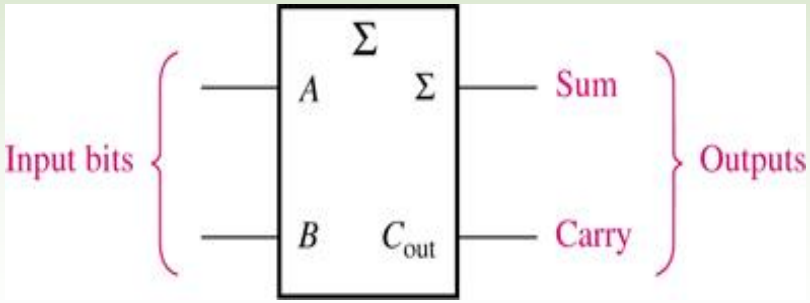
6장. 조합 논리의 기능

- 가산기 (1, 2)
- 비교기(3)
- 디코더(4)
- 인코더(5)
- 코드변환기(6)
- 멀티플렉서(데이터 선택기)(7)
- 디멀티플렉서(8)
- 패리티 생성기/검출기(9)

기본 가산기(Basic Adders)

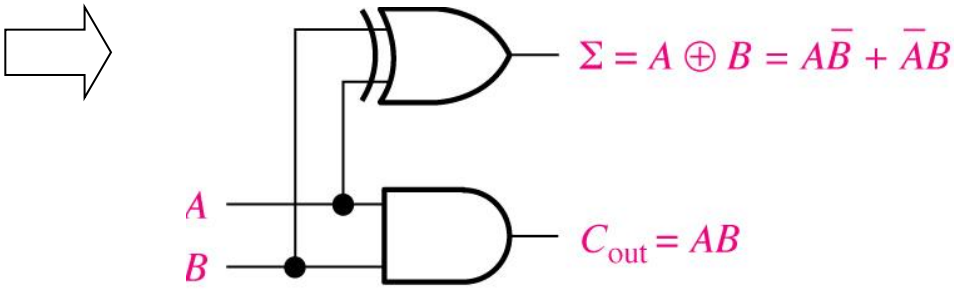
- 반가산기

  * 2개의 입력 비트,
    2개의 출력 비트(합, 캐리)



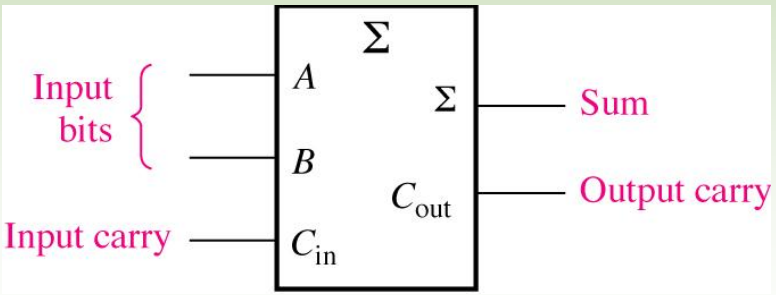
반가산기의 진리표

A	B	Σ	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

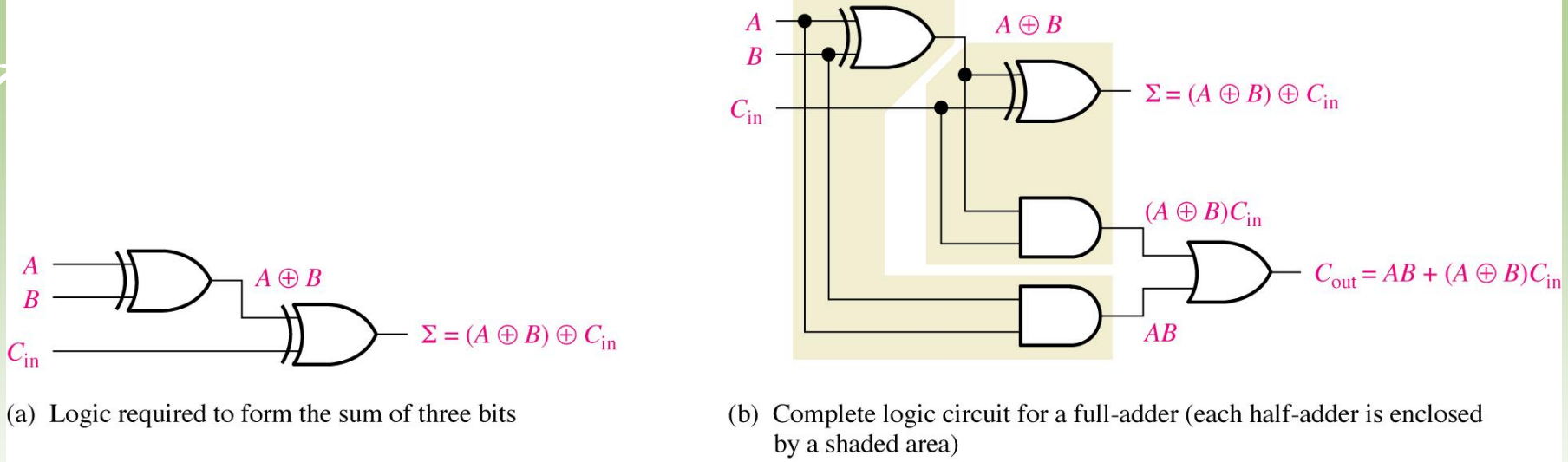


기본 가산기 (Basic Adders)

- 전가산기
  - * 2개의 입력 비트, 한 개의 캐리 입력
    2개의 출력 비트(합, 캐리)

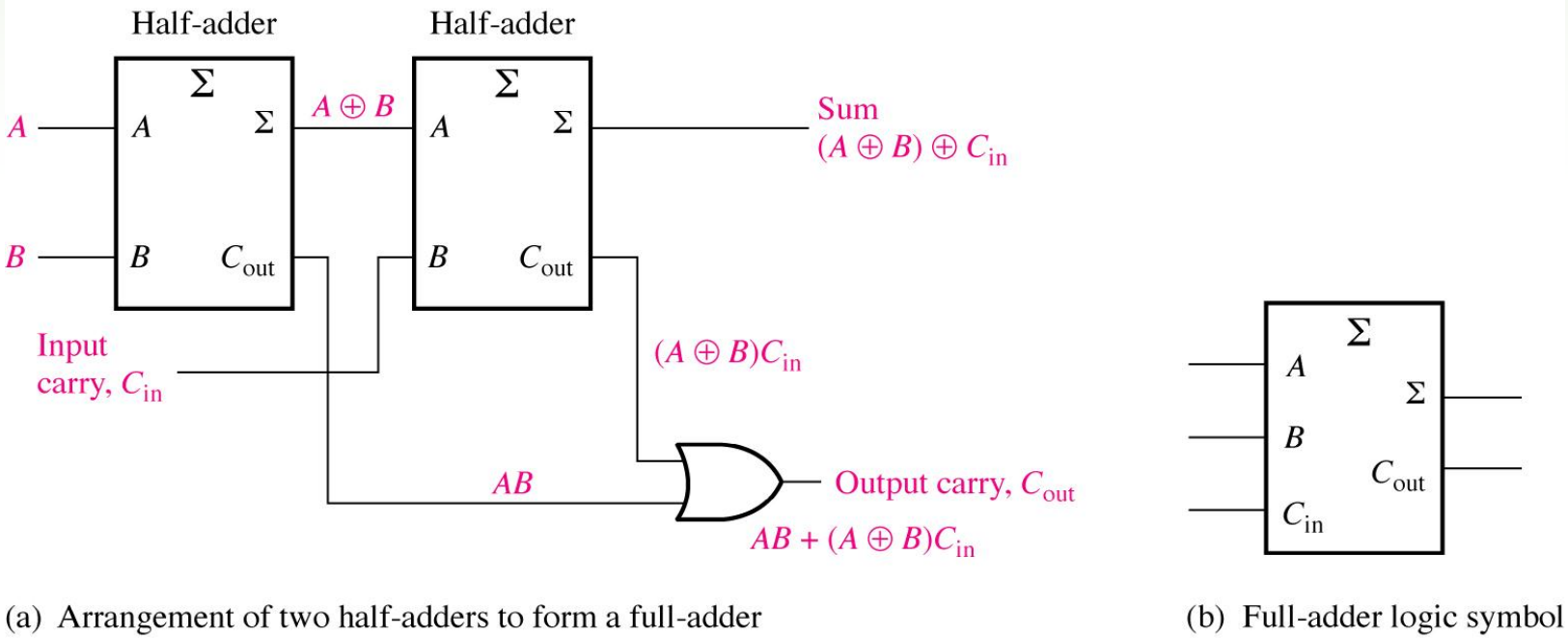


A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



(a) Logic required to form the sum of three bits

(b) Complete logic circuit for a full-adder (each half-adder is enclosed by a shaded area)



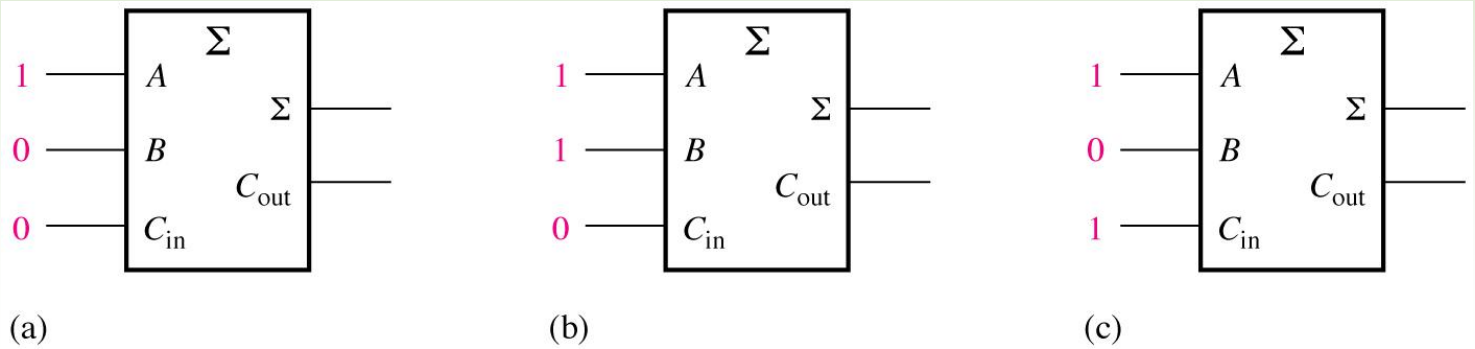
(a) Arrangement of two half-adders to form a full-adder

(b) Full-adder logic symbol

Figure 6-5 Full-adder implemented with half-adders.

기본 가산기 (Basic Adders)

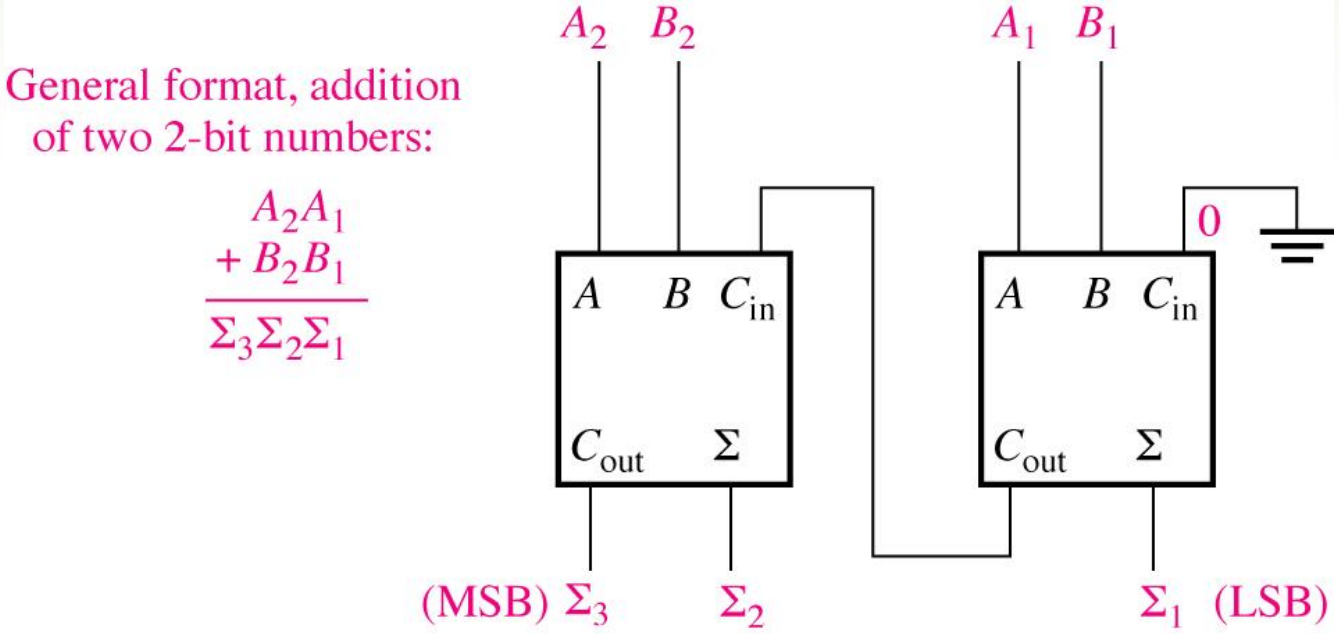
* 예제 6-1) 출력을 구하라.



병렬 2진 가산기(Parallel Binary Adders)

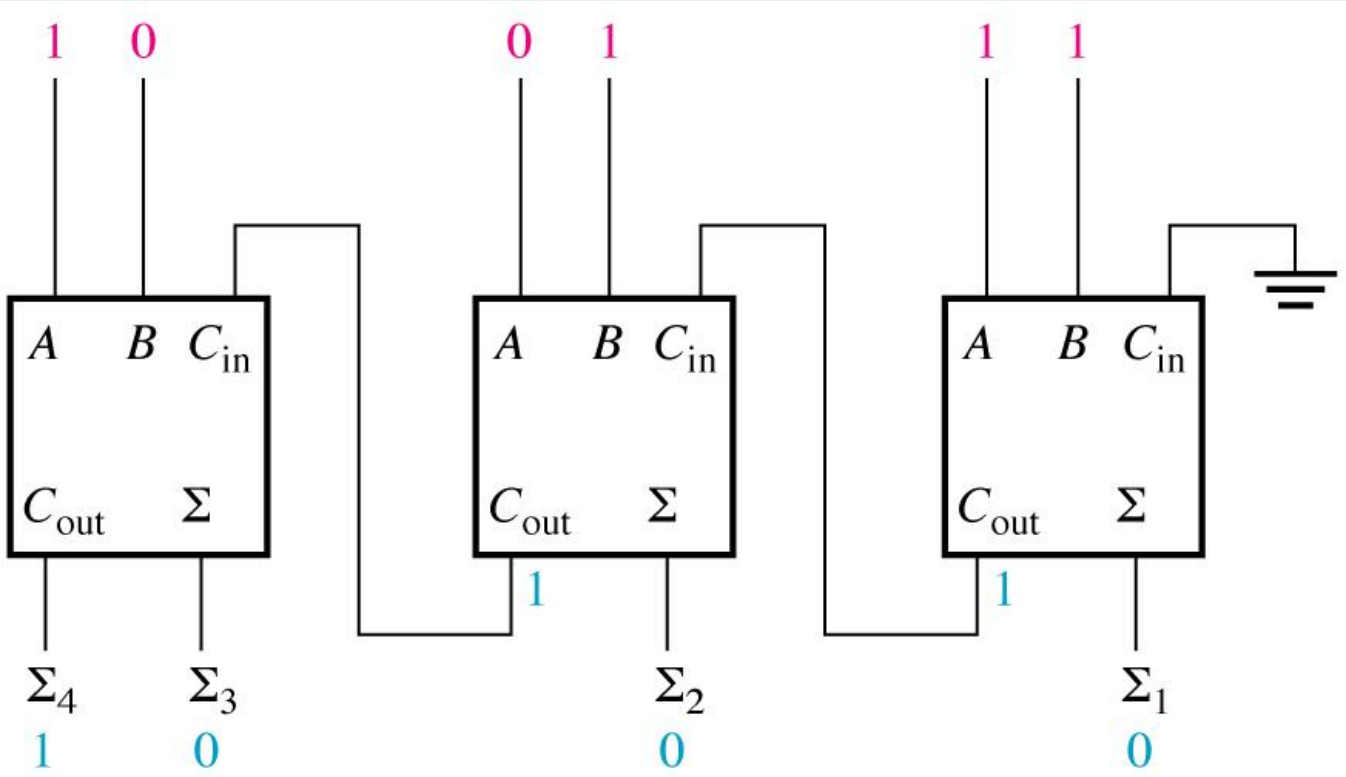
- 2-비트 병렬 가산기

  * 2개의 전가산기 사용 (LSB는 반가산기 사용 가능)



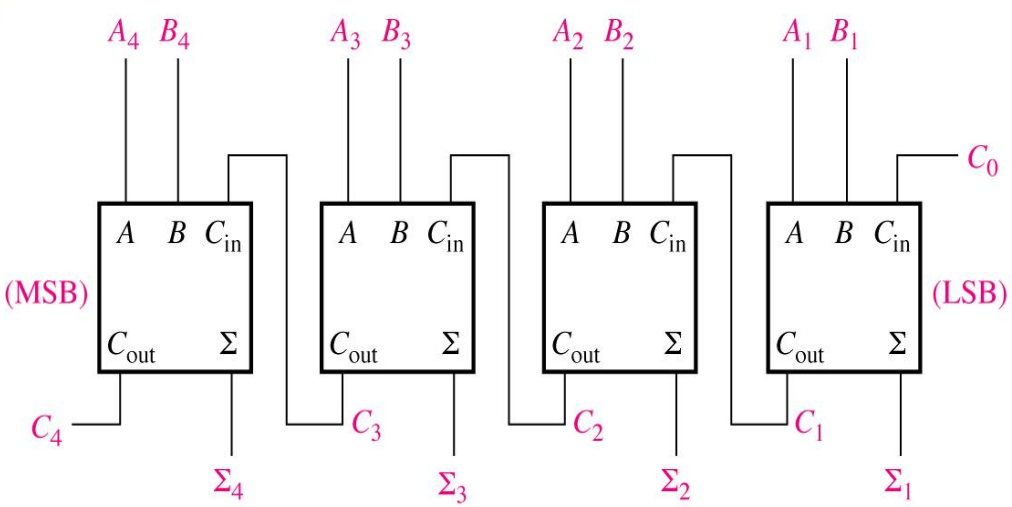
병렬 2진 가산기 (Parallel Binary Adders)

* 예제 6-2) 2진수 101 + 011

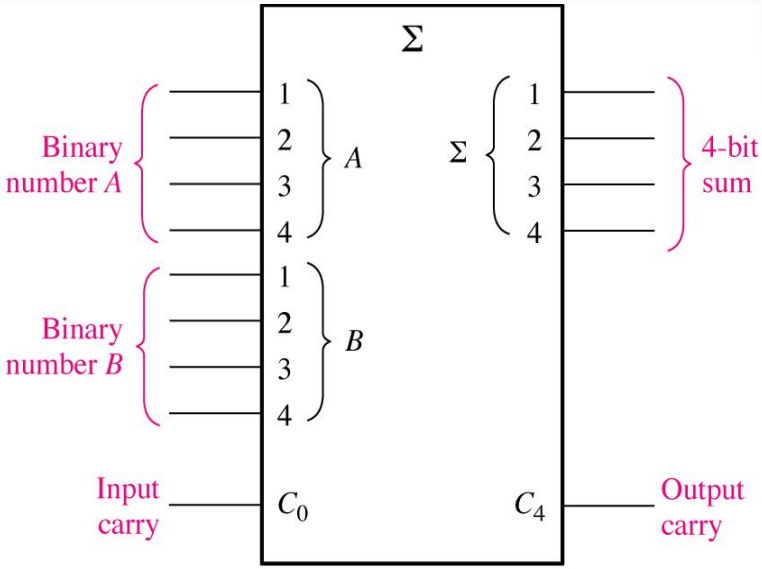


병렬 2진 가산기 (Parallel Binary Adders)

- 4-비트 병렬 가산기
  - * 4비트 : nibble
  - * 4개의 전가산기로 구성



(a) Block diagram



(b) Logic symbol

병렬 2진 가산기 (Parallel Binary Adders)

* 4비트 병렬 가산기의 진리표

C_{n-1}	A_n	B_n	Σn	Cn
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

예제 6-3) 입력캐리($Cn-1$) = 0,
A = 1100, B=1100 일 때, 합과 캐리는?

n=1 일 때, A_1 =0, B_1 =0, C_0 =0
=> Σ_1 =0, C_1 =0

n=2 일 때, A_2 =0, B_2 =0, C_1 =0
=> Σ_2 =0, C_2 =0

n=3 일 때, A_3 =1, B_3 =1, C_2 =0
=> Σ_3 =0, C_3 =1

n=4 일 때, A_4 =1, B_4 =1, C_3 =1
=> Σ_4 =1, C_4 =1(출력캐리)

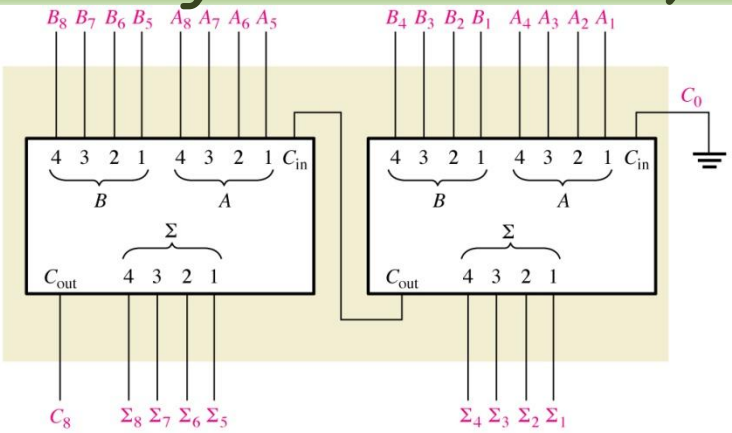
=> 최종 합 : 11000

병렬 2진 가산기 (Parallel Binary Adders)

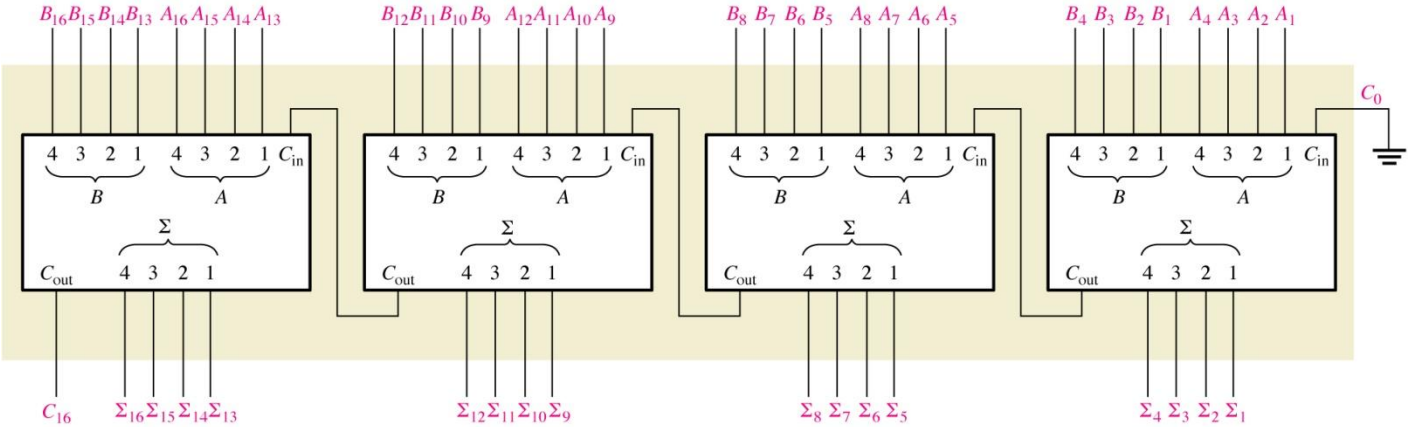
* 가산기 확장(Cascading)

* 4비트 가산기 2개
=> 8비트 가산기

* 4비트 가산기 4개
=> 16비트 가산기



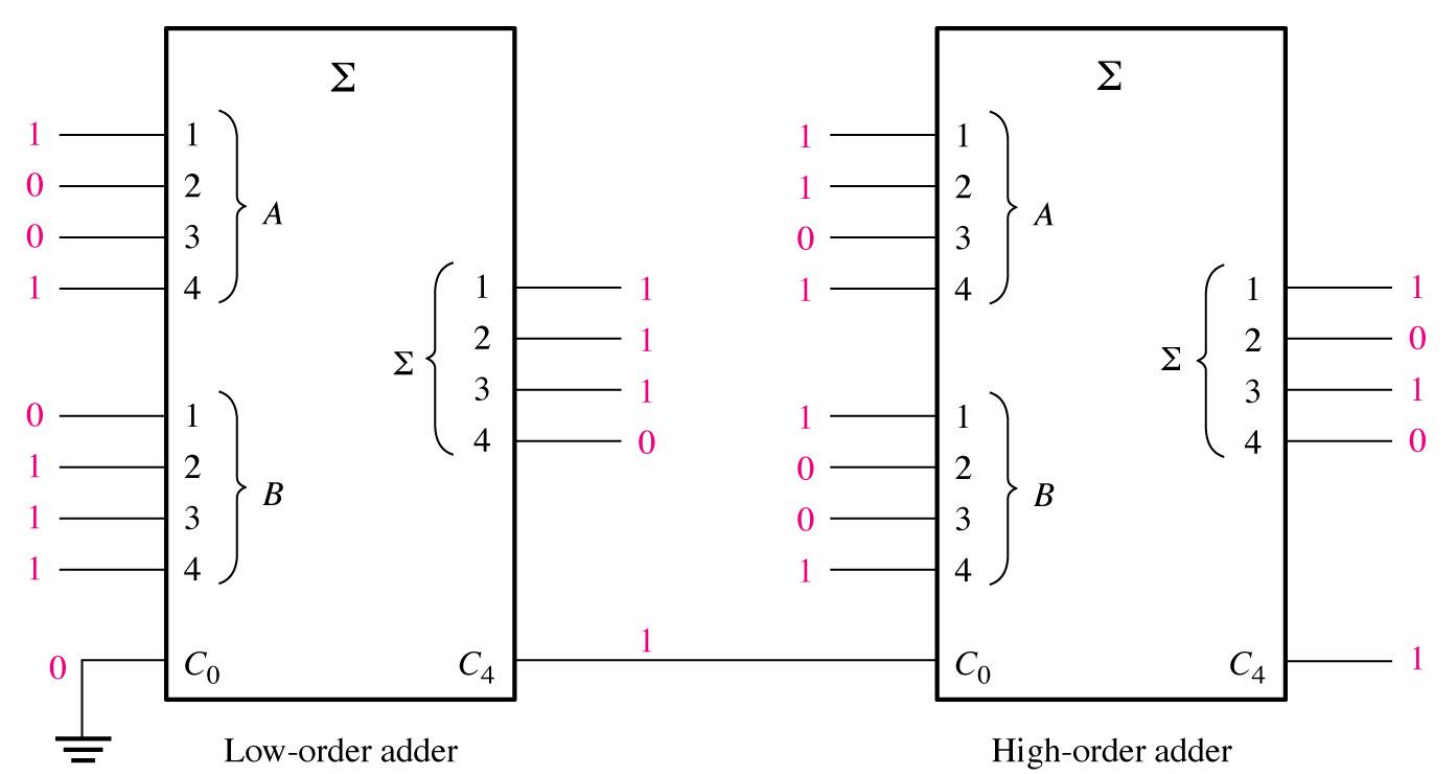
(a) Cascading of two 4-bit adders to form an 8-bit adder



(b) Cascading of four 4-bit adders to form a 16-bit adder

병렬 2진 가산기 (Parallel Binary Adders)

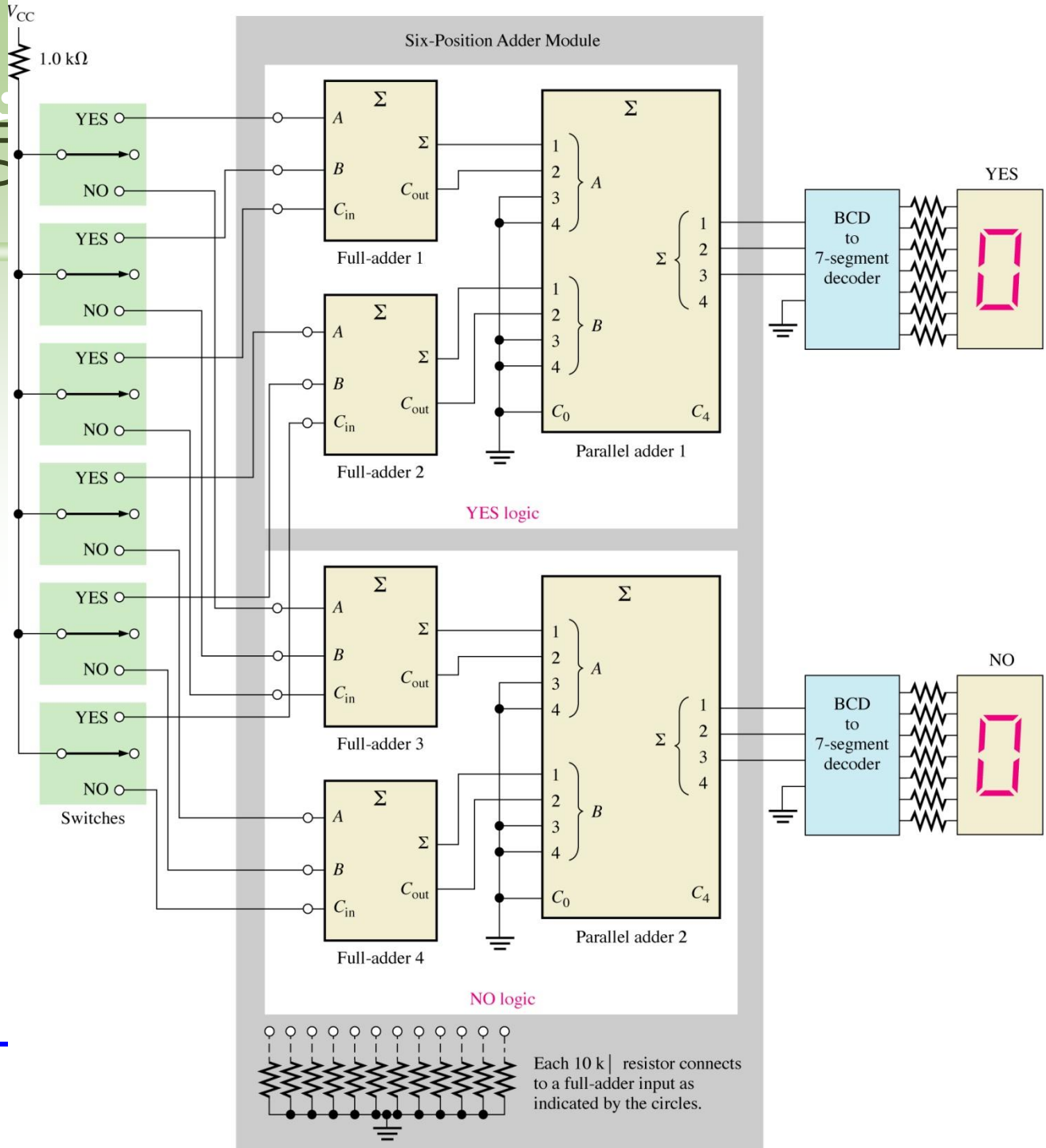
예제6-4) 2개의 4비트 병렬 가산기를 사용한 8비트 병렬 가산기 구성.
A=10111001, B=10011110 => 101010111



응용

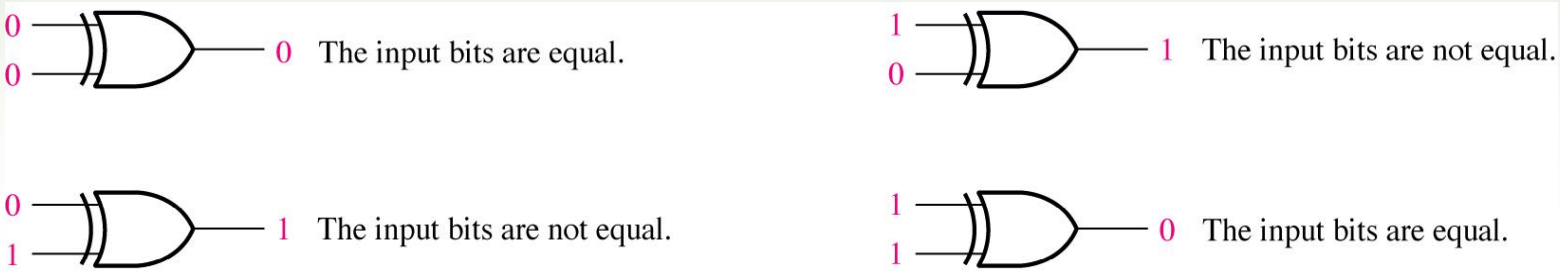
* 응용 예제 ;
 6명의 찬/반 집계기

* 3개 합/전가산기

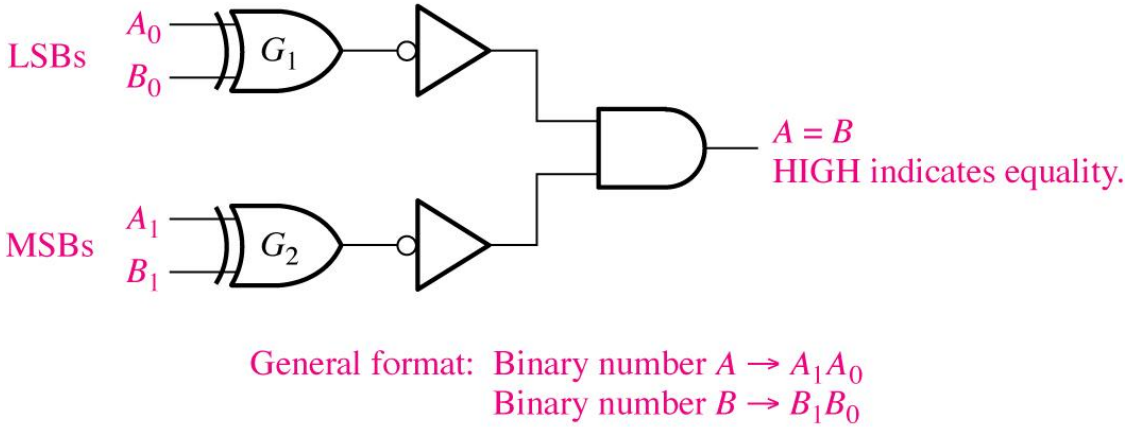


비교기(Comparators)

- 등가 비교기

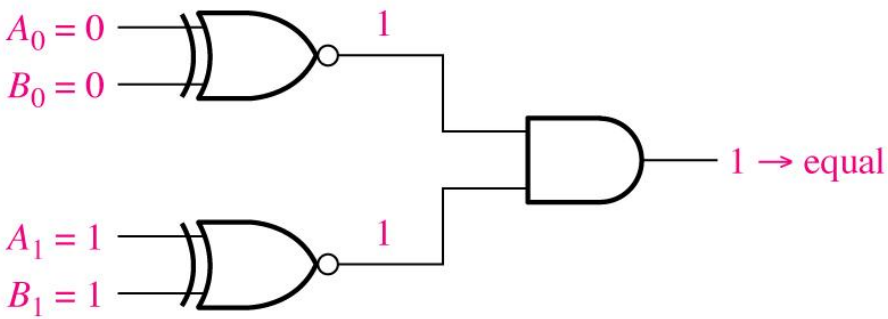


* 두개의 2비트 수의 등가 비교

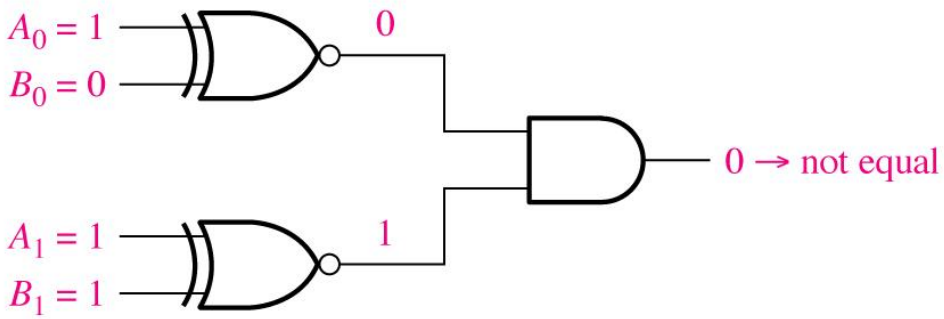


비교기 (Comparators)

* 예제 6-5) (a) 10과 10 (b) 11과 10

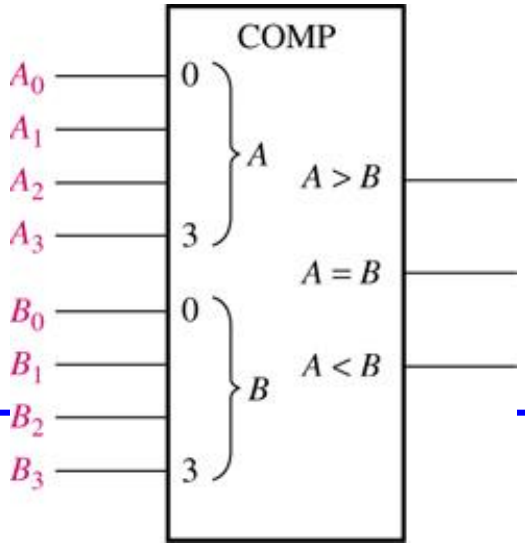


(a)



(b)

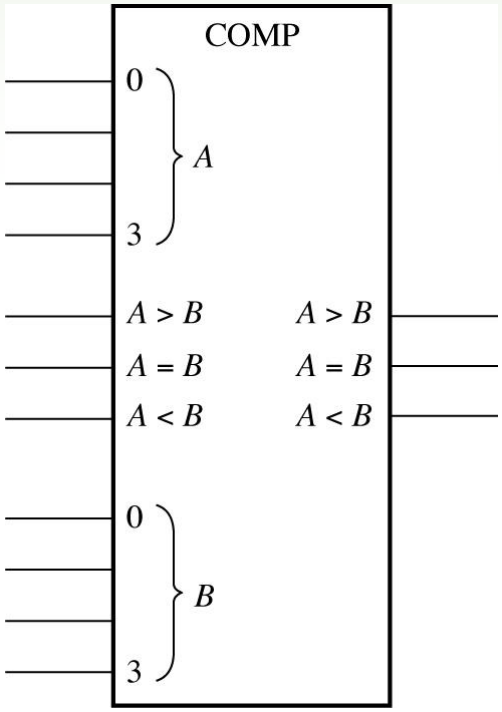
- 부등가 비교기



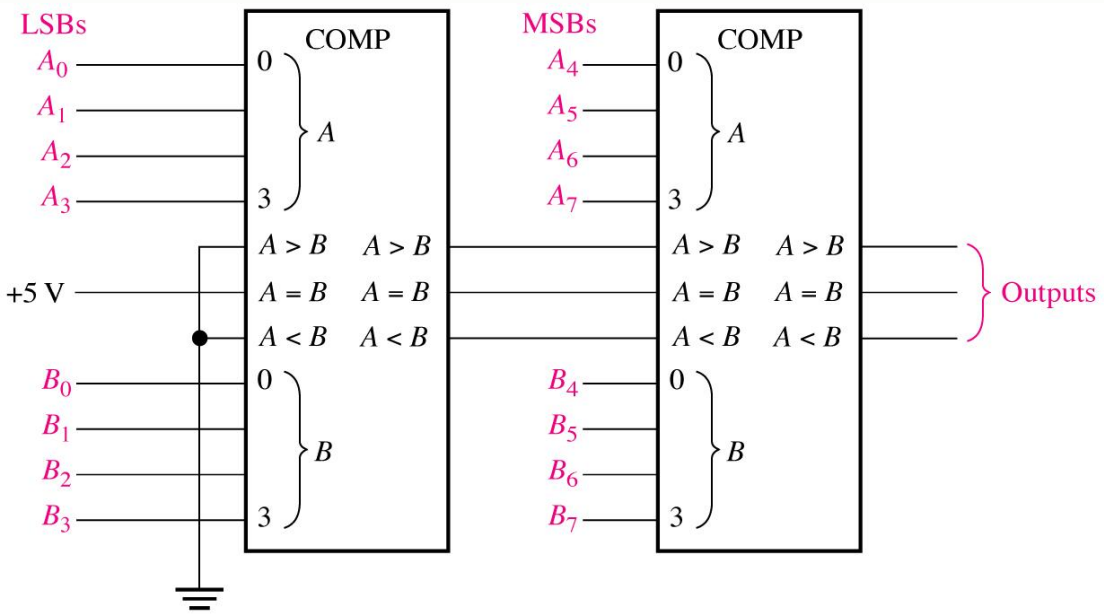
비교기 (Comparators)

* 비교기의 캐스케이딩 연결

=> 비교되는 비트수를 늘린다. 비교기의 출력이 다음 비교기의 입력으로.



예6-7) 4비트 비교기를 사용하여 두개의 8비트 수 크기 비교.

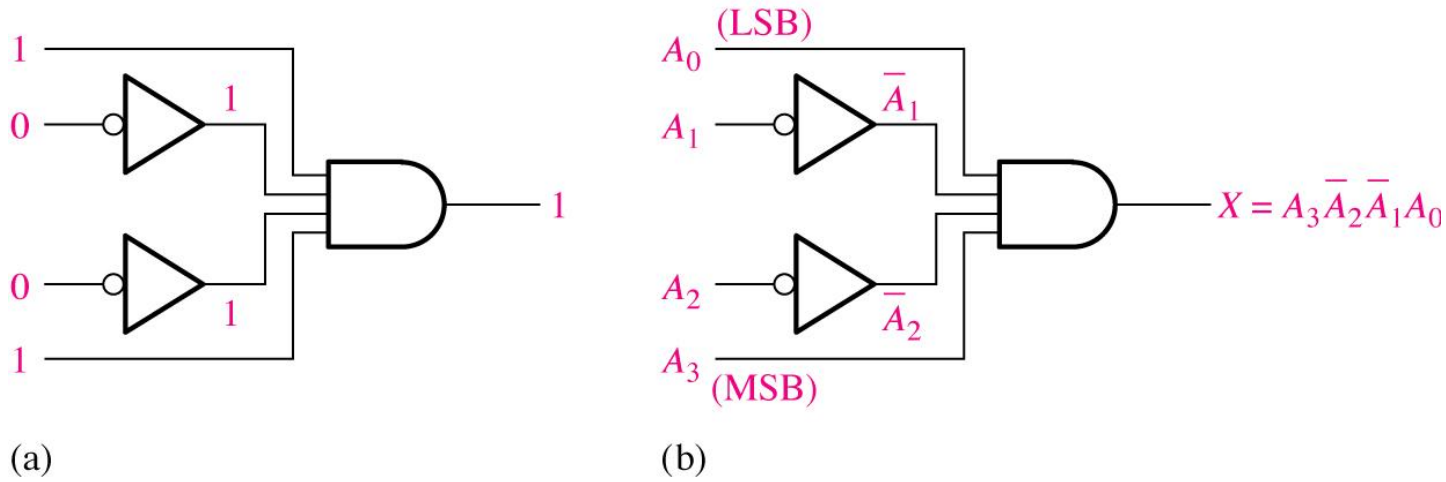


디코더(Decoders)

* 디코더 :

기본적인 2진 디코더

* 입력이 1001 일 때, HIGH를 출력하는 디코더의 예
* 입력이 1001 일 때, LOW를 출력하려면, AND 게이트 대신 NAND 게이트 사용.

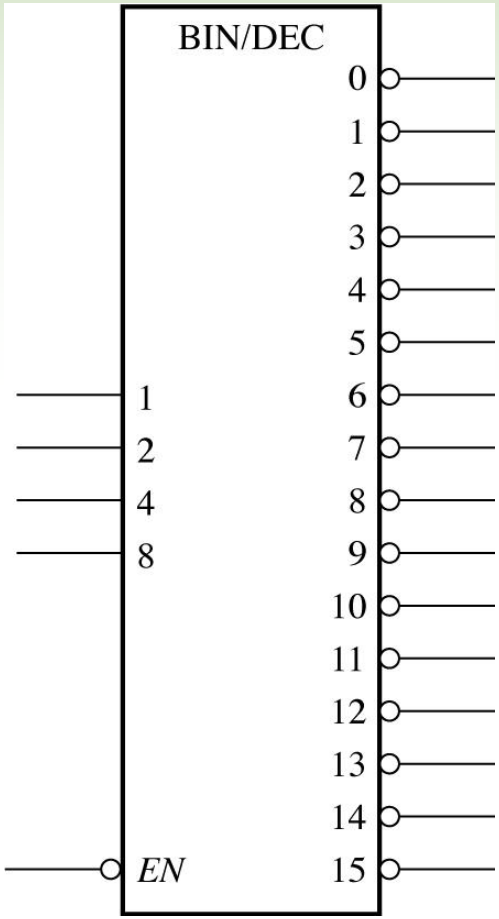


디코더(Decoders)

4비트 디코더

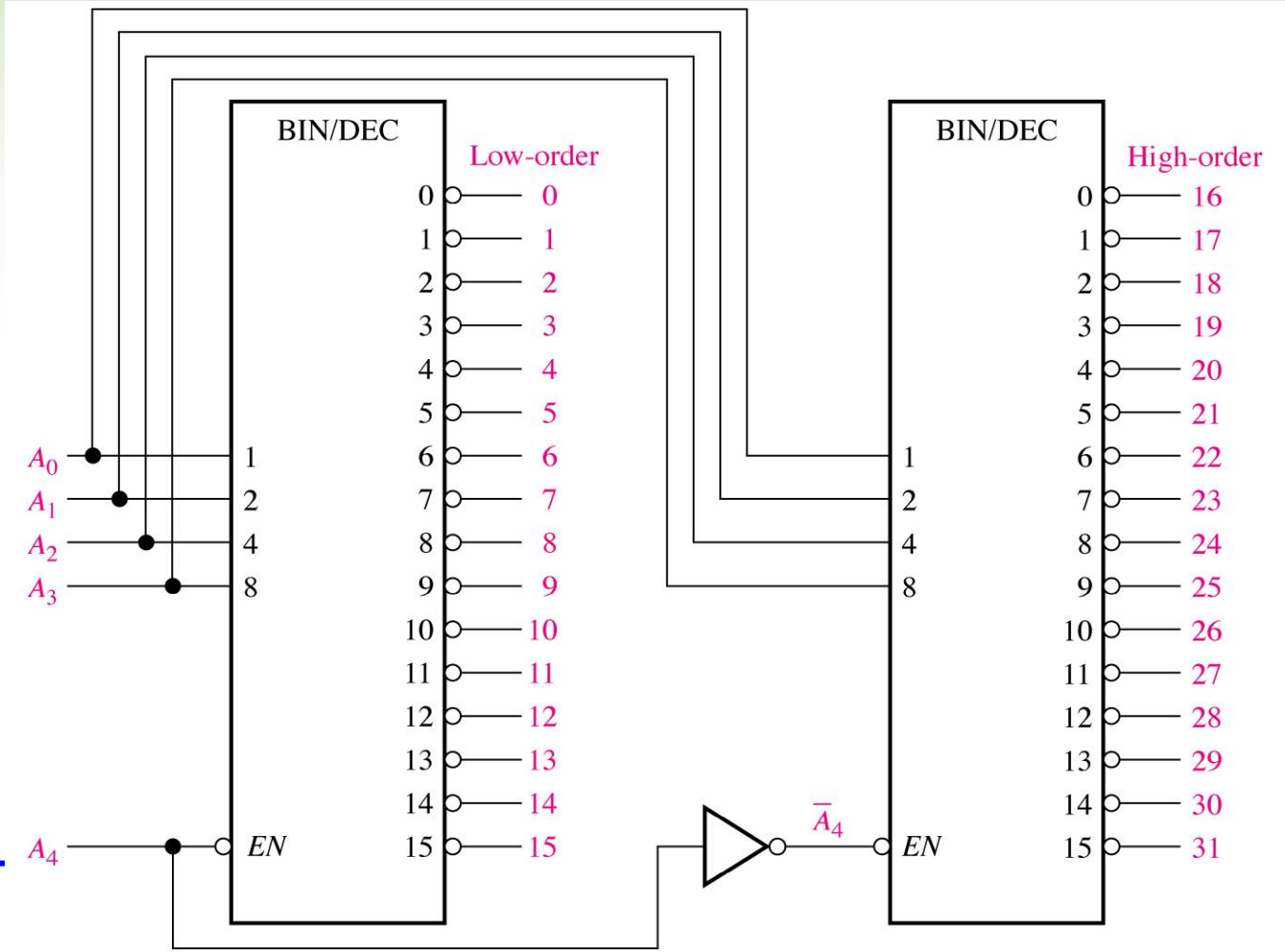
* 4비트의 모든 가능한 조합 검출
  => 16개의 디코딩 게이트 필요.

- 4-line-to-16-line decoder ;
  4개의 입력과 16개의 출력
- 1-of-16 decoder ;
  1개의 입력 코드에 대해
  16개 출력 중 하나가 active.



디코더(Decoders)

* 예제 6-9) 2개의 4-line-to-16-line decoder 를 사용하여, 5비트 수를 디코딩.



디코더

- 응용 예제

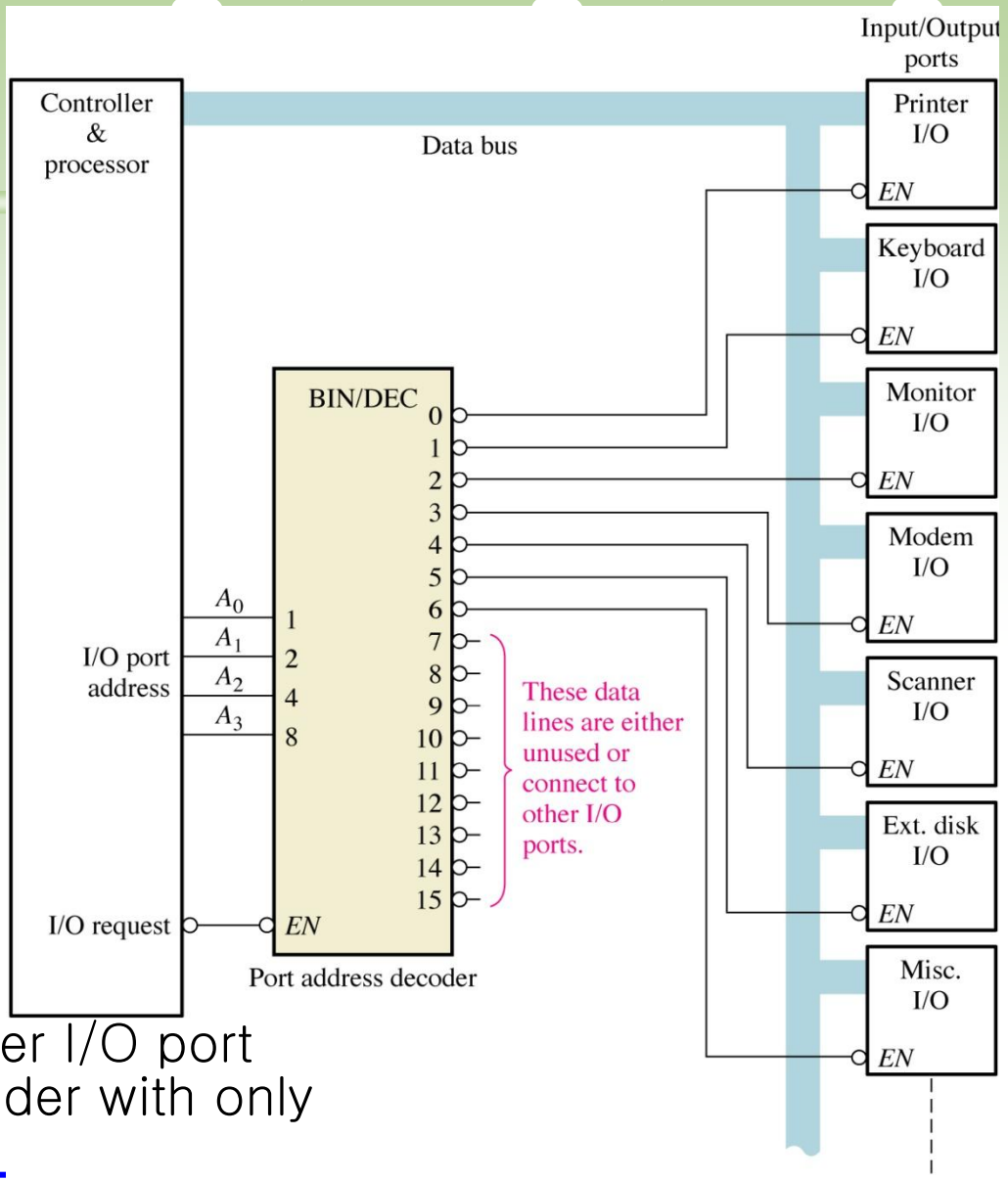
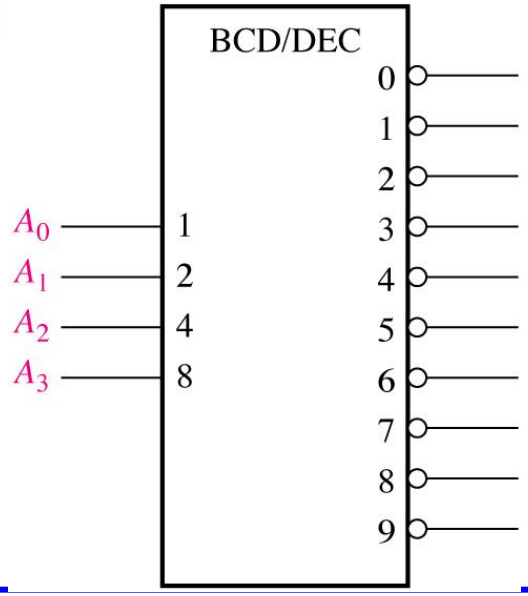


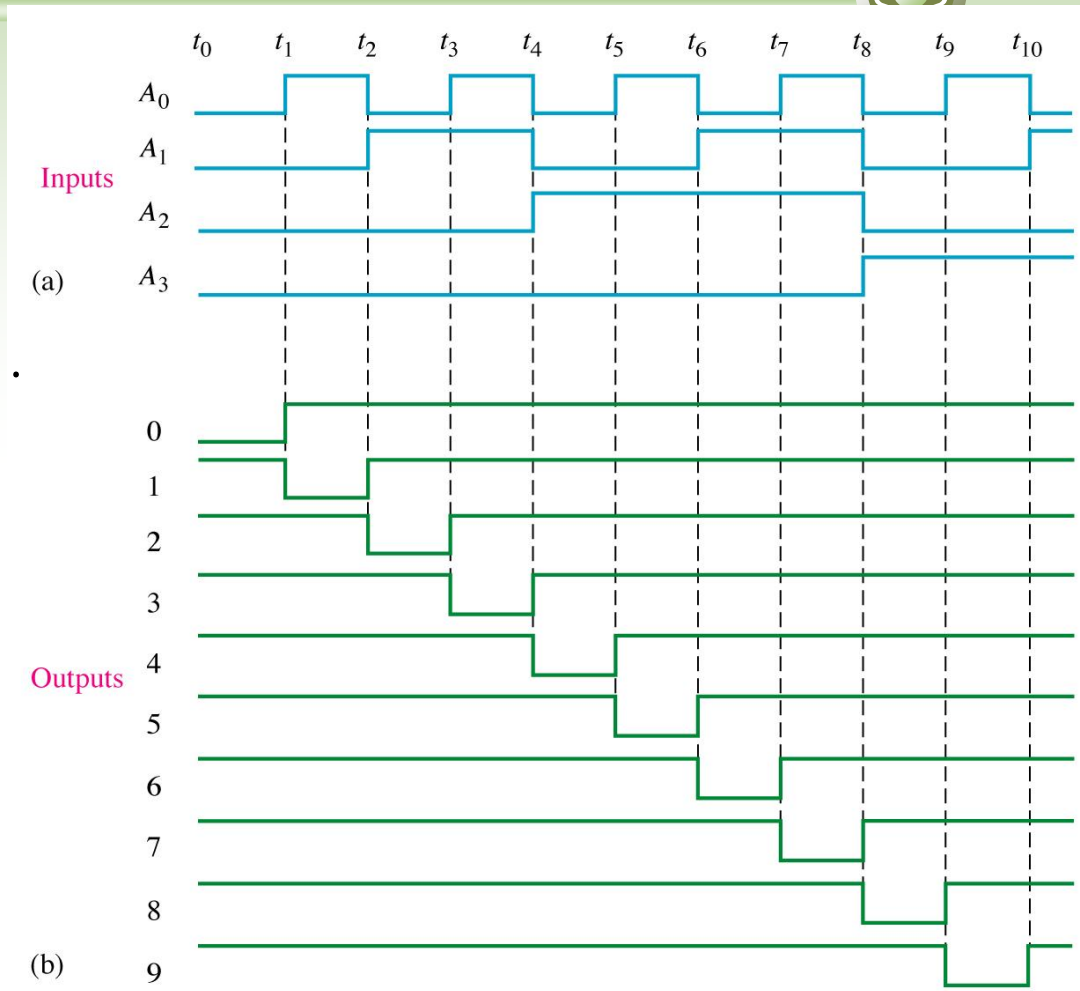
Figure 6-24 A simplified computer I/O port system with a port address decoder with only four address lines shown.

디코더(Decoders)

- BCD-to-10진 디코더

* 1-of-16 디코더와 다른점은 0-9의 10개 디코딩만 필요한 점.

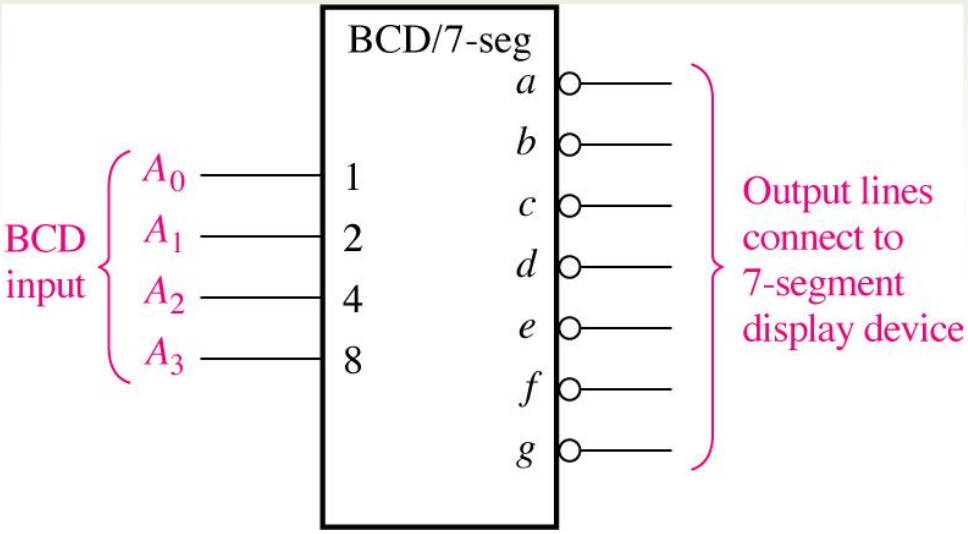




디코더(Decoders)

BCD-to-7-세그먼트 디코더

- 입력 : BCD 코드
- 출력 : 7-세그먼트 구동 출력

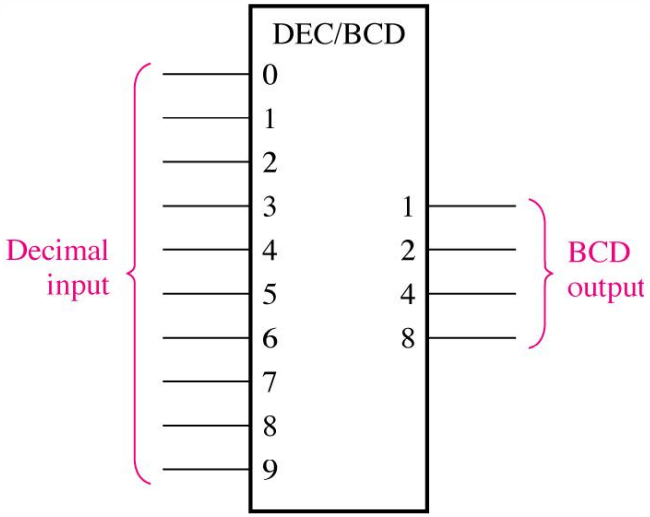


* 부가 기능 ;

- 램프 테스트 : 모든 세그먼트 켜기
- 제로 억제 : 불필요한 0 끄기
  030.080 => 30.08

인코더(Encoders)

- 10진-to-BCD 인코더



BCD 코드

10진	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

인코더(Encoders)

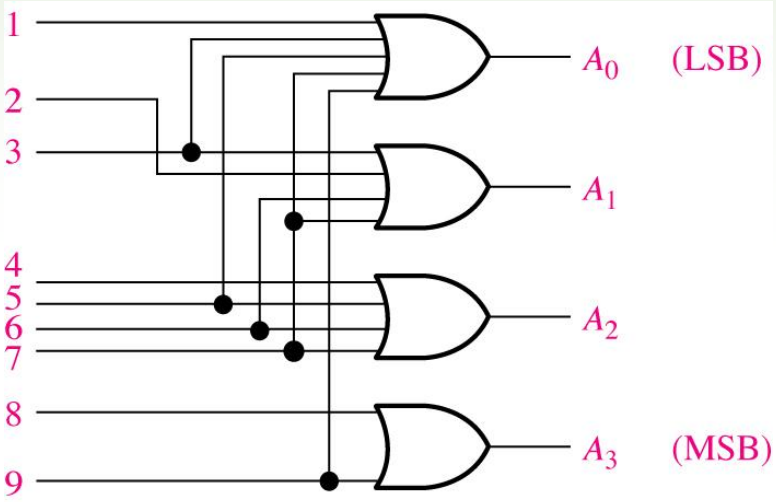
* 10진-to-BCD 인코더의 기본적인 논리도

$A_3 = 8+9$

$A_2 = 4+5+6+7$

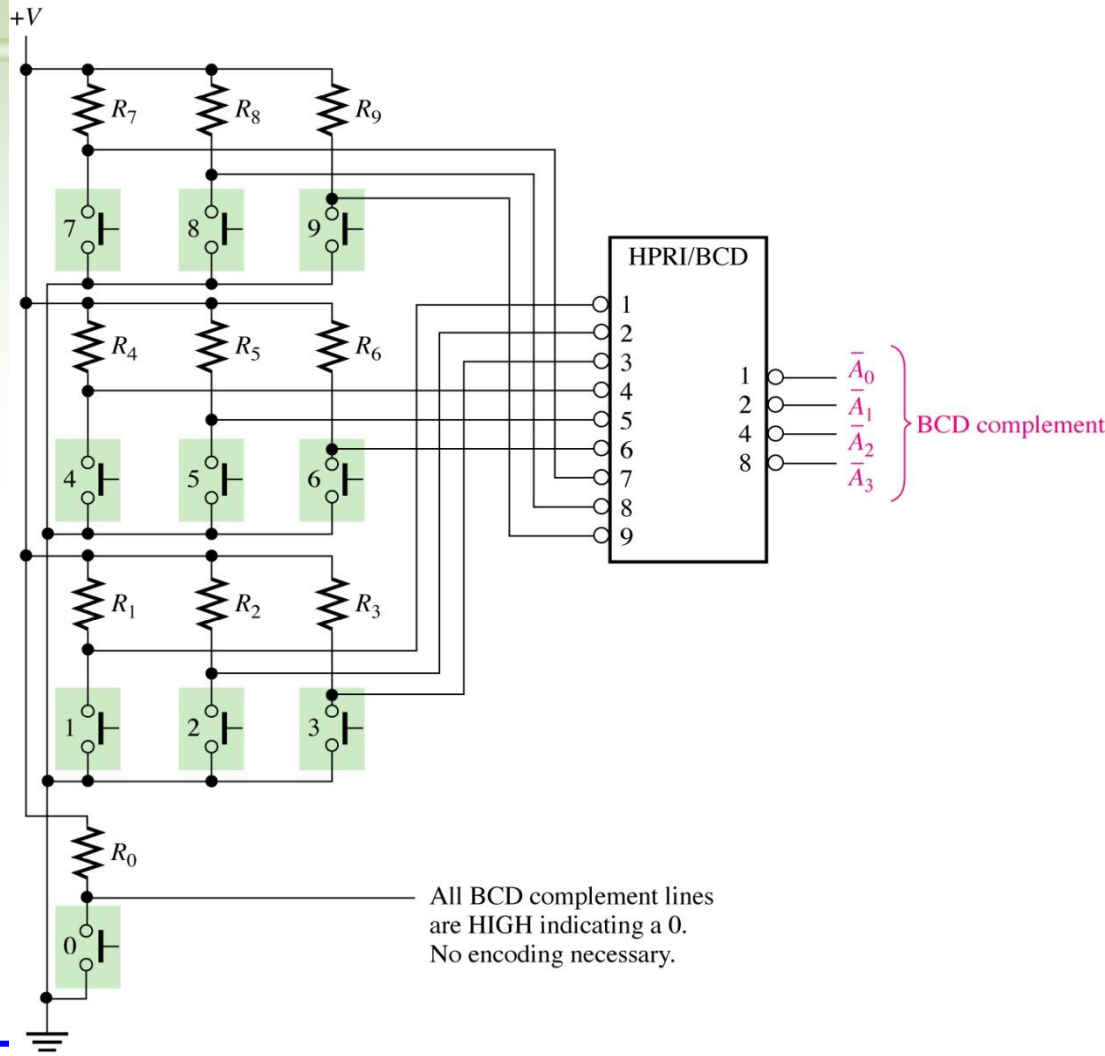
$A_1 = 2+3+6+7$

$A_0 = 1+3+5+7+9$



인코더(Encoders)

* 응용 예제 : 키보드 인코더

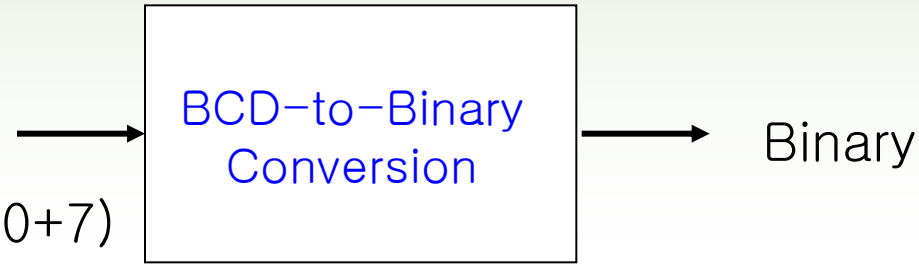


코드 변환기(Code Converters)

- BCD-to-2진 변환

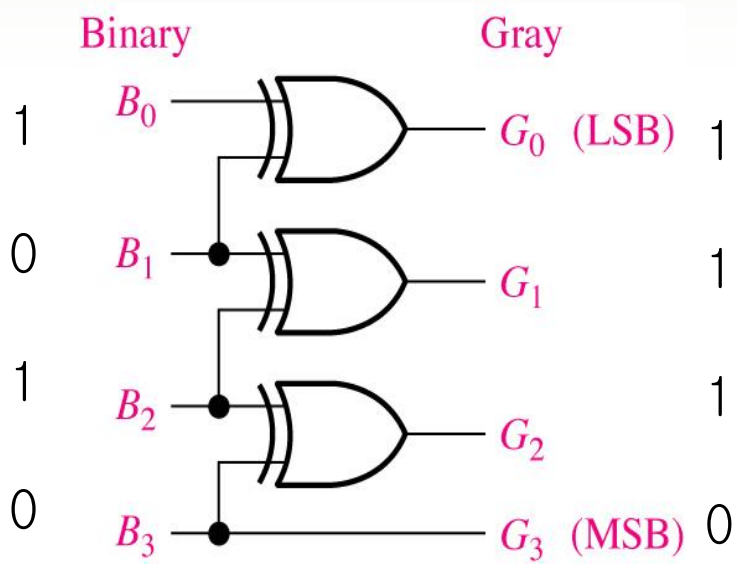
BCD ;

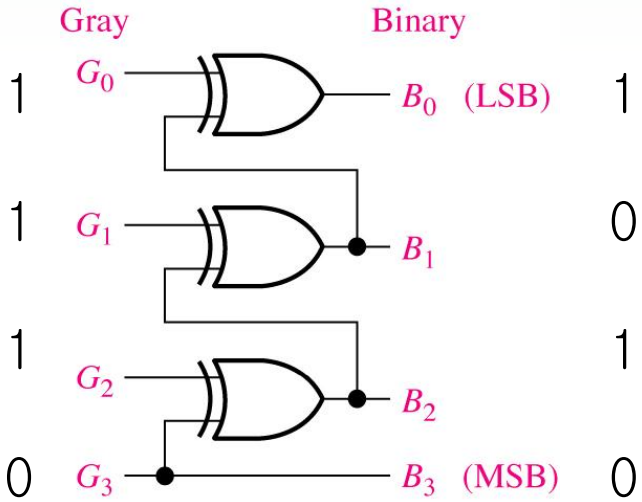
0 0 1 0 0 1 1 1 (27=20+7)



코드 변환기(Code Converters)

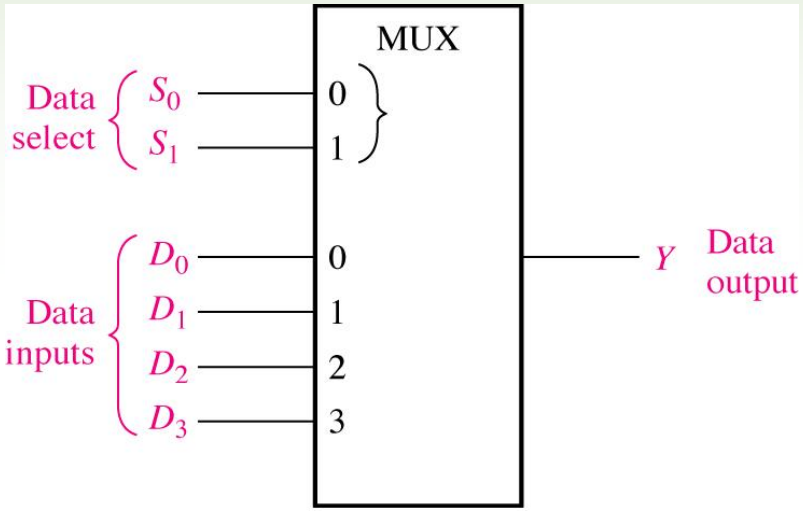
- 2진-to-그레이
- 그레이-to-2진





멀티플렉서(데이터 선택기)

- 멀티플렉서(Multiplexer:MUX)



Data Select Inputs S_1	S_0	Input Selected Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

* 2개의 선택 비트로 4개의 데이터-입력 라인 중 하나 선택

멀티플렉서(데이터 선택기)

* $Y = D_0(S_1)'(S_0)' + D_1(S_1)'S_0 + D_2S_1(S_0)' + D_3S_1S_0$

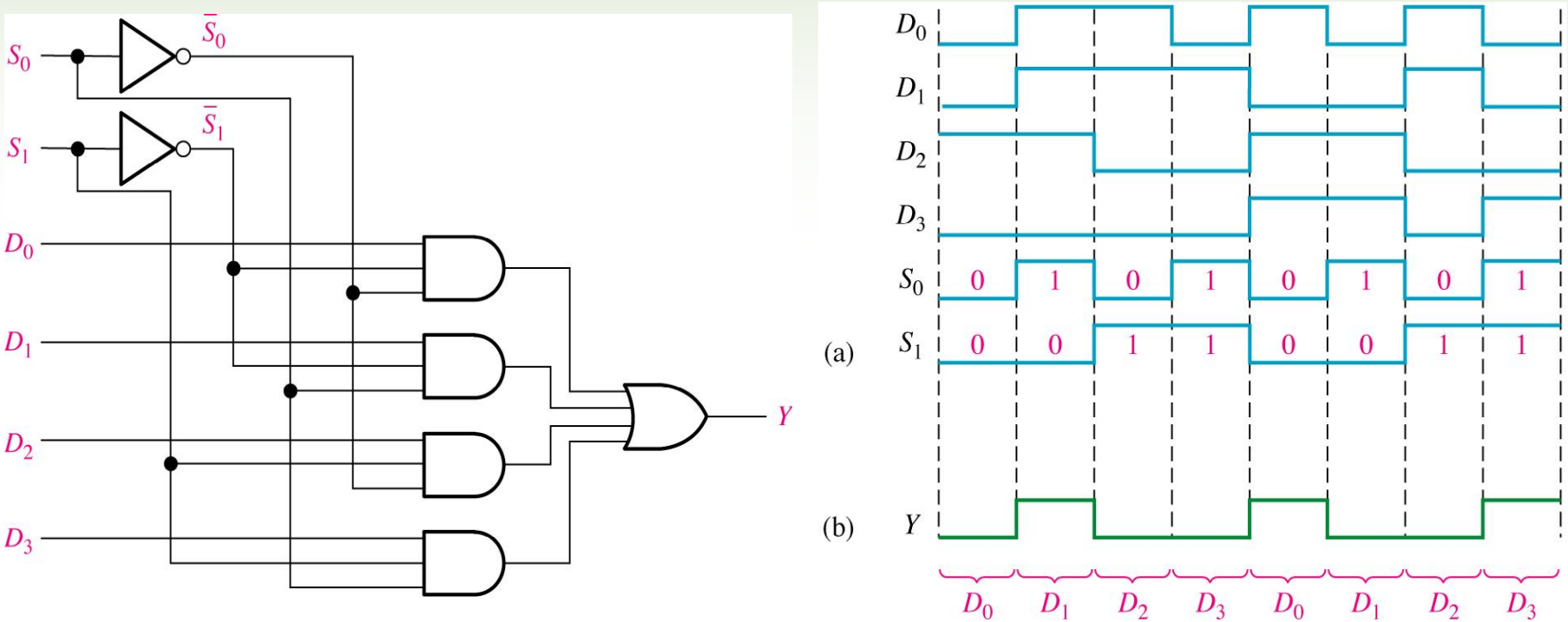
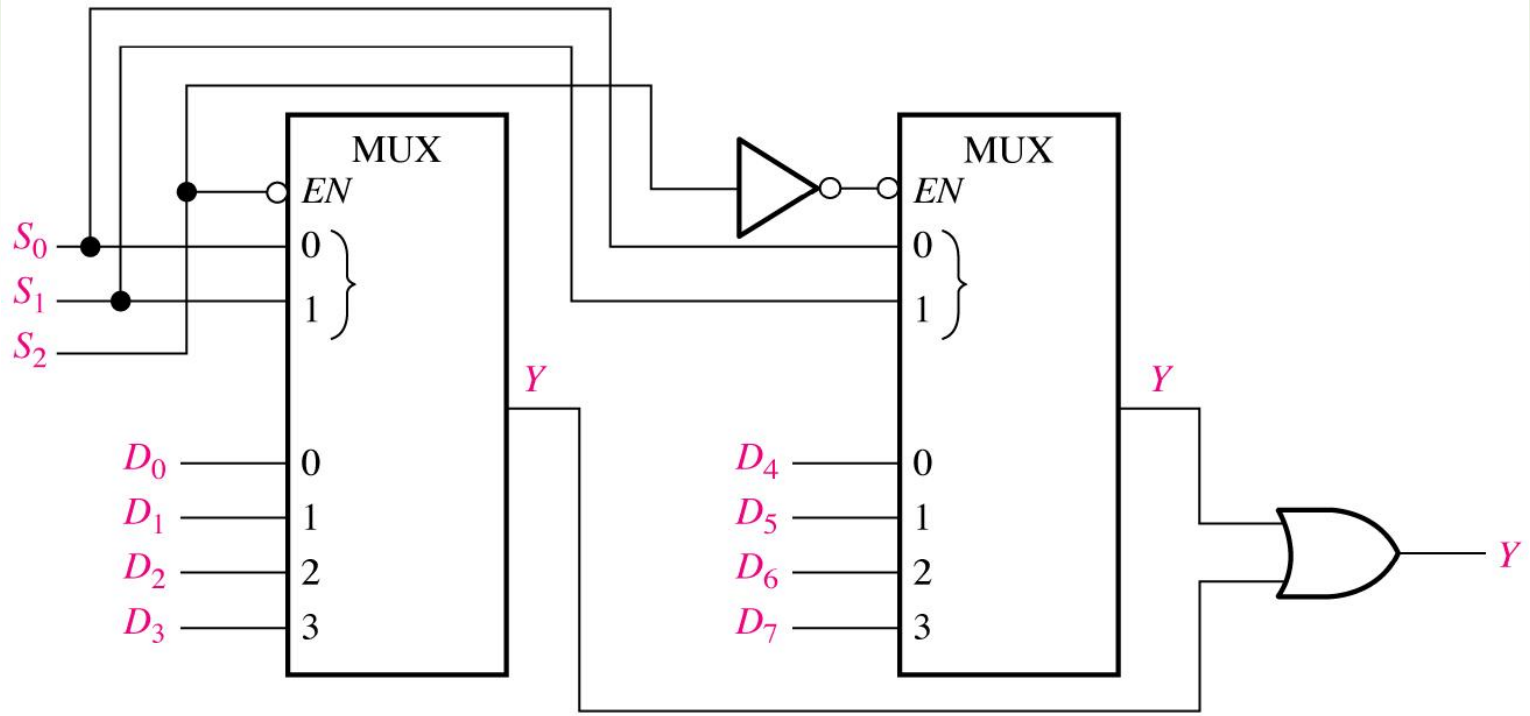


Figure 6-35 Logic diagram and timing diagram for a 4-input multiplexer.

멀티플렉서(데이터 선택기)

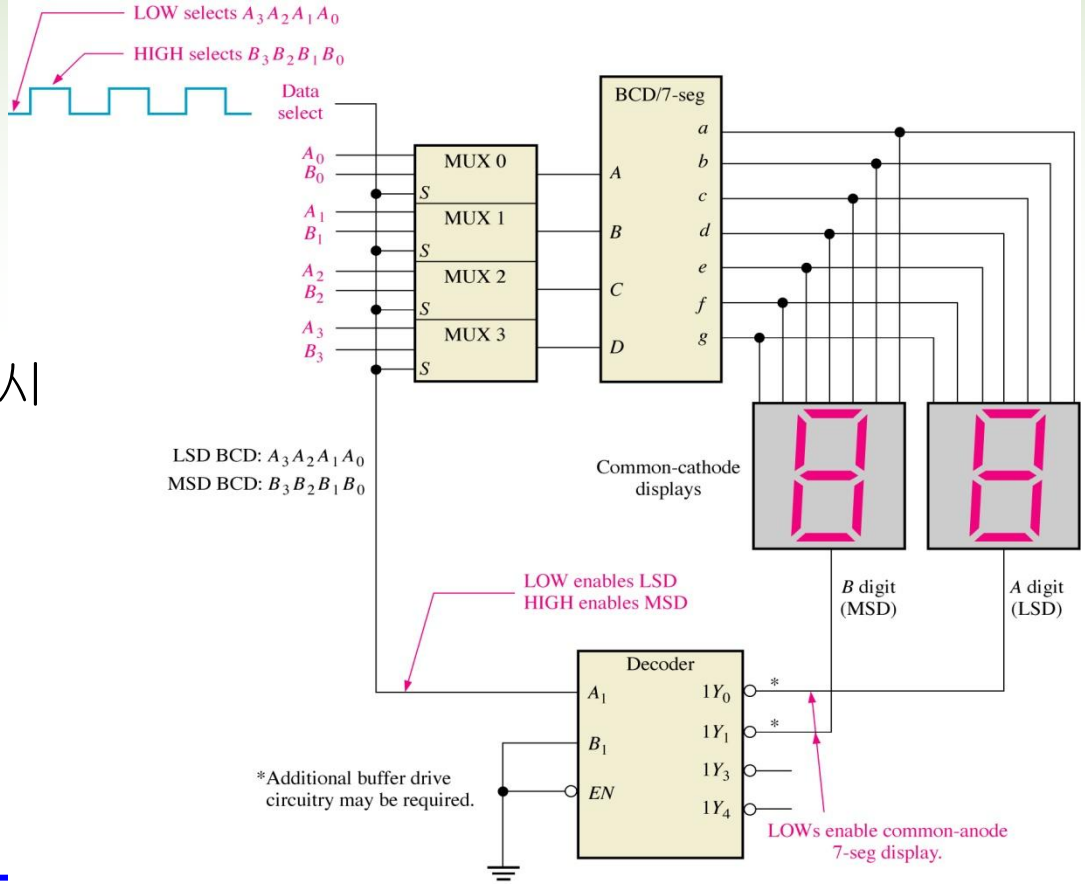
* 예제 6-15) 2개의 1-of-4 MUX를 사용하여 1-of-8 MUX 구성하라.



멀티플렉서(데이터 선택기)

* 응용 예제 : 7-세그먼트 디스플레이 멀티플렉서

- ⇒ 2자리 숫자 표시
- ⇒ 두 개의 BCD 숫자 : A, B
- ⇒ 4개의 1-of-2 MUX
- ⇒ S=0이면 A, Y_0선택 ->A 표시
- ⇒ S=1이면 B , Y_1 선택 ->B 표시

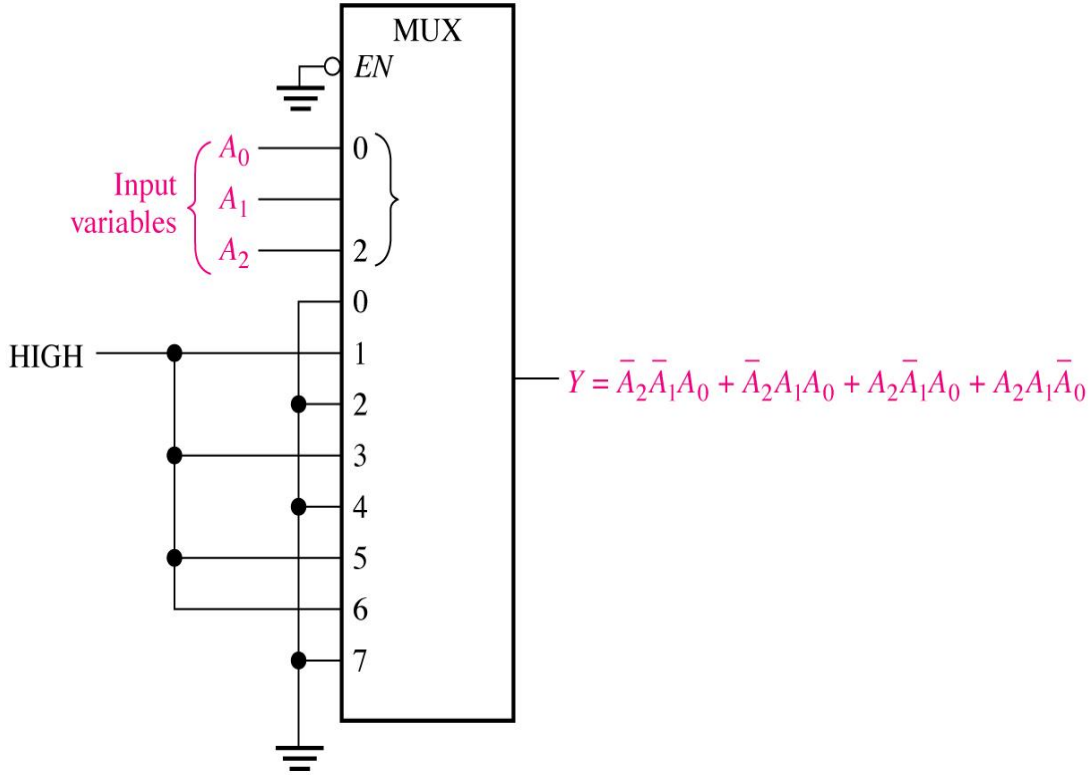


멀티플렉서(데이터 선택기)

* **응용 예제 : 논리 함수 발생기** : 곱의 합 형태의 조합 논리 함수 발생 가능.

* 예제 6-16) 8 입력 MUX를 사용하여 다음 논리 함수를 구성

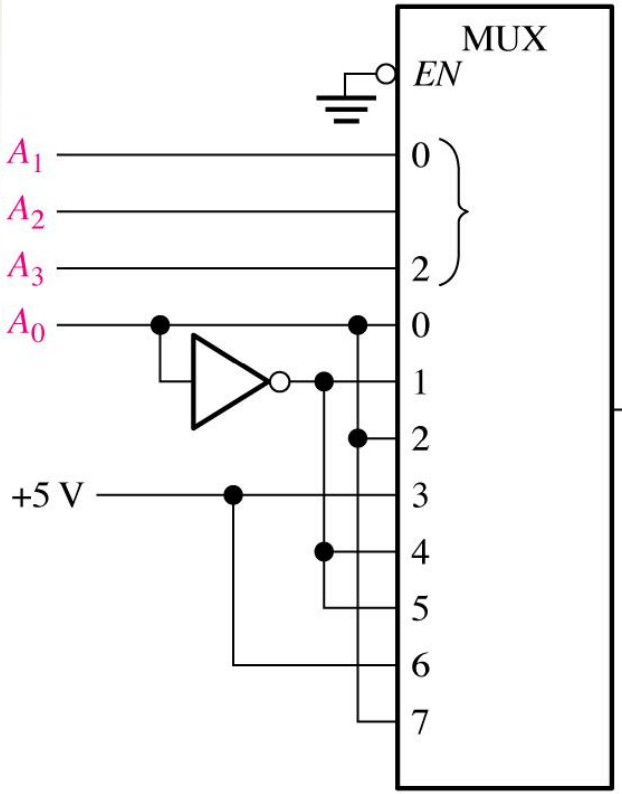
Inputs A_2 A_1 A_0	Output Y
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	0



멀티플렉서(데이터 선택기)

* 예제 6-17) 8 입력 MUX를 사용하여 다음 논리 함수를 구성

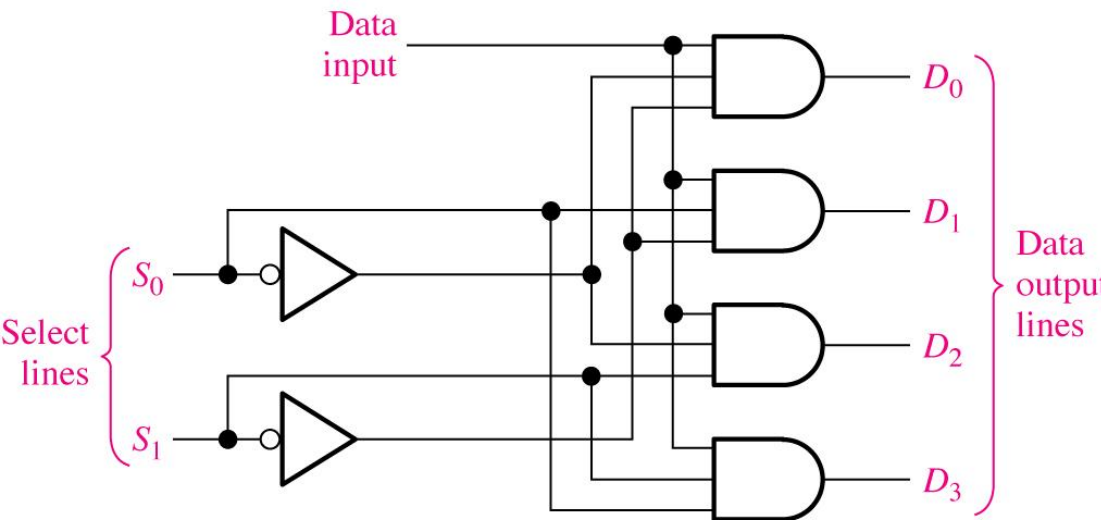
Inputs $A_3A_2A_1A_0$	Output Y
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 0 1 1	0
0 1 0 0	0
0 1 0 1	1
0 1 1 0	1
0 1 1 1	1
1 0 0 0	1
1 0 0 1	0
1 0 1 0	1
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	0
1 1 1 1	1

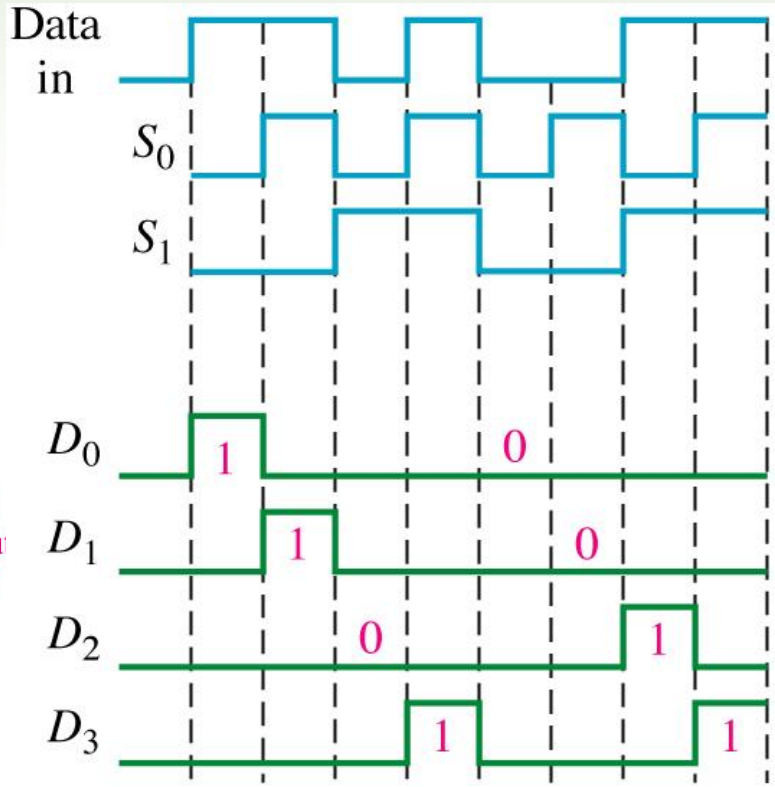


$$Y = \bar{A}_3\bar{A}_2\bar{A}_1A_0 + \bar{A}_3\bar{A}_2A_1\bar{A}_0 + \bar{A}_3A_2\bar{A}_1A_0 + \bar{A}_3A_2A_1\bar{A}_0 + \bar{A}_3A_2A_1A_0 + A_3\bar{A}_2\bar{A}_1\bar{A}_0 + A_3\bar{A}_2A_1\bar{A}_0 + A_3A_2\bar{A}_1\bar{A}_0 + A_3A_2\bar{A}_1A_0 + A_3A_2A_1A_0$$

디멀티플렉서(데이터 분배기)

* 한 개의 선으로 들어오는 정보를 정해진 수의 출력 라인으로 분배

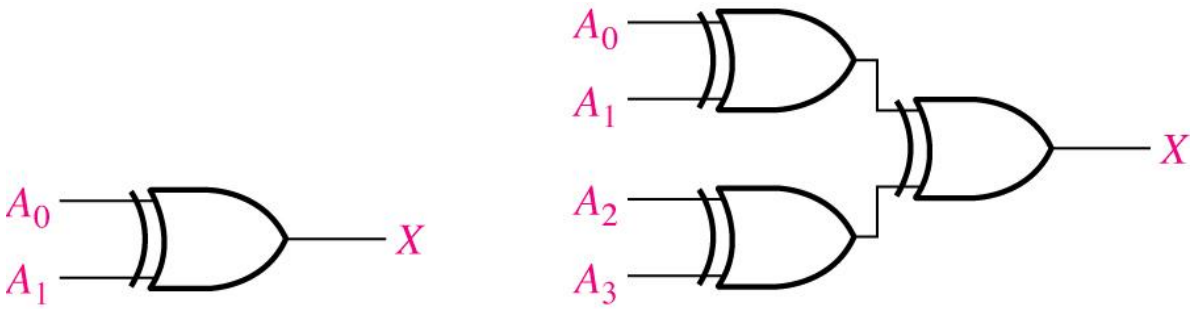




패리티 생성기/검출기

* 패리티 방법 : 데이터 전송시 전송 오류를 검출하기 위한 방법

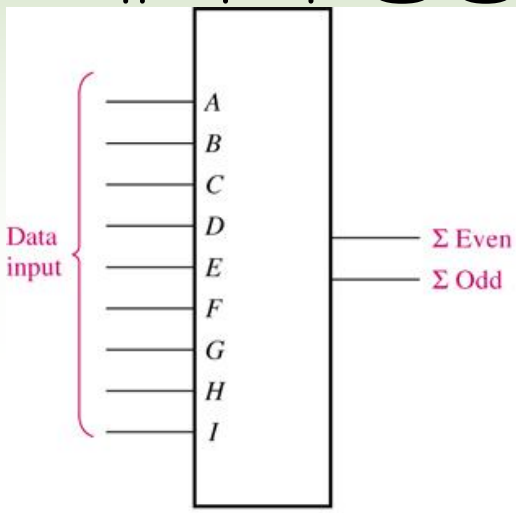
* **기본적인 패리티 논리**
  - ⇒ 짝수 개의 1의 합은 0이고, 홀수 개의 1의 합은 1
  - ⇒ 짝수 패리티인지 홀수 패리티인지는 모든 비트를 더하면 알 수 있다.



(a) Summing of two bits (b) Summing of four bits

패리티 생성기/검출기

- 9비트 패리티 생성기/검출기



(a) Traditional logic symbol

Number of Inputs A–I That Are HIGH	Outputs Σ Even	Σ Odd
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

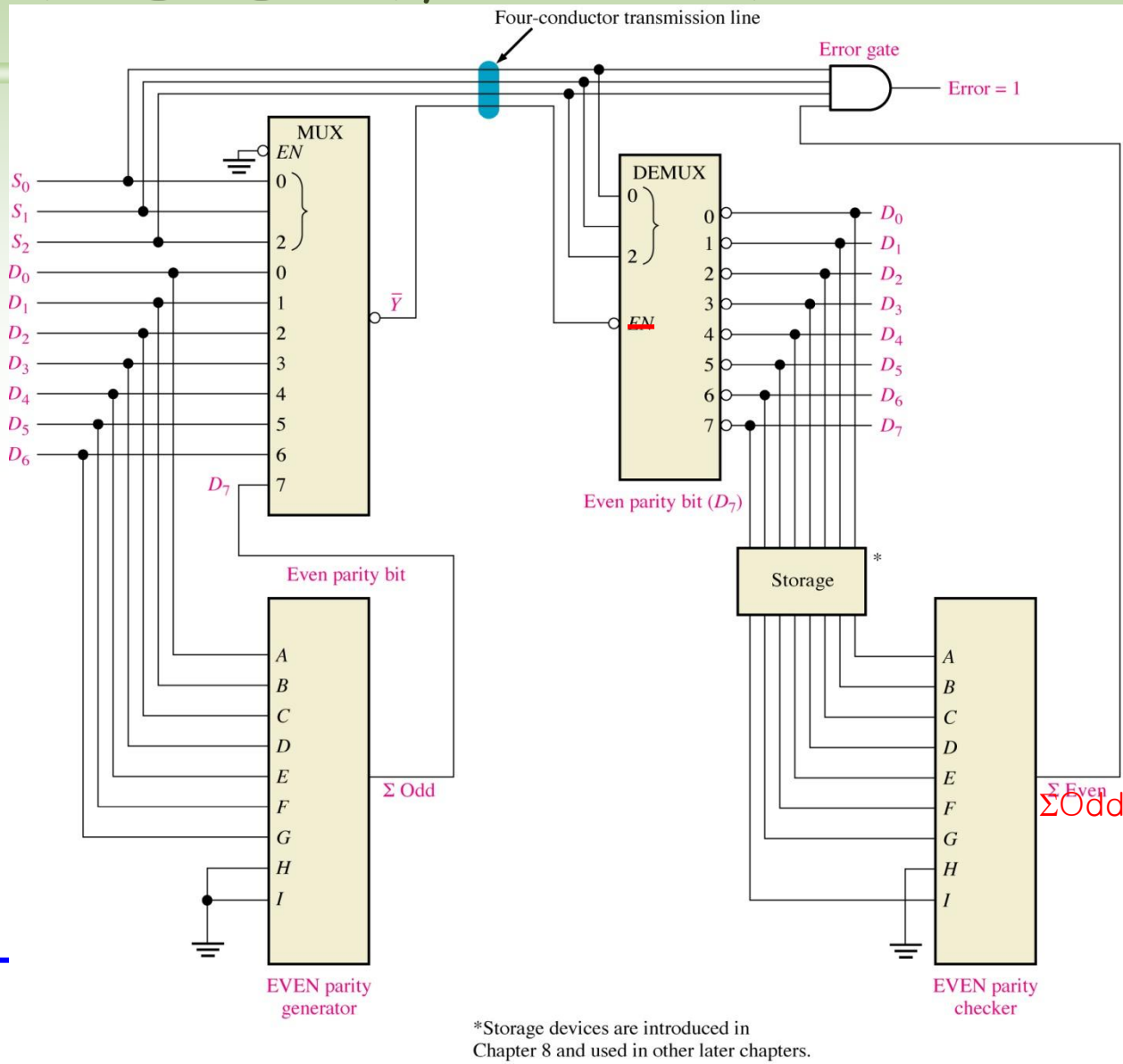
(b) Function table

* 짝수 패리티 검출기 : 1의 개수가 짝수이면 ΣEven 은 H -> 정상
  1의 개수가 홀수이면 ΣEven 은 L -> 에러
* 짝수 패리티 생성기 : 1의 개수가 짝수이면 ΣOdd 은 L
  1의 개수가 홀수이면 ΣOdd 은 H

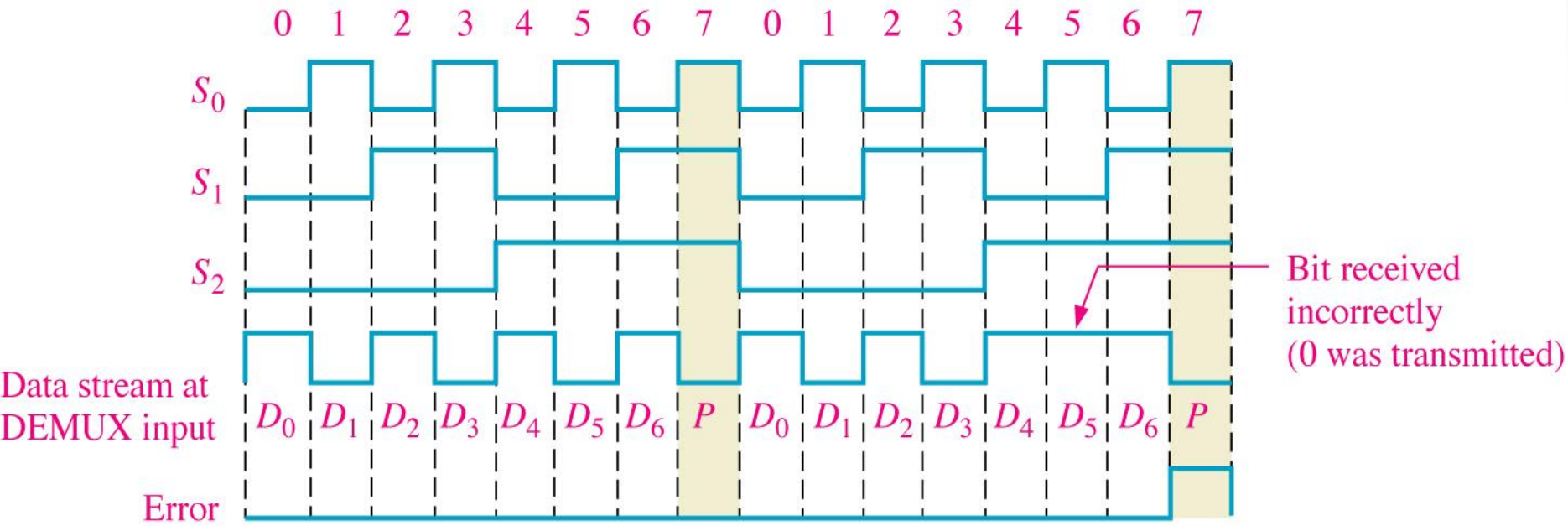
⇨ Σodd 출력에서 패리티 비트 발생

패리티 생성기/검출기

- 에러 검출이 가능한 데이터 전송 시스템



패리티 생성기/검출기


0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
S_0
S_1
S_2
Data stream at DEMUX input
D_0 D_1 D_2 D_3 D_4 D_5 D_6 P D_0 D_1 D_2 D_3 D_4 D_5 D_6 P
Error
Bit received incorrectly (0 was transmitted)