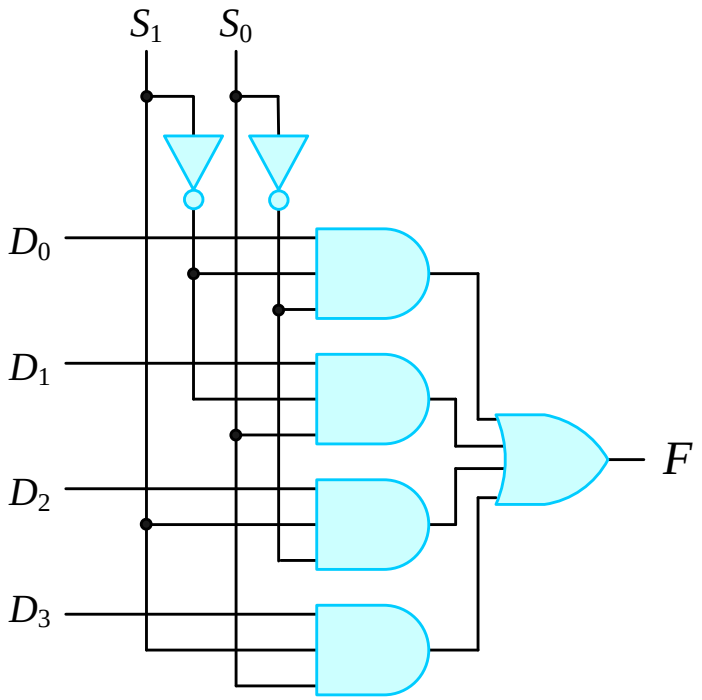
선택선	출력
S	F
0	D_0
1	D_1



멀티플렉서(Multiplexer, MUX)

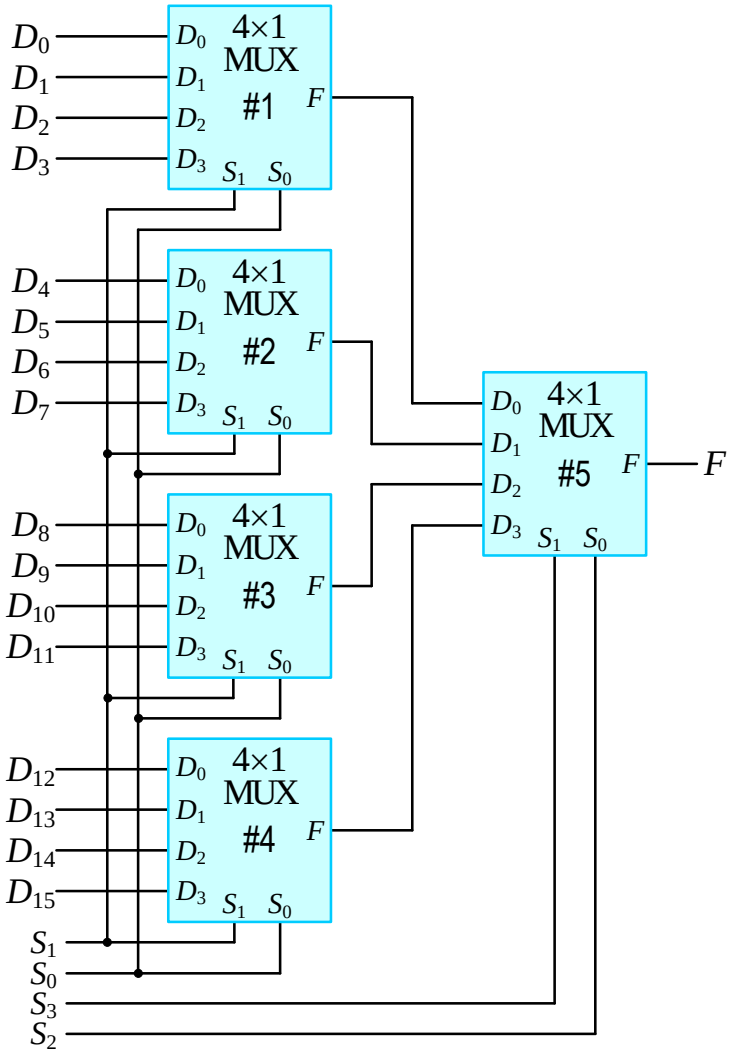
- 4×1 MUX
  - 논리식: $F = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$
  - 진리표 및 논리회로

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



멀티플렉서(Multiplexer, MUX)

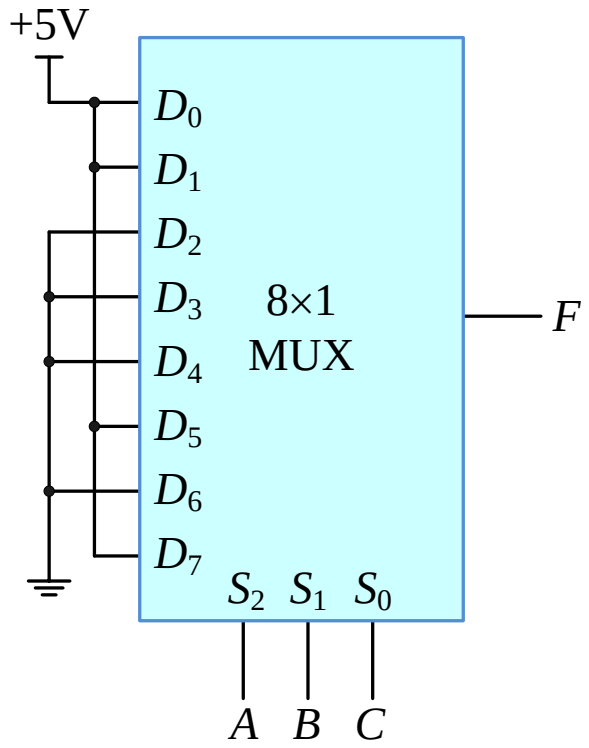
- 16×1 MUX
  - 4×1 MUX를 5개 활용하여 구성 가능
  - MUX #1: $\{S_3S_2S_1S_0\} = 0000 \sim 0011$
  - MUX #2: $\{S_3S_2S_1S_0\} = 0100 \sim 0111$
  - MUX #3: $\{S_3S_2S_1S_0\} = 1000 \sim 1011$
  - MUX #4: $\{S_3S_2S_1S_0\} = 1100 \sim 1111$
  - MUX #5: $\{S_3S_2S_1S_0\} = 00XX \sim 11XX$



멀티플렉서(Multiplexer, MUX)

- 멀티플렉서를 이용한 조합회로 구현
  - 예: $F(A,B,C) = \Sigma m(0,1,5,7)$
    - 진리표

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)

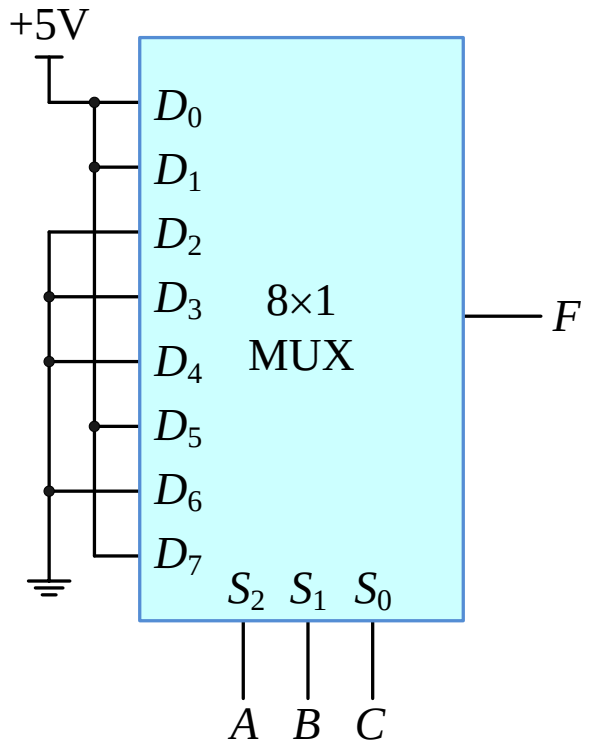


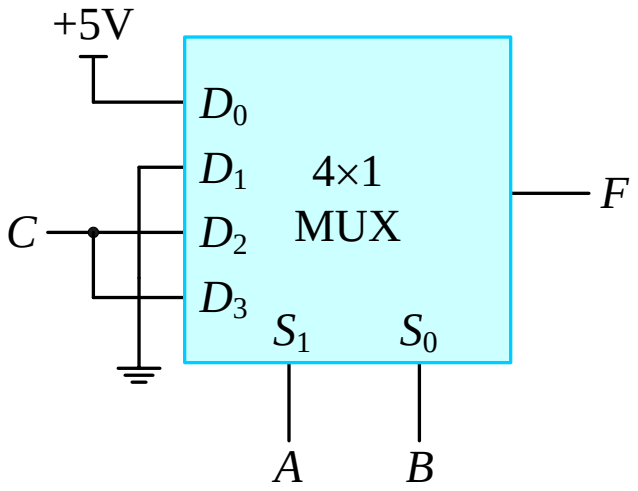
    - 8×1 MUX 사용
      - A, B, C를 선택 신호에 연결
      - minterm에 대응하는 $D_0 \sim D_7$ 중 $F = 0, 1$을 각각 GND와 VDD로 연결

멀티플렉서(Multiplexer, MUX)

- 멀티플렉서를 이용한 조합회로 구현
  - 예: $F(A,B,C) = \Sigma m(0,1,5,7)$
    - 진리표

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)



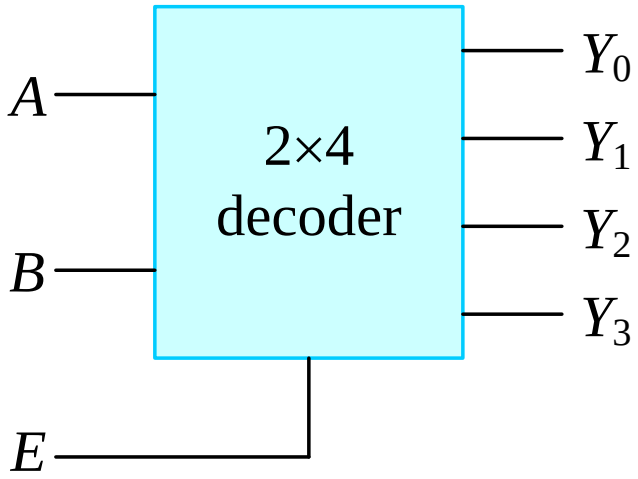


    - 4×1 MUX 사용
      - A, B를 선택 신호에 연결
      - $D_0 \sim D_3$ 중 $F = 0, 1, \bar{C}, C$을 각각 GND, VDD, $\bar{C}$, C와 연결

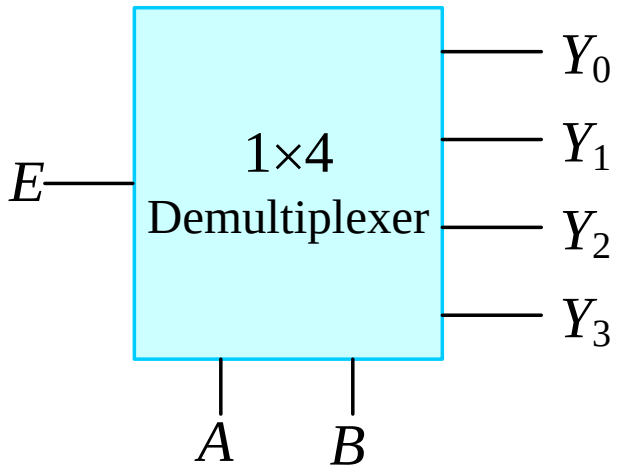
디멀티플렉서(Demultiplexer, DEMUX)

- 한 개의 입력을 n개의 출력 중 선택된 한 곳에 전달

- Decoder vs. DEMUX



<2×4 디코더>



<1×4 디멀티플렉서>

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

  - enable 신호를 가진 $n \times 2^n$ decoder = 1×2^n DEMUX

코드 변환기

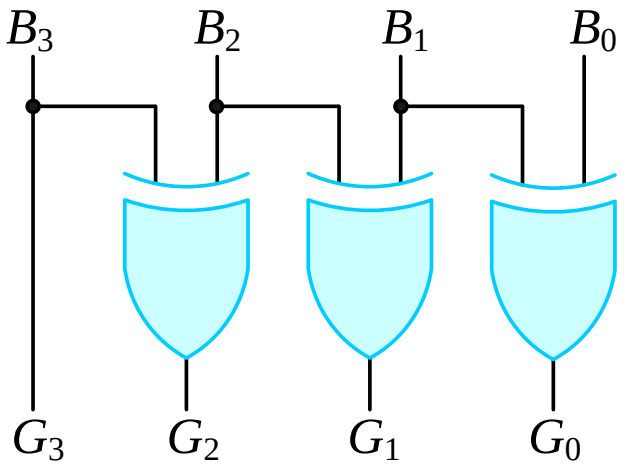
- 2진코드-그레이코드

2진 코드				그레이 코드			
B_3	B_2	B_1	B_0	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

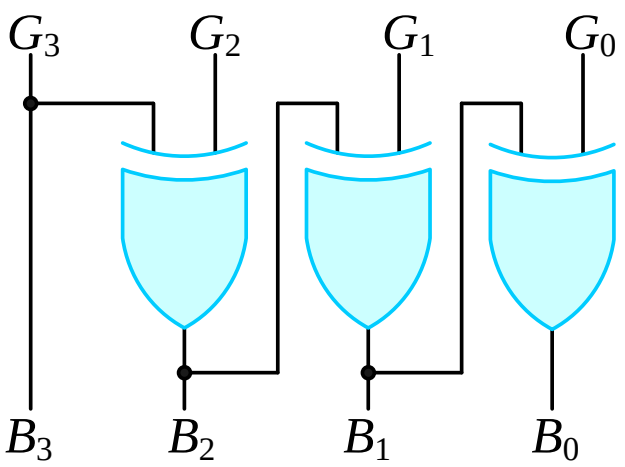
코드 변환기

- 2진코드-그레이코드

2진 코드				그레이 코드			
B_3	B_2	B_1	B_0	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



<2진코드 → 그레이코드>

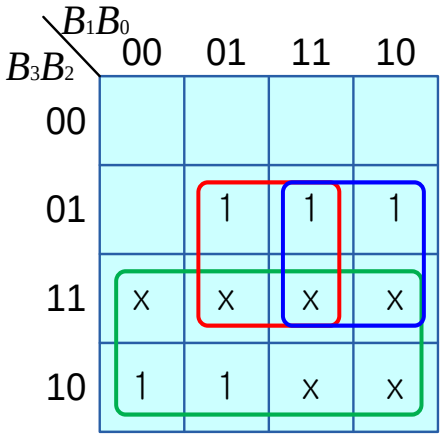


<그레이코드 → 2진코드>

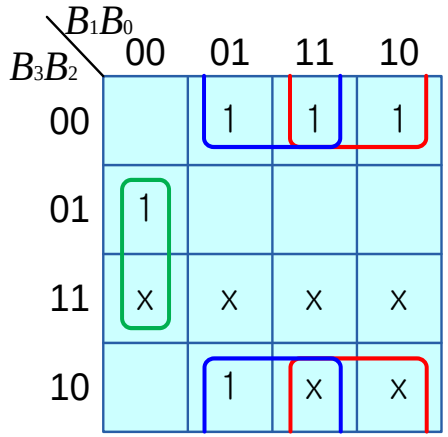
코드 변환기

- BCD코드-3초과 코드

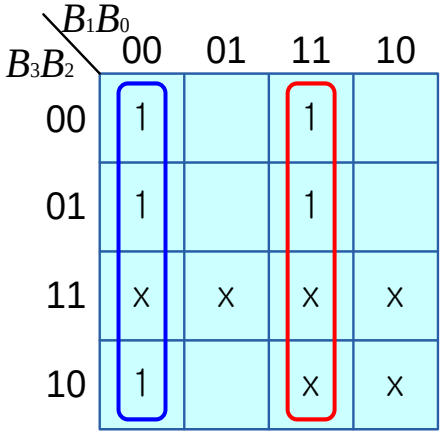
2진 코드(입력)				3초과 코드(출력)			
B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



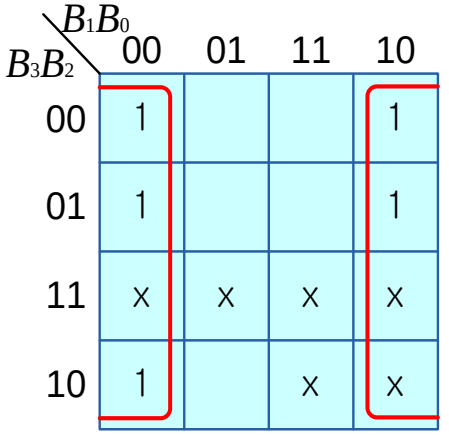
$$E_3 = B_3 + B_2B_1 + B_2B_0$$



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



$$E_1 = \overline{B}_1\overline{B}_0 + B_1B_0 = \overline{B_1 \oplus B_0}$$

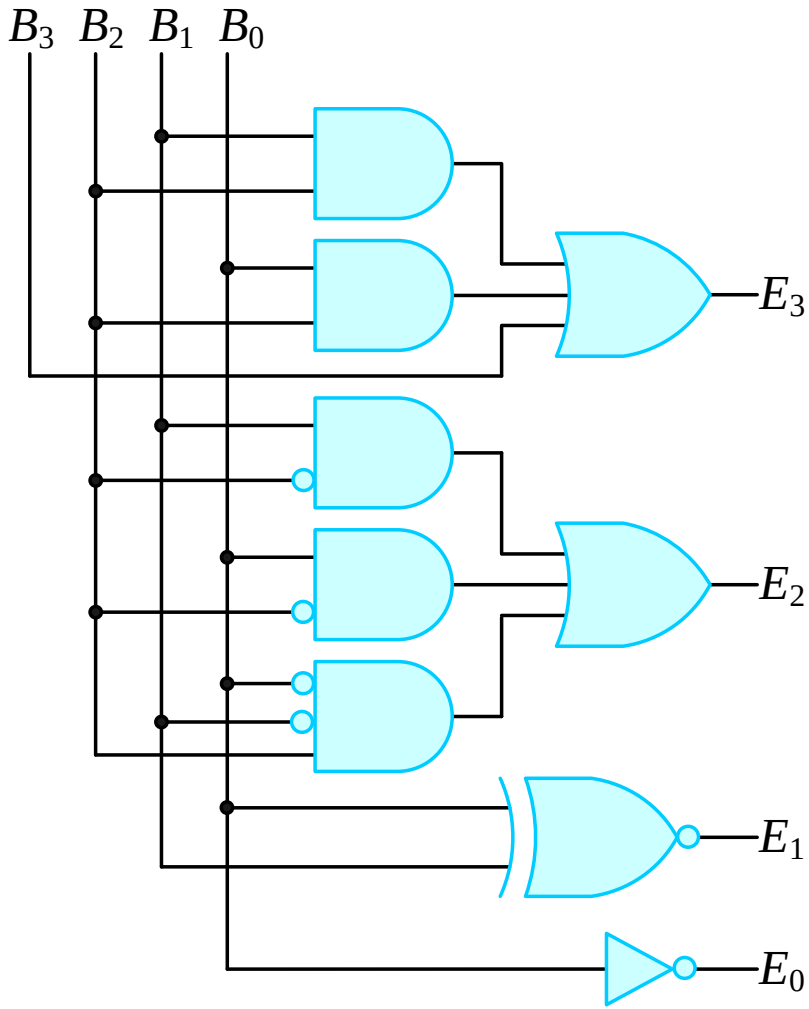


$$E_0 = \overline{B}_0$$

코드 변환기

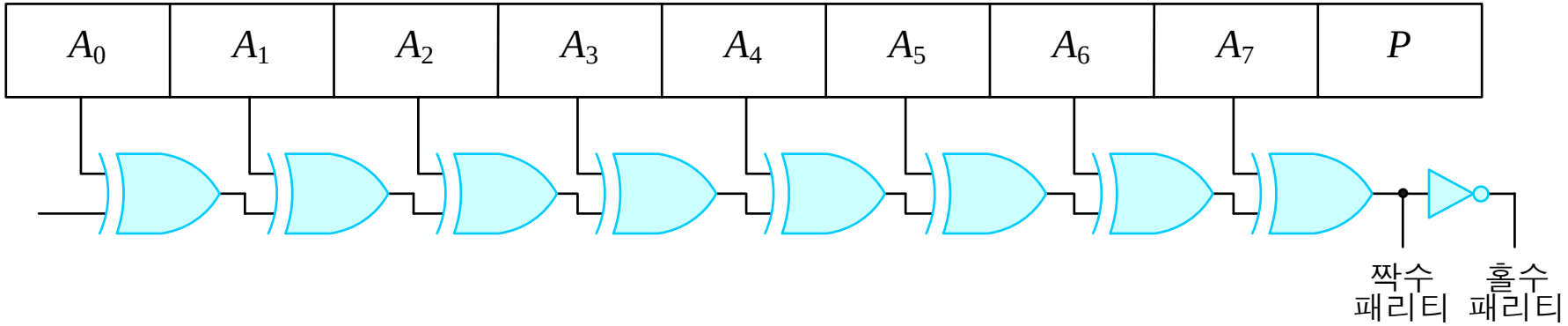
- BCD코드-3초과 코드

2진 코드(입력)				3초과 코드(출력)			
B_3	B_2	B_1	B_0	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



패리티 발생기/검출기

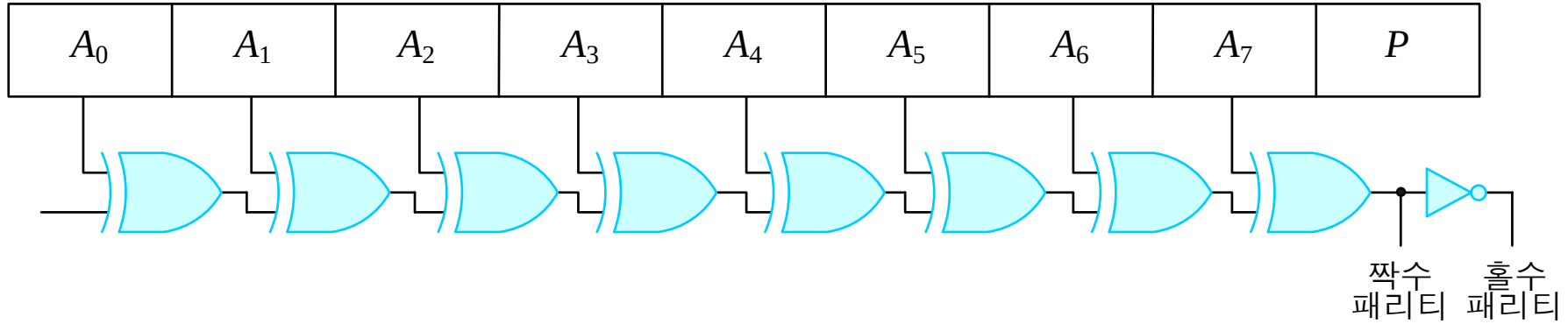
- 패리티 비트의 생성
  - 짝수 패리티: 1의 개수[a]가 짝수 → 데이터 비트 XOR
  - 홀수 패리티: 1의 개수[a]가 홀수 → 데이터 비트 XNOR



[a]데이터 비트와 패리티 비트를 모두 포함하였을 때

패리티 발생기/검출기

- 패리티 비트의 생성
  - 짝수 패리티: 1의 개수[a]가 짝수 → 데이터 비트 XOR
  - 홀수 패리티: 1의 개수[a]가 홀수 → 데이터 비트 XNOR



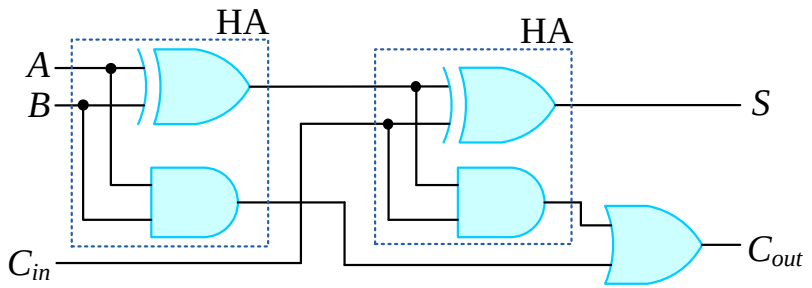
- 오류 검출
  - 짝수 패리티: 오류[b] 발생 시 1의 개수[a]가 홀수 → XOR한 값 = 1
  - 홀수 패리티: 오류[b] 발생 시 1의 개수[a]가 짝수 → XOR한 값 = 0

[a]데이터 비트와 패리티 비트를 모두 포함하였을 때

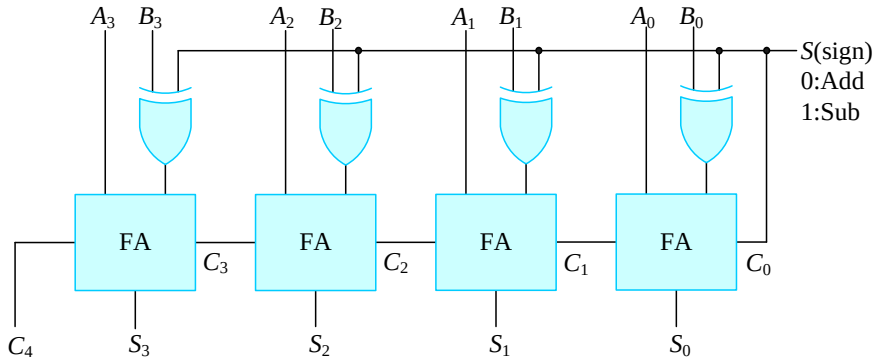
[b]한 비트에서 오류가 발생하였을 때를 의미, 홀수 개의 비트에서 오류가 발생하는 경우도 포함

Summary

- 가산기
  - HA: 두 비트 A, B 덧셈 → 출력 S, C
  - FA: 세 비트 A, B, C_{in} 덧셈 → 출력 S, C_{out}
    - 2개의 HA와 OR 게이트로 구성 가능
  - 병렬가감산기: FA를 여러 개 병렬로 연결
    - XOR 게이트를 추가하여 뺄셈까지 수행 가능
  - CLA: carry 별도 계산 → 빠른 덧셈



<FA>



<4-bit 가감산기>

Summary

- 비교기
  - $A \neq B$: A와 B의 같은 위치의 비트끼리 XOR → 결과 OR
  - $A = B$: A와 B의 같은 위치의 비트끼리 XOR → 결과 NOR
  - $A > B$: 예) 2비트 비교

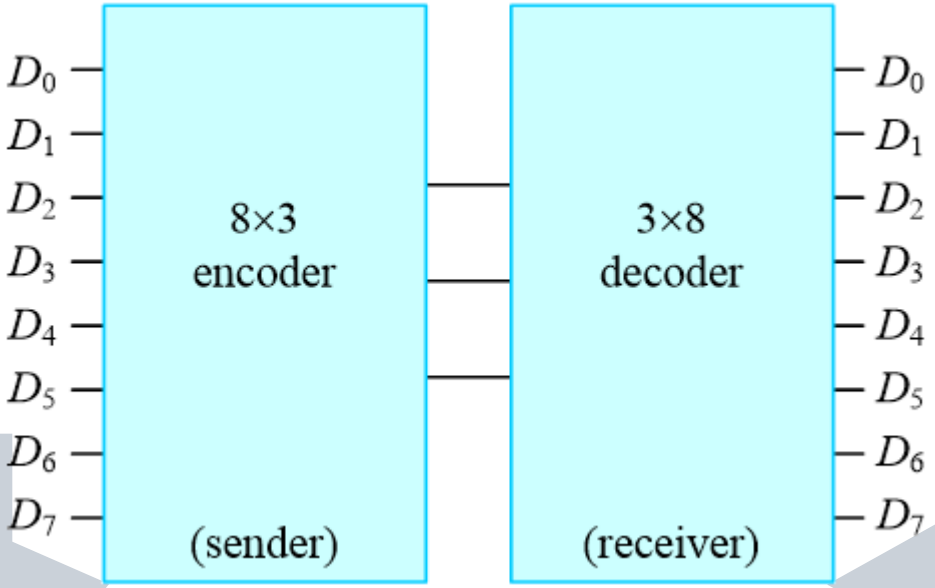
$\{A_1A_0\}$ \ $\{B_1B_0\}$	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

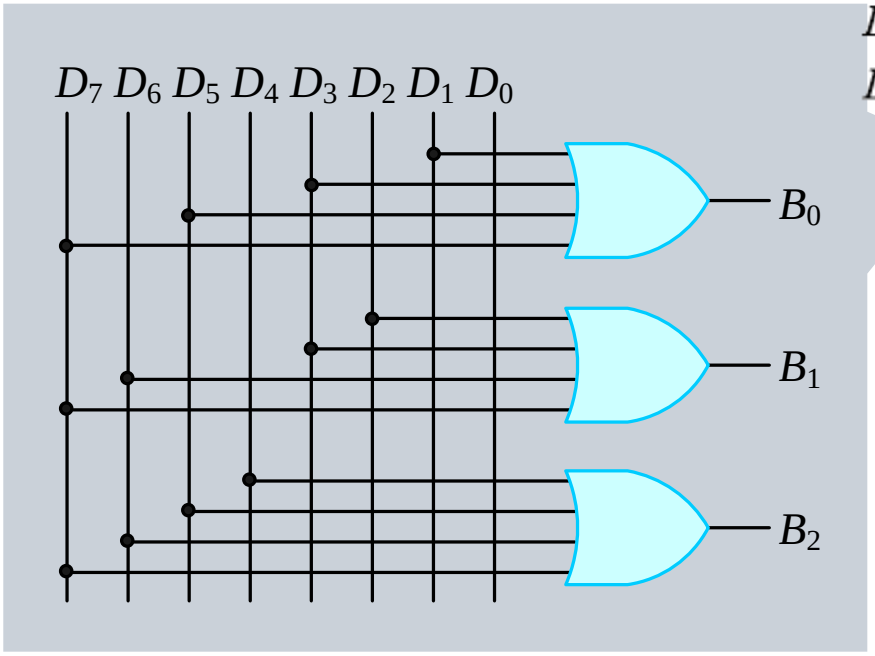
$$A > B = A_1\overline{B_1} + A_1A_0\overline{B_0} + A_0\overline{B_1}\,\overline{B_0}$$

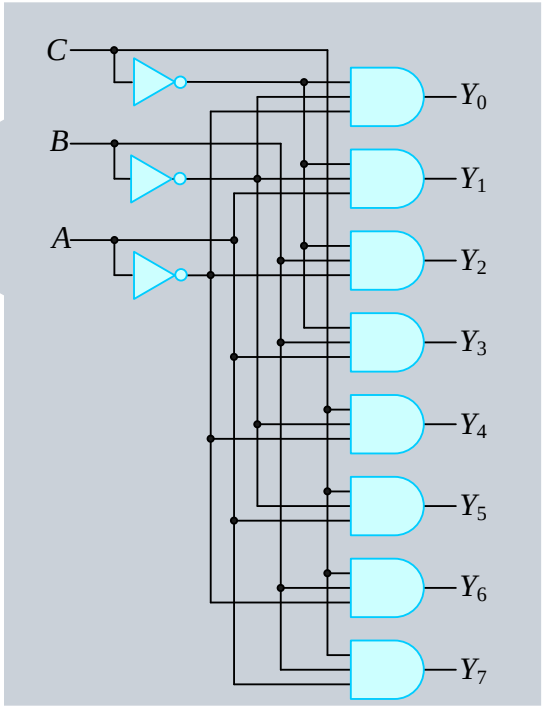
  - $A < B = \overline{(A > B)} \cdot (A \neq B)$

Summary

- Encoder vs. Decoder: $2^n \rightarrow n$ vs. $n \rightarrow 2^n$

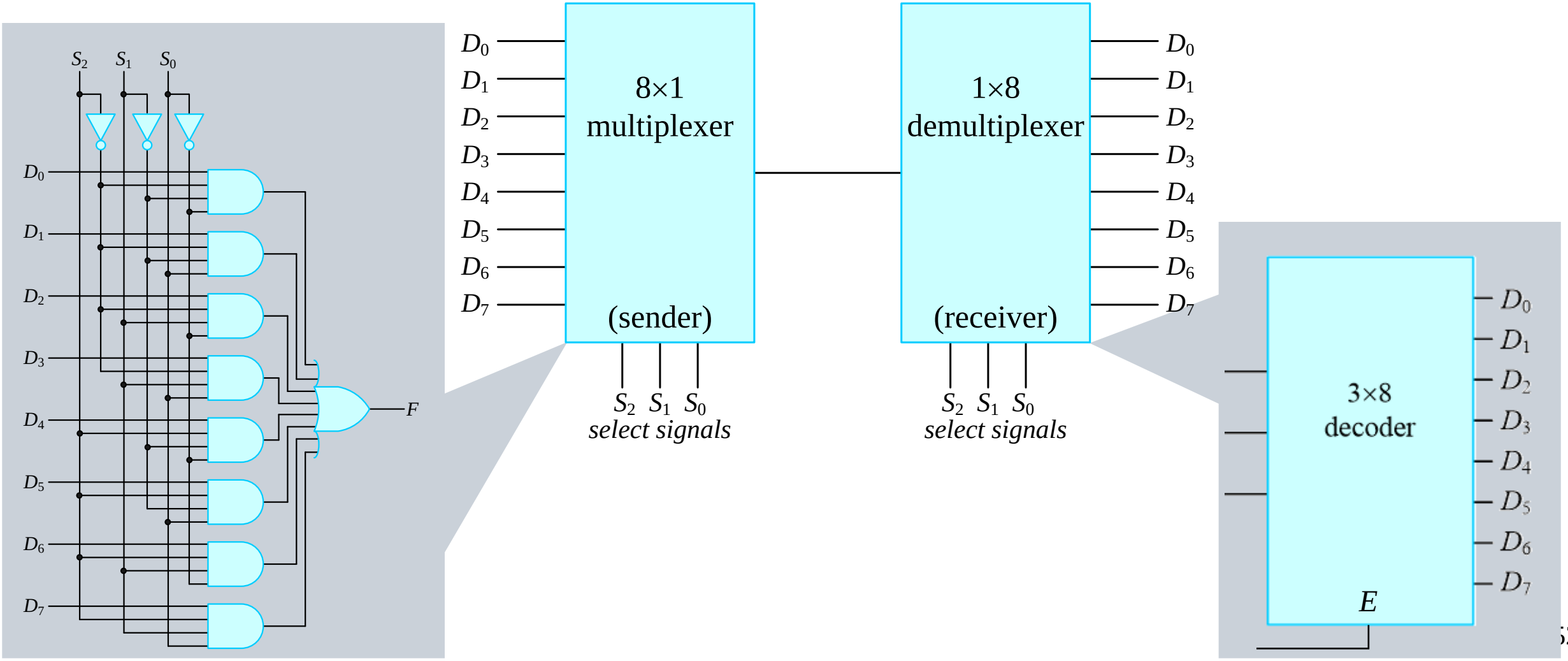






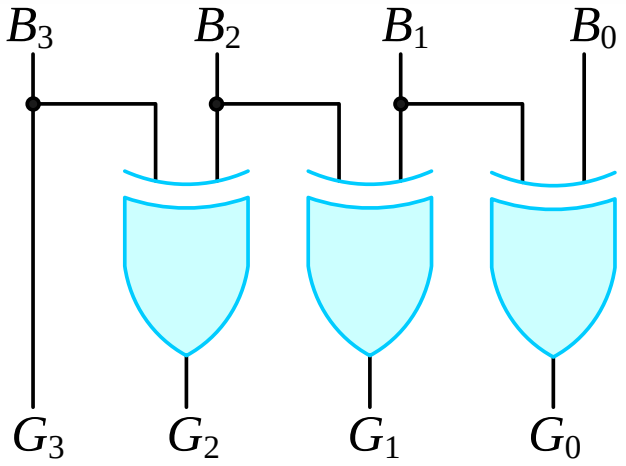
Summary

- MUX vs. DEMUX: $2^n \rightarrow 1$ vs. $1 \rightarrow 2^n$

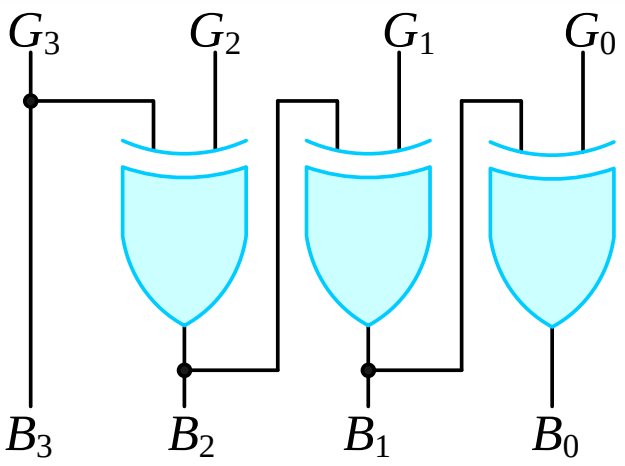


Summary

- 코드 변환기
  - 2진코드-그레이코드:



<2진코드 → 그레이코드>



<그레이코드 → 2진코드>

- 패리티 발생기/검출기 (짝수 패리티 기준)
  - 패리티 비트의 생성: 데이터 비트 부분을 XOR
  - 오류 검출: 모든 비트 XOR한 결과가 1이면 오류 발생

- Etc.: 7-Segment decoder, BCD-3초과 코드