

디지털논리회로 (Digital Logic Circuit)

- Chapter 5

Chapter 5 Designing Combinational Systems

5.1 Iterative (반복) 시스템 ex) ALUs

5.2 Binary Decoders

5.3 Encoders and Priority Encoders

5.4 Multiplexer (Mux) & Demultiplexers (Demux)

5.5 Tree-State Gates

5.6 Gate Arrays – ROM, PLA, PALs..

Chapter 5 Designing Combinational Systems

5.1 Iterative (반복) 시스템 ex) ALUs

- Adder 설계
- Subtractor/Adder 설계
- **Comparator(비교기) 설계**

5.2 Binary Decoders

5.3 Encoders and Priority Encoders

5.4 Multiplexer (Mux) & Demultiplexers (Demux)

5.5 Tree-State Gates

5.6 Gate Arrays – ROM, PLA, PALs..

같은 값인가? 아닌가?

비교기 - '같다' '다르다'

XOR LOGIC

XOR Symbol



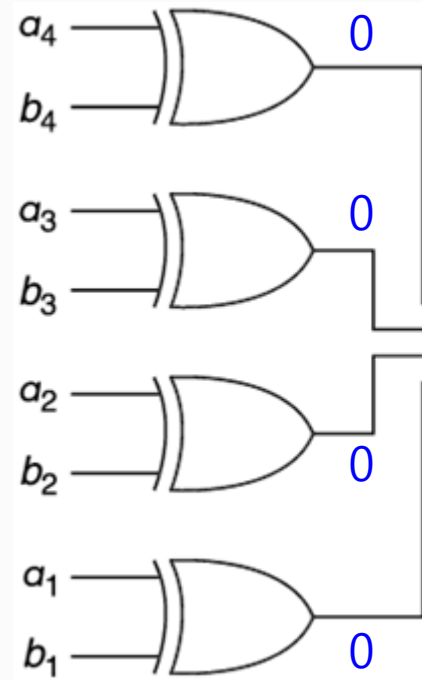
0 XOR 0 = 0	Same Bits
1 XOR 1 = 0	Same Bits
1 XOR 0 = 1	Different Bits
0 XOR 1 = 1	Different Bits

a: 1100

b: 1100



1 bit 비교



4 bit 비교

같은가?
YES
=1

비교기 - '같다' '다르다'

XOR LOGIC

XOR Symbol



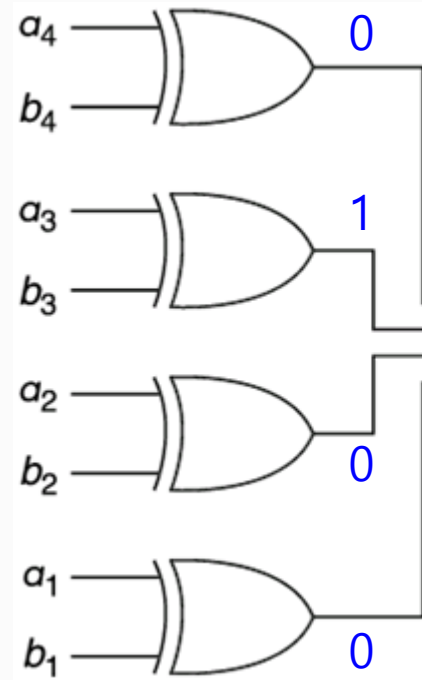
0 XOR 0 = 0	Same Bits
1 XOR 1 = 0	Same Bits
1 XOR 0 = 1	Different Bits
0 XOR 1 = 1	Different Bits

a: 1100

b: 1000



1 bit 비교



4 bit 비교

같은가?
NO
=0

비교기 - '같다' '다르다'

XOR LOGIC

XOR Symbol

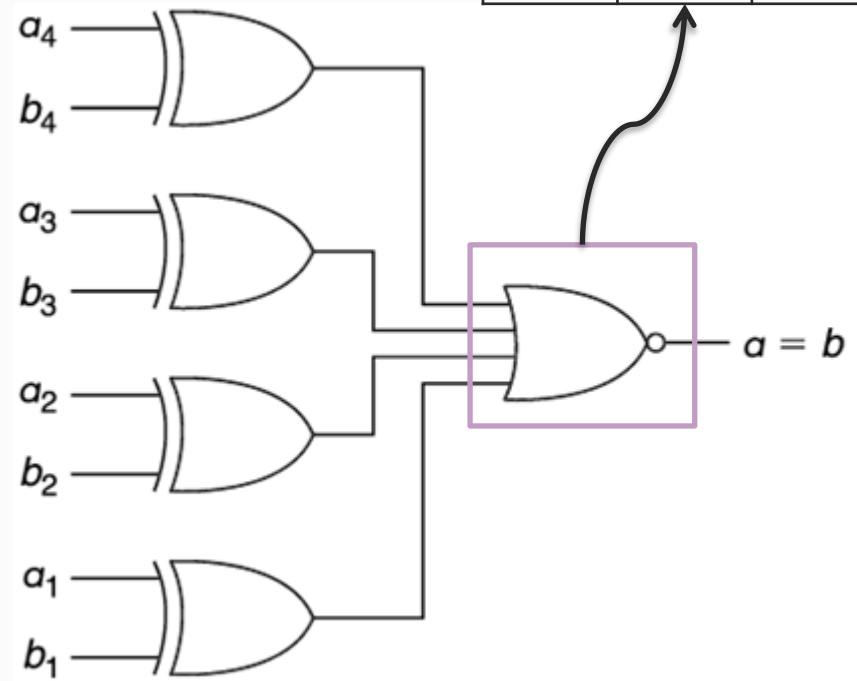


0 XOR 0 = 0 Same Bits
1 XOR 1 = 0 Same Bits
1 XOR 0 = 1 Different Bits
0 XOR 1 = 1 Different Bits



1 bit 비교

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

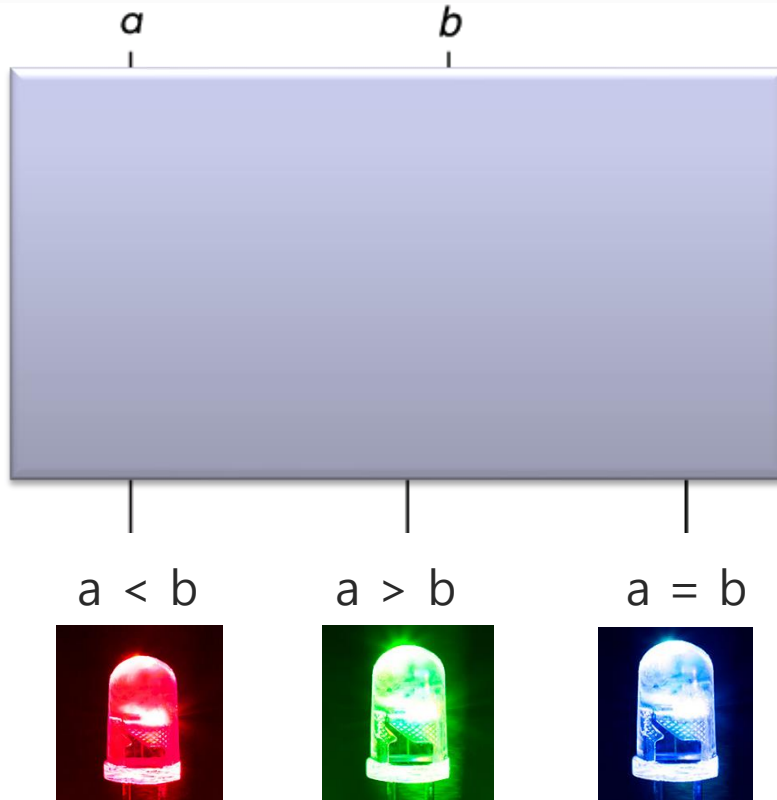


4 bit 비교

같은 값인가?

큰가? 작은가?

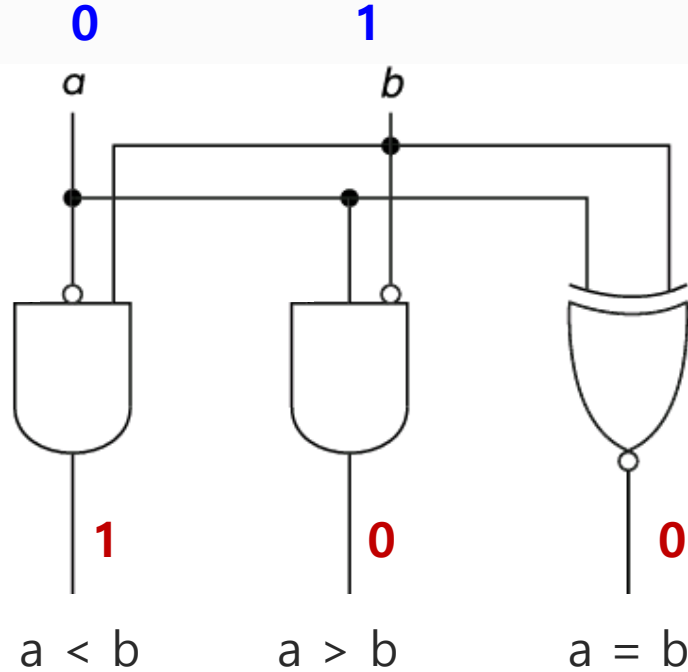
비교기 - '크다' '같다' '작다'



2개 입력,
3개 출력에 대한
진리표 그려서 구현

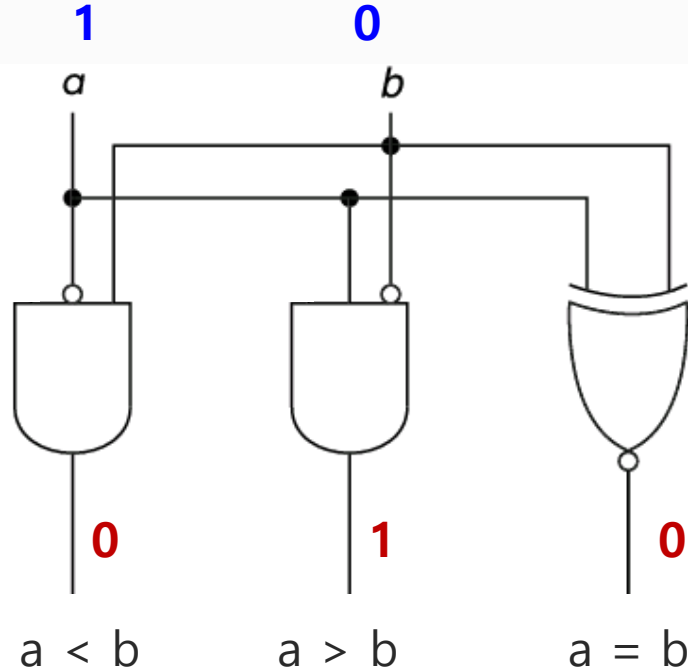
1 bit 비교

비교기 - '크다' '같다' '작다'



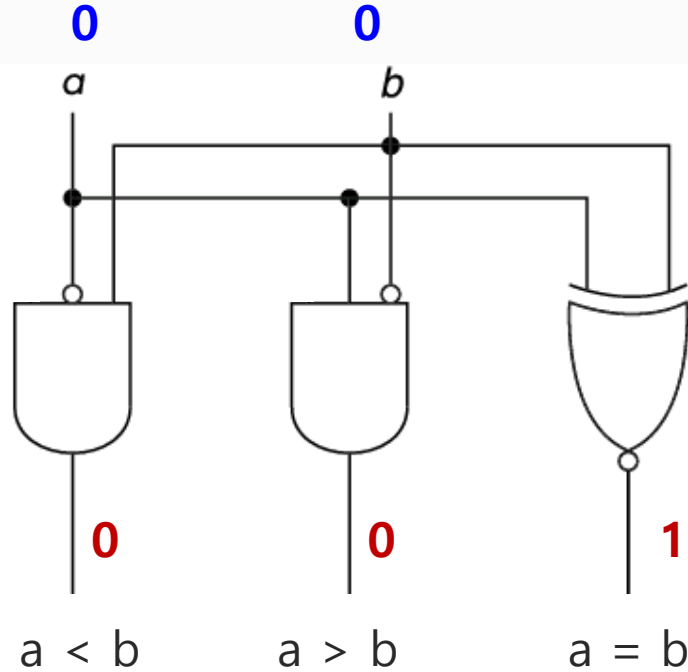
1 bit 비교

비교기 - '크다' '같다' '작다'



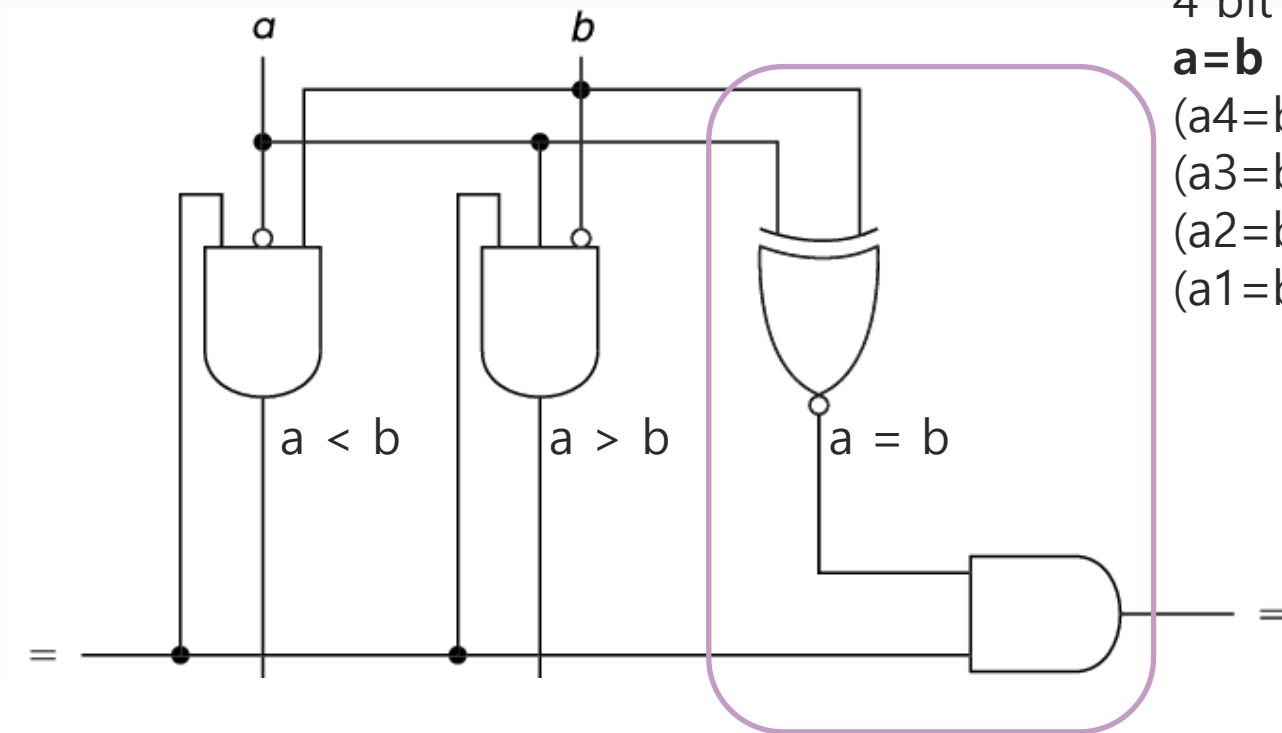
1 bit 비교

비교기 - '크다' '같다' '작다'

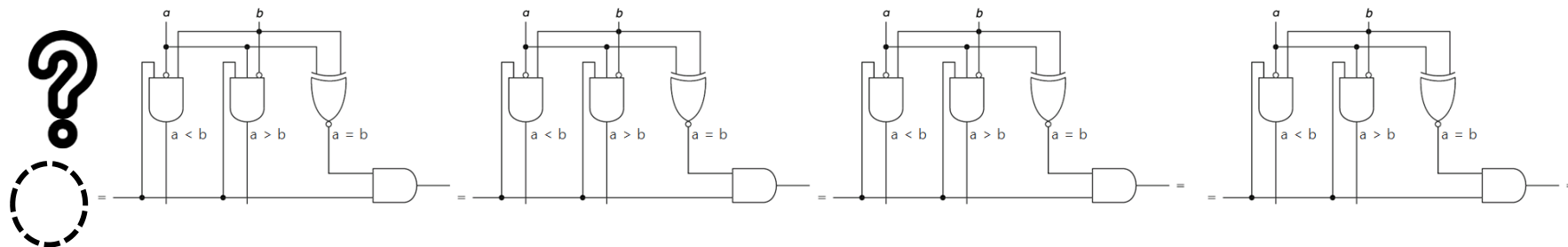


1 bit 비교

비교기 - '크다' '같다' '작다'

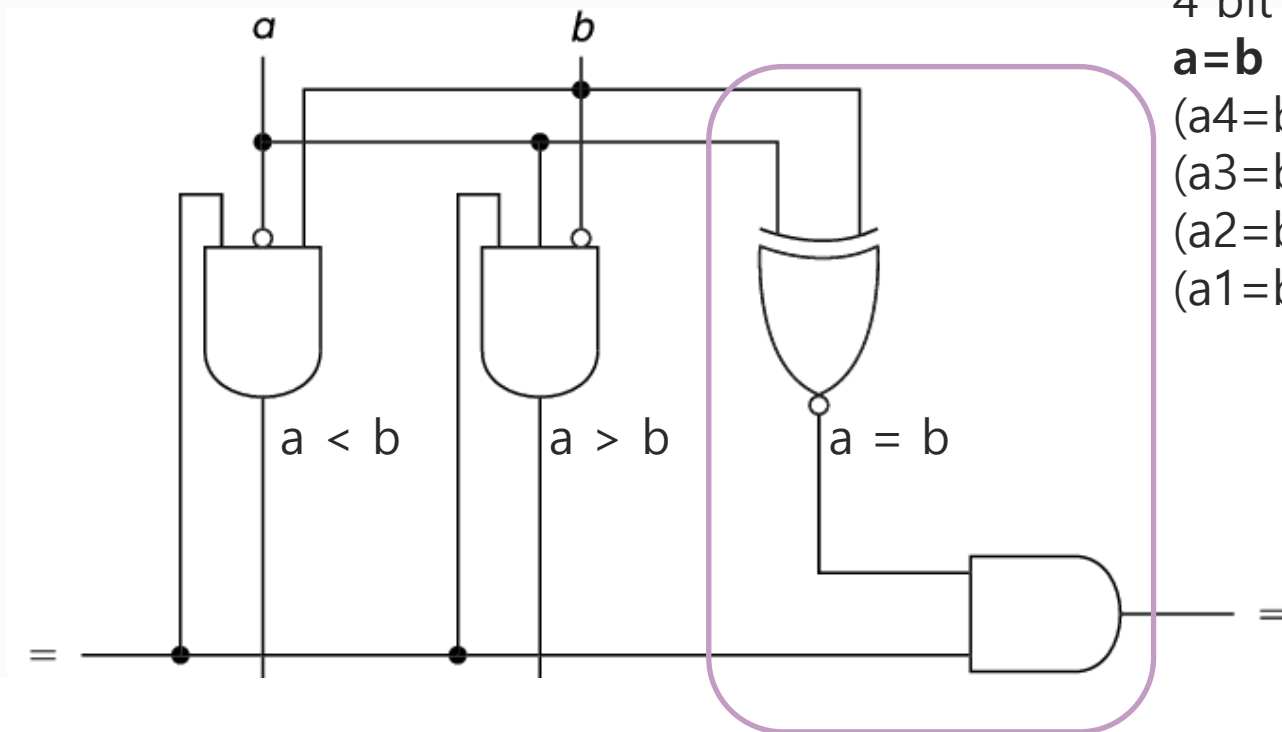


상위비트부터
비교해 나가는
구현

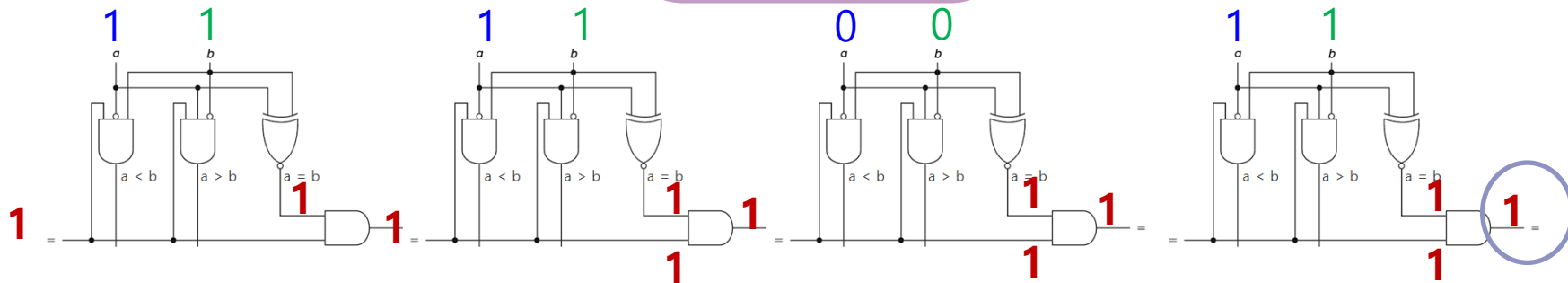


multi bit 비교 (직렬로 연결)

비교기 - '크다' '같다' '작다'

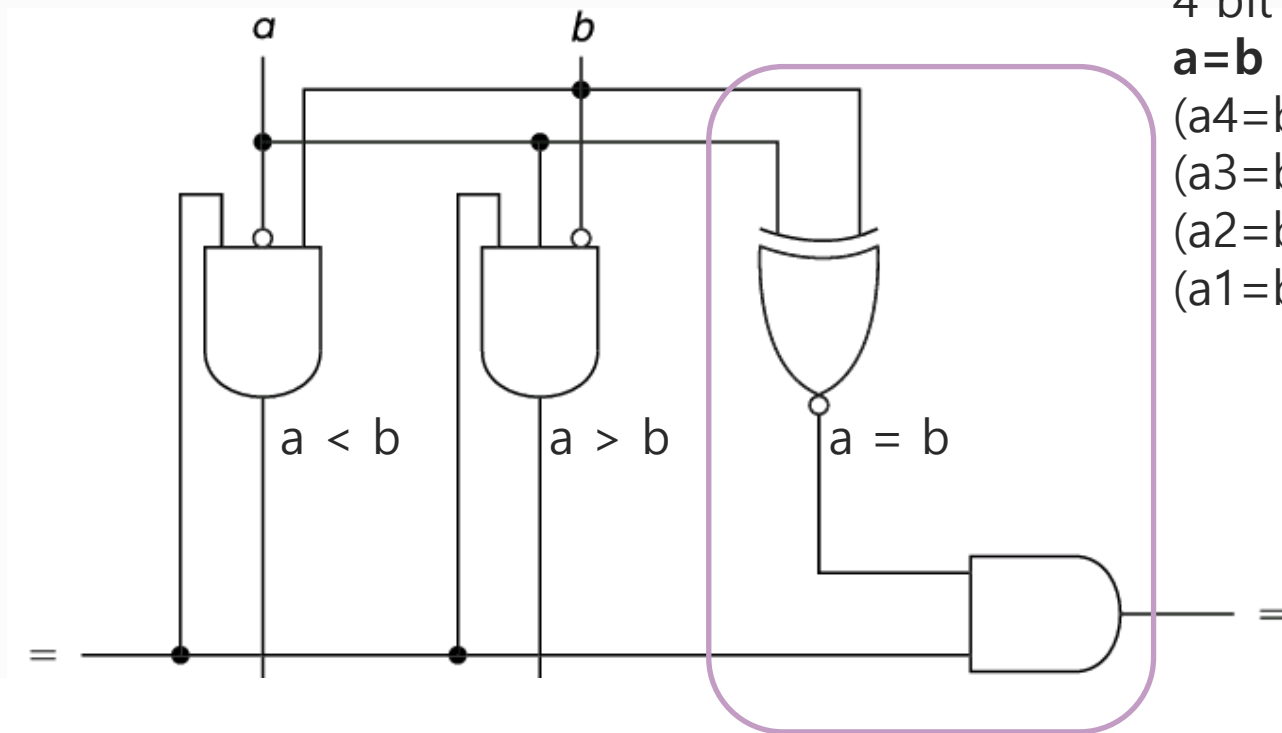


a: 1101
b: 1101



multi bit 비교 (직렬로 연결)

비교기 - '크다' '같다' '작다'

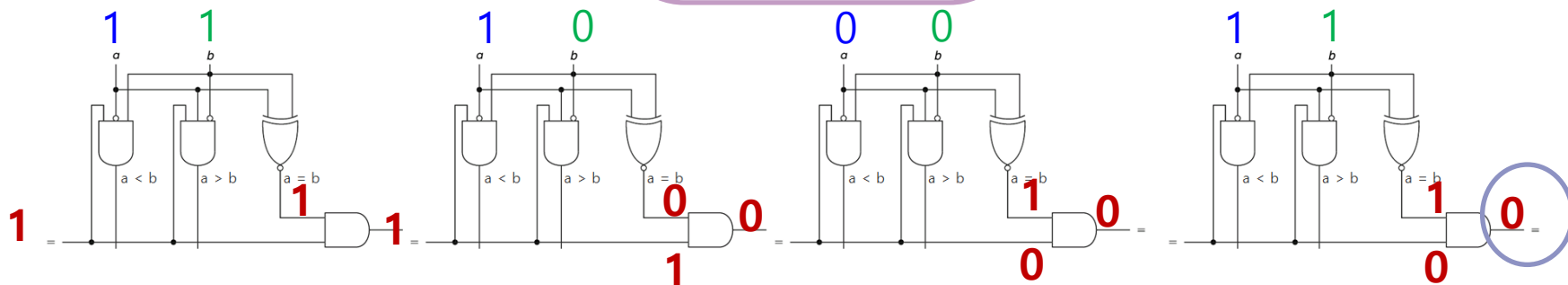


4 bit 비교

$a = b$

($a_4 = b_4$) and
($a_3 = b_3$) and
($a_2 = b_2$) and
($a_1 = b_1$)

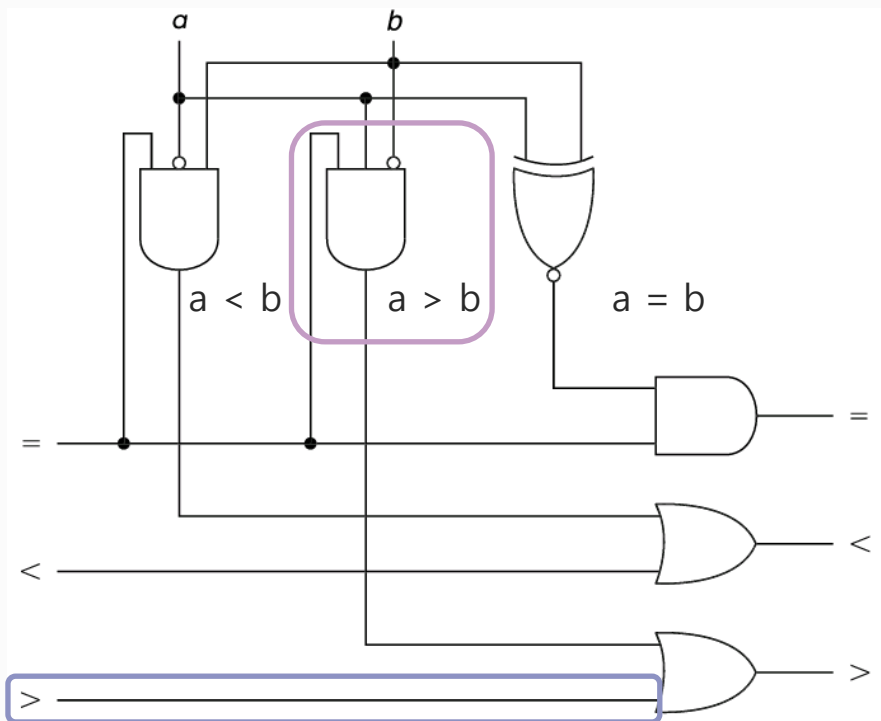
a: 1101
b: 1001



multi bit 비교 (직렬로 연결)

비교기 - '크다' '같다' '작다'

역시 상위비트부터
비교해 나가는
구현



multi bit 비교 (직렬로 연결)

n bit 비교

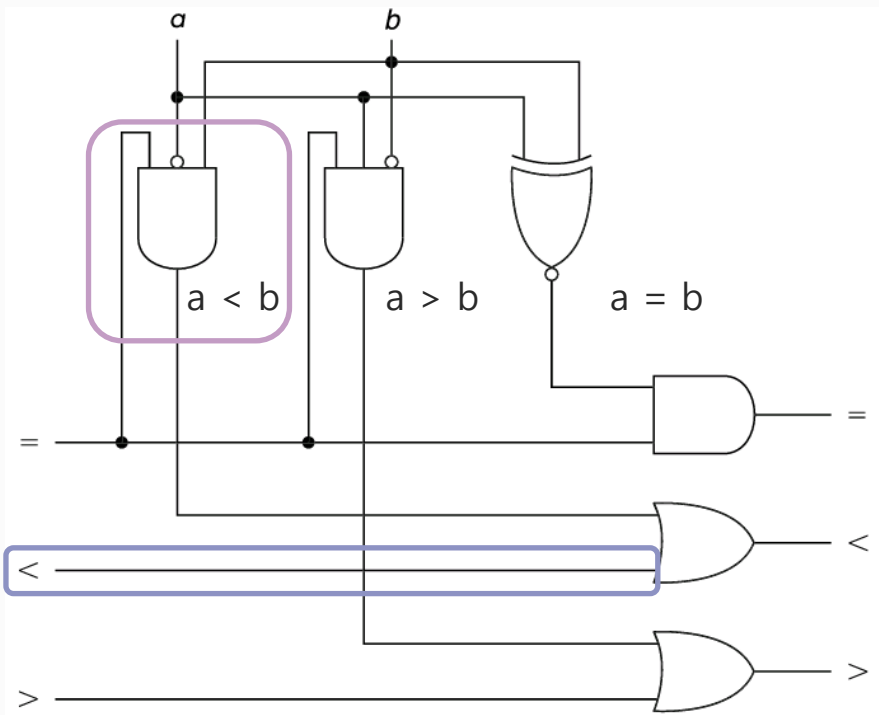
a > b

$$\left\{ \begin{array}{l} (a_{n+1} > b_{n+1}) \text{ or} \\ (a_{n+1} = b_{n+1} \text{ and } a_n > b_n) \end{array} \right.$$

n-bit a, b 비교에서

- 1) 상위 비트 a_{n+1} 가 크면 a가 크다고 결정
- 2) 상위비트에서 a, b가 똑같은 경우
현재 비트 a_n 이 크면 a가 크다고 할 수 있음

비교기 - '크다' '같다' '작다'



multi bit 비교 (직렬로 연결)

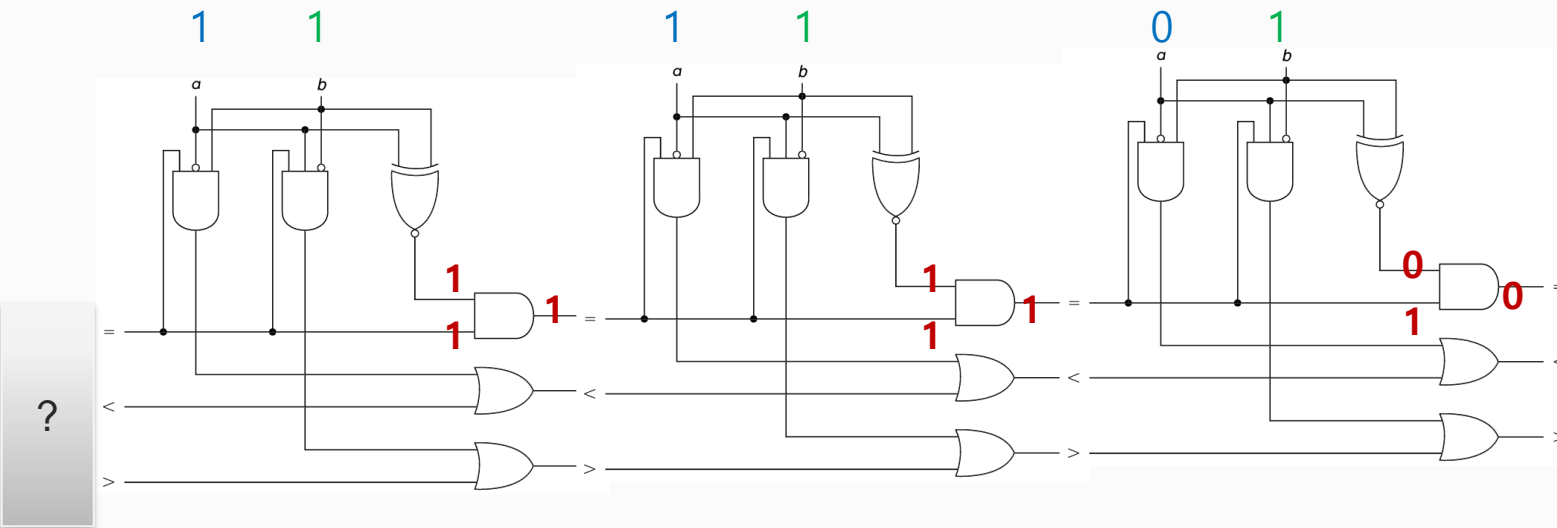
n bit 比較

$$a < b$$
$$\left\{ \begin{array}{l} (a_{n+1} < b_{n+1}) \text{ or} \\ (a_{n+1} = b_{n+1} \text{ and } a_n < b_n) \end{array} \right.$$

n-bit a, b 비교에서

- 1) 상위 비트 b_{n+1} 가 크면 b 가 크다고 결정
- 2) 상위비트에서 a, b 가 똑같은 경우
현재 비트 b_n 이 크면 b 가 크다고 할 수 있음

3-bit 비교기 - '크다' '같다' '작다'

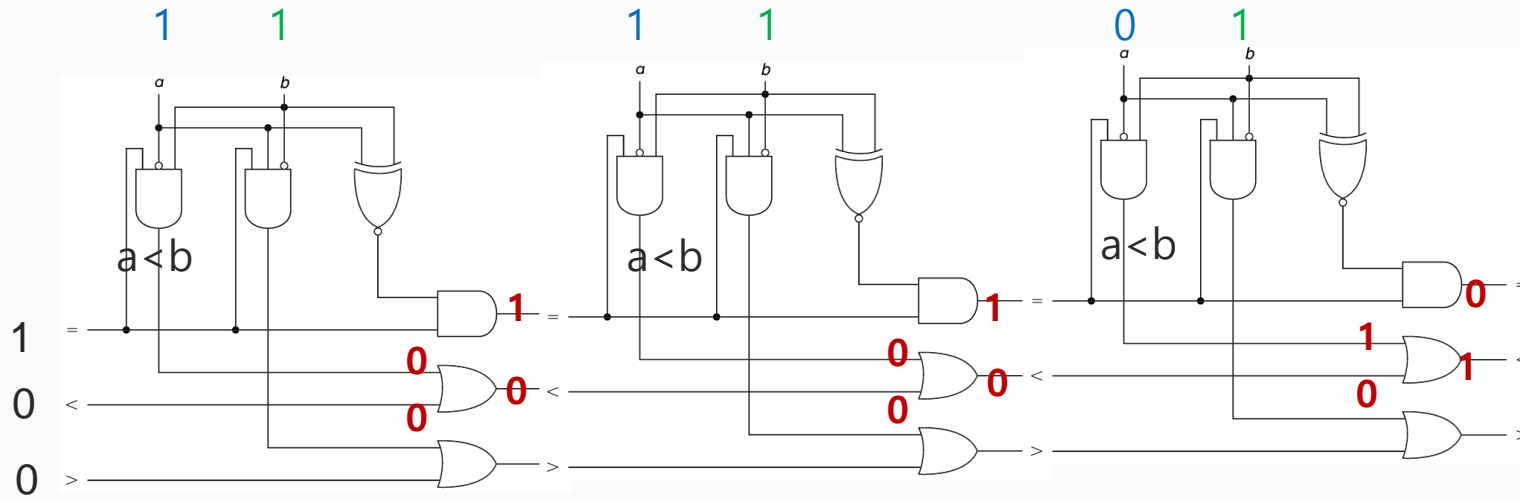


비교

a: 1 1 0 (6_{10})

b: 1 1 1 (7_{10})

3-bit 비교기 - '크다' '같다' '작다'

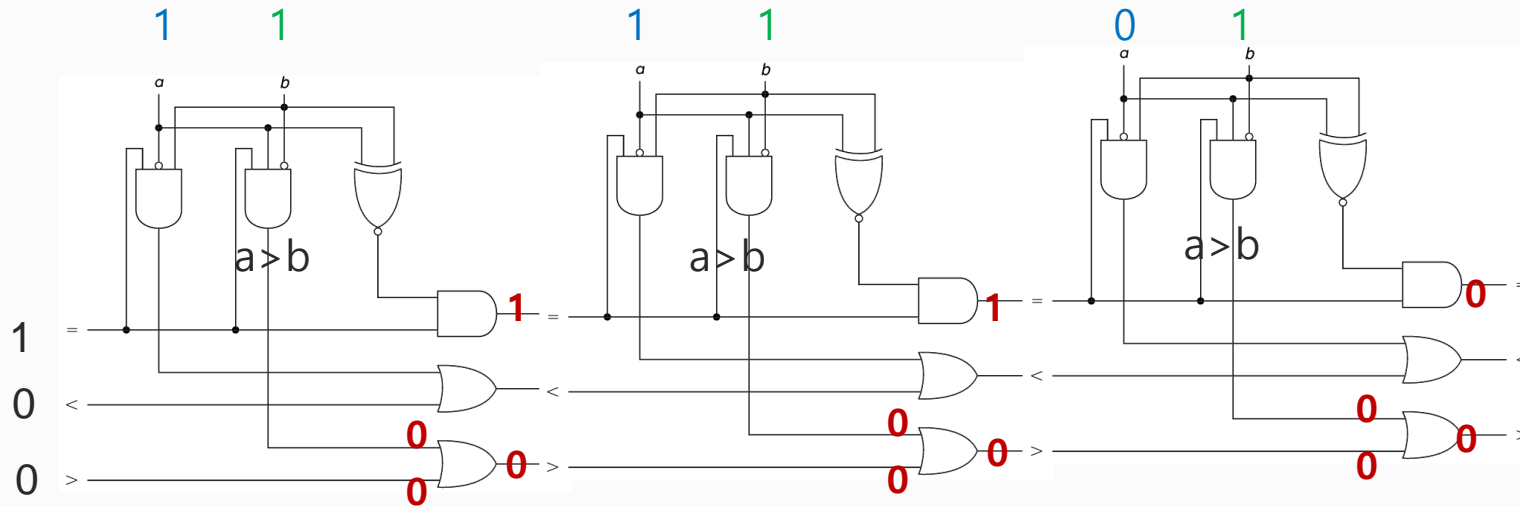


비교

a: 1 1 0 (6_{10})

b: 1 1 1 (7_{10})

3-bit 비교기 - '크다' '같다' '작다'



비교
a: 1 1 0 (6_{10})
b: 1 1 1 (7_{10})

Chapter 5 Designing Combinational Systems

5.1 Iterative (반복) 시스템 ex) ALUs

5.2 Binary Decoders

5.3 Encoders and Priority Encoders

5.4 Multiplexer (Mux) & Demultiplexers (Demux)

5.5 Tree-State Gates

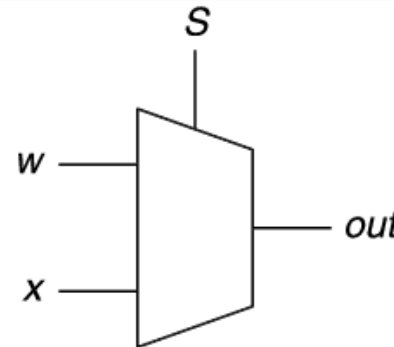
5.6 Gate Arrays – ROM, PLA, PALs..

어떤 입력을 고를까?

Multiplexer (Mux) – *2 way mux*

N개의 데이터 입력 중 하나를 선택하여 출력으로 보낸다.

<2-way Mux>

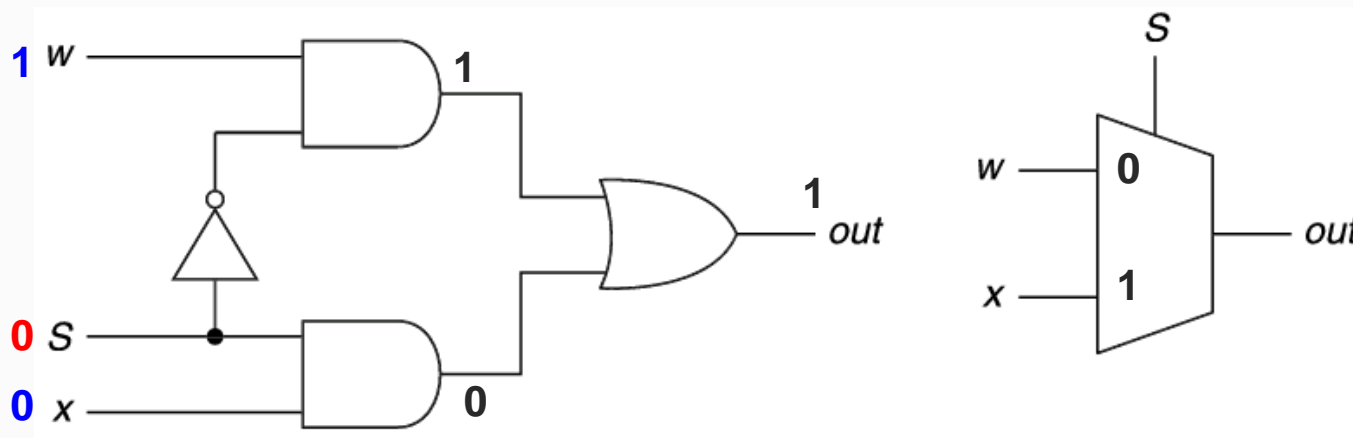


If ($s==0$)	$out = w$
else	$out = x$

Multiplexer (Mux) – 2 way mux

N개의 데이터 입력 중 하나를 선택하여 출력으로 보낸다.

<2-way Mux>

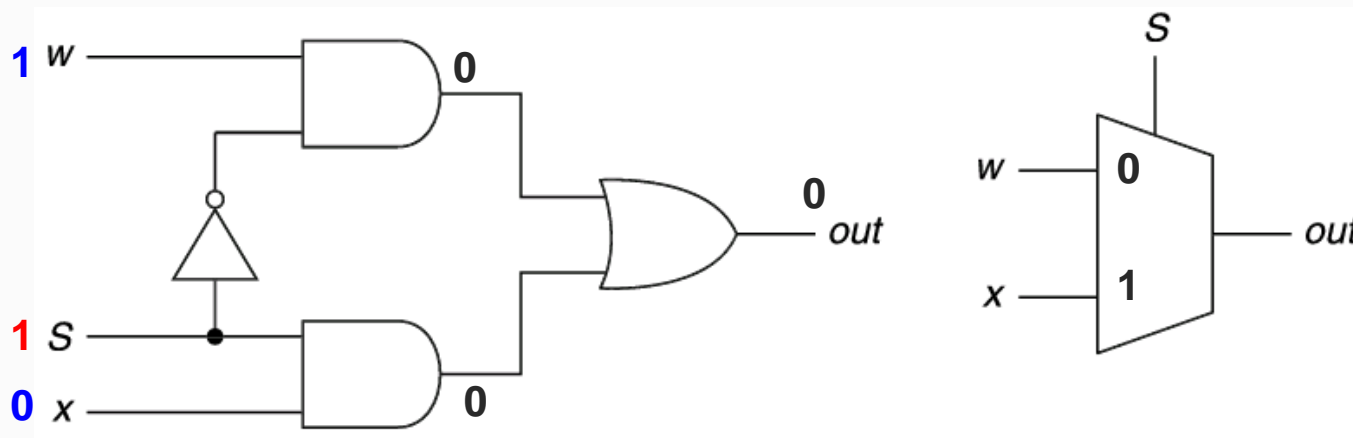


If ($s==0$) $out = w$
else $out = x$

Multiplexer (Mux) – 2 way mux

N개의 데이터 입력 중 하나를 선택하여 출력으로 보낸다.

<2-way Mux>

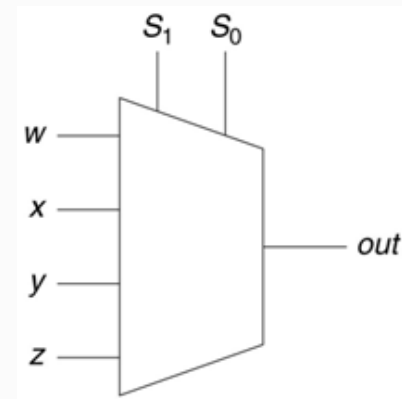
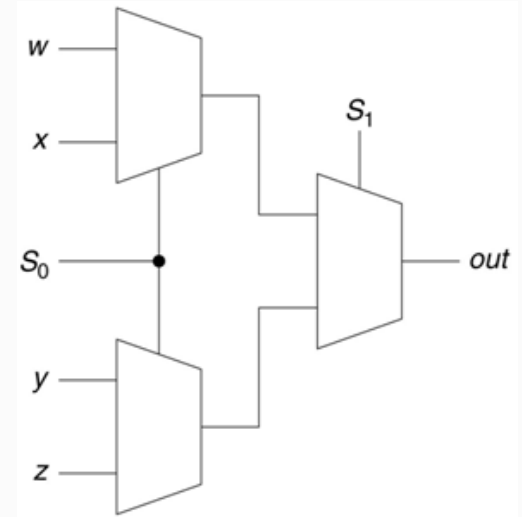
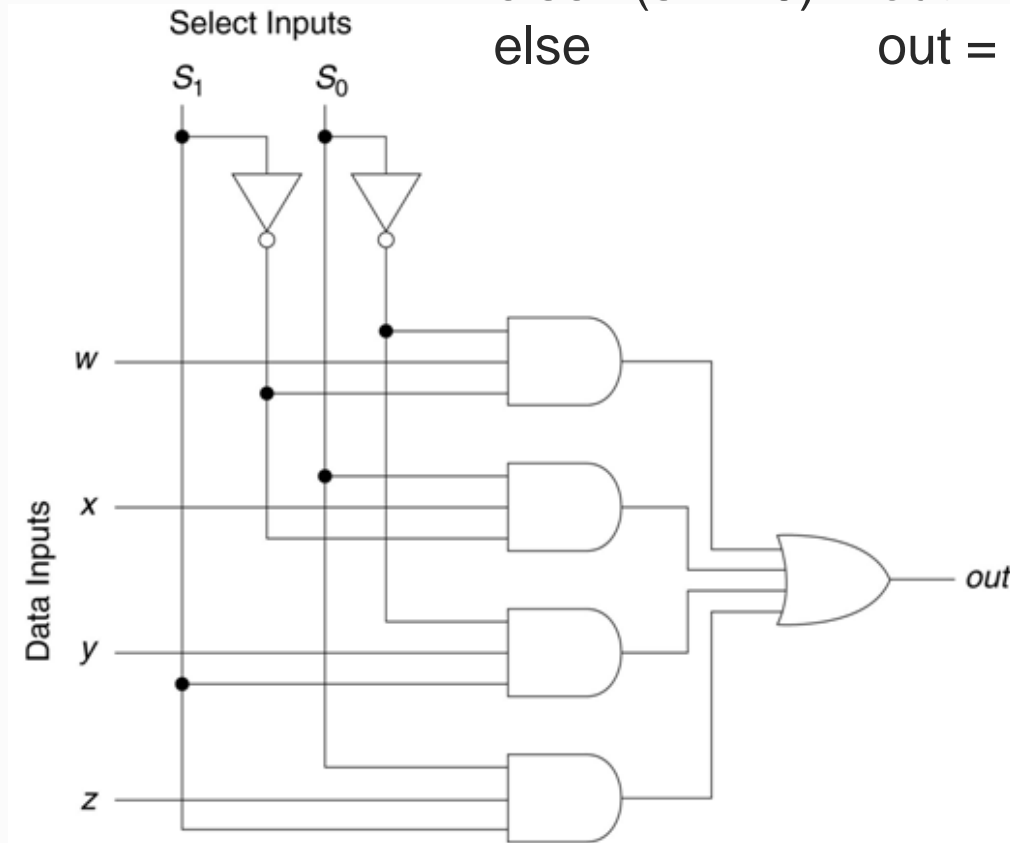


If ($s==0$) $out = w$
else $out = x$

Multiplexer (Mux) – 4 way mux

<4-way Mux>

if (s==00) out = w
else if (s==01) out = x
else if (s==10) out = y
else out = z



어떤 입력을 고를까?

입력이 1 bit가 아니고

4 bit 숫자라면? 16 bit 숫자라면?

Multiplexer (Mux) – *multi-bit mux*

<Multi-bit Mux>

16 bit 숫자 4개 중 한 개를 선택

- 1) 1000101010100010
- 2) 1010101110001010
- 3) 1010001010101010
- 4) 1001101011100010

4개 중 어떤 것?

Multiplexer (Mux) – *multi-bit mux*

<Multi-bit Mux>

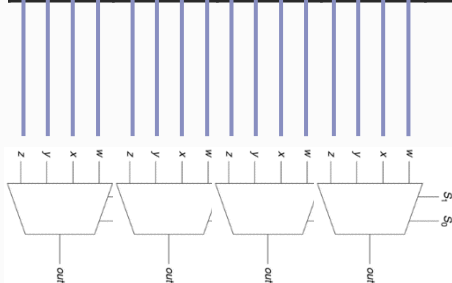
16 bit 숫자 4개 중 한 개를 선택

- 1) 1000101010100010
- 2) 1010101110001010
- 3) 1010001010101010
- 4) 1001101011100010

4개 중 어떤 것?

숫자1
숫자2
숫자3
숫자4

In 15	In 14	In 13	In 12	In 11	In 10	In 9	In 8	In 7	In 6	In 5	In 4	In 3	In 2	In 1	In 0
1	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0
1	0	1	0	1	0	1	1	1	0	0	0	1	0	1	0
1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	0
1	0	0	1	1	0	1	0	1	1	1	0	0	0	1	0



...

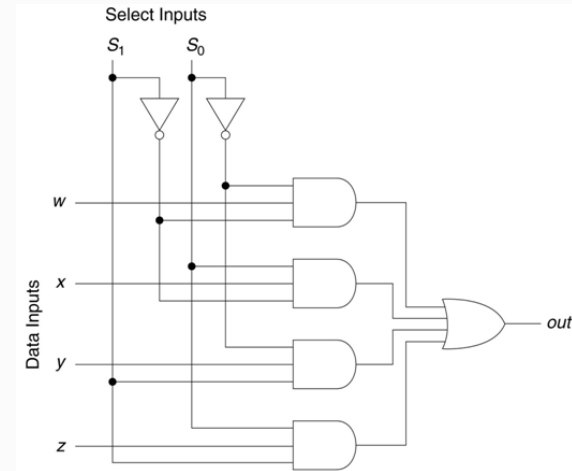
각 비트마다
MUX 할당

Multiplexer (Mux) – *multi-bit mux*

<Multi-bit Mux>

16 bit 숫자 4개 중 한 개를 선택

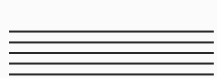
- 4개 중 1개 고르는 것이므로 4:1 mux 필요
- 비트수가 16이니까 16개의 4:1 mux 사용



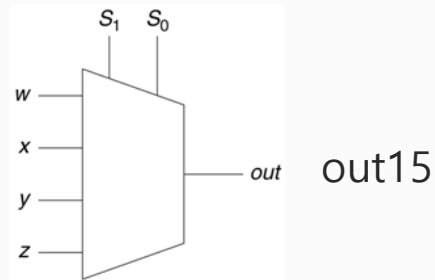
....

....

- 64(4 X 16)개의 3-입력 AND 게이트
- 16 (1x 16) 개의 4-입력 OR 게이트



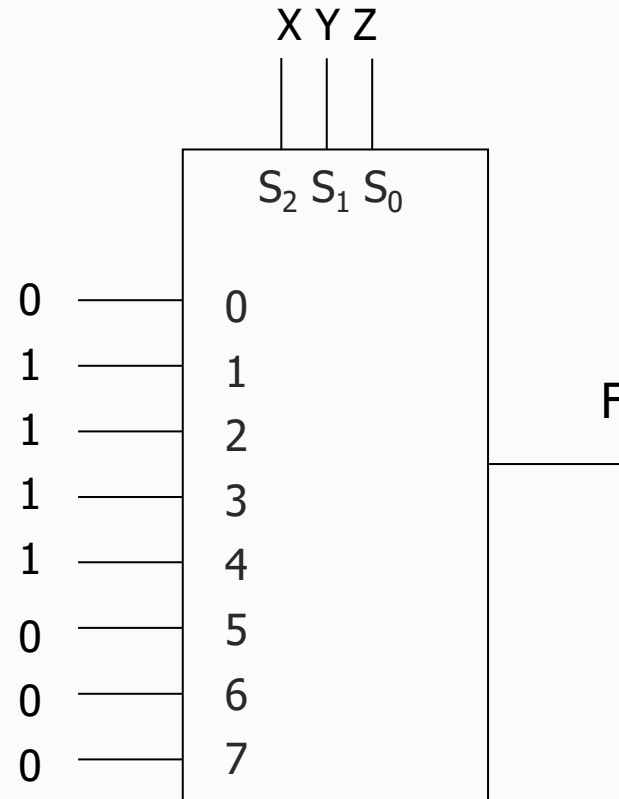
4개 숫자의
in15 bit (MSB)



Mux의 또다른 쓸모: 함수 구현

예제 5.5

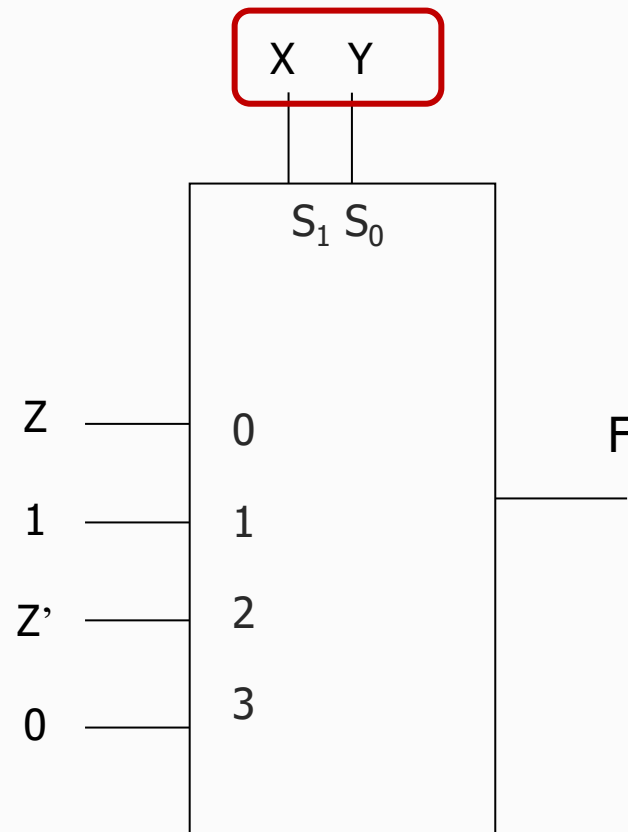
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0



Mux의 또다른 쓸모: 함수 구현

예제 5.5 좀 더 작은 Mux를 사용하도록 간략화 할 방법은?

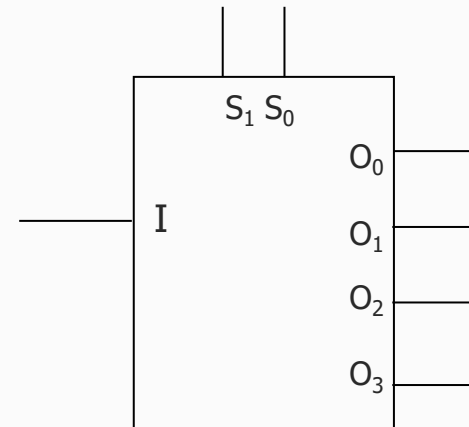
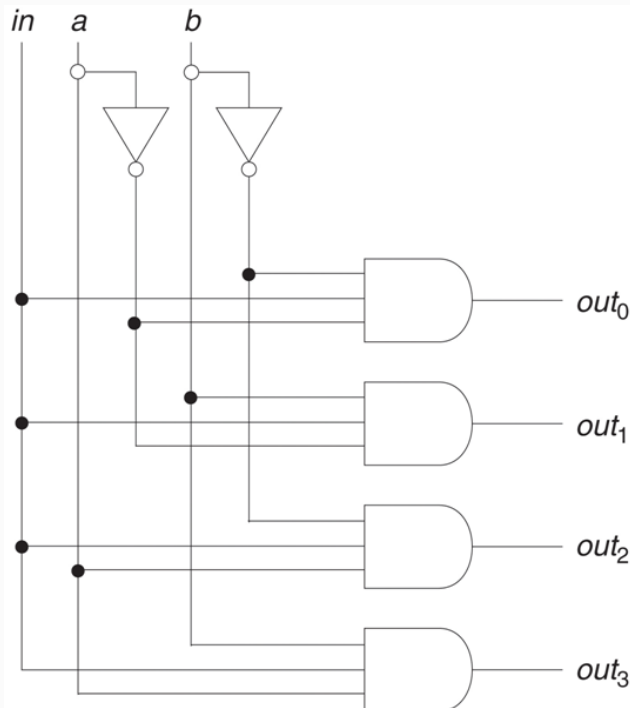
X	Y	Z	F	
0	0	0	0	$F = Z$
0	0	1	1	
0	1	0	1	$F = 1$
0	1	1	1	
1	0	0	1	$F = Z'$
1	0	1	0	
1	1	0	0	$F = 0$
1	1	1	0	



Demultiplexer (Demux)

- 멀티플렉서의 역함수 역할
- 입력을 어느 출력으로 보낼지 결정

S_1	S_0	$O_0 \ O_1 \ O_2 \ O_3$			
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I



Chapter 5 Designing Combinational Systems

5.1 Iterative (반복) 시스템 ex) ALUs

5.2 Binary Decoders

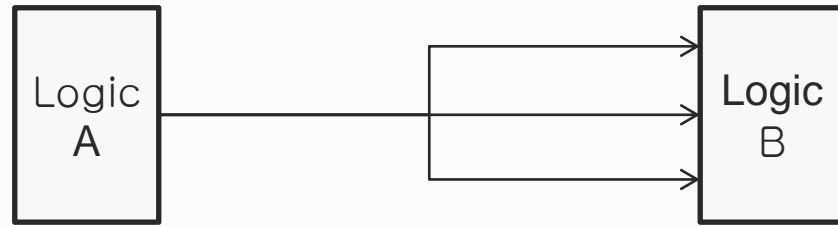
5.3 Encoders and Priority Encoders

5.4 Multiplexer (Mux) & Demultiplexers (Demux)

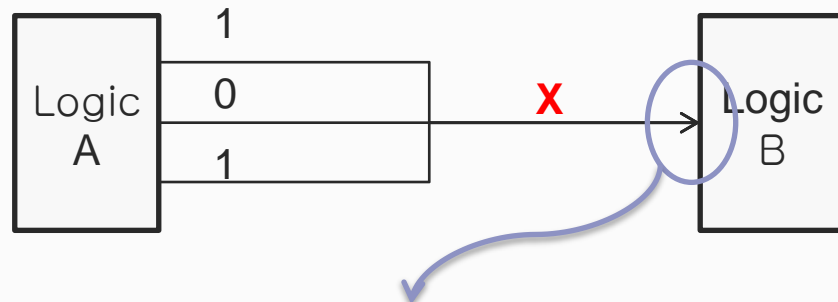
5.5 Three-State Gates

5.6 Gate Arrays – ROM, PLA, PALs..

단일 출력 → 다중 입력



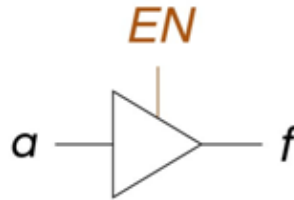
다중 출력 → 단일 입력



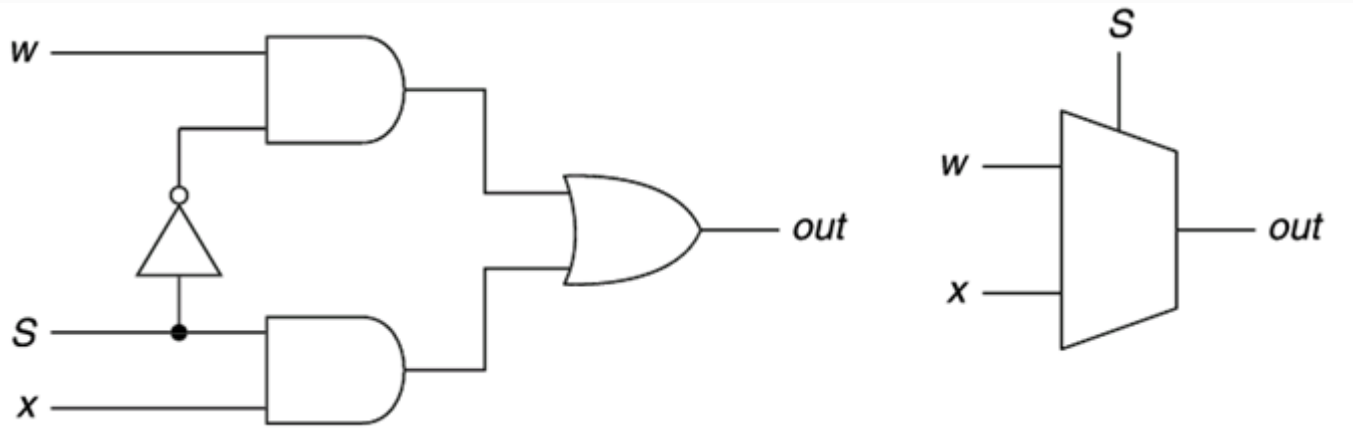
Three-State (3-상태) Gate

3 state: 0, 1 and Z

<i>EN</i>	<i>a</i>	<i>f</i>
0	0	Z
0	1	Z
1	0	0
1	1	1



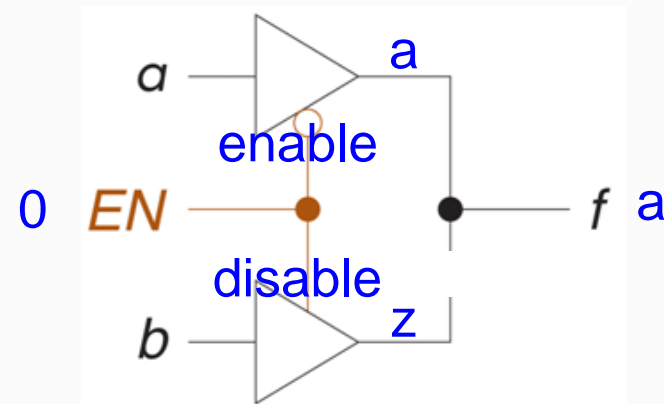
Three-State (3-상태) Gate로 Mux 구현



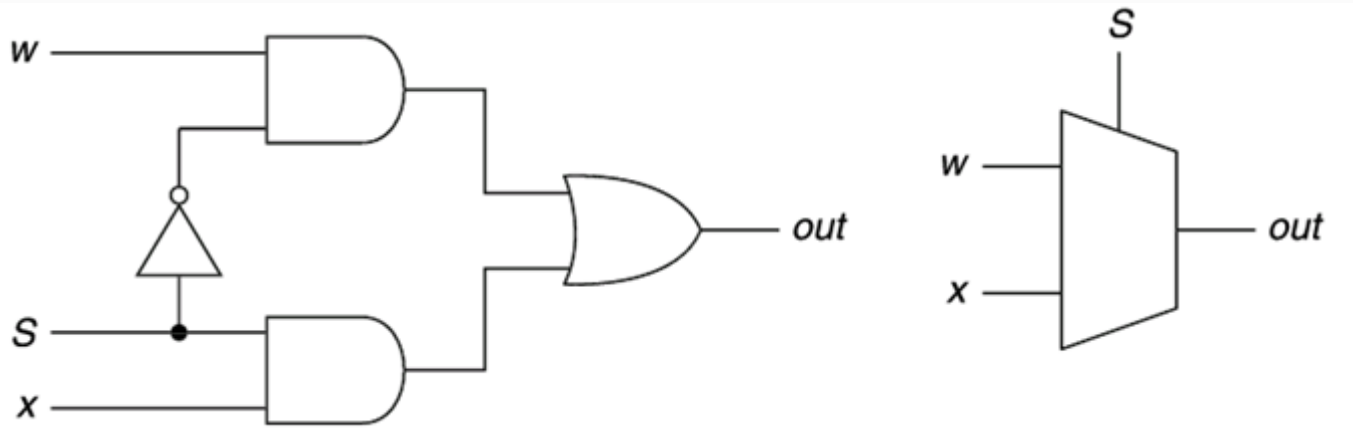
- 3-상태(three-state) 게이트로 OR 게이트 없는 멀티플렉서 구현 가능
- 2-to-1 멀티플렉서

$$\Rightarrow f = a \cdot EN' + b \cdot EN$$

if($EN == 0$) $f = a$
else $f = b$

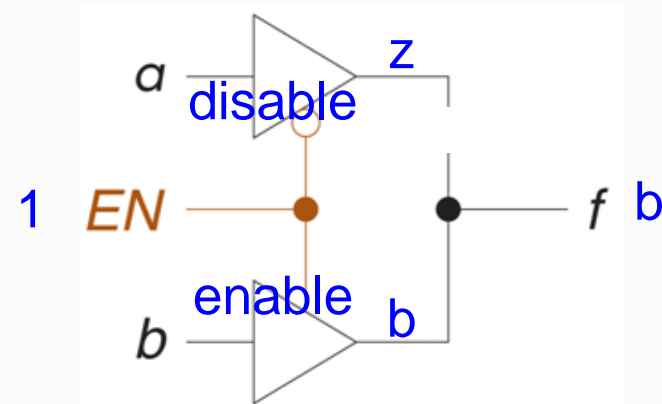


Three-State (3-상태) Gate로 Mux 구현



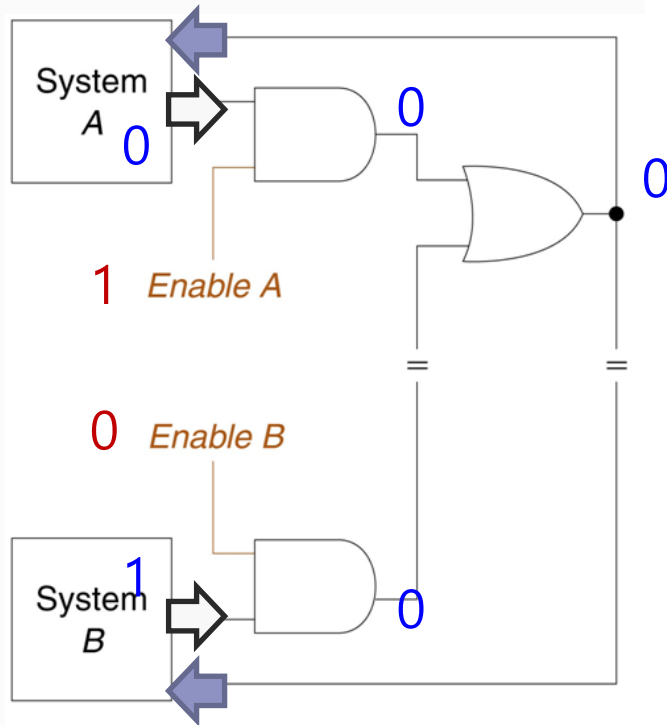
- 3-상태(three-state) 게이트로 OR 게이트 없는 멀티플렉서 구현 가능
- 2-to-1 멀티플렉서

$$\Rightarrow f = a \cdot EN' + b \cdot EN$$



Bus 구현

1-비트 버스 구현 - Mux type으로 구현

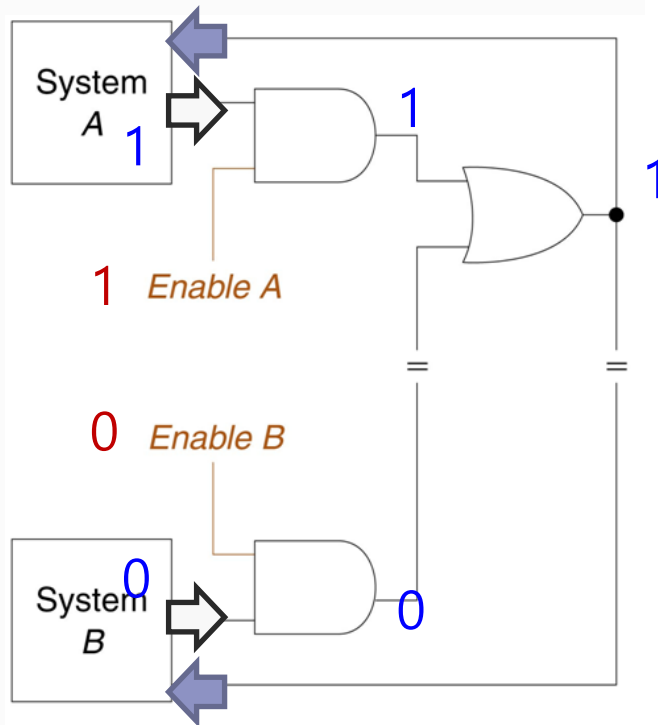


(a) Using AND/OR Gates

버스에 A와 B 중
누가 출력한 데이터가
돌아다닐 것인가

Bus 구현

1-비트 버스 구현 - Mux type으로 구현

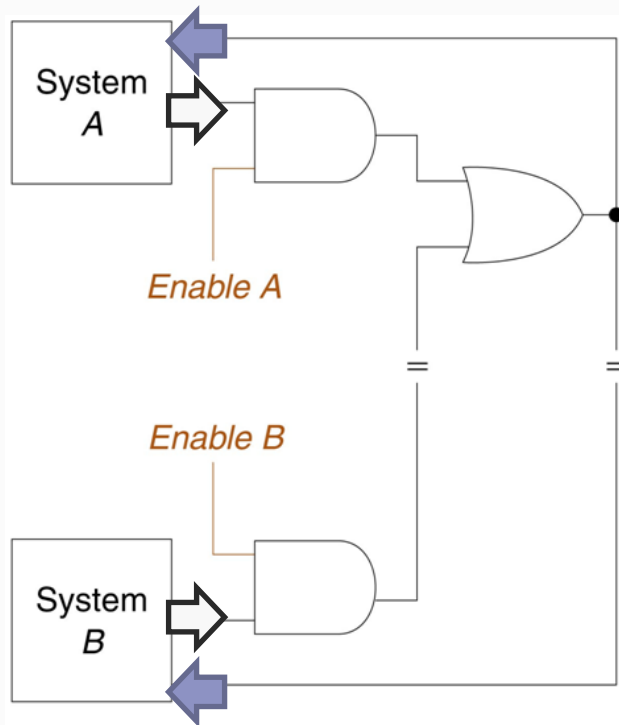


(a) Using AND/OR Gates

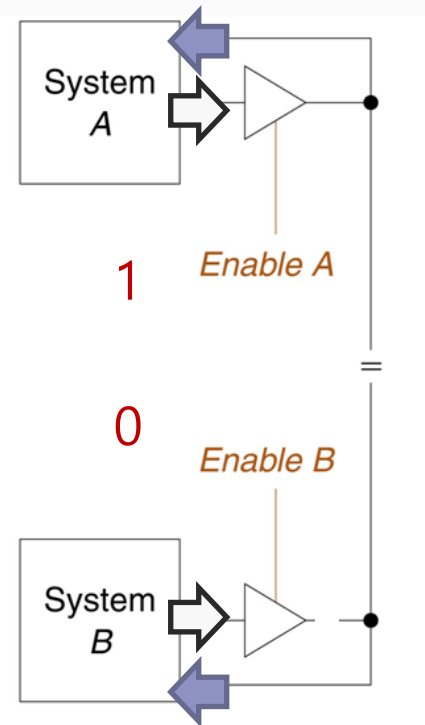
버스에 A와 B 중
누가 출력한 데이터가
돌아다닐 것인가

Three-State (3-상태) Gate로 Bus 구현

1-비트 버스 구현 - tri-state 로 구현



(a) Using AND/OR Gates

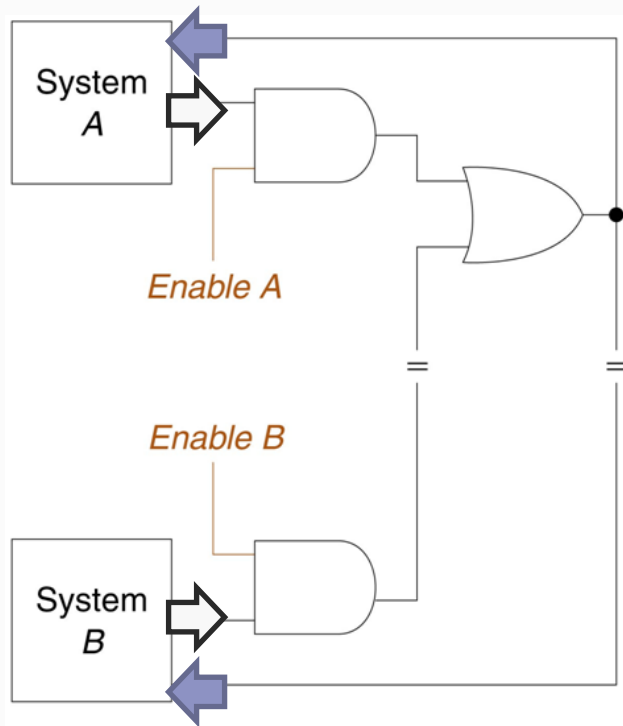


(b) Using Three-State Gates

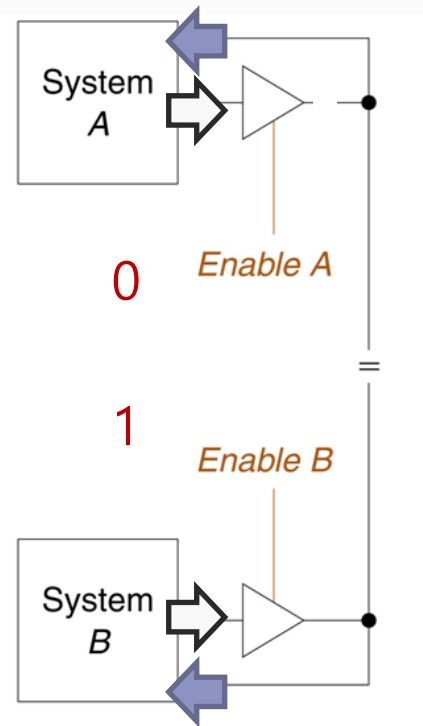
Conflict?

Three-State (3-상태) Gate로 Bus 구현

1-비트 버스 구현 - tri-state 로 구현



(a) Using AND/OR Gates



(b) Using Three-State Gates

Chapter 5 Designing Combinational Systems

5.1 Iterative (반복) 시스템 ex) ALUs

5.2 Binary Decoders

5.3 Encoders and Priority Encoders

5.4 Multiplexer (Mux) & Demultiplexers (Demux)

5.5 Tree-State Gates

5.6 Gate Arrays – ROM, PLA, PALs..

디코더

- 디코더는 코딩된 입력 신호에 의해 **여러 출력 선 중 하나를 선택하는 장치**
=> 일반적으로, 입력은 n비트 이진수이고 2^n 의 출력 선이 존재
- 2-입력(4-출력) **active high** 출력 디코더에 대한 진리표
=> 활성화되는 출력은 1이고 비활성화되는 출력은 0

<i>a</i>	<i>b</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

* **active low**: 활성화되는 출력은 0이고 비활성화되는 출력은 1 인 경우

2 입력 디코더 - *active high*

- 2-입력(4-출력) active high 출력 디코더
- 각 출력은 2변수 함수의 minterm들과 일치

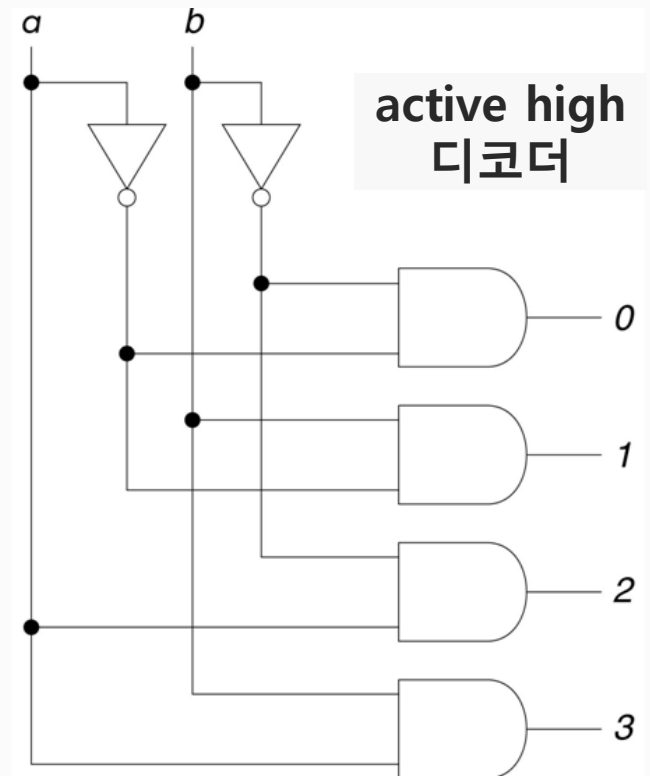
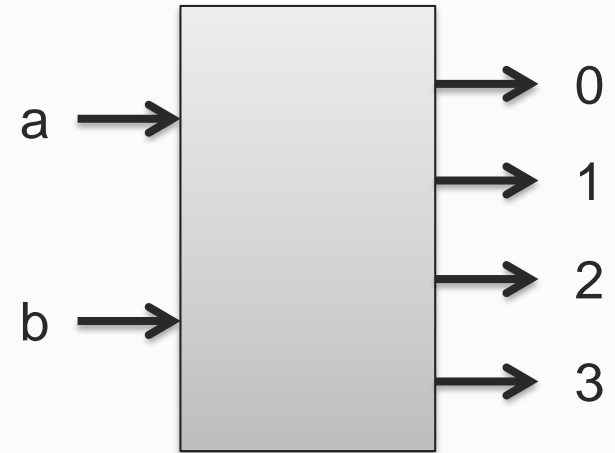
출력0 : $a'b'$

출력1 : $a'b$

출력2 : ab'

출력3 : ab

<i>a</i>	<i>b</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



2 입력 디코더 - *active low*

- active low 출력 형식의 디코더 회로와 진리표

출력0 : $(a'b')'$

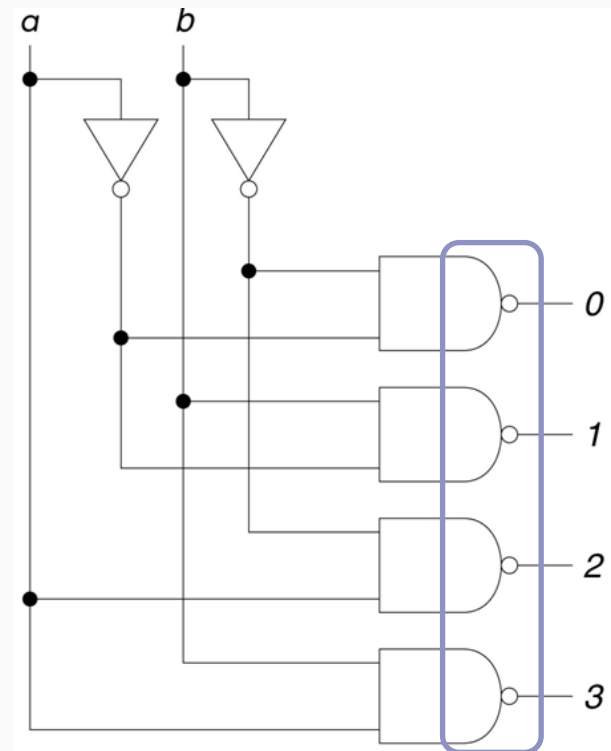
출력1 : $(a'b)$

출력2 : (ab')

출력3 : (ab)

<i>a</i>	<i>b</i>	0	1	2	3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

active low 디코더

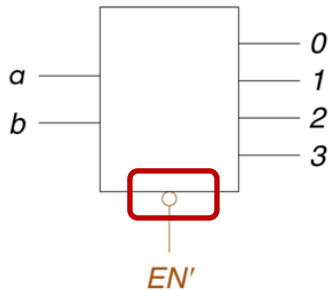


2 입력 디코더 *with Enable*(활성화 신호)

- 예) **active low 활성화 신호**가 있는 active high 출력 디코더 (진리표, 블록 다이어그램, 회로)
- active low 신호는 블록 다이어그램에서 **작은 원(bubble)**으로 표시

<i>EN'</i>	<i>a</i>	<i>b</i>	0	1	2	3
1	X	X	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

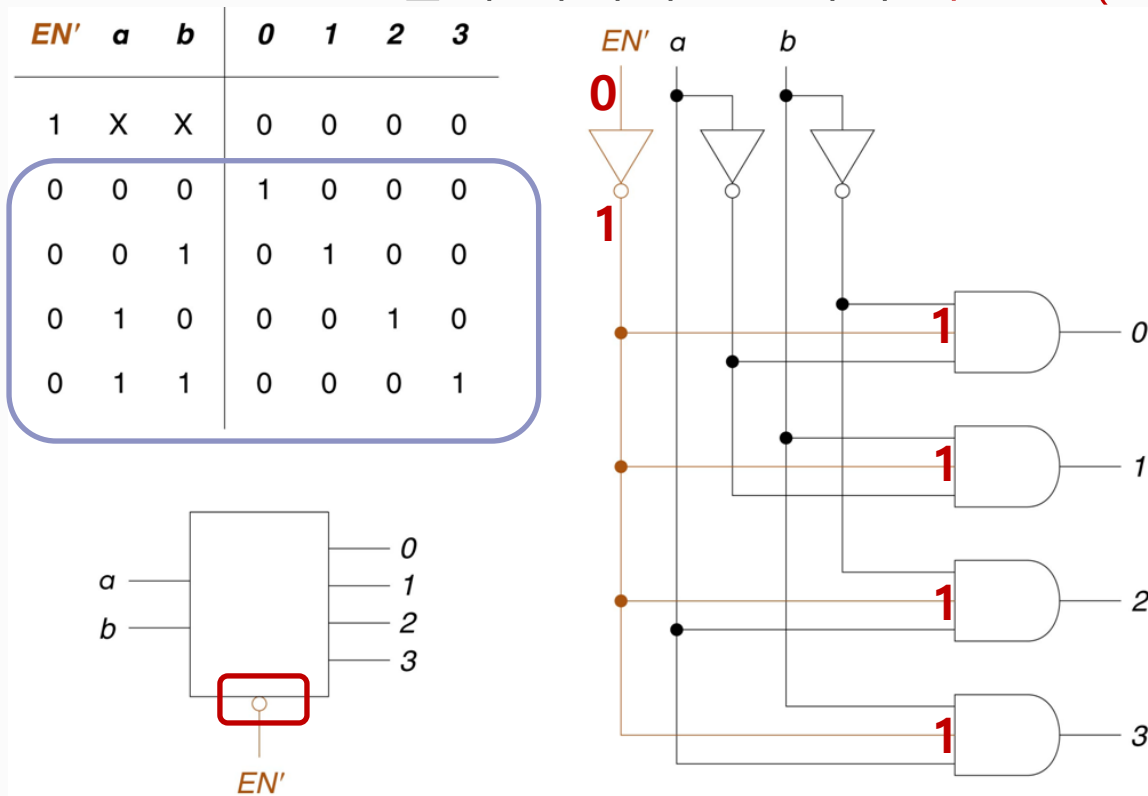
Enable signal=0일 경우에만 동작



Q. Active low 활성화 신호가 active high 활성화 신호보다 좋은 경우는?
어떤 점을 고려해서 active low/high를 선택하게 될까?

2 입력 디코더 *with Enable*(활성화 신호)

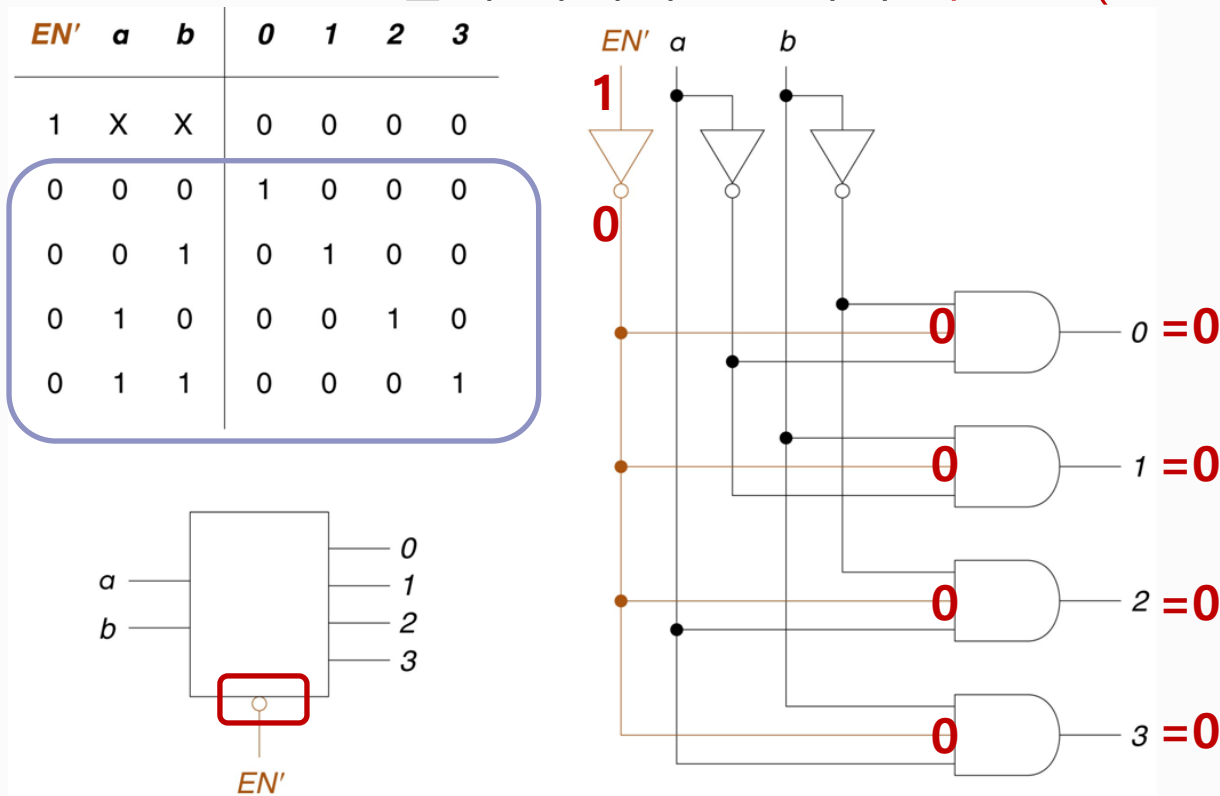
- 예) **active low 활성화 신호**가 있는 active high 출력 디코더 (진리표, 블록 다이어그램, 회로)
- active low 신호는 블록 다이어그램에서 **작은 원(bubble)**으로 표시



Q. Active low 활성화 신호가 active high 활성화 신호보다 좋은 경우는? 어떤 점을 고려해서 active low/high를 선택하게 될까?

2 입력 디코더 *with Enable*(활성화 신호)

- 예) **active low 활성화 신호**가 있는 active high 출력 디코더 (진리표, 블록 다이어그램, 회로)
- active low 신호는 블록 다이어그램에서 **작은 원(bubble)**으로 표시



Q. Active low 활성화 신호가 active high 활성화 신호보다 좋은 경우는? 어떤 점을 고려해서 active low/high를 선택하게 될까?

74138 (3-to-8 decoder)

- active low 출력과 3개의 동작 입력

=> active high : EN1
active low : EN2', EN3'

=> **EN1 = 1 and EN2' = 0 and EN3' = 0 일 때 동작**

- 입력은 C, B, A(C가 최상위 비트)로 표기

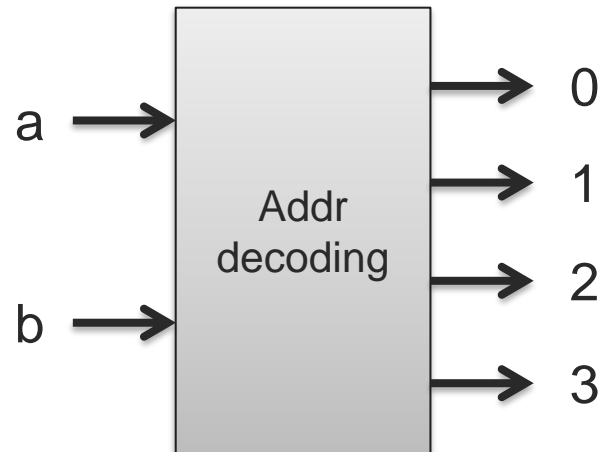


Decoder를 어디에 사용해 볼까?

- 1) Address decoding : 장치 선택
- 2) 논리함수 구현하기

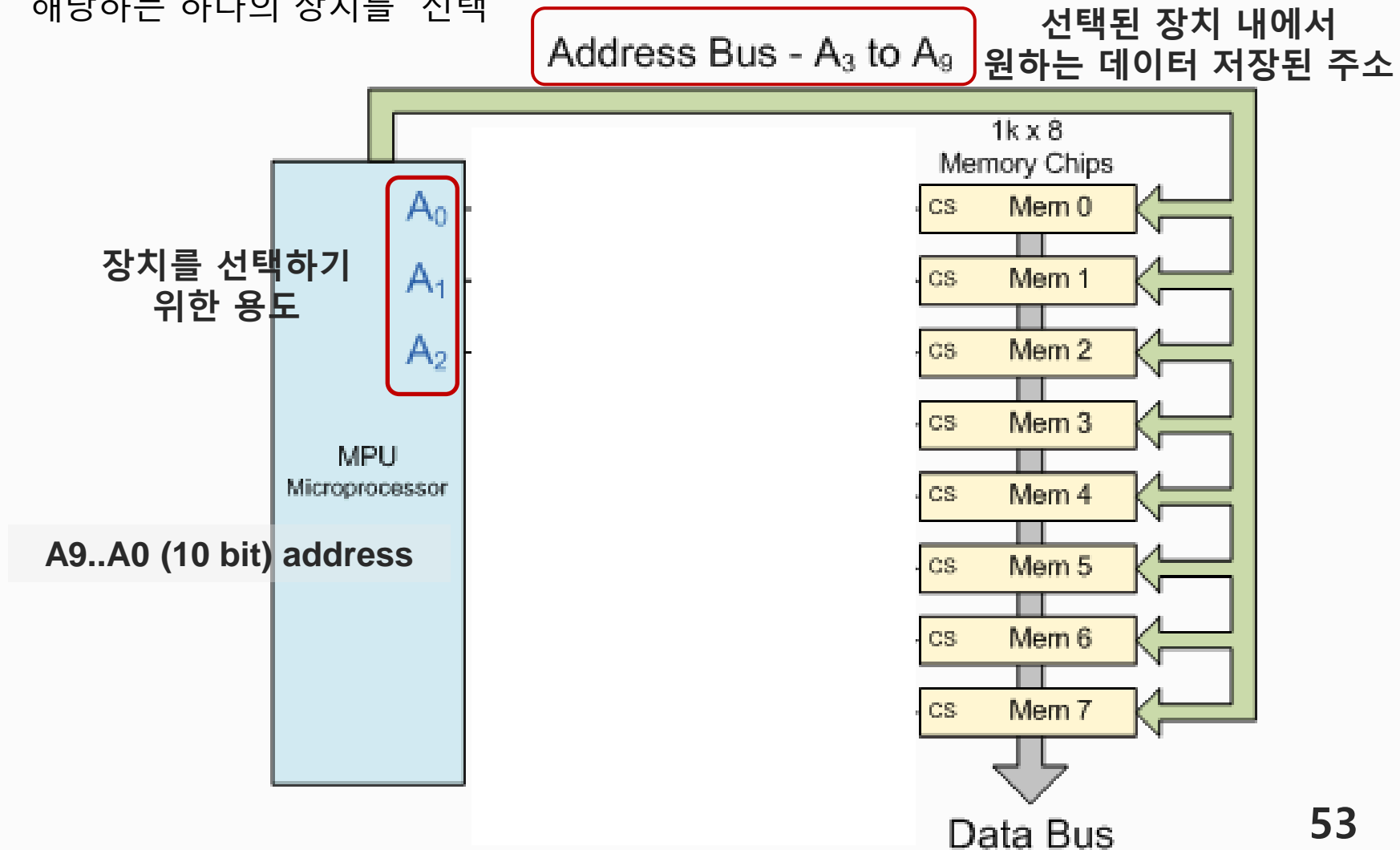
Decoder의 응용 - *addr decoding*

- 각각 고유의 주소를 갖고 있는 많은 장치 중에서 하나를 선택하는 것
- 주소는 디코더의 입력이 되고 하나의 출력이 활성화되면 그 주소에 해당하는 하나의 장치를 선택



Decoder의 응용 - *addr decoding*

- 각각 고유의 주소를 갖고 있는 많은 장치 중에서 하나를 선택하는 것
- 주소는 디코더의 입력이 되고 하나의 출력이 활성화되면 그 주소에 해당하는 하나의 장치를 선택



2301	2302
2201	2202
2101	2102
2001	2002
1901	1902
1801	1802
1701	1702
1601	1602
1501	1502
1401	1402
1301	1302
1201	1202
1101	1102
1001	1002
901	902
801	802
701	702
601	602
501	502
401	402
301	302
필로티	필로티
필로티	필로티

101동

2801	2802	2803	2804
2701	2702	2703	2704
2601	2602	2603	2604
2501	2502	2503	2504
2401	2402	2403	2404
2301	2302	2303	2304
2201	2202	2203	2204
2101	2102	2103	2104
2001	2002	2003	2004
1901	1902	1903	1904
1801	1802	1803	1804
1701	1702	1703	1704
1601	1602	1603	1604
1501	1502	1503	1504
1401	1402	1403	1404
1301	1302	1303	1304
1201	1202	1203	1204
1101	1102	1103	1104
1001	1002	1003	1004
901	902	903	904
801	802	803	804
701	702	703	704
601	602	603	604
501	502	503	504
401	402	403	404
301	302	303	304
필로티	202	203	204
필로티	102	103	104

102동

2701	2702	2703	2704
2601	2602	2603	2604
2501	2502	2503	2504
2401	2402	2403	2404
2301	2302	2303	2304
2201	2202	2203	2204
2101	2102	2103	2104
2001	2002	2003	2004
1901	1902	1903	1904
1801	1802	1803	1804
1701	1702	1703	1704
1601	1602	1603	1604
1501	1502	1503	1504
1401	1402	1403	1404
1301	1302	1303	1304
1201	1202	1203	1204
1101	1102	1103	1104
1001	1002	1003	1004
901	902	903	904
801	802	803	804
701	702	703	704
601	602	603	604
501	502	503	504
401	402	403	404
301	302	303	304
201	202	203	204
101	102	103	104

103동

2501	2502	2503	2504
2401	2402	2403	2404
2301	2302	2303	2304
2201	2202	2203	2204
2101	2102	2103	2104
2001	2002	2003	2004
1901	1902	1903	1904
1801	1802	1803	1804
1701	1702	1703	1704
1601	1602	1603	1604
1501	1502	1503	1504
1401	1402	1403	1404
1301	1302	1303	1304
1201	1202	1203	1204
1101	1102	1103	1104
1001	1002	1003	1004
901	902	903	904
801	802	803	804
701	702	703	704
601	602	603	604
501	502	503	504
401	402	403	404
301	302	303	304
201	필로티	필로티	204
101	필로티	필로티	104

104동

양산부동산114닷컴
http://www.ysland114.com

2301	2302
2201	2202
2101	2102
2001	2002
1901	1902
1801	1802
1701	1702
1601	1602
1501	1502
1401	1402
1301	1302
1201	1202
1101	1102
1001	1002
901	902
801	802
701	702
601	602
501	502
401	402
301	302
201	202
101	102

105동

2801	2802	2803	2804
2701	2702	2703	2704
2601	2602	2603	2604
2501	2502	2503	2504
2401	2402	2403	2404
2301	2302	2303	2304
2201	2202	2203	2204
2101	2102	2103	2104
2001	2002	2003	2004
1901	1902	1903	1904
1801	1802	1803	1804
1701	1702	1703	1704
1601	1602	1603	1604
1501	1502	1503	1504
1401	1402	1403	1404
1301	1302	1303	1304
1201	1202	1203	1204
1101	1102	1103	1104
1001	1002	1003	1004
901	902	903	904
801	802	803	804
701	702	703	704
601	602	603	604
501	502	503	504
401	402	403	404
301	302	303	304
필로티	202	203	204
필로티	102	103	104

106동

2701	2702	2703	2704
2601	2602	2603	2604
2501	2502	2503	2504
2401	2402	2403	2404
2301	2302	2303	2304
2201	2202	2203	2204
2101	2102	2103	2104
2001	2002	2003	2004
1901	1902	1903	1904
1801	1802	1803	1804
1701	1702	1703	1704
1601	1602	1603	1604
1501	1502	1503	1504
1401	1402	1403	1404
1301	1302	1303	1304
1201	1202	1203	1204
1101	1102	1103	1104
1001	1002	1003	1004
901	902	903	904
801	802	803	804
701	702	703	704
601	602	603	604
501	502	503	504
401	402	403	404
301	302	303	304
201	202	203	204
101	102	103	104

107동

		2603	2604
		2503	2504
2401	2402	2403	2404
2301	2302	2303	2304
2201	2202	2203	2204
2101	2102	2103	2104
2001	2002	2003	2004
1901	1902	1903	1904
1801	1802	1803	1804
1701	1702	1703	1704
1601	1602	1603	1604
1501	1502	1503	1504
1401	1402	1403	1404
1301	1302	1303	1304
1201	1202	1203	1204
1101	1102	1103	1104
1001	1002	1003	1004
901	902	903	904
801	802	803	804
701	702	703	704
601	602	603	604
501	502	503	504
401	402	403	404
301	302	303	304
201	202	203	204
101	102	103	104

108호



XXX - YYYY
XXX동 YYYY호

총 7개의 숫자 사용하여 장치(동)와
장치 내의 메모리 위치(호)를 표시

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

2bit: 장치 선택

2bit: 장치 내부의 공간 위치 선택

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

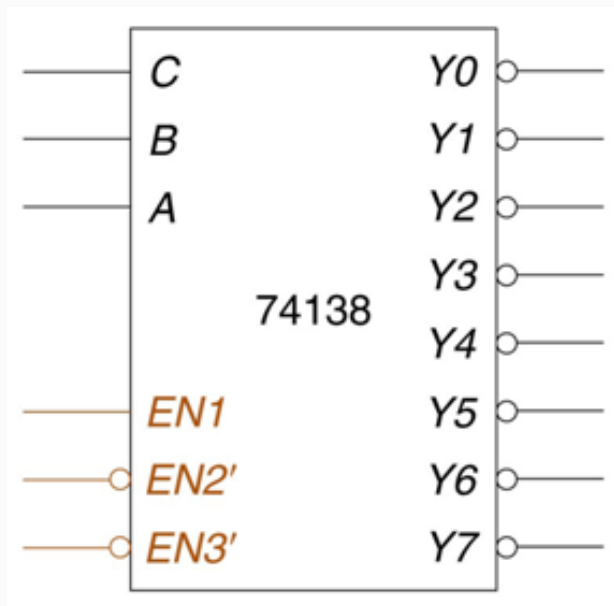
1bit: 장치 선택

3bit: 장치 내부의 공간 위치 선택

Decoder의 응용 - *addr decoding*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 동작 입력(enable신호)을 이용하여 디코드

예제 5.1 74138 디코더를 사용하여 **32개의 장치** 중 하나를 선택하기 위한 회로를 설계하라



5개 입력 32개 출력
decoder가
있으면 좋겠는데.

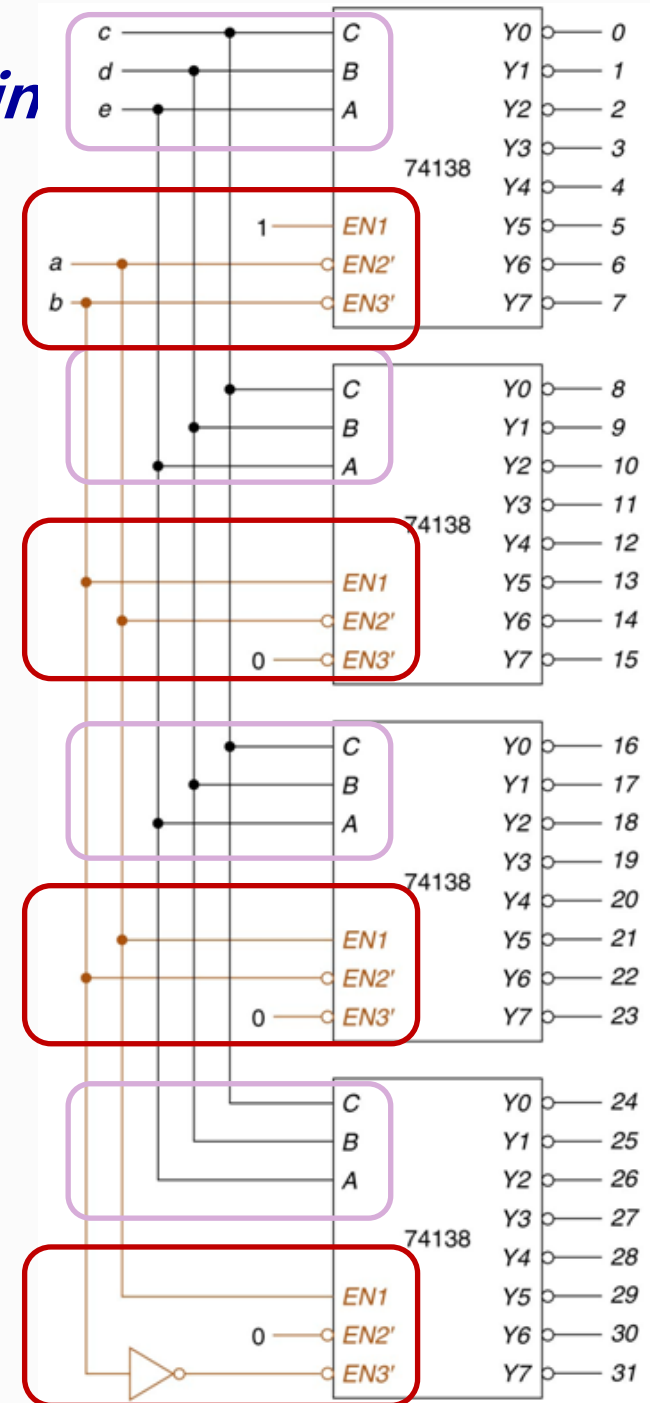
만약 없다면?

Decoder의 응용 - *addr decodin*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 동작 입력(enable신호)을 이용하여 디코드

예제 5.1 74138 디코더를 사용하여 **32개의 장치** 중 하나를 선택하기 위한 회로를 설계하라

- 4개의 동일한 디코더를 사용
- 주소가 **a, b, c, d, e**로 주어졌다면,
 - c, d, e 비트가 4개의 디코더에 공통으로 사용 (C, B, A의 순서)
→ **각 장치 내의 위치 선택 (8가지 출력)**
 - a, b는 4개의 디코더 중에서 1개를 선택
→ **장치 선택 (Chip select)**

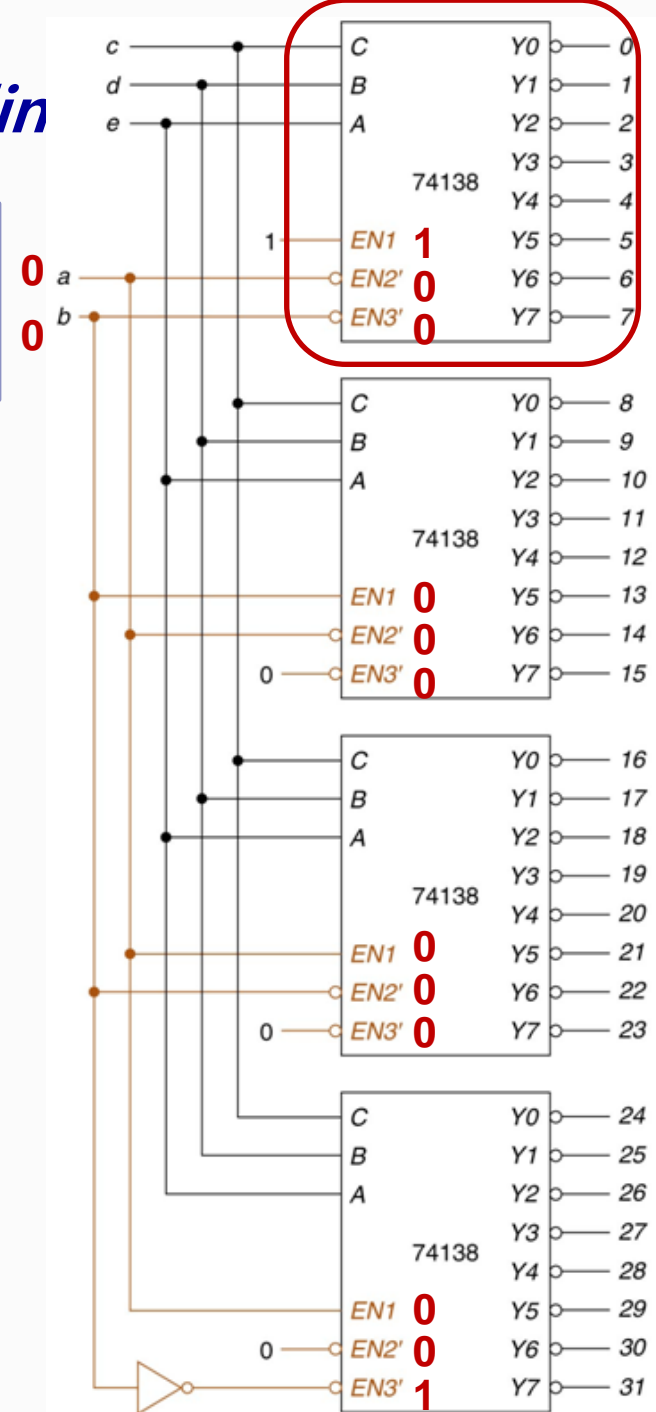


Decoder의 응용 - *addr decodin*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 동작 입력(enable신호)을 이용하여 디코드

예제 5.1 74138 디코더를 사용하여 **32개의 장치** 중 하나를 선택하기 위한 회로를 설계하라

a, b로 4개의 디코더 중 하나를 활성화 시킬 수 있음

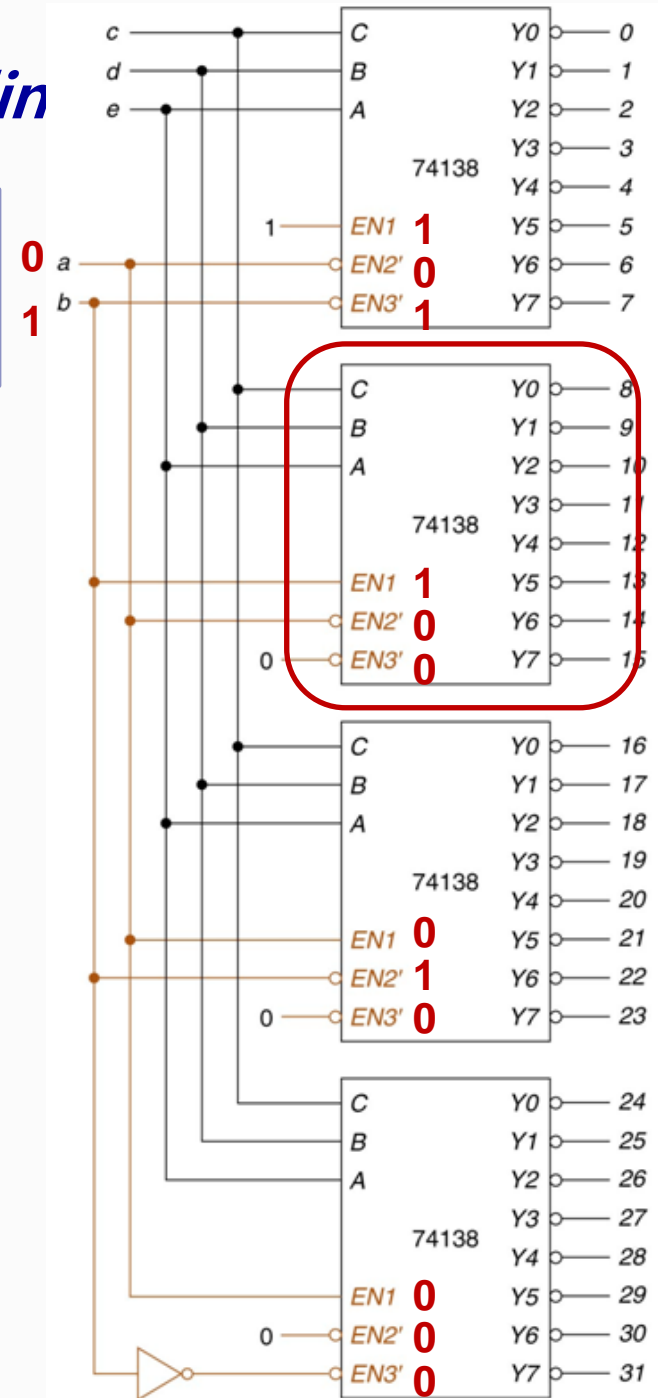


Decoder의 응용 - *addr decodin*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 동작 입력(enable신호)을 이용하여 디코드

예제 5.1 74138 디코더를 사용하여 **32개의 장치** 중 하나를 선택하기 위한 회로를 설계하라

a, b로 4개의 디코더 중 하나를 활성화 시킬 수 있음

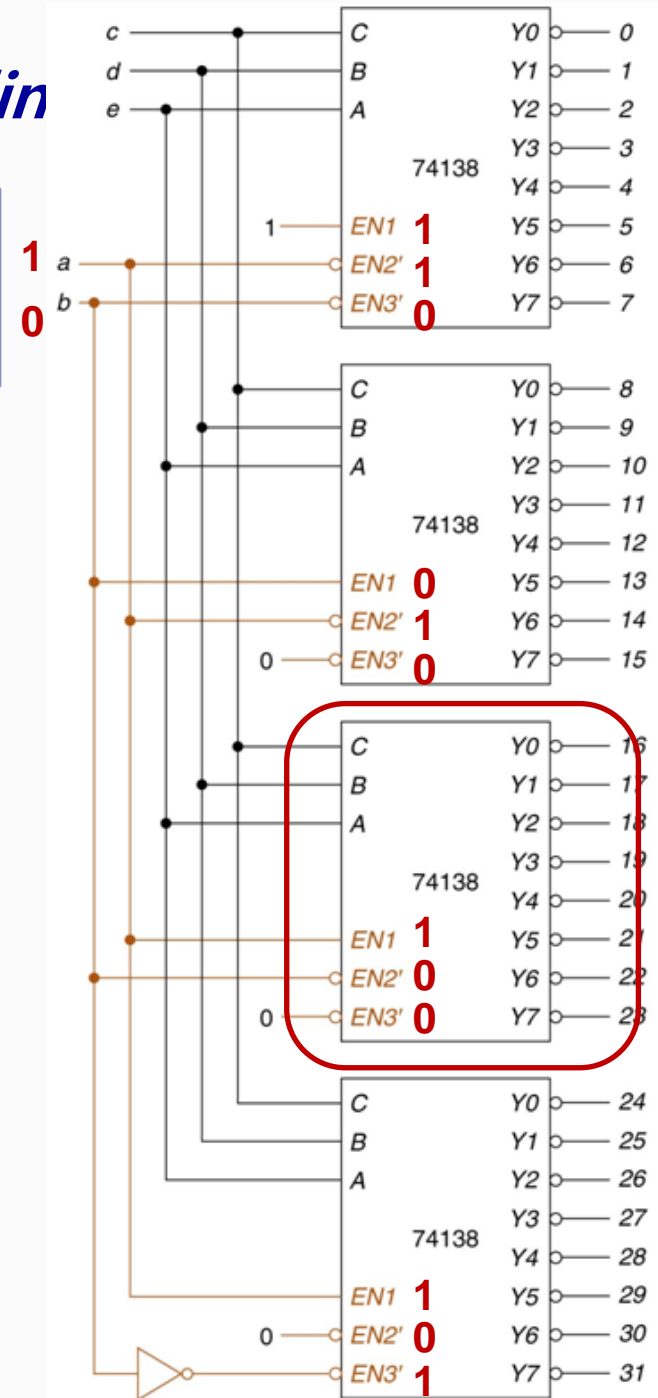


Decoder의 응용 - *addr decodin*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 동작 입력(enable신호)을 이용하여 디코드

예제 5.1 74138 디코더를 사용하여 **32개의 장치** 중 하나를 선택하기 위한 회로를 설계하라

a, b로 4개의 디코더 중 하나를 활성화 시킬 수 있음

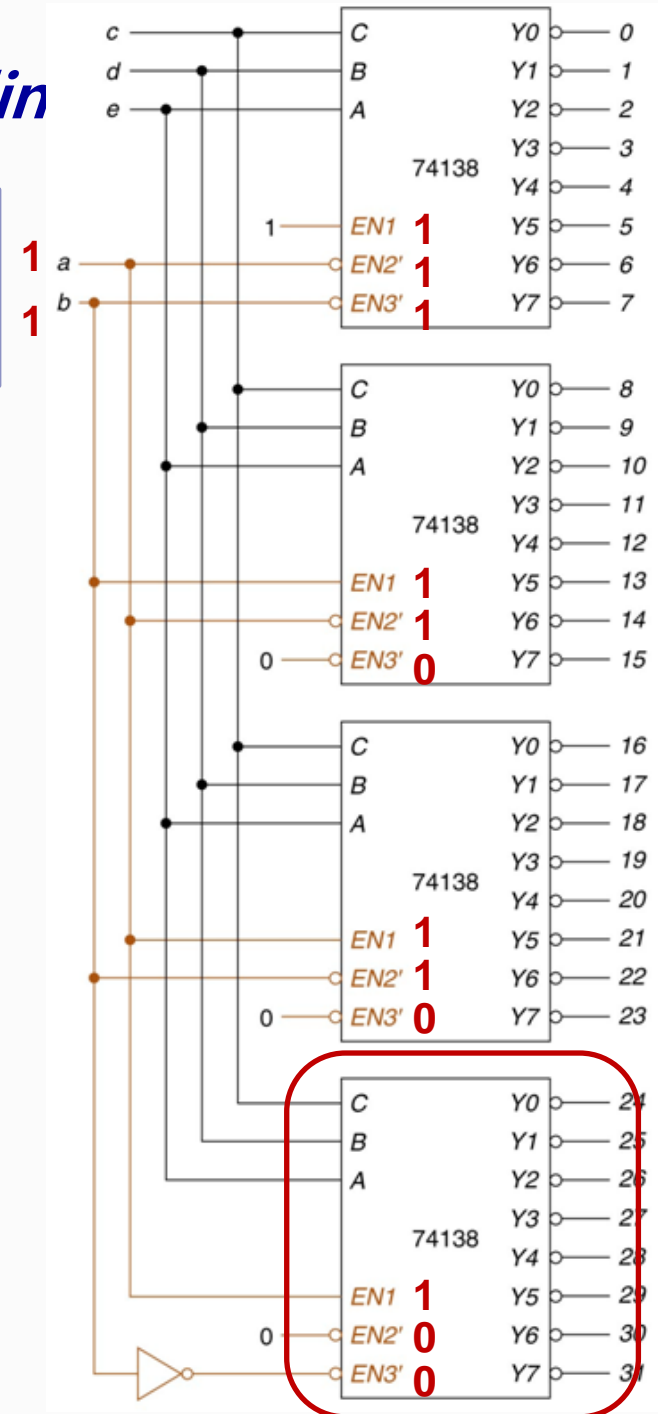


Decoder의 응용 - *addr decodin*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 동작 입력(enable신호)을 이용하여 디코드

예제 5.1 74138 디코더를 사용하여 **32개의 장치** 중 하나를 선택하기 위한 회로를 설계하라

a, b로 4개의 디코더 중 하나를 활성화 시킬 수 있음



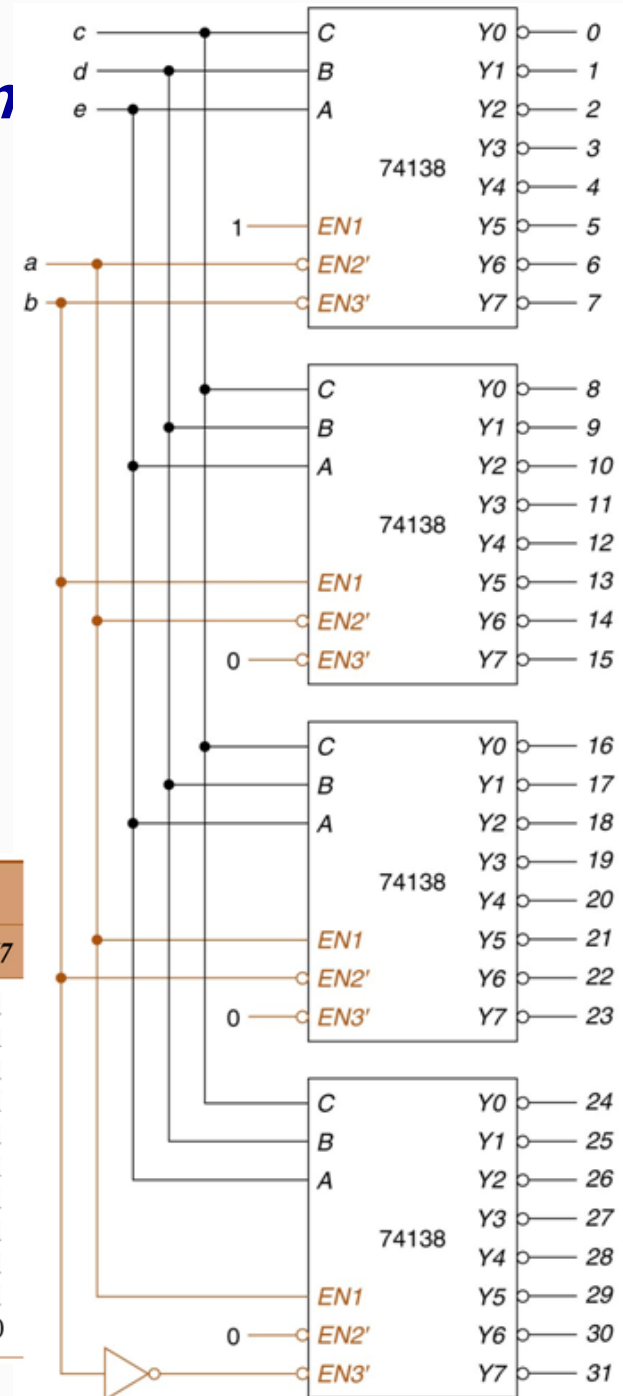
Decoder의 응용 - *addr decodin*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 동작 입력(enable신호)을 이용하여 디코드

예제 5.1 74138 디코더를 사용하여 **32개의 장치** 중 하나를 선택하기 위한 회로를 설계하라
c, d, e로 활성화된 칩의 8가지 출력 중 하나를 active 하게 만들 수 있음

Enables			Inputs			Outputs							
<i>EN1</i>	<i>EN2'</i>	<i>EN3'</i>	<i>C</i>	<i>B</i>	<i>A</i>	<i>Y0</i>	<i>Y1</i>	<i>Y2</i>	<i>Y3</i>	<i>Y4</i>	<i>Y5</i>	<i>Y6</i>	<i>Y7</i>
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

1 0 0



Decoder의 응용 - *addr decoding*

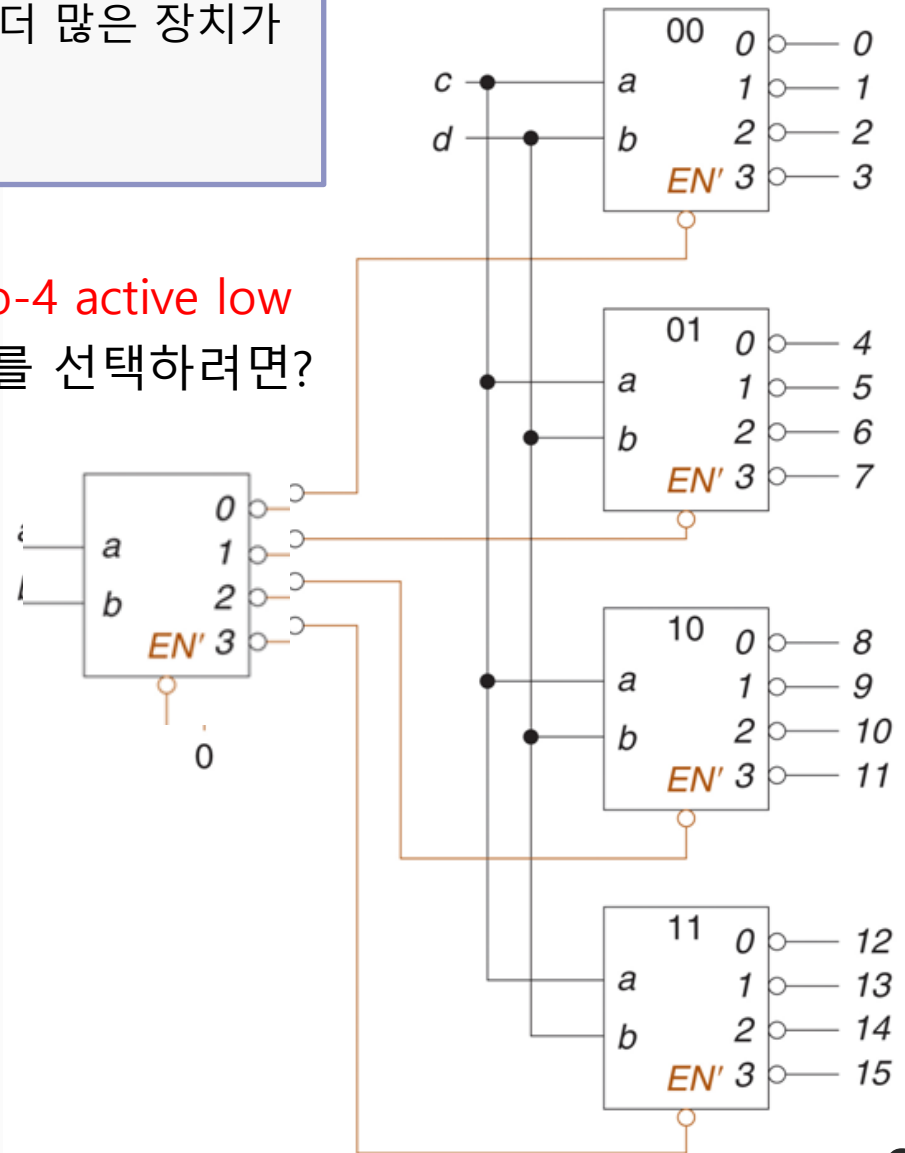
- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 추가적인 디코더를 사용

예제 5.2 Active low En을 갖는 2-to-4 active low 디코더가 있다. 16개 장치 중 하나를 선택하려면?

16개 출력?

4-to-16 decoder가
필요하겠군..

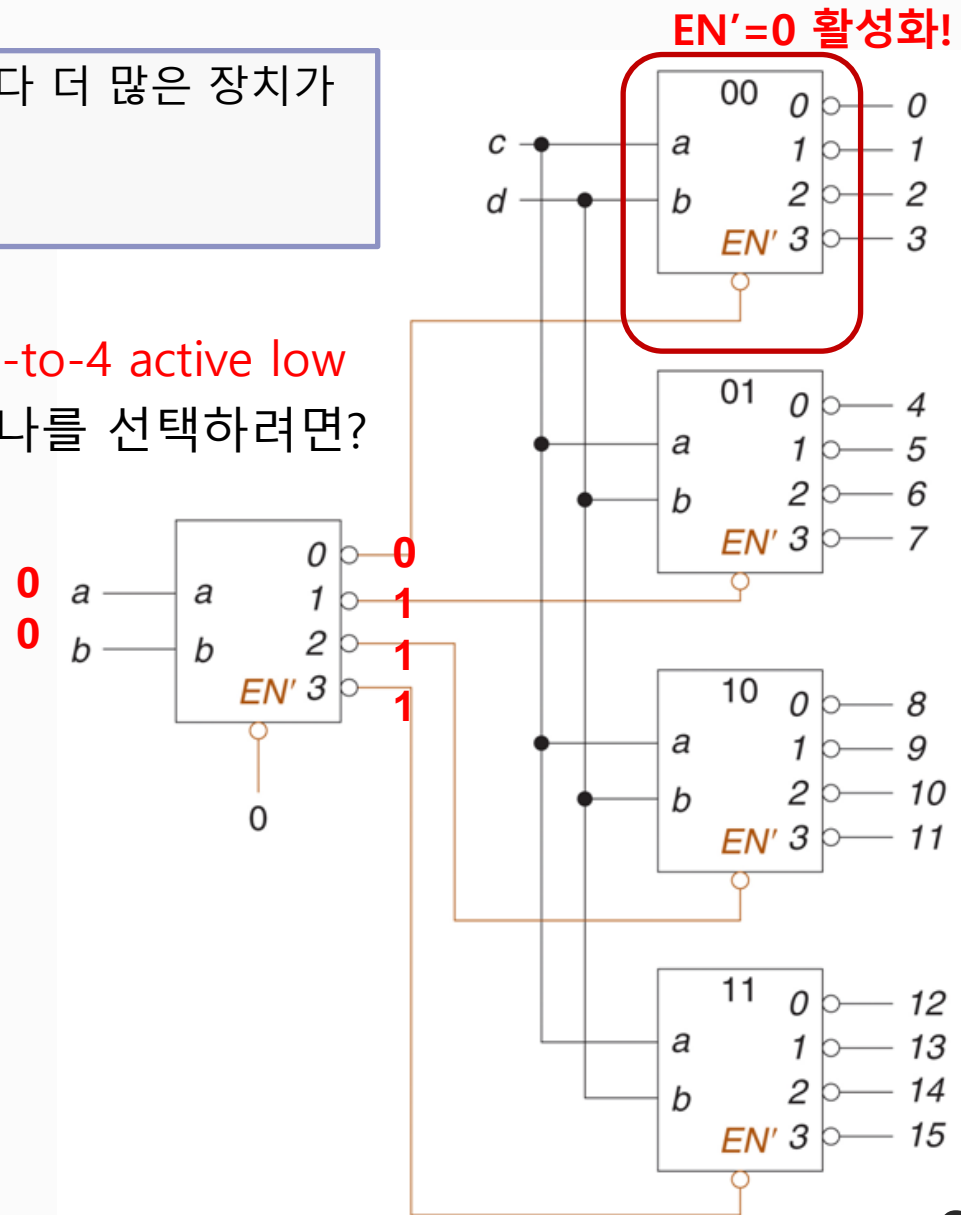
또 없어요???



Decoder의 응용 - *addr decoding*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 추가적인 디코더를 사용

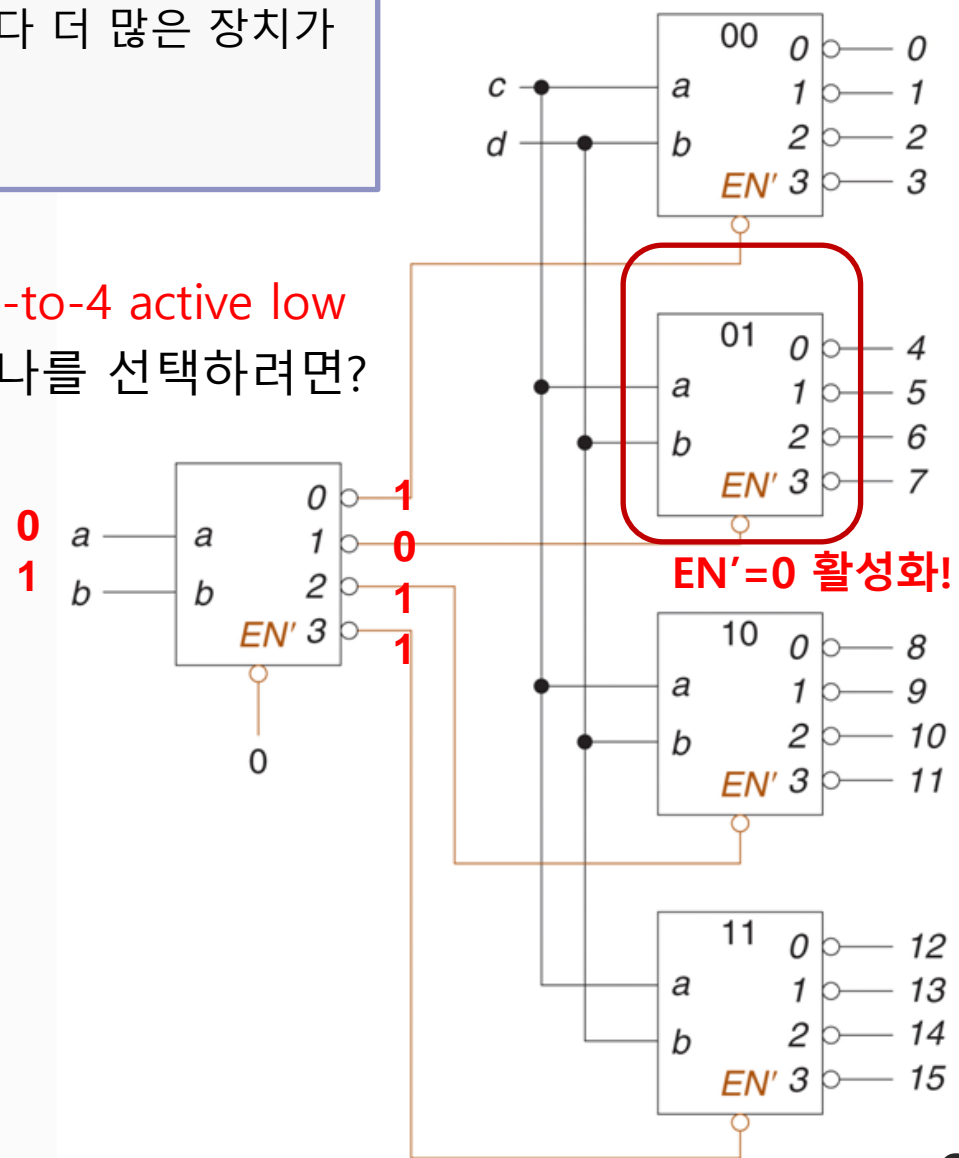
예제 5.2 Active low En을 갖는 2-to-4 active low 디코더가 있다. 16개 장치 중 하나를 선택하려면?



Decoder의 응용 - *addr decoding*

- 하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
 - 추가적인 디코더를 사용

예제 5.2 Active low En을 갖는 2-to-4 active low 디코더가 있다. 16개 장치 중 하나를 선택하려면?



- Decoder를 통해서 address을 주면 장치를 선택할 수 있다.

- 주어진 Decoder에서..

하나의 디코더로 선택할 수 있는 것보다 더 많은 장치가 있는 경우
→ **Chip select(CS)**로 구현

- 1) Enable 신호 활용 : 동작 입력(enable신호)을 이용하여
Chip Select 하기

- 2) Chip select하기 위한 추가적인 decoder 사용하기

Decoder를 어디에 사용해 볼까?

- 1) Address decoding : 장치 선택
- 2) 논리함수 구현하기

Decoder의 응용 - 논리 함수 구현

- 디코더 각각의 **active high** 출력은 그 함수의 **minterm**과 일치
=> 논리함수 구현에 필요한 것은 해당된 출력들을 연결할 OR 게이트

Decoder의 응용 - 논리 함수 구현

- 2-입력(4-출력) active high 출력 디코더
- 각 출력은 2변수 함수의 minterm들과 일치

출력0 : $a'b'$

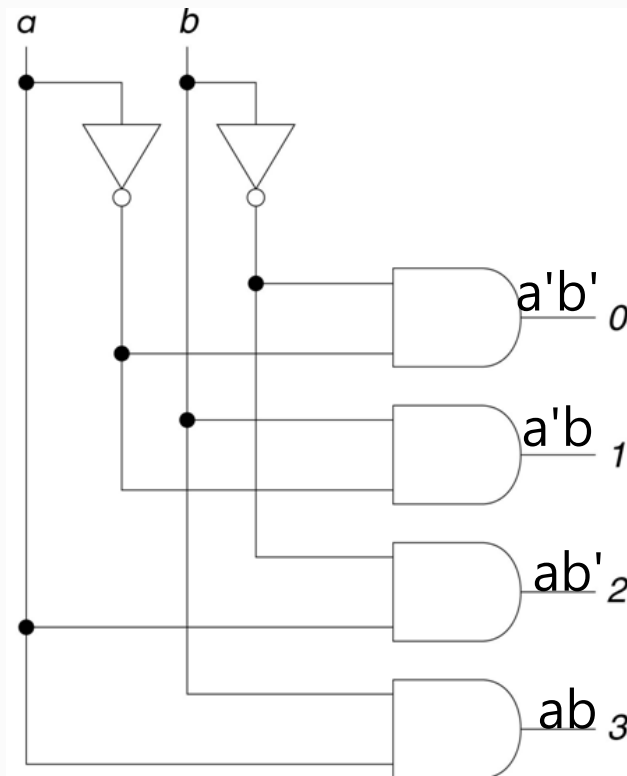
출력1 : $a'b$

출력2 : ab'

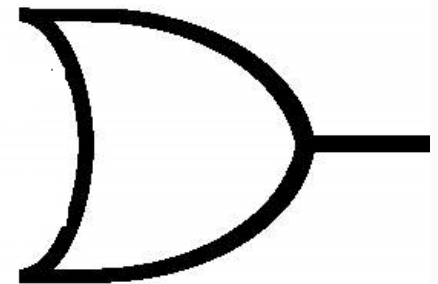
출력3 : ab

<i>a</i>	<i>b</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

active high 디코더



넷 중 하나라도
1이면 출력 1



Decoder의 응용 - 논리 함수 구현

- 디코더 각각의 **active high** 출력은 그 함수의 **minterm**과 일치
=> 논리함수 구현에 필요한 것은 해당된 출력들을 연결할 OR 게이트
- **active low** 출력 디코더의 경우는 OR 게이트를 NAND 게이트로 변경
=> AND-OR 회로에서 **NAND-NAND 회로로 변경**
$$f = ab + cd = \overline{\overline{ab + cd}} = \overline{\overline{ab} \cdot \overline{cd}}$$

Decoder의 응용 - 논리 함수 구현

- active low 출력 형식의 디코더 회로와 진리표

출력0 : $(a'b')'$

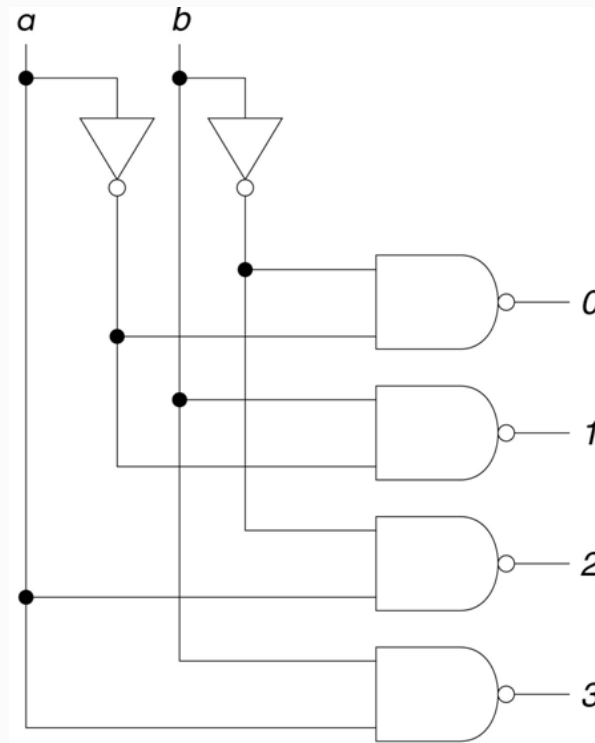
출력1 : $(a'b)$

출력2 : (ab')

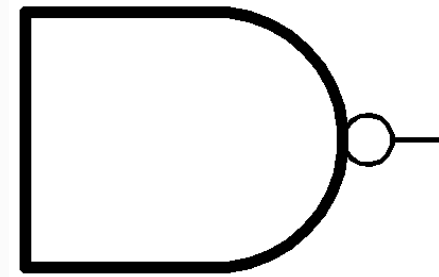
출력3 : (ab)

<i>a</i>	<i>b</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

active low 디코더



넷 중 하나라도
0이라면 출력 1



Decoder의 응용 - 논리 함수 구현

- 디코더 각각의 **active high** 출력은 그 함수의 **minterm**과 일치
=> 논리함수 구현에 필요한 것은 해당된 출력들을 연결할 OR 게이트
- **active low** 출력 디코더의 경우는 OR 게이트를 NAND 게이트로 변경
=> AND-OR 회로에서 **NAND-NAND 회로로 변경**
$$f = ab + cd = \overline{\overline{ab + cd}} = \overline{\overline{ab} \cdot \overline{cd}}$$
- 같은 입력들로 구성되는 **둘 이상의 함수에 대해서도 하나의 디코더만 필요**
=> 각각의 출력 함수에 대해서는 하나의 OR 또는 NAND가 필요

Decoder의 응용 - 논리 함수 구현

예제 5.3 논리함수 구현 - 다중 출력 구현

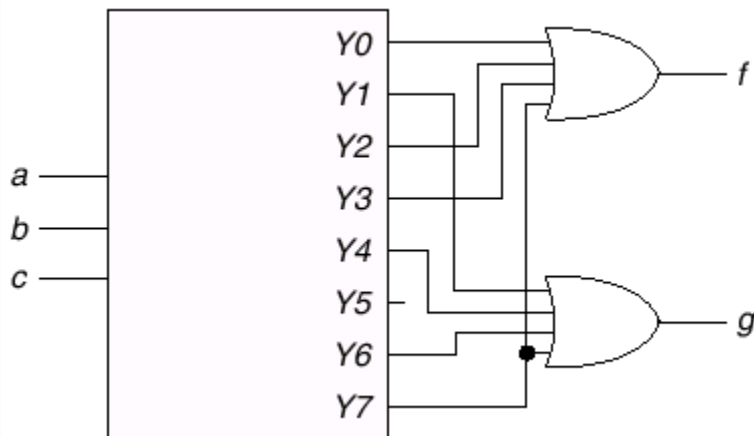
$$f(a, b, c) = \sum m(0, 2, 3, 7)$$

$$g(a, b, c) = \sum m(1, 4, 6, 7)$$

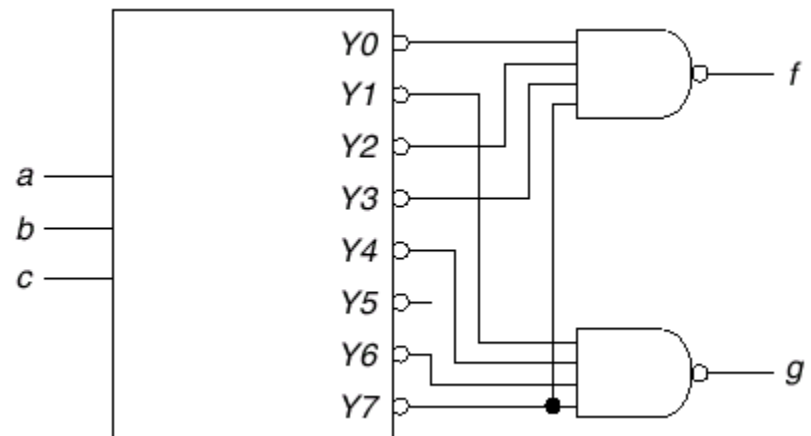
함수를 **디코더를 사용하여** 설계하라.

- 위 함수들은 아래에 보이는 디코더 회로 중 하나로 구현 가능.

Active high decoder



Active low decoder



Decoder의 응용 - 논리 함수 구현

예제 5.4 다음 함수에 대하여, 아래의 4개의 디코더와 3개의 NAND gate (4 입력)로만 설계.

$a \ b$		$c \ d$			
		00	01	11	10
$c \ d$	00	1	1		1
	01				
	11	1	1	1	1
	10	1			1

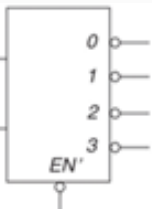
f

$a \ b$		g			
		00	01	11	10
$c \ d$	00	1		1	1
	01	1	1	1	1
	11			1	
	10	1			1

$a \ b$		h			
		00	01	11	10
$c \ d$	00		1	1	1
	01				
	11			1	
	10	1	1	1	1

f, g, h 에 대하여 9개, 10개, 8개의 minterm에 대하여 1

Active
Low
디코더



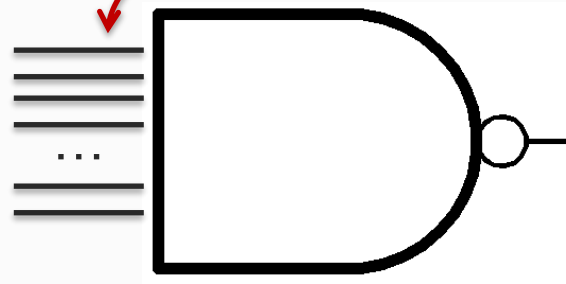
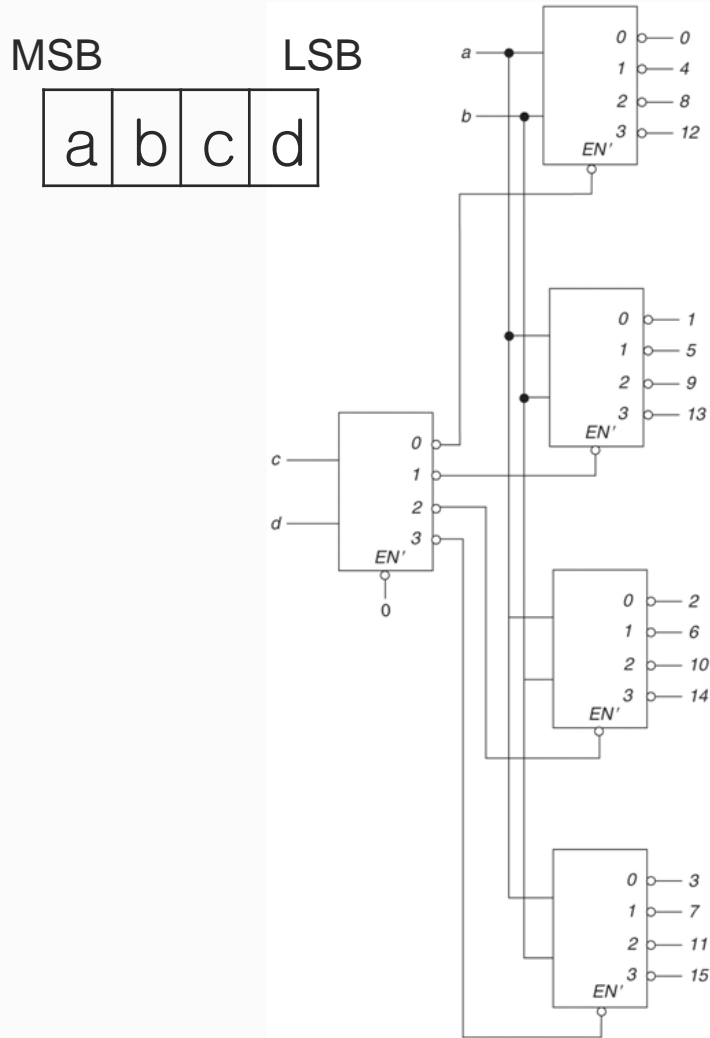
En'	a	b	0	1	2	3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

K-map에서 16개의 cell 중
1이 표시되어 있는 cell 들을
선택하기 위해
디코더를 사용
2-to-4 decoder 4개로
4-to-16 decoder 구현하는 문제

Decoder의 응용 - 논리 함수 구현

예제 5.4 다음 함수에 대하여, 아래의 4개의 디코더와 3개의 NAND gate (4 입력)로만 설계.

예제 5.2 방식으로 단순하게 먼저 시작해 보자. **F는 9개, g는 10개, h는 8개의 NAND input 필요**



$c d$		$a b$			
		00	01	11	10
$c d$	00	1	1		1
	01				
	11	1	1	1	1
	10	1			1

f

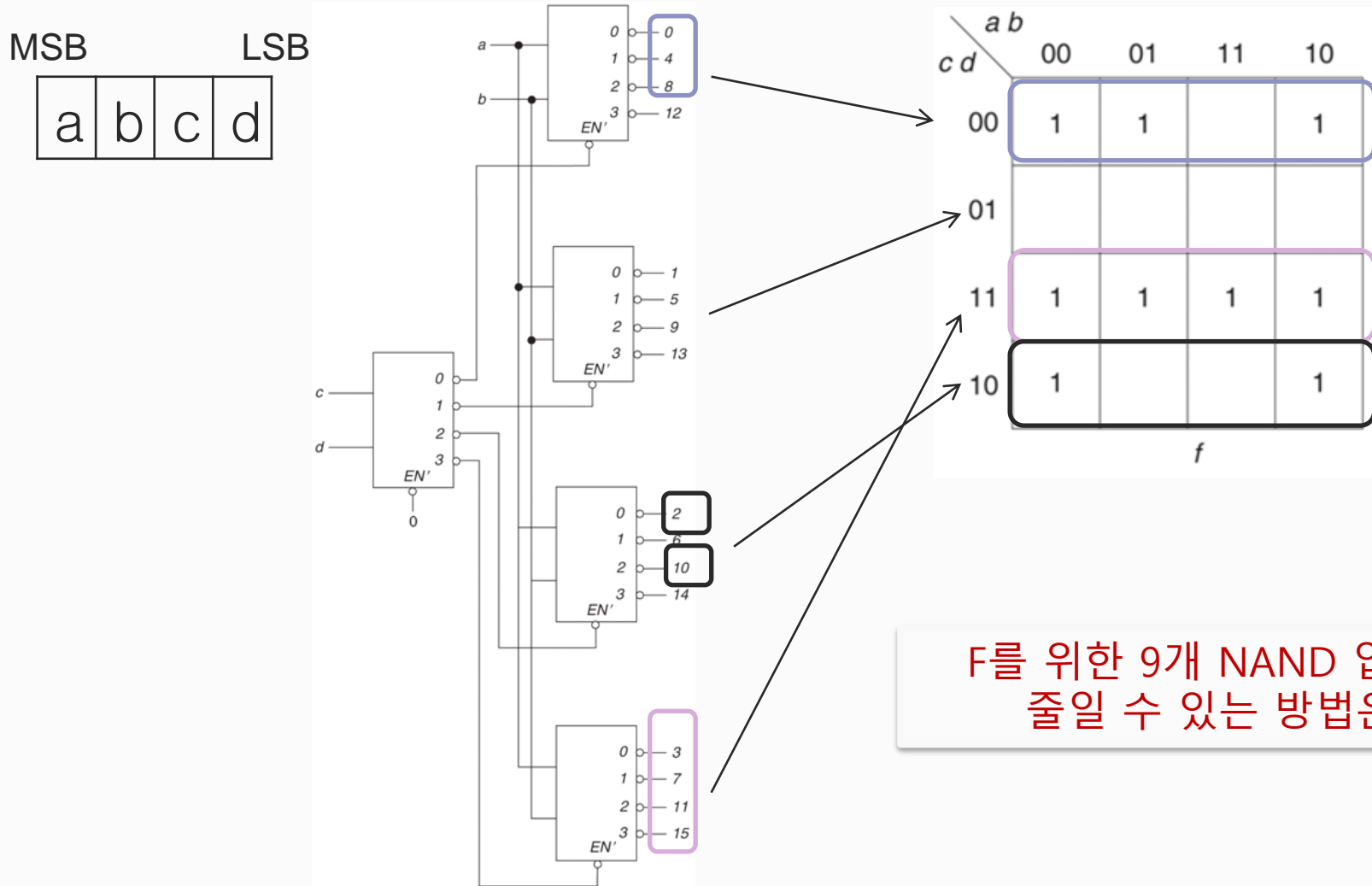
$a \backslash b$		$a \ b$			
		00	01	11	10
$c \ d$	00	1		1	1
	01	1	1	1	1
	11			1	
	10	1			1
		a			

$c d \backslash a b$		h			
		00	01	11	10
$c d$	00		1	1	1
	01				
	11			1	
	10	1	1	1	1

Decoder의 응용 - 논리 함수 구현

예제 5.4 다음 함수에 대하여, 아래의 4개의 디코더와 3개의 NAND gate (4 입력)로만 설계.

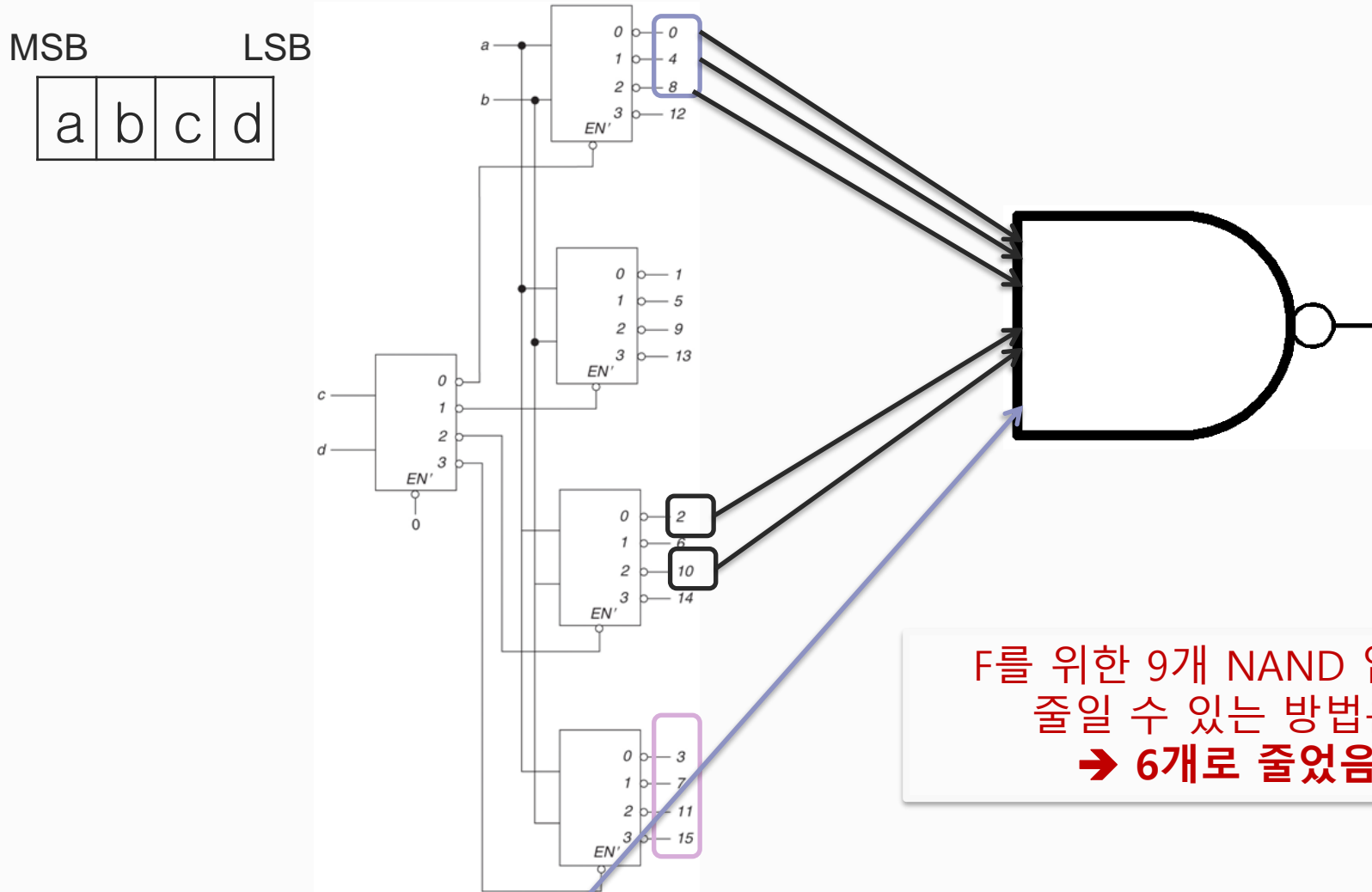
예제 5.2 방식으로 단순하게 먼저 시작해 보자. F는 9개, g는 10개, h는 8개의 NAND input 필요



Decoder의 응용 - 논리 함수 구현

예제 5.4 다음 함수에 대하여, 아래의 4개의 디코더와 3개의 NAND gate (4 입력)로만 설계.

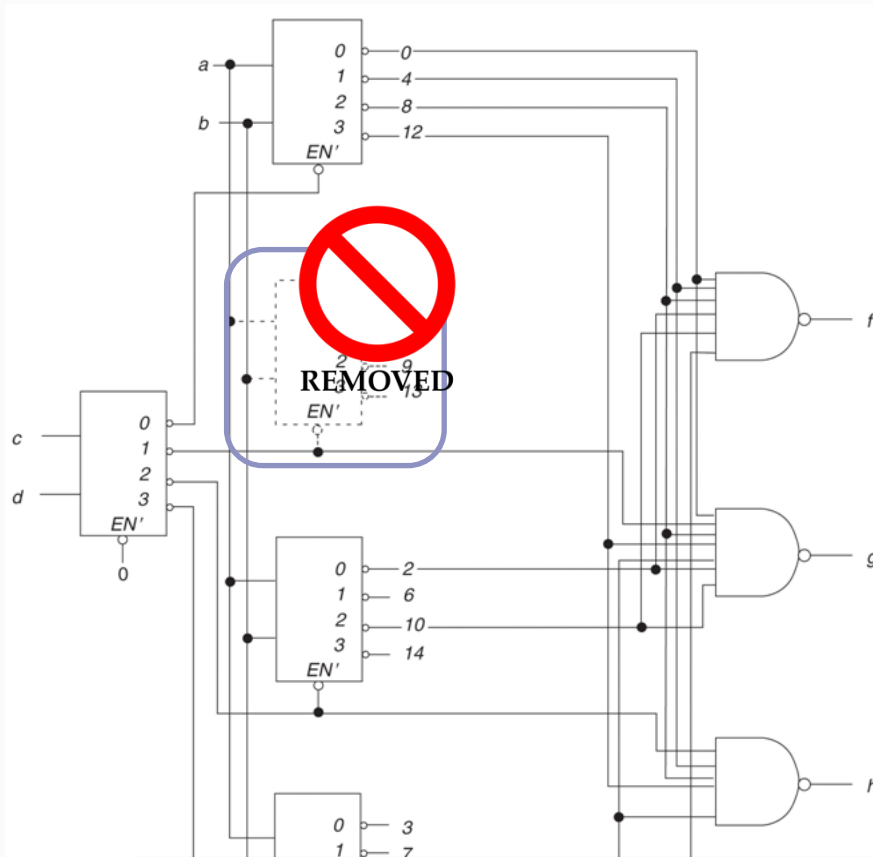
예제 5.2 방식으로 단순하게 먼저 시작해 보자. F는 9개, g는 10개, h는 8개의 NAND input 필요



Decoder의 응용 - 논리 함수 구현

예제 5.4 다음 함수에 대하여, 아래의 4개의 디코더와 3개의 **NAND gate (4 입력)**로만 설계.

이제 F는 6개, g는 7개, h는 5개의 NAND 입력 필요



0,4,8,2,10,
3,7,11,5,

0,1,5,9,13,8,12,
15,2,10

12,6,10,14,
4,8,12,16

a b		00	01	11	10
c d	00	1	1		1
	01				
	11	1	1	1	1
	10	1			1

f

a b		00	01	11	10
c d	00	1		1	1
	01	1	1	1	1
	11			1	
	10	1			1

g

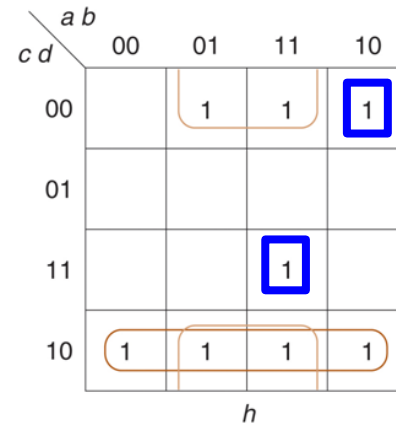
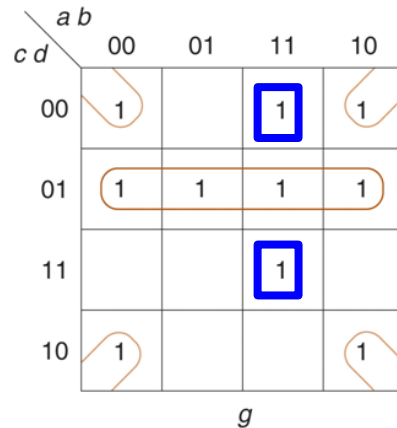
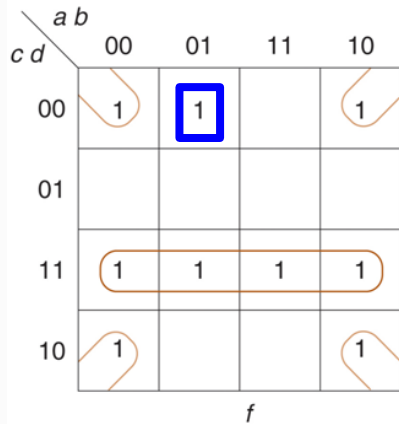
a b		00	01	11	10
c d	00		1	1	1
	01				
	11			1	
	10	1	1	1	1

h

그러나..
4개 NAND 입력을 만들려면 또
다른 최적화(Simplification) 필요.

Decoder의 응용 - 논리 함수 구현

예제 5.4 다음 함수에 대하여, 아래의 4개의 디코더와 3개의 **NAND gate (4 입력)**로만 설계.



CD=11

CD=01

CD=10

BD=00

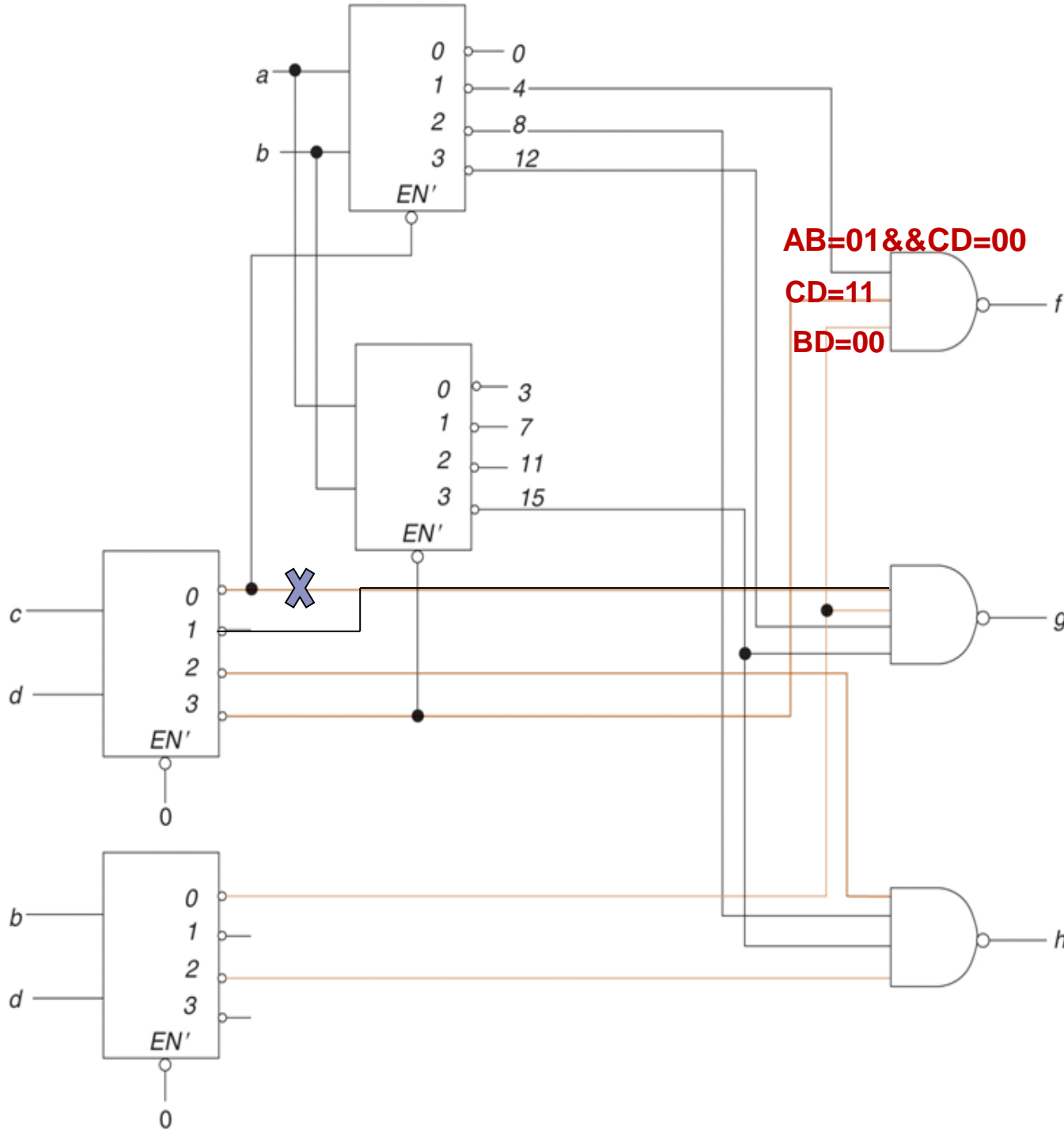
BD=00

BD=10

AB값은 상관 없음

AC값은 상관 없음

남은 1들은 $CD=00$ or $CD=11$ 일 때 다양한 AB값들을 가짐



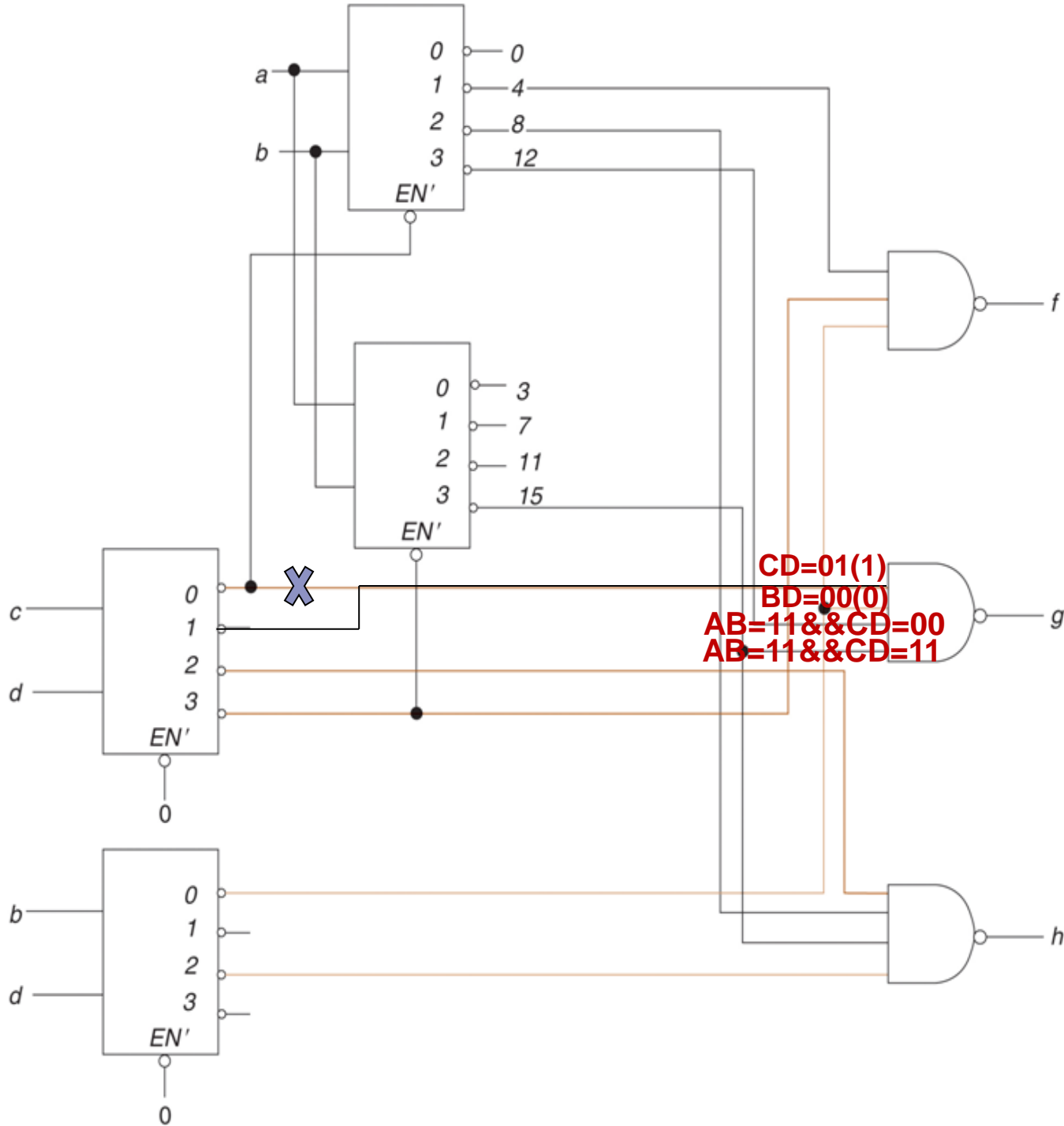
		$a\ b$			
$c\ d$		00	01	11	10
	00	1	1		1
	01				
	11	1	1	1	1
	10	1			1

f

CD=11 (3)

BD=00 (0)

AB=01 && CD=00



		$a\ b$			
$c\ d$		00	01	11	10
	00	1		1	1
	01	1	1	1	1
	11			1	
	10	1			1

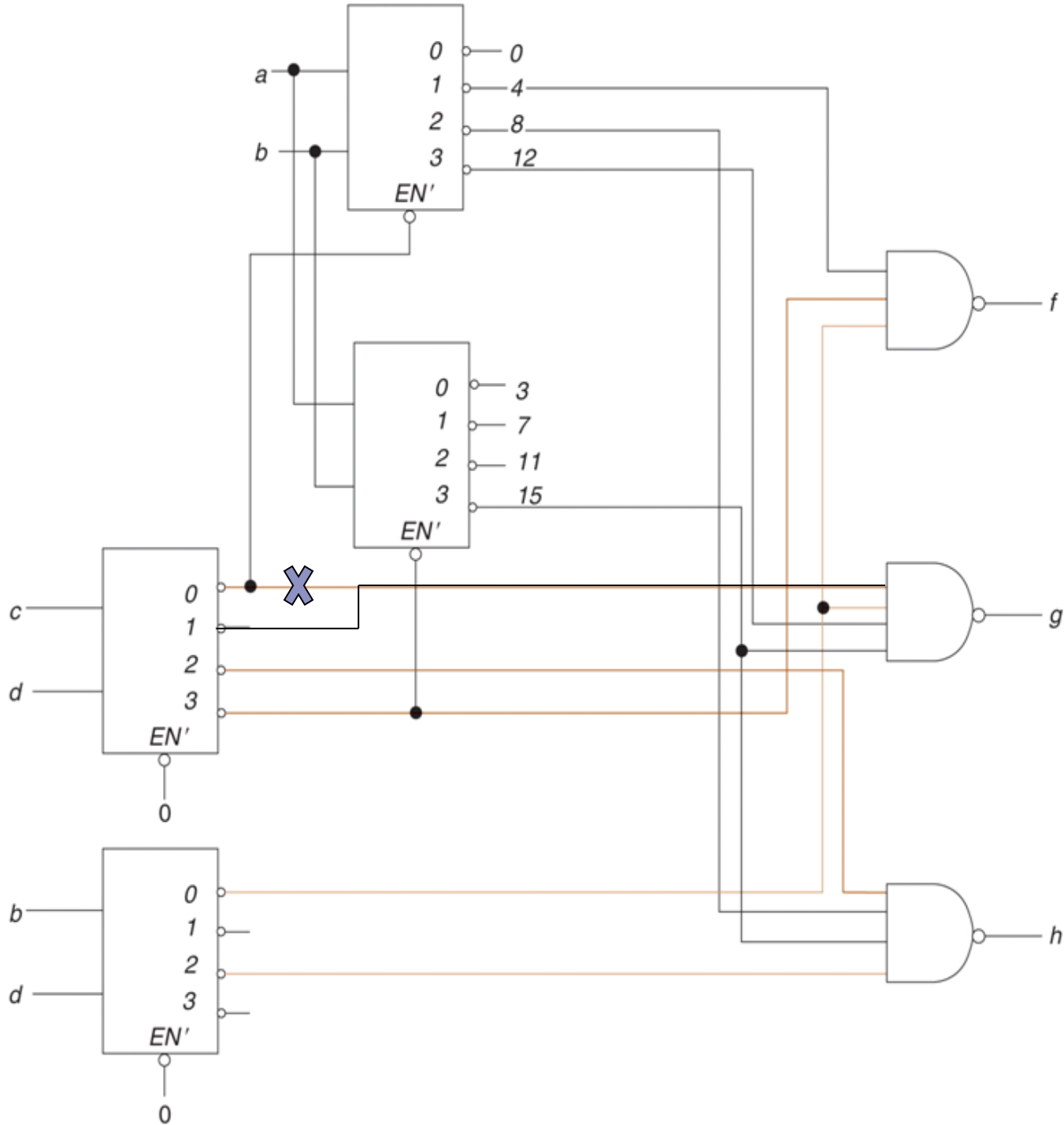
g

$CD=01(1)$

$BD=00(0)$

$AB=11 \& \& CD=00$

$AB=11 \& \& CD=11$



$AB=10 \& \& CD=00$

$AB=11 \& \& CD=11$

		a b			
c d		00	01	11	10
00			1	1	1
01					
11				1	
10		1	1	1	1

h

$CD=10(2)$

$BD=10(2)$

•논리 함수 (Truth table, switching algebra, K-map)가 주어졌을 때 구현하기

1) Chapter 2에서 배운 방법 : and-or 로 two-level 로직 구현

2) 오늘 배운 방법 : decoder 로 구현

- decoder는 모든 minterm들을 출력으로 갖고 있음. 따라서 원하는 minterm, 즉 논리함수에서 1로 원하는 minterm들만 선택해서 OR 또는 NAND로 묶기만 하면 됨

- 리소스에 제약이 있는 경우?
K-map으로 간략화해서 구현하기

Chapter 5 Designing Combinational Systems

5.1 Iterative (반복) 시스템 ex) ALUs

5.2 Binary Decoders

5.3 Encoders and Priority Encoders

5.4 Multiplexer (Mux) & Demultiplexers (Demux)

5.5 Tree-State Gates

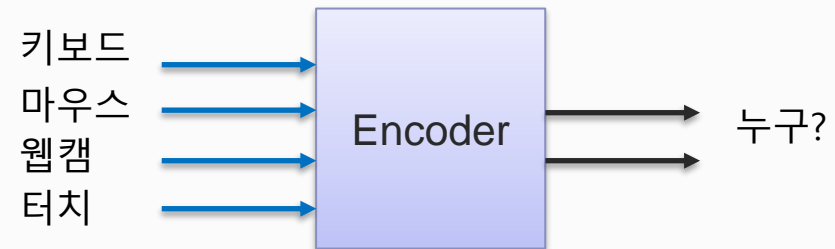
5.6 Gate Arrays – ROM, PLA, PALs..

Decoder와 Encoder

- 디코더는 **코딩된 입력 신호에** 의해 여러 출력 선(장치) 중 하나를 선택하는 장치
- 인코더는 **여러 입력 선(입력 장치) 중 하나가 신호를 보낼 때 장치 명을 코딩해서 출력**

EX)

A_0	A_1	A_2	A_3	Z_0	Z_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



- 식


$$Z_0 = A_2 + A_3$$

$$Z_1 = A_1 + A_3$$

단 하나의 입력만 1이 된다는
가정하에 성립 가능

Encoder

장치00과 어떤 입력도 없는 상태를 구별할 필요가 있을 때..



A_0	A_1	A_2	A_3	Z_0	Z_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

• 식

$$Z_0 = A_2 + A_3$$

$$Z_1 = A_1 + A_3$$

A_0	A_1	A_2	A_3	Z_0	Z_1	N
0	0	0	0	0	0	1
1	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	1	0	1	0	0
0	0	0	1	1	1	0

• 식

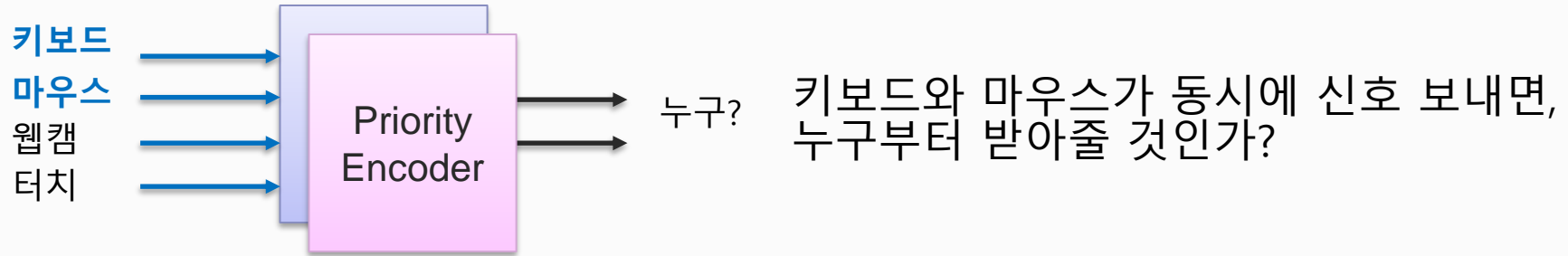
$$Z_0 = A_2 + A_3$$

$$Z_1 = A_1 + A_3$$

$$N = A'_0 A'_1 A'_2 A'_3$$

$$= (A_0 + A_1 + A_2 + A_3)'$$

Priority Encoder (우선 순위 인코더)



- 하나 이상의 입력이 동시에 일어날 수 있다면, 우선 순위가 필요
=> 우선순위가 가장 높은 장치 번호에 대한 code 출력
- 우선순위는 일반적으로 내림차순(또는 오름차순)으로 정렬되어 최우선 순위에 가장 큰(또는 가장 작은) 입력 번호가 주어짐

8-input Priority Encoder

우선 순위가 점점 높아짐

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Z_0	Z_1	Z_2	NR
0	0	0	0	0	0	0	0	X	X	X	1
X	X	X	X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

$$NR = A_0' A_1' A_2' A_3' A_4' A_5' A_6' A_7'$$

$$Z_0 = A_4 + A_5 + A_6 + A_7$$

$$Z_1 = A_6 + A_7 + (A_2 + A_3) A_4' A_5'$$

$$Z_2 = A_7 + A_5 A_6' + A_3 A_4' A_6' + A_1 A_2' A_4' A_6'$$

8-input Priority Encoder

우선 순위가 점점 높아짐

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Z_0	Z_1	Z_2	NR
0	0	0	0	0	0	0	0	X	X	X	1
X	X	X	X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

$$NR = A'_0 A'_1 A'_2 A'_3 A'_4 A'_5 A'_6 A'_7$$

$$Z_0 = A_4 + A_5 + A_6 + A_7$$

$$Z_1 = A_6 + A_7 + (A_2 + A_3)A'_4 A'_5$$

$$Z_2 = A_7 + A_5 A'_6 + A_3 A'_4 A'_6 + A_1 A'_2 A'_4 A'_6$$

8-input Priority Encoder

우선 순위가 점점 높아짐

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Z_0	Z_1	Z_2	NR
0	0	0	0	0	0	0	0	X	X	X	1
X	X	X	X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

$$NR = A'_0 A'_1 A'_2 A'_3 A'_4 A'_5 A'_6 A'_7$$

$$Z_0 = A_4 + A_5 + A_6 + A_7$$

$$Z_1 = A_6 + A_7 + (A_2 + A_3)A'_4 A'_5$$

$$Z_2 = A_7 + A_5 A'_6 + A_3 A'_4 A'_6 + A_1 A'_2 A'_4 A'_6$$

8-input Priority Encoder

우선 순위가 점점 높아짐

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Z_0	Z_1	Z_2	NR
0	0	0	0	0	0	0	0	X	X	X	1
X	X	X	X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

$$NR = A'_0 A'_1 A'_2 A'_3 A'_4 A'_5 A'_6 A'_7$$

$$Z_0 = A_4 + A_5 + A_6 + A_7$$

$$Z_1 = A_6 + A_7 + (A_2 + A_3)A'_4 A'_5$$

$$Z_2 = A_7 + A_5 A'_6 + A_3 A'_4 A'_6 + A_1 A'_2 A'_4 A'_6$$