Combinational Logic Design p.128

일반적으로 컴퓨터 설계에서 매우 유용하게 사용되는 조합회로 블록 들에 대해 알아본다.

Decoder , Encoder, Multiplexer

1

Rudimentary Logic Functions p.114 기본논리함수

- 값고정(value fixing)
- 전달(transferring)
- 보수화(inverting)
- 유효화(enabling).. 나중에
- 단일 변수 X에 대한 함수들
  - 값 고정(0, 1)
  - 전달(X)
  - 반전(X') : not gate

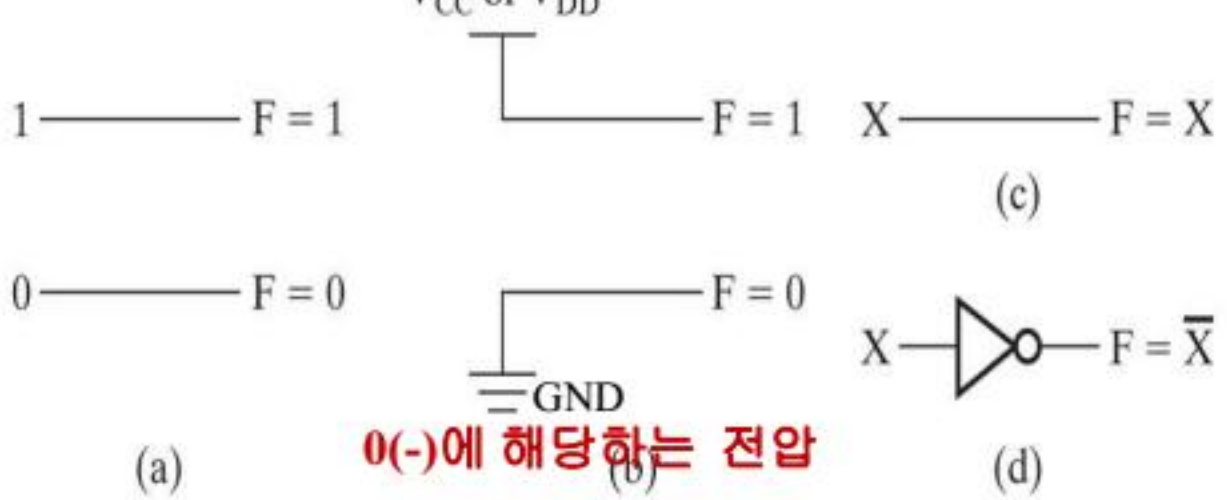
X	F = 0	F = X	F = $\bar{X}$	F = 1
0	0	0	1	1
1	0	1	0	1

2

p.115
(양의논리에서)

X	F = 0	F = X	F = $\bar{X}$	F = 1
0	0	0	1	1
1	0	1	0	1

1(+)에 해당하는 전압

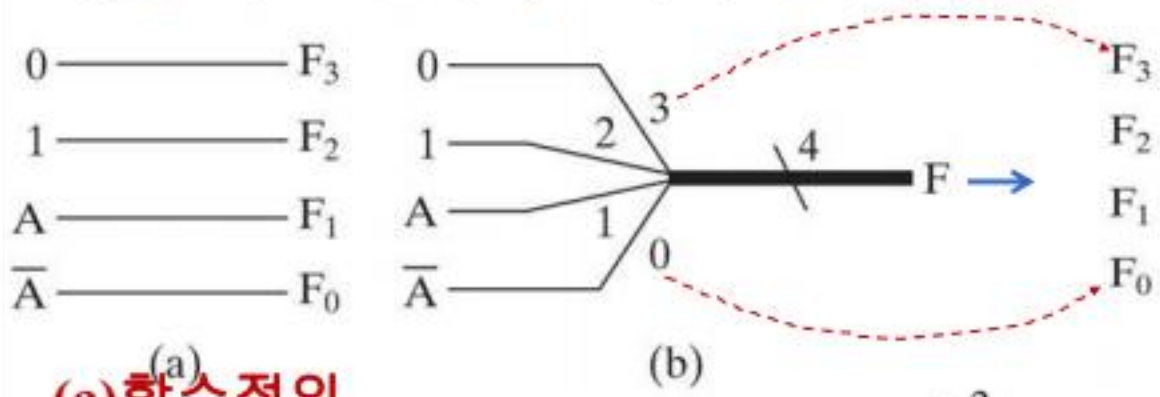


0(-)에 해당하는 전압

3

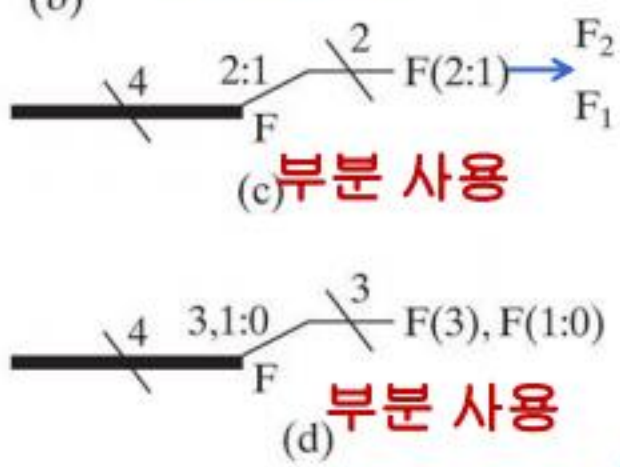
3

다중 비트 함수(도식화 방법) p.116



(a)함수정의

정의 후 (b)와 같이
도식화할 수 있다.



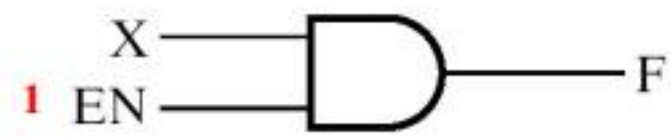
5

5

유효화(Enabling) p.118

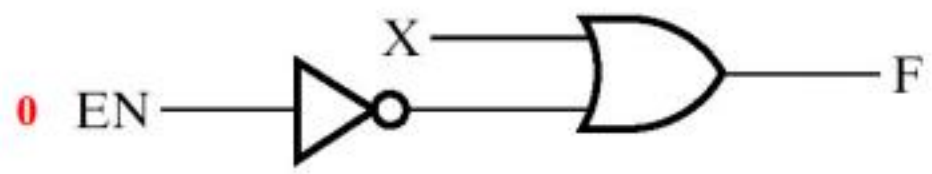
- Enabling : 입력이 출력으로 전달된다 는 의미
  - an input signal ➔ an output
- 그림 a에서 if EN =? , 유효한가?
- 그림 b에서 if EN =? , 유효한가?

유효화 예)



(a)

유효하지 않다 = x의 전달을 가로막는다.



(b)

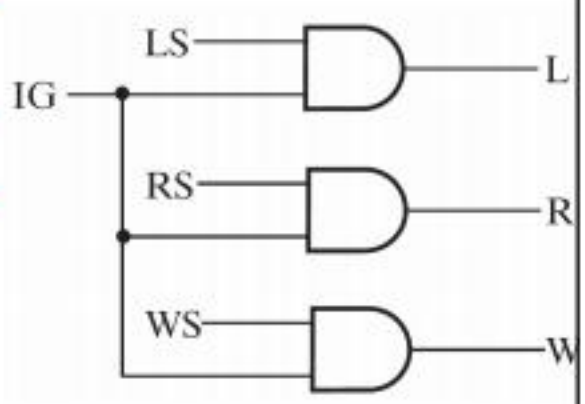
6

6

유효화(Enabling) p.118

- 스위치 유효화

Input Switches				Accessory Control		
IG	LS	RS	WS	L	R	W
0	X	X	X	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	1	0	0
1	1	0	1	1	0	1
1	1	1	0	1	1	0
1	1	1	1	1	1	1



7

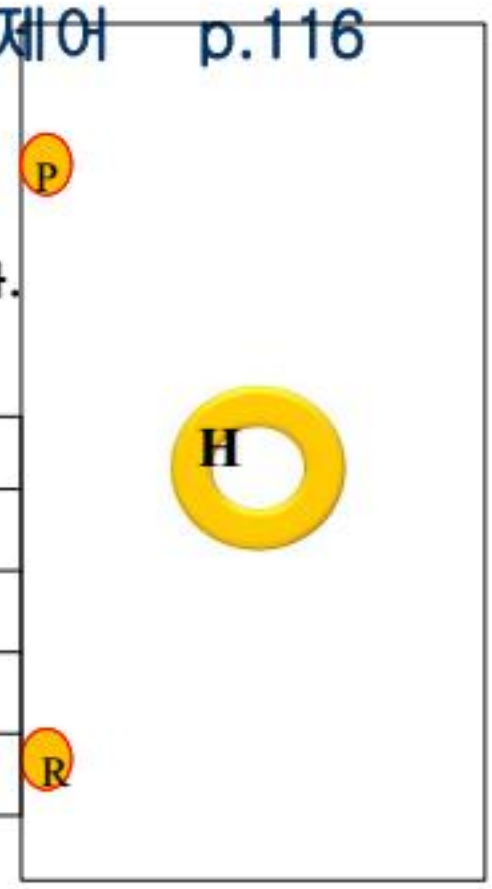
7

예)값고정을 사용한 조명제어 p.116

- M1 : P만이 조명을 켜고 끈다.
- M2 : R만이 조명을 켜고 끈다.
- M0 : P 또는 R이 조명을 켜고 끈다.

H(P,R) = ?

P	R	H(M0)	H(M1)	H(M2)
0	0			
0	1			
1	0			
1	1			



8

8

조명제어 함수의 기능 정의

Mode:		M_0	M_1	M_2
P	R	$H = \bar{P}R + P\bar{R}$	H = P	H = R
0	0	0	0	0
0	1	1	0	1
1	0	1	1	0
1	1	0	1	1

9

9

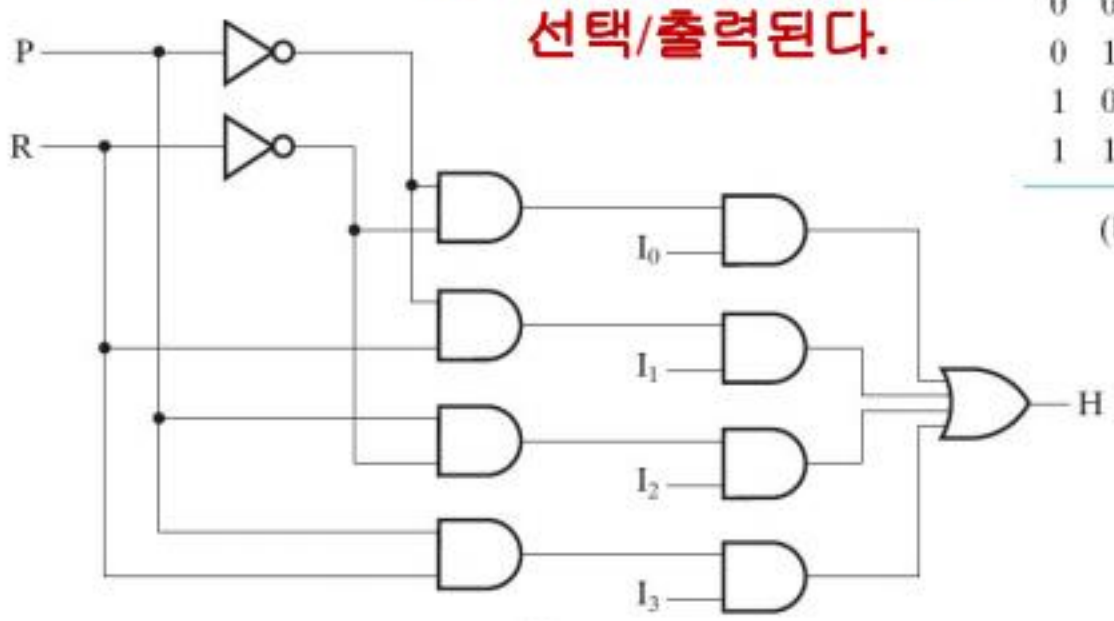
값 고정을 사용한 회로 (value fixing) p.117

- $H(P,R,I_0,I_1,I_2,I_3)=P'R'I_0+P'RI_1+PR'I_2+PRI_3$

P,R에 따라 고정값 I가 선택/출력된다.

P	R	H
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b)



(a)

10

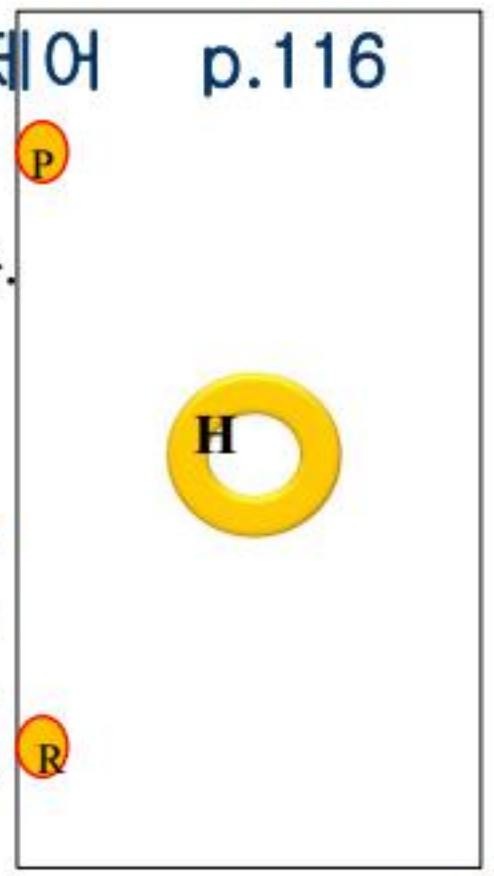
10

값고정을 사용한 조명제어 p.116

- M1 : P만이 조명을 켜고 끈다.
- M2 : R만이 조명을 켜고 끈다.
- M0 : P 또는 R이 조명을 켜고 끈다.

H(P,R) = ?

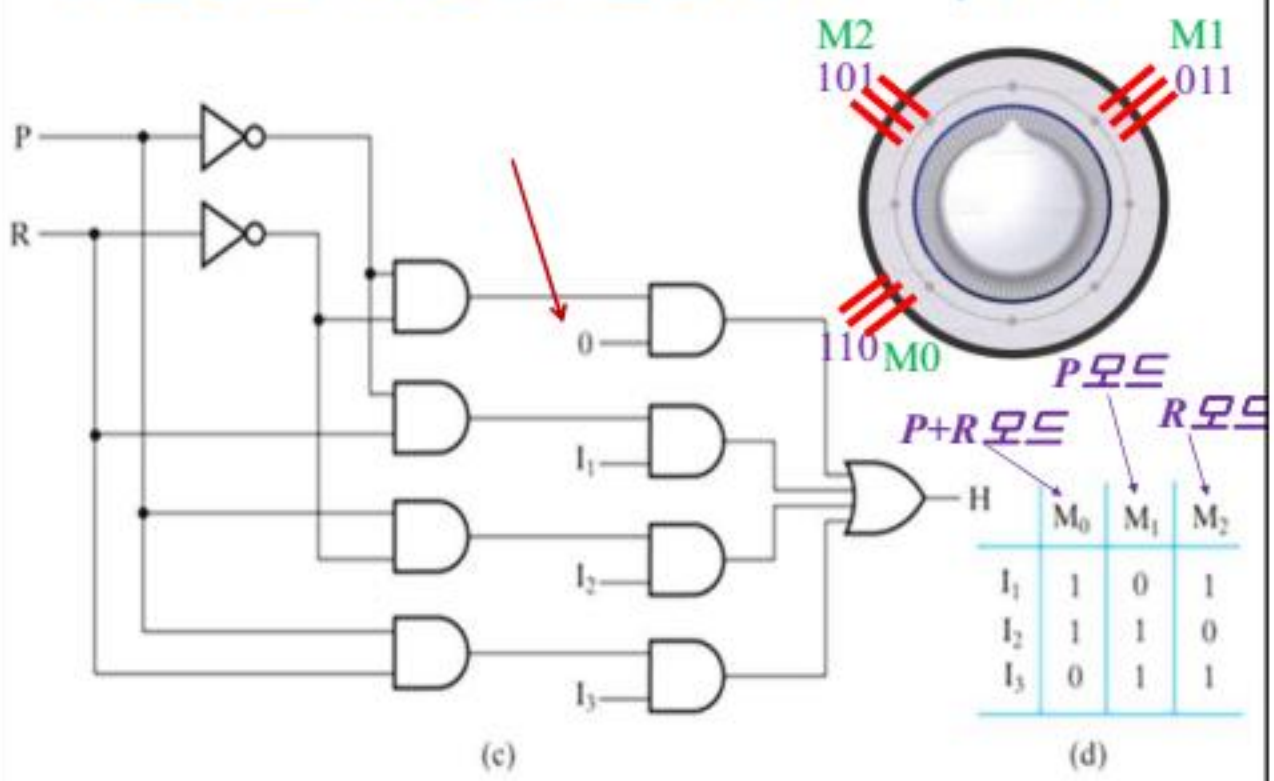
P	R	H(M0)	H(M1)	H(M2)	
0	0	0	0	0	I_0
0	1	1	0	1	I_1
1	0	1	1	0	I_2
1	1	0	1	1	I_3



11

11

값고정을 사용한 조명제어 p.117



	M_0	M_1	M_2
I_1	1	0	1
I_2	1	1	0
I_3	0	1	1

12

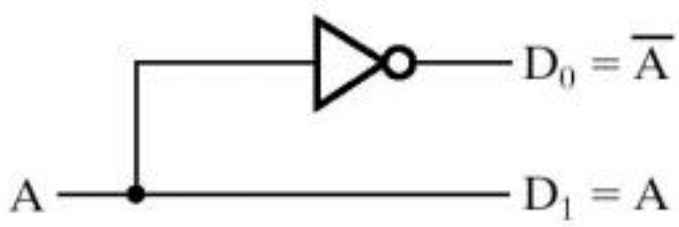
12

Decoding p.120

- decoder
  - n비트로 코딩된 2진 정보를 최대 2^n개의 출력으로 변환하는 조합회로
  - n개의 입력 ⇒ 2^n개의 출력
  - n-to-m라인 디코더($m < 2^n$)
- 1-to-2 decoder

A	D_0	D_1
0	1	0
1	0	1

(a)



(b)

13

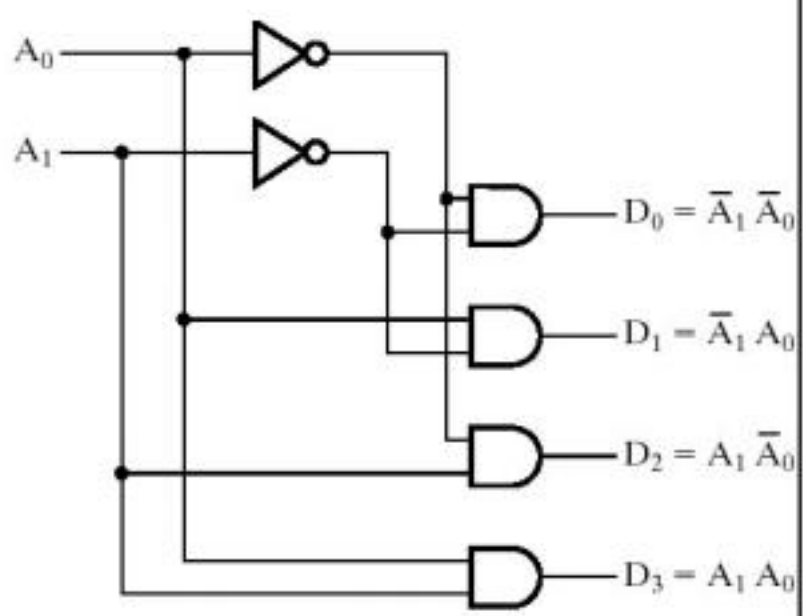
13

Decoding p.120

- 2-to-4 decoder

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)

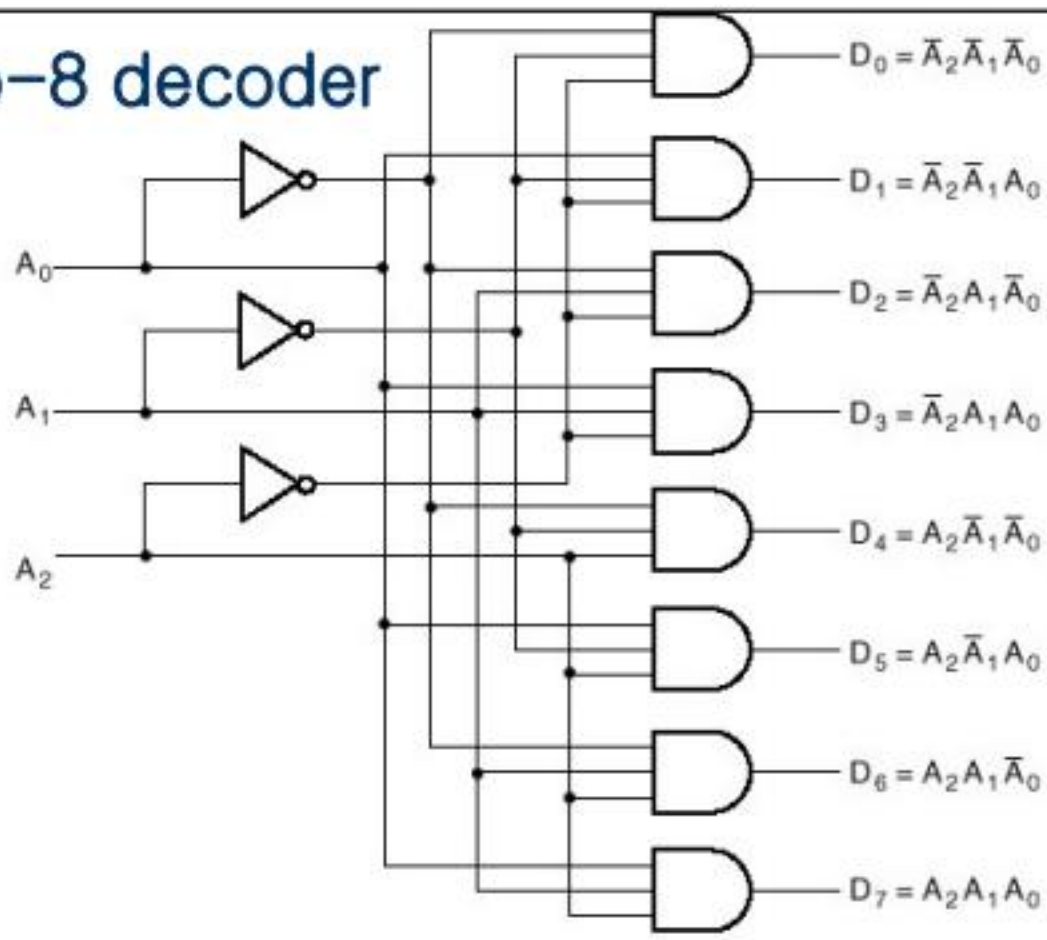


(b)

14

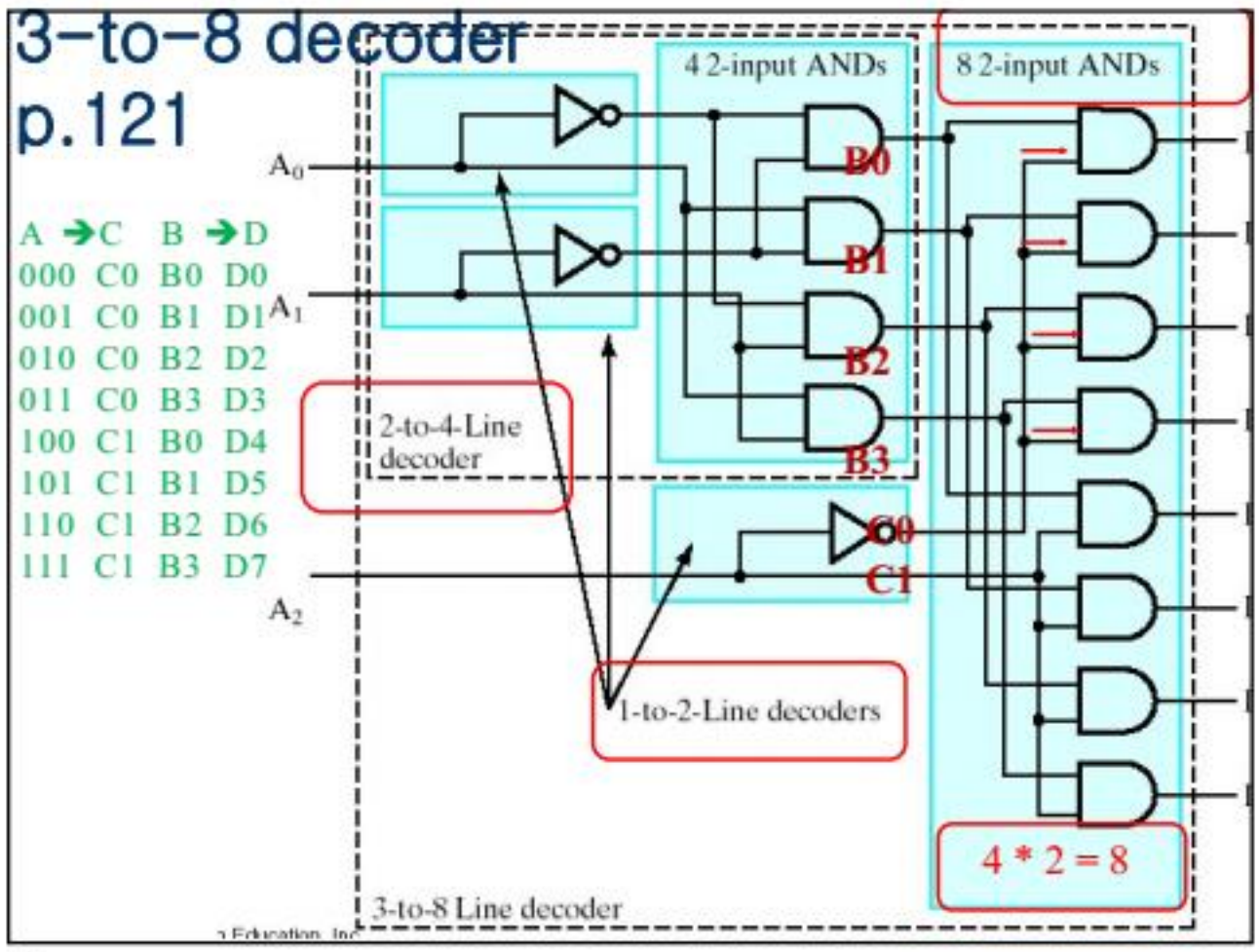
14

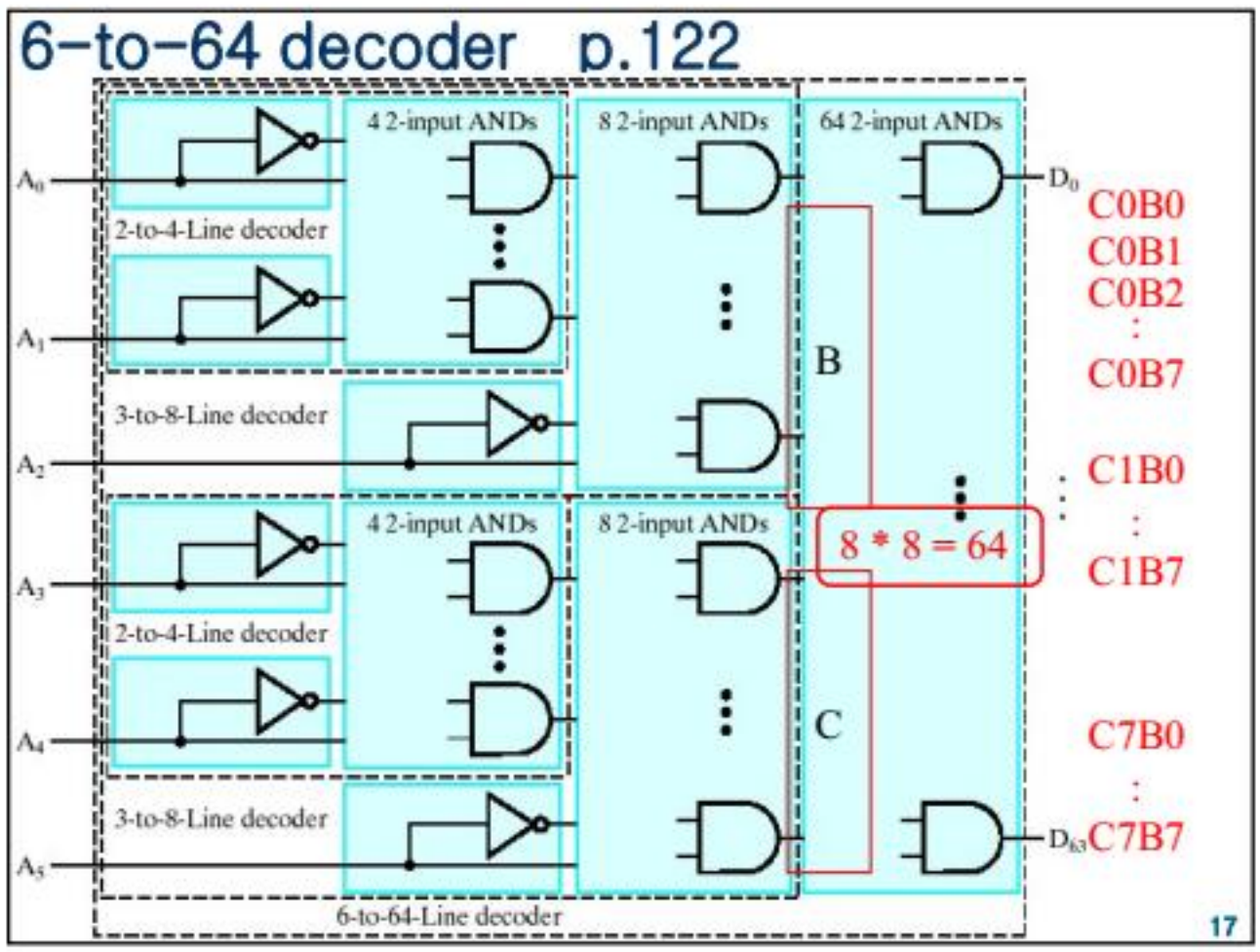
3-to-8 decoder



15

15


3-to-8 decoder
p.121
A → C B → D
000 C0 B0 D0
001 C0 B1 D1
010 C0 B2 D2
011 C0 B3 D3
100 C1 B0 D4
101 C1 B1 D5
110 C1 B2 D6
111 C1 B3 D7
A0
A1
A2
4 2-input ANDs
8 2-input ANDs
B0
B1
B2
B3
C0
C1
2-to-4-Line decoder
1-to-2-Line decoders
4 * 2 = 8
3-to-8 Line decoder


6-to-64 decoder p.122
A0
A1
A2
A3
A4
A5
2-to-4-Line decoder
3-to-8-Line decoder
4 2-input ANDs
8 2-input ANDs
64 2-input ANDs
B
C
8 * 8 = 64
D0
D63
C0B0
C0B1
C0B2
C0B7
C1B0
C1B7
C7B0
C7B7
6-to-64-Line decoder
17

6-to-64 decoder (비용 절감 측면)

P.123

- D0 = C0 B0 (000 000)
- D1 = C0 B1 (000 001)
- D2 = C0 B2 (000 010)
- :
- D7 = C0 B7 (000 111)
- D8 = C1 B0 (001 000)
- D9 = C1 B1 (001 001)
- :
- D15 = C1 B7 (001 111)
- D16 = C2 B0 (010 000)
- :
- D63 = C7 B7 (111 111)

Gate input cost 비교

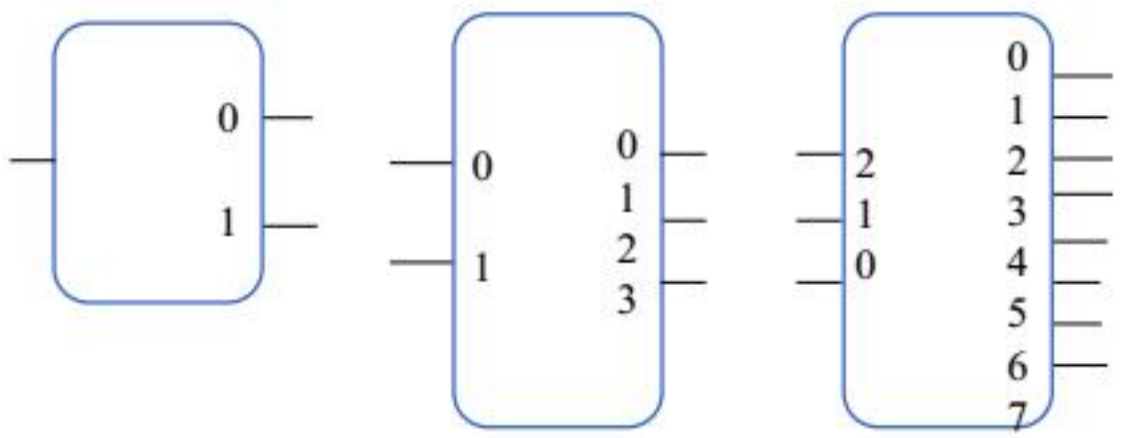
6입력 and gate 사용의 경우
6 + (6*64) = 390

그림 3-20의 경우
6 + 2(2*4)+2(2*8) + 2* 64
= 182

18

18

Decoder 1X2 2X4 3X8



19

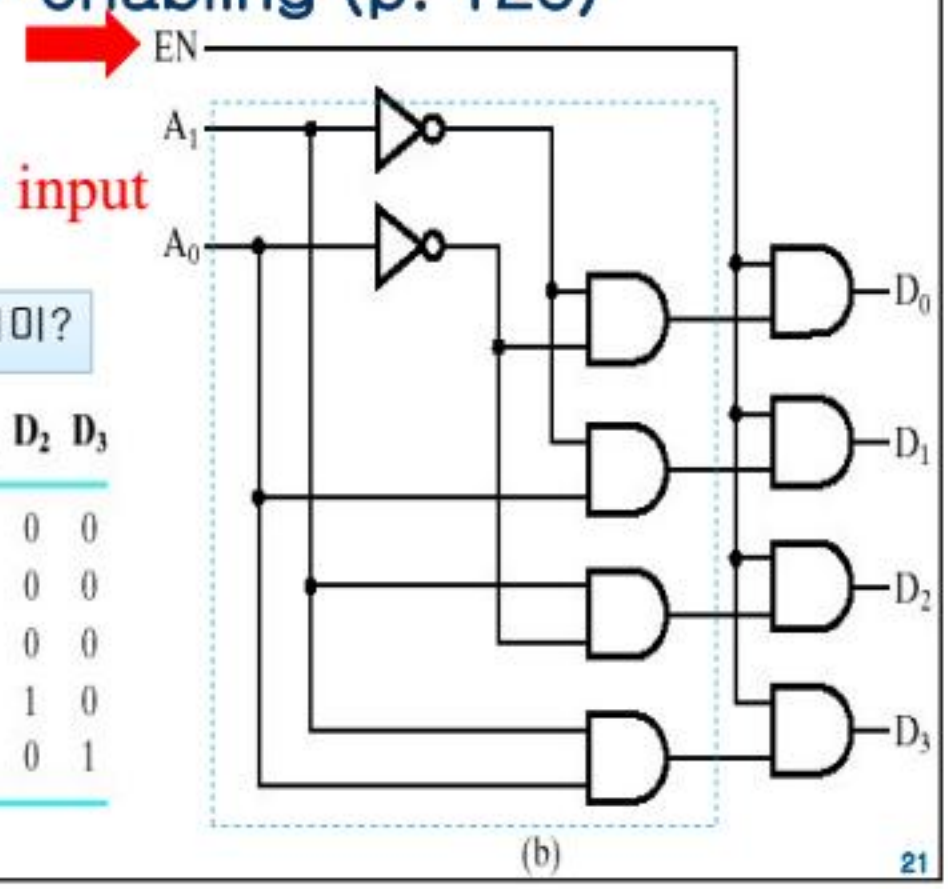
19

Function Blocks의 또 다른 적용

20

20

Decoder + enabling (p. 123)



입력에서 X의 의미?

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a) (b)

21

21

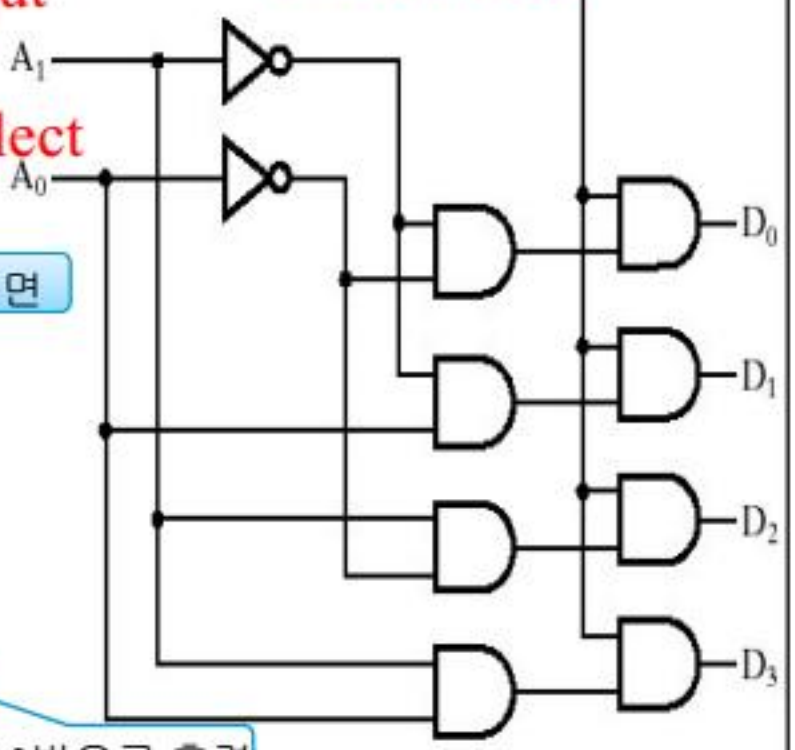
deMultiplexer (p. 124)

1 X 4 deMUX

input

하나의 정보(input)를
여러 개의 출력 중
하나로 전달.

select



관점을 바꾸면

input S_1 S_0

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

1을 3번으로 출력

(a) (b)

22

22

해석(관점)에 따라 동일한 회로를 다르게 부를 수 있다. (p. 124)

- 해석 1 :

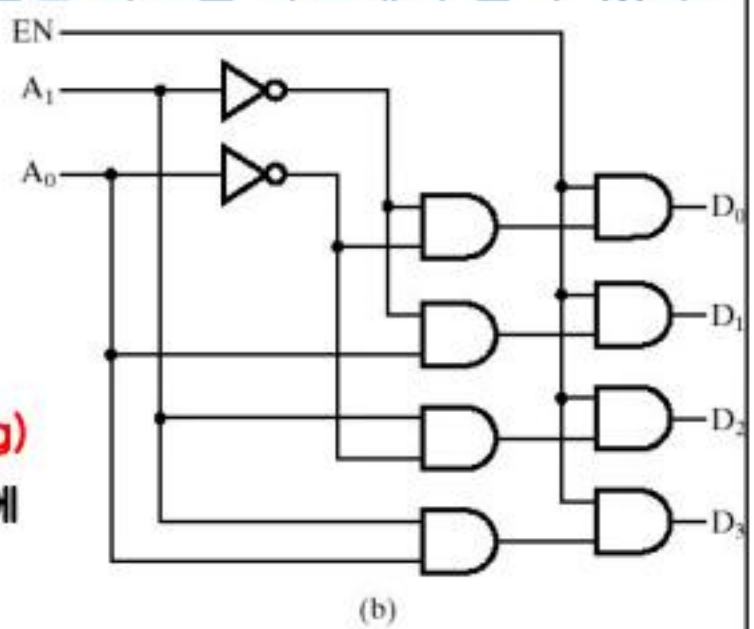
  EN = 0 이면 디코더는 동작하지 않는다.

  EN = 1 이면 디코더가 동작한다.

  (decoder with enabling)

- 해석 2 : A0,A1의 선택에 따라 입력 신호 EN 이 4개의 출력 중 하나로 전달된다.

  (deMultiplexer)



(b)

23

23

Decoder-Based Combinational Circuits p.126

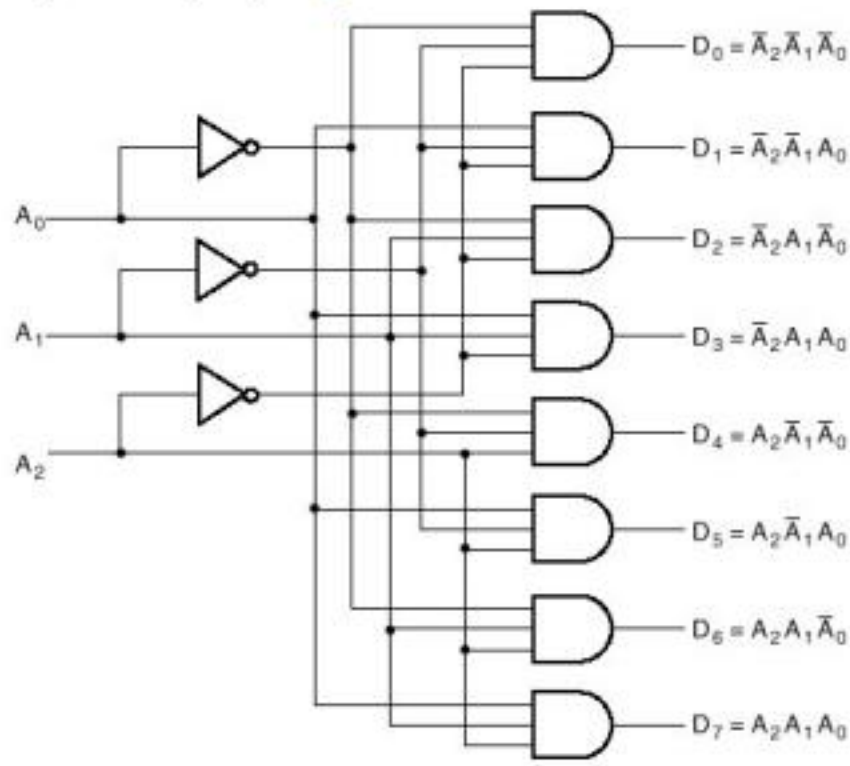
- 디코더 :
  - 2^n개의 최소항 제공
  - 모든 부울 함수는 최소항의 합으로 표현할 수 있다.
  - 즉, 디코더와 OR 게이트로 모든 부울함수가 구현가능.

- 활용 예
  - 많은 개수의 출력이 필요하고 각 출력의 구성이 적은 수의 최소항으로 구성된 경우, 디코더를 이용한 조합회로 설계가 유용하다.

24

24

기본 디코더 구성



25

25

Binary Adder p.127 진리표완성?

X	Y	Z	C	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

26

26

Binary Adder p.127 진리표완성?

X	Y	Z	C	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

$S(X,Y,Z)=\sum m(1,2,4,7)$

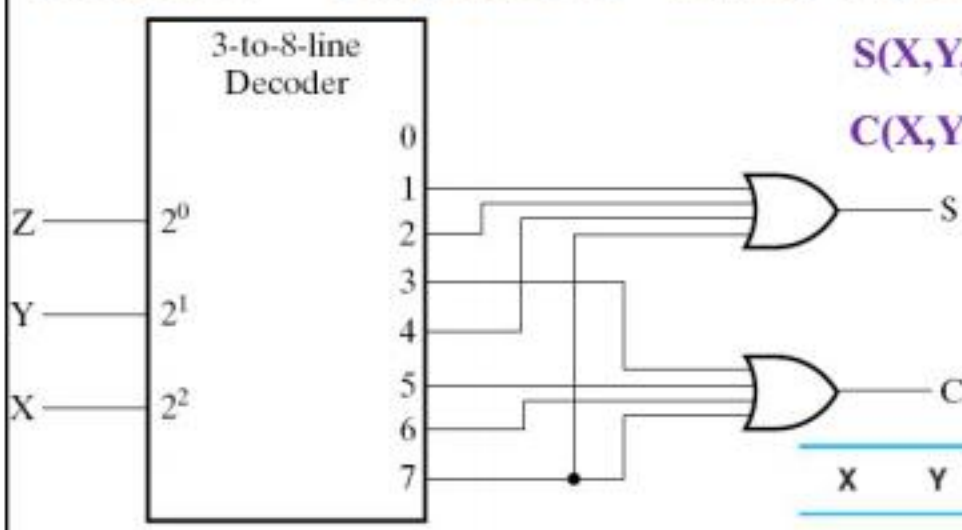
$C(X,Y,Z)=\sum m(3,5,6,7)$

* truth table = function table

28

28

Decoder + OR gates ➔ Binary Adder p.139



$S(X,Y,Z)=\sum m(1,2,4,7)$

$C(X,Y,Z)=\sum m(3,5,6,7)$

X	Y	Z	C	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

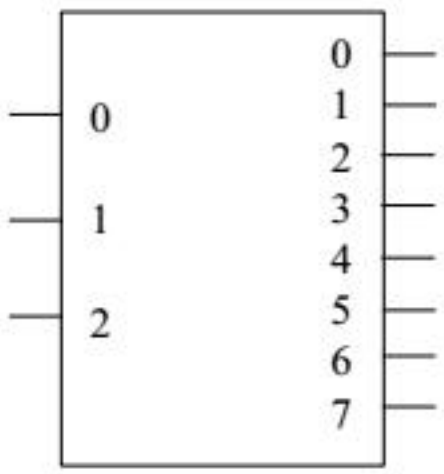
29

연습(디코더 기반 조합회로)

- GrayCode –to–BCD converter

를 3X8 decoder 와 or gates를 이용하여 설계하시오.

- 입력변수 3개 : A,B,C
- 출력변수 4개 : a,b,c,d



	Gray X	Y	Z	BCD abcd
0	0	0	0	0000
1	1	0	0	0001
2	1	0	1	0010
3	1	1	1	0011
4	1	1	0	0100
5	0	1	0	0101
6	0	1	1	0110
7	0	0	1	0111

m4

30

Encoder p.128

Decoder : 3 X 8
Encoder : 8 X 3

- decoder와 반대동작
  - 2^n개보다 적거나 같은 입력 ➔ n개의 출력
  - 의미) 입력선에 따라 2진코드를 생성한다.
- Encoder's Limitation
  Only one input can be active(1) at any given time

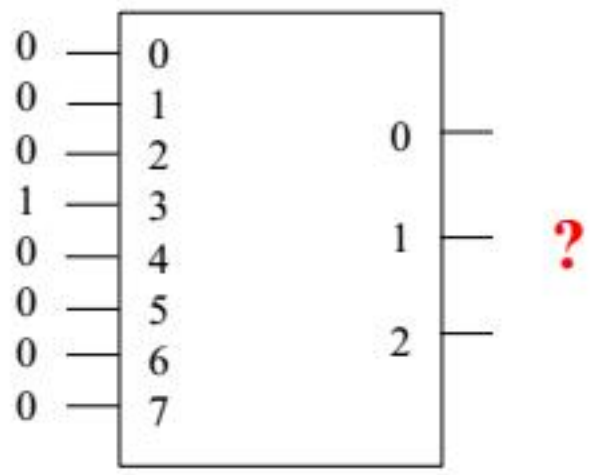
Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

32

32

encoder (p.128)

- 입력선에 따른 2진코드 생성

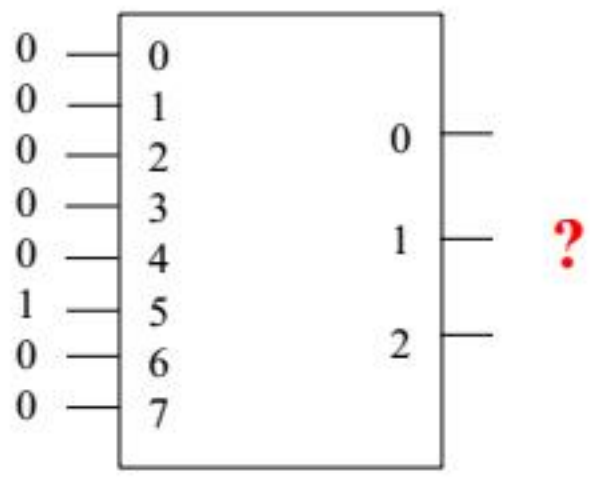


33

33

encoder (p.128)

- 입력선에 따른 2진코드 생성

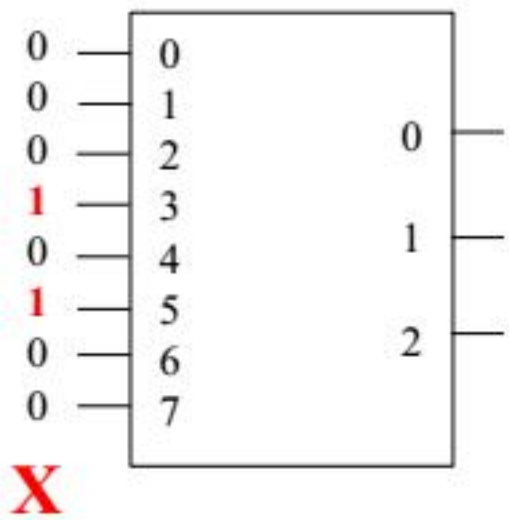


34

34

encoder (p.128)

- Encoder's Limitation : Only one input can be active(1) at any given time



35

35

Encoder p.128

- 예) octal-to binary encoder
  - $A_0 = D_1 + D_3 + D_5 + D_7$
  - $A_1 = D_2 + D_3 + D_6 + D_7$
  - $A_2 = D_4 + D_5 + D_6 + D_7$

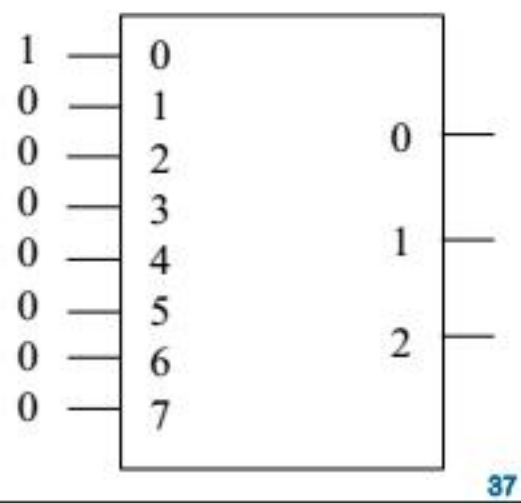
Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

36

36

Priority Encoder (우선순위-) p.129

- 문제 1) 부정확한 입력(1개 이상의 입력)
  - 해결1) 모호함을 없애기 위해 입력에 대한 우선 순위를 두자.
    <u>큰 첨자의 입력</u>(큰 값)에 우선순위를 둔다.
- 문제 2) 0의 출력에 대한 두가지 해석
  - D_0가 1일때의 유효한 출력
  - 모든 입력이 0일때의 무응답
    - (유효한 출력 / 무응답) 의 구분을 위한 출력 V 를 추가하자.



37

37

Priority Encoder (우선순위-) p.129

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V *valid*
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

- 입력이 하나 이상의 1을 가질때 V=1
- 무응답일때 V=0 (유효하지 않다.)

38

38

Priority Encoder 설계(간소화)

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

- A1 = ??
- A0 = ??
- V = ??

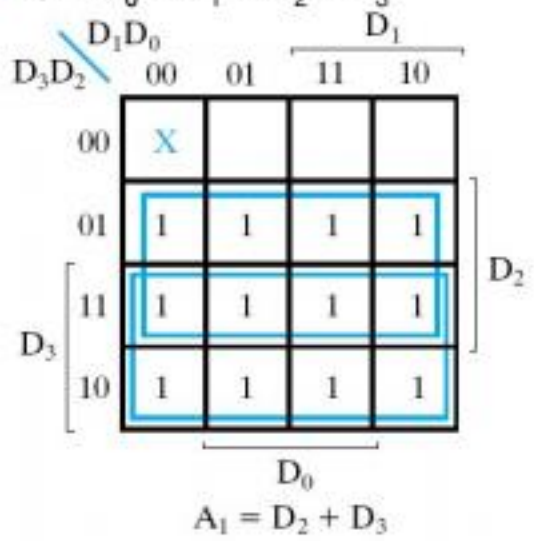
39

39

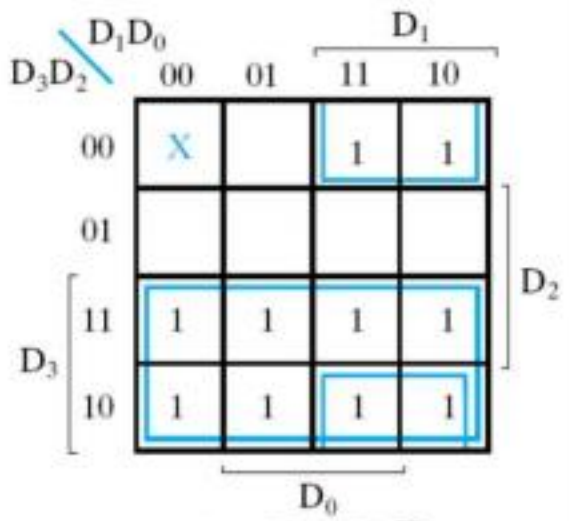
우선 순위 인코더는 회로의 단순함이 특징이다.

우선순위 인코더 p.130

- **우선 순위 인코더**
  - $A_0 = D_3 + D_1\overline{D_2}$
  - $A_1 = D_2 + D_3$
  - $V = D_0 + D_1 + D_2 + D_3$



$A_1 = D_2 + D_3$



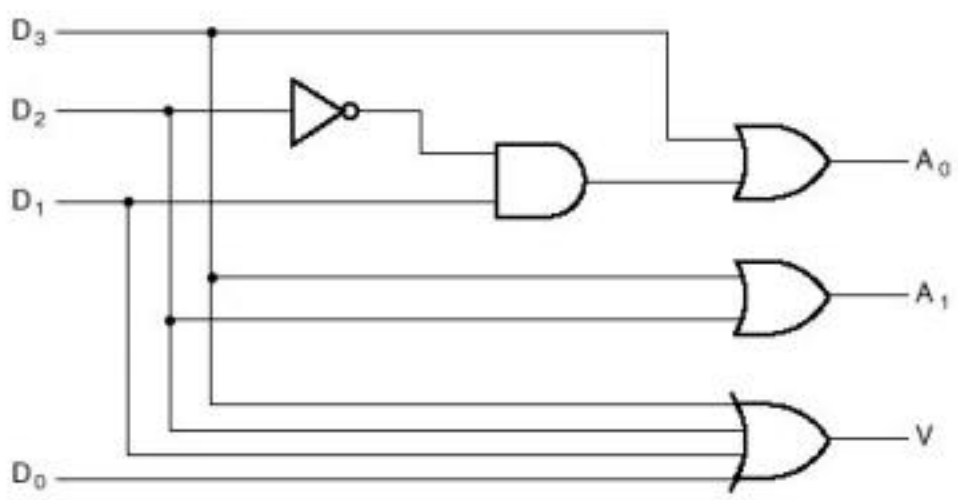
$A_0 = D_3 + D_1\overline{D}_2$

40

우선 순위 인코더는 회로의 단순함이 특징이다.

우선순위 인코더 p.130

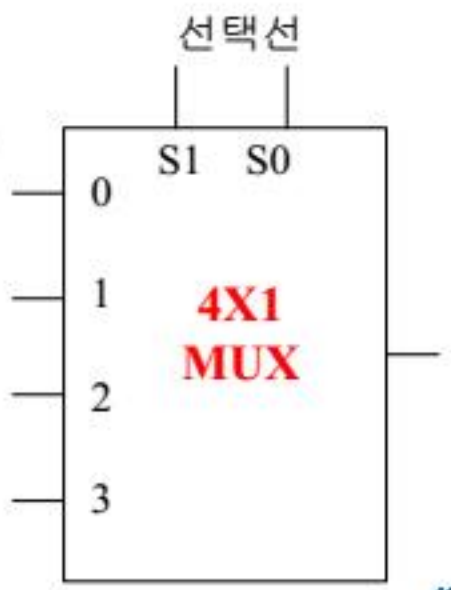
- **우선 순위 인코더**
  - $A_0 = D_3 + D_1\overline{D_2}$
  - $A_1 = D_2 + D_3$
  - $V = D_0 + D_1 + D_2 + D_3$



41

Multiplexer p.131

- 시스템간 통신에서 여러 입력 중 하나를 선택하는 기능이 필요.
- 멀티플렉서 (multiplexer, MUX)
  - 입력
    - 2^n의 다수의 입력선
    - n개의 선택 정보(selection input)
  - 출력 : 단일 출력선
  - 용도 : data selector 역할



42

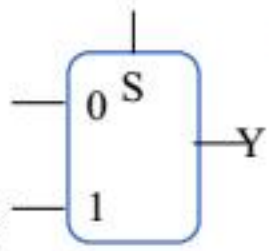
Multiplexer p.131

- 예1) 2X1 MUX
  - 2개의 입력 중 하나를 선택해 출력
  - 2개의 입력
  - 1개의 출력
  - ?개의 선택정보
- 예2) 8X1 MUX
  - 8개의 입력 중 하나를 선택해 출력
  - 8개의 입력
  - 1개의 출력
  - ? 개의 선택정보

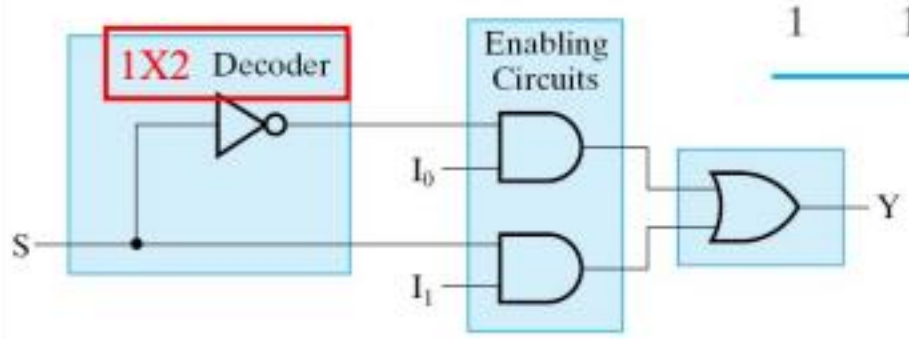
43

Multiplexer' p.132

- 2-to-1 MUX
  - $Y = S' I_0 + S I_1$
  - 해석)
    - If S=0, I_0 출력
    - If S=1, I_1 출력



S	I_0	I_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



S	Y
0	I_0
1	I_1

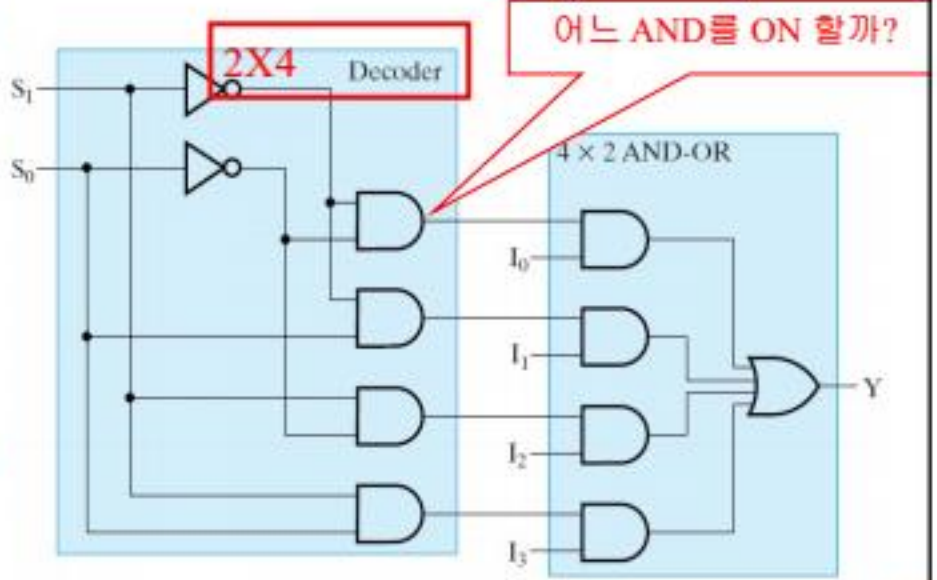
44

n-to-1 line MUX p.133

- N-to-1 line MUX에서 선택입력선은 몇 개?

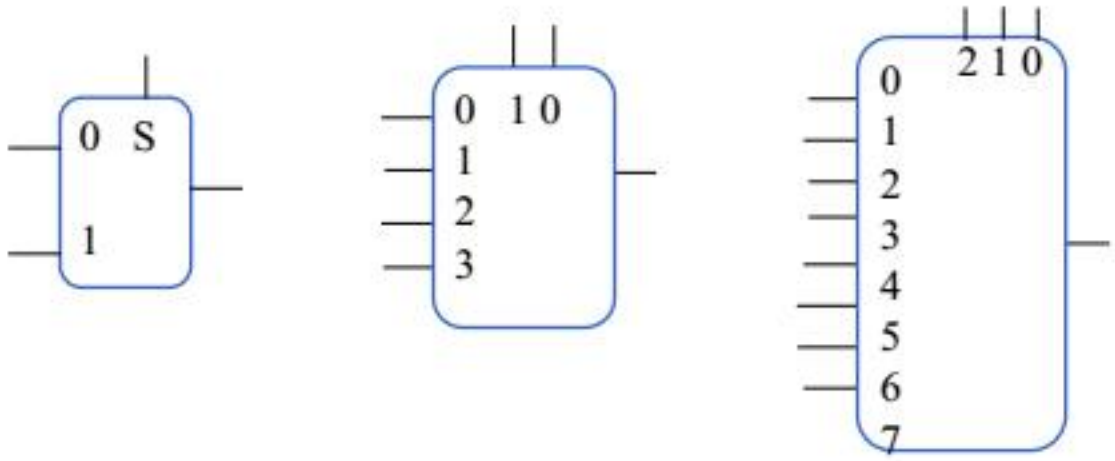
4 X 1 MUX

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



45

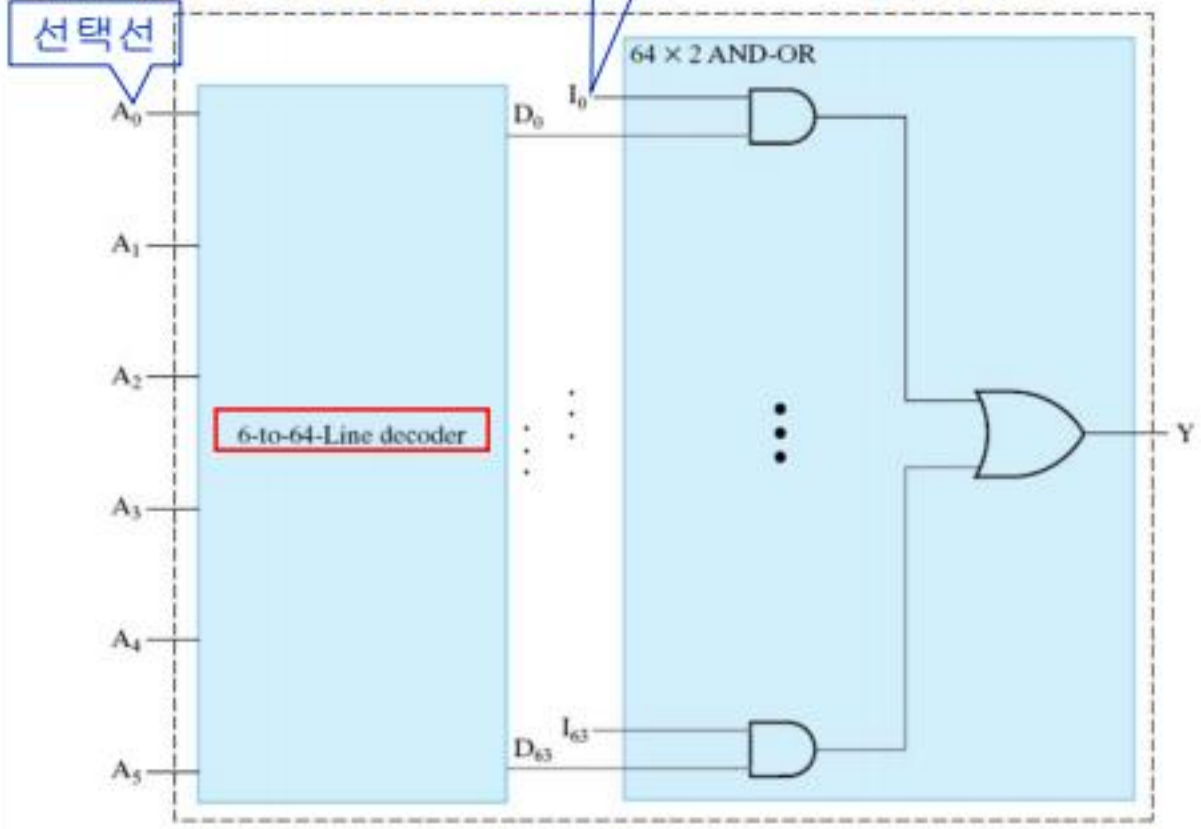
MUX 2X1 4X1 8X1



46

46

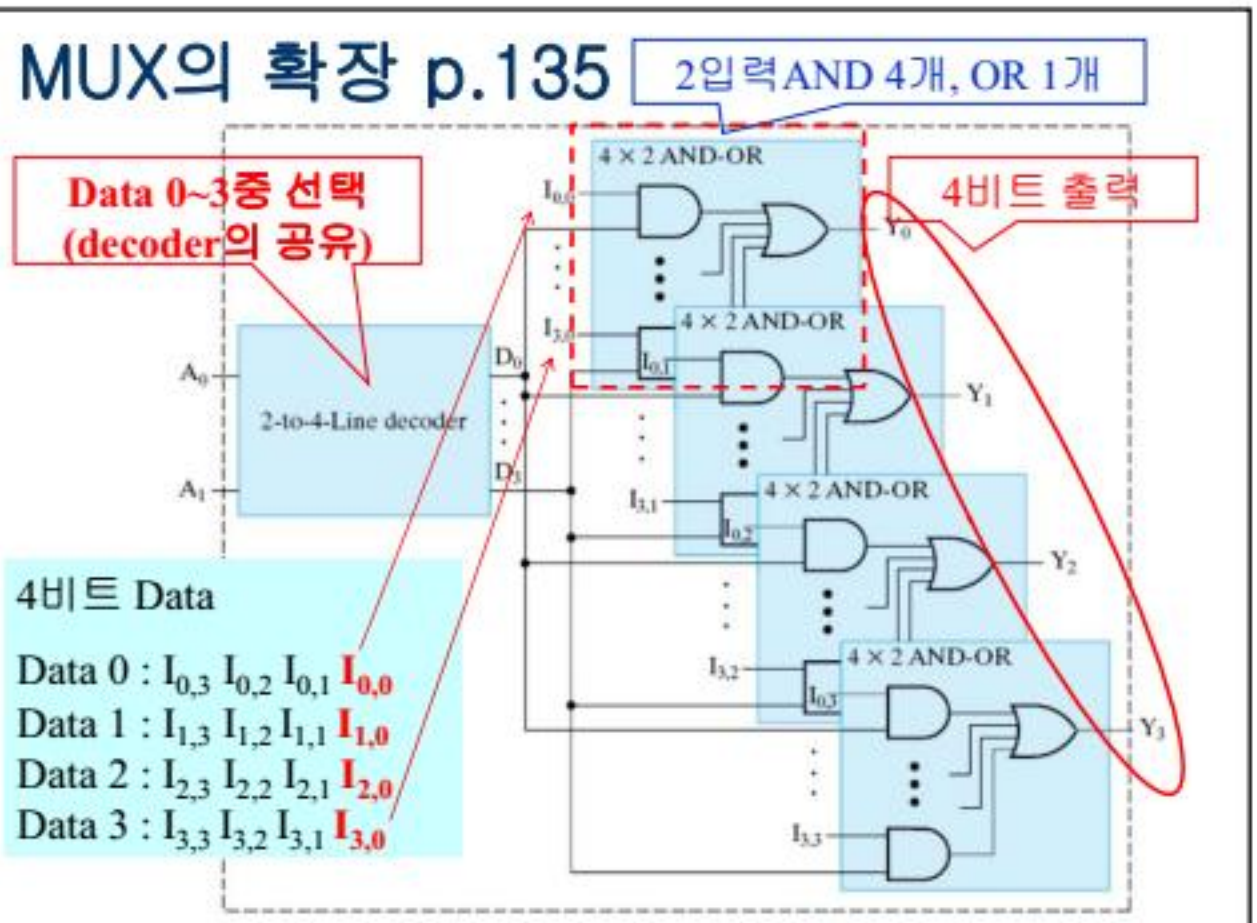
64 X 1 MUX 입력선 p.134



47

47

MUX의 확장 p.135

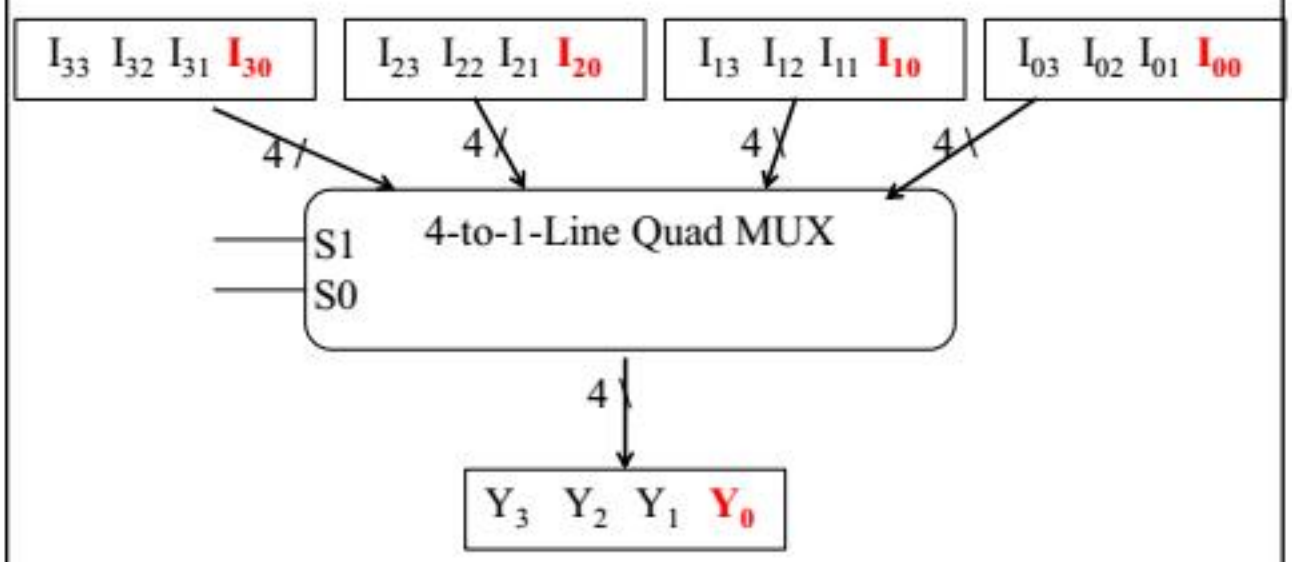


48

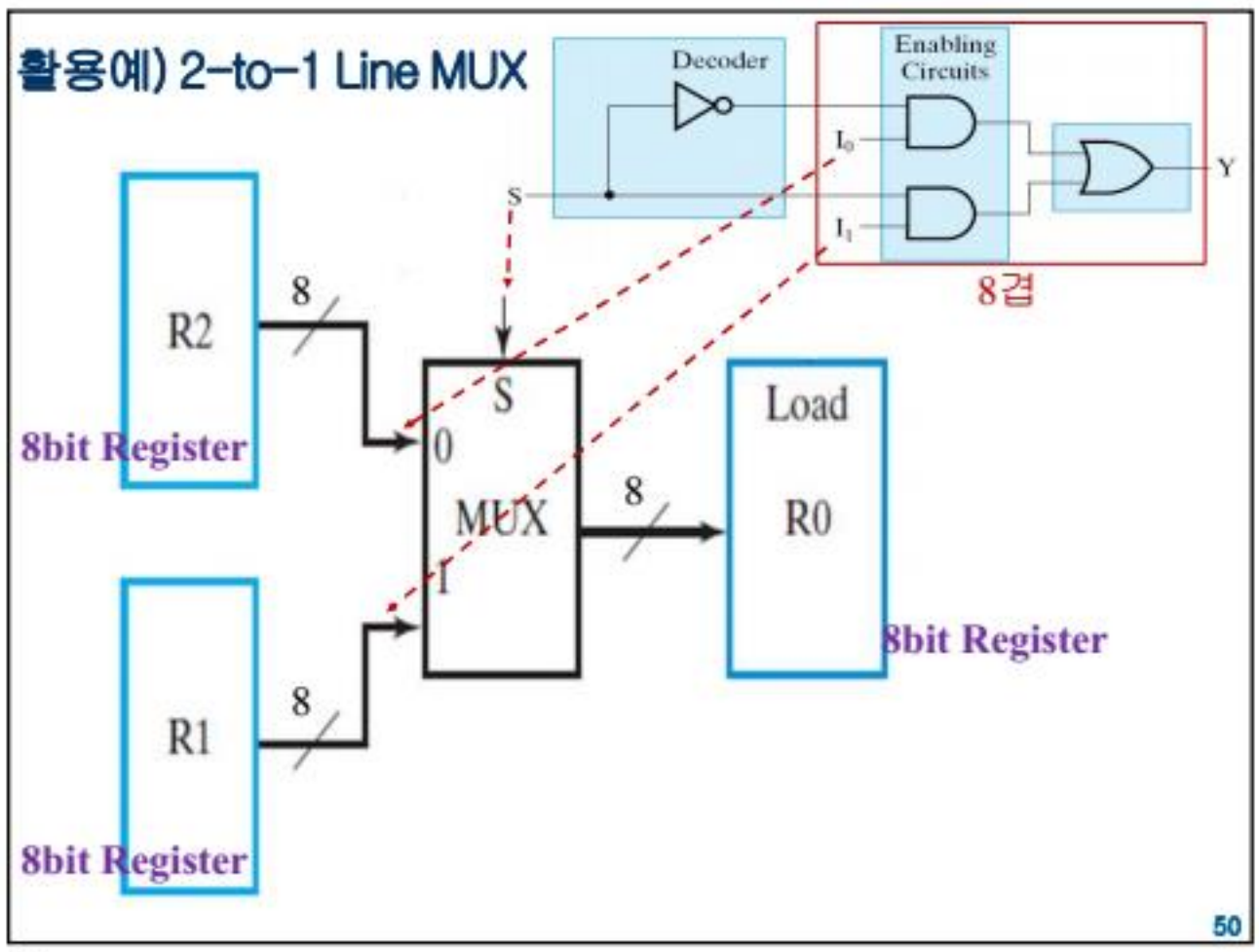
4-to-1-Line Quad MUX

p.135

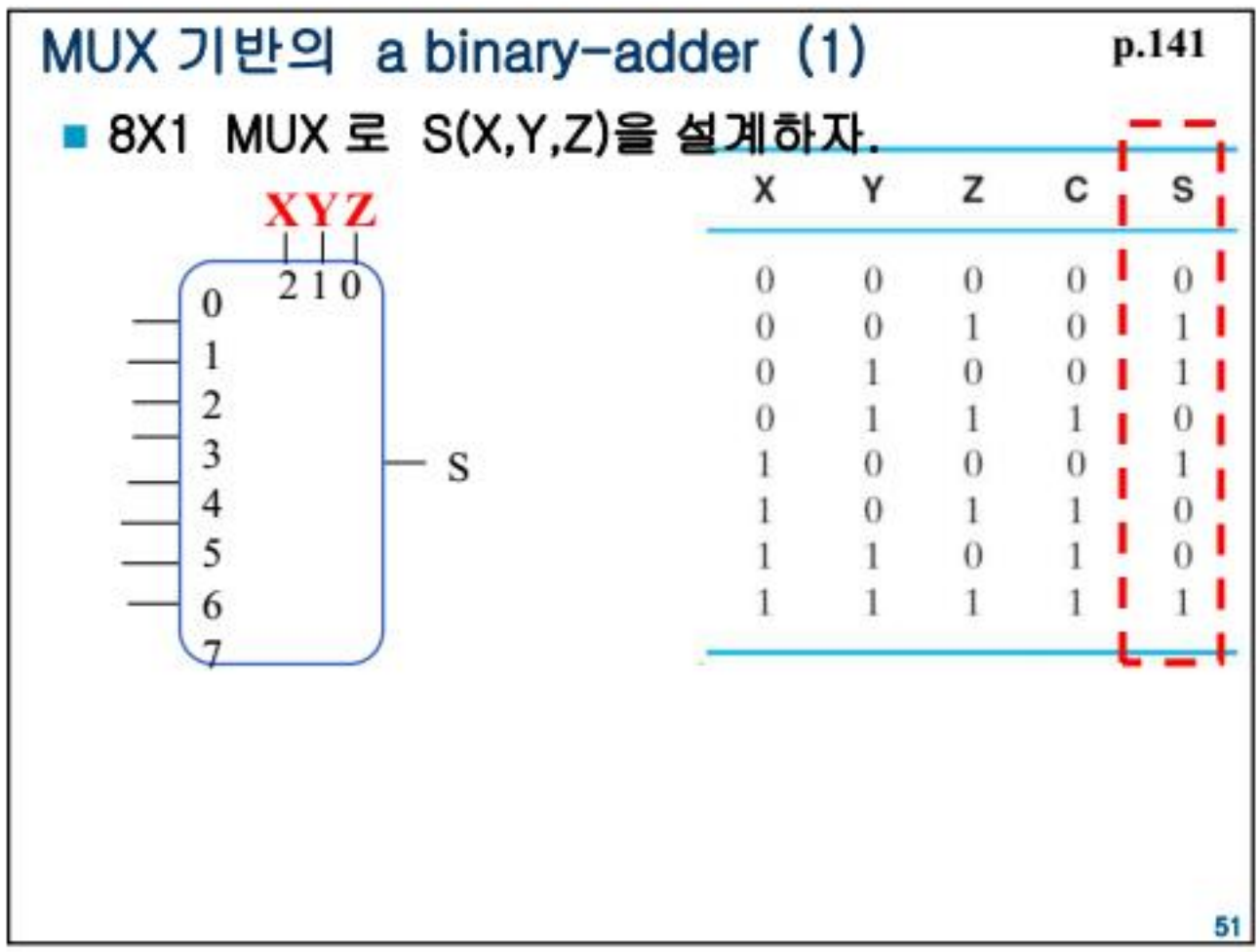
- 4비트 데이터가 4개 있다. 이 중 하나를 선택한다.



49


활용예) 2-to-1 Line MUX
Decoder
Enabling Circuits
I_0
I_1
S
Y
8겹
R2
8bit Register
8
S
0
MUX
1
8
Load
R0
8bit Register
R1
8
8bit Register
50

50

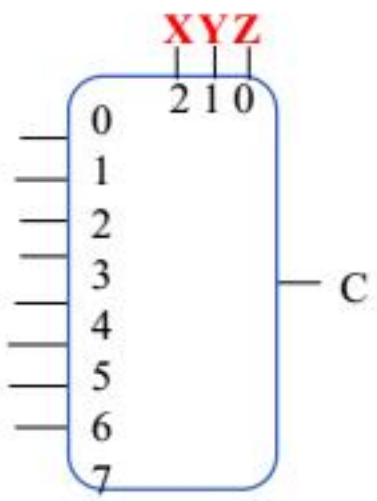

MUX 기반의 a binary-adder (1)
p.141
8X1 MUX 로 S(X,Y,Z)을 설계하자.
X Y Z
2 1 0
0
1
2
3
4
5
6
7
S
X Y Z C S
0 0 0 0 0
0 0 1 0 1
0 1 0 0 1
0 1 1 1 0
1 0 0 0 1
1 0 1 1 0
1 1 0 1 0
1 1 1 1 1
51

51

MUX 기반의 a binary-adder (1)

p.141

- 8X1 MUX 로 C(X,Y,Z)을 설계하자.



X	Y	Z	C	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

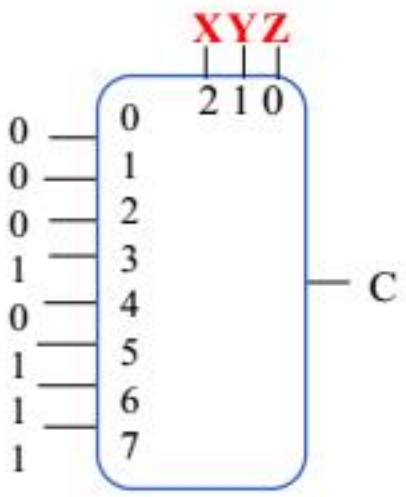
52

52

MUX 기반의 a binary-adder (1)

p.141

- 8X1 MUX 로 C(X,Y,Z)을 설계하자.



X	Y	Z	C	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

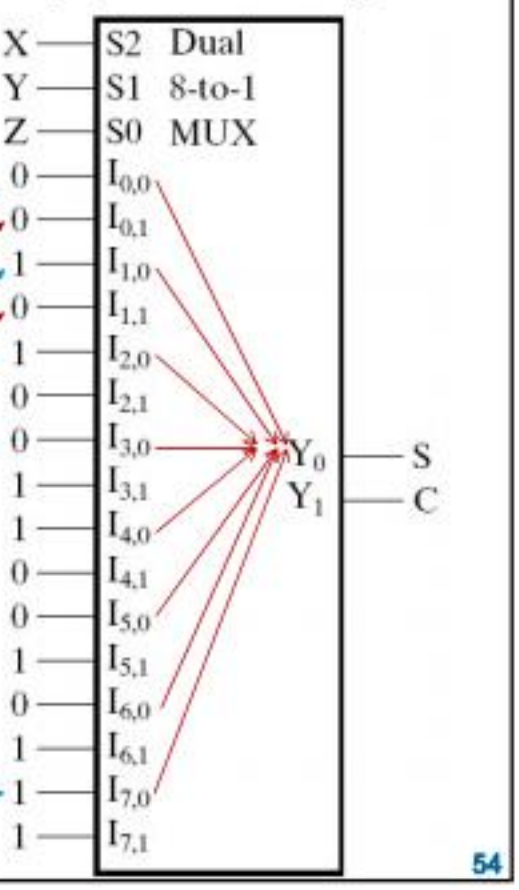
53

53

MUX 기반의 a binary-adder (1) p.141

- 2비트 8-to-1 MUX
  - 해석) 8-to-1 MUX 2쌍
  - 동작)
    - XYZ에 (100)이 입력되면 ($I_{4,0}$, $I_{4,1}$)의 2비트가 출력

X	Y	Z	C	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

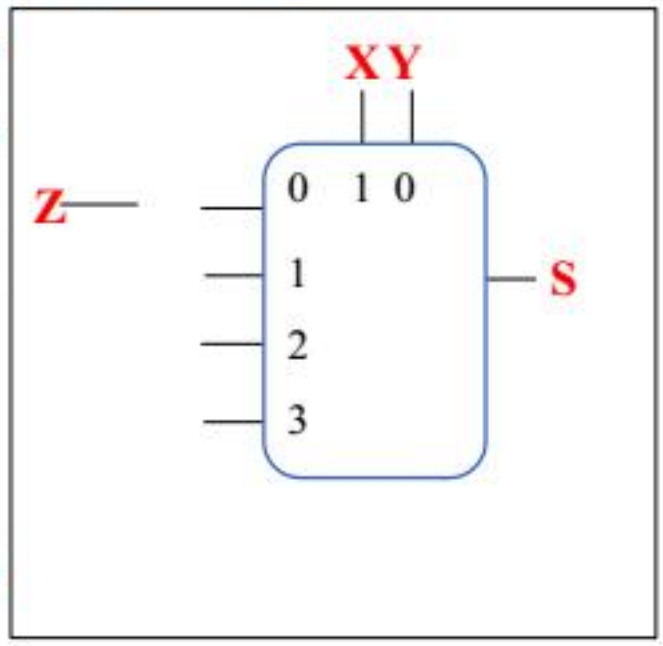


54

54

4X1 MUX of a binary-adder (2) p.141

X	Y	Z	C	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

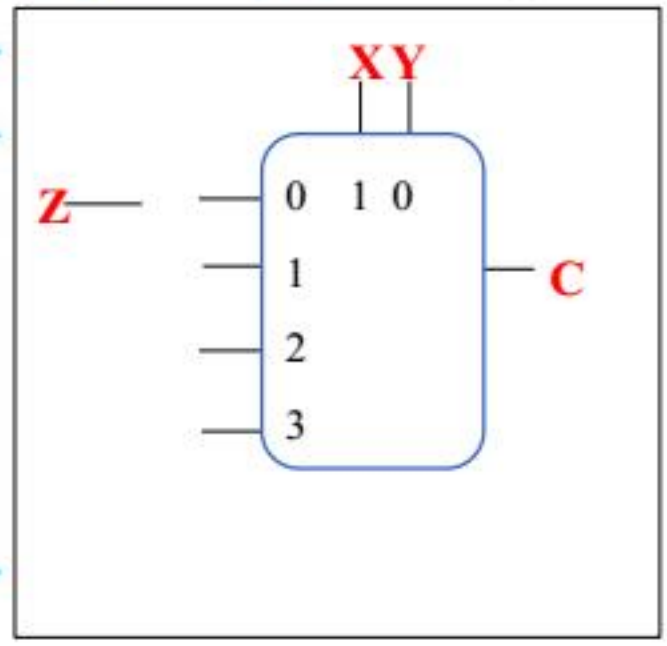


55

55

4X1 MUX of a binary-adder (2) p.141

X	Y	Z	C	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

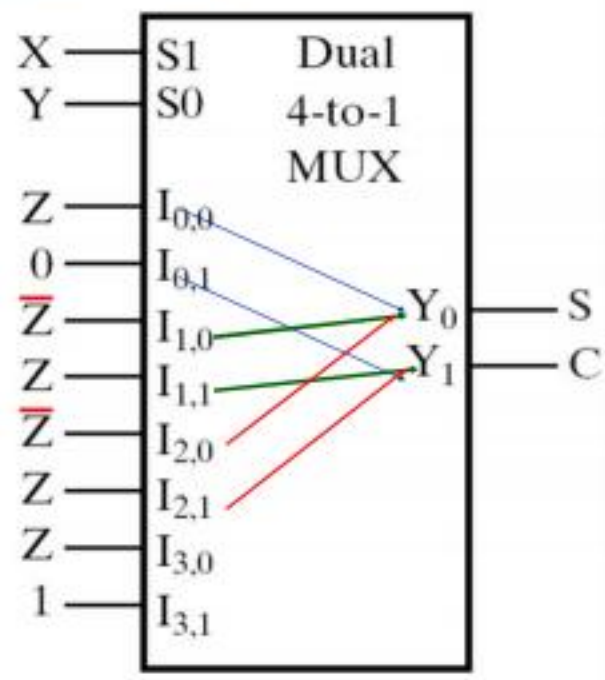


56

4X1 MUX of a binary-adder (2) p.141

X	Y	Z	C	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

Y1 Y0



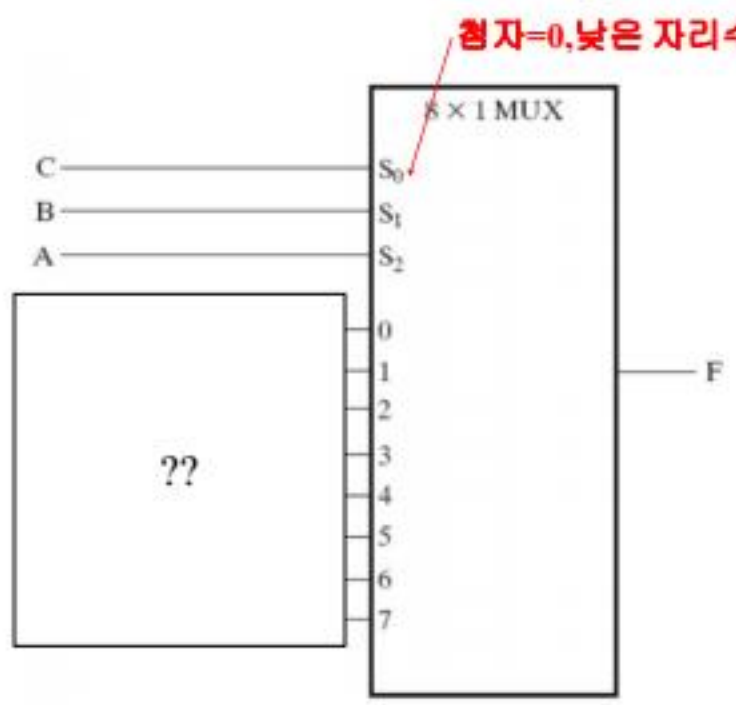
입력을 꼭 z만 이용할 필요는 없다. 생각의 편의상 진리표의 낮은 값 변수를 이용할 뿐

57

57

8X1 MUX 활용 => 4입력함수의 구현 p.142

A	B	C	D	F	
0	0	0	0	0	??
0	0	0	1	1	
0	0	1	0	0	
0	0	1	1	1	
0	1	0	0	1	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

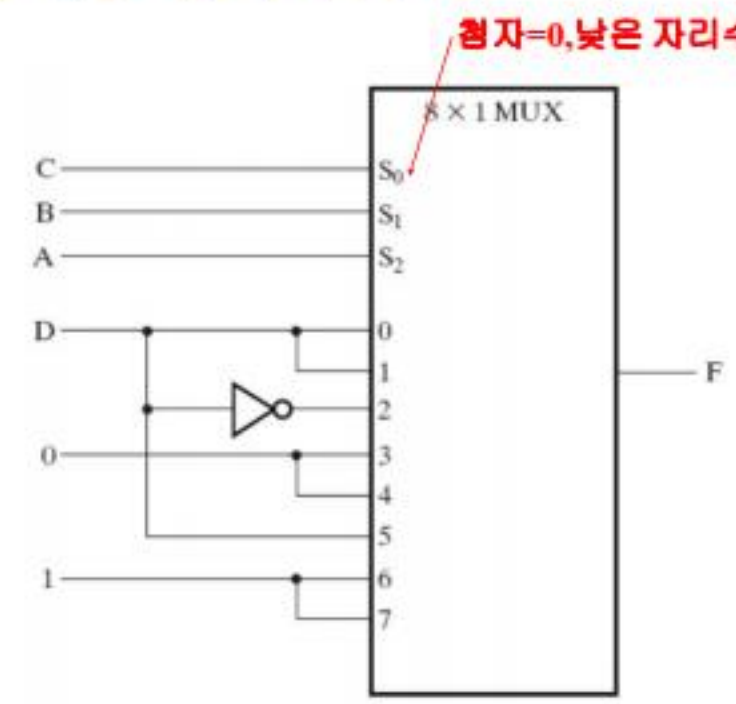


58

58

8X1 MUX 활용 => 4입력함수의 구현 p.142

A	B	C	D	F	
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	$F = \overline{D}$
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = 1
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	

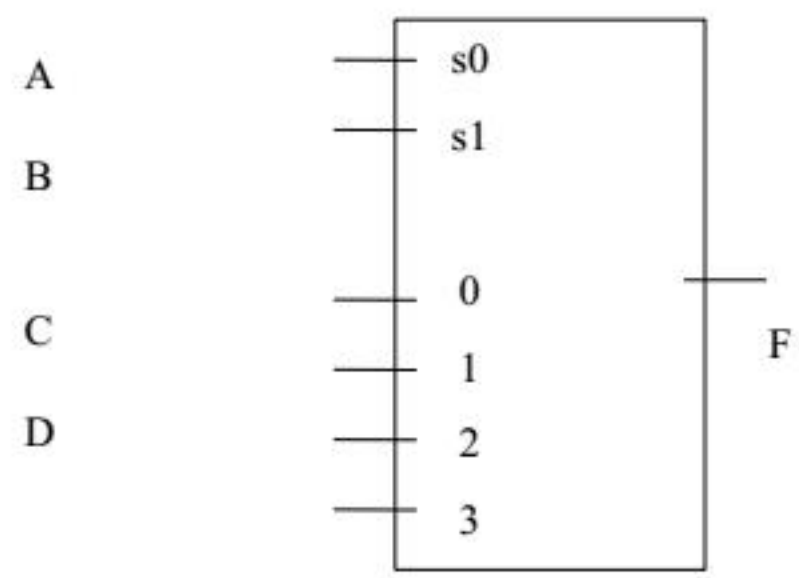


59

59

4X1 MUX 활용 => 4입력함수의 구현 p.142

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



60

60

연습 P.180

- 3-47 4-to-1 라인 멀티플렉서와 외부게이트들 (And, Or, Not)로 된 부울 함수를 구현하라.

  $F(A,B,C,D) = \sum m(1,3,4,11,12,13,14,15)$

  단, A,B를 선택선과 연결하라.

61

61

정리

- 기본 함수 블럭들 : 컴퓨터 설계에서 유용하게 사용되는 조합회로들
  - enable 기능
  - Decoder
    - Decoder + enabling
    - Decoder를 이용한 부울 함수
  - Encoder
    - Priority encoder
  - Multiplexer
    - MUX를 이용한 부울 함수

62