UNIT 2
MUX, DECODER AND ENCODER, ROM

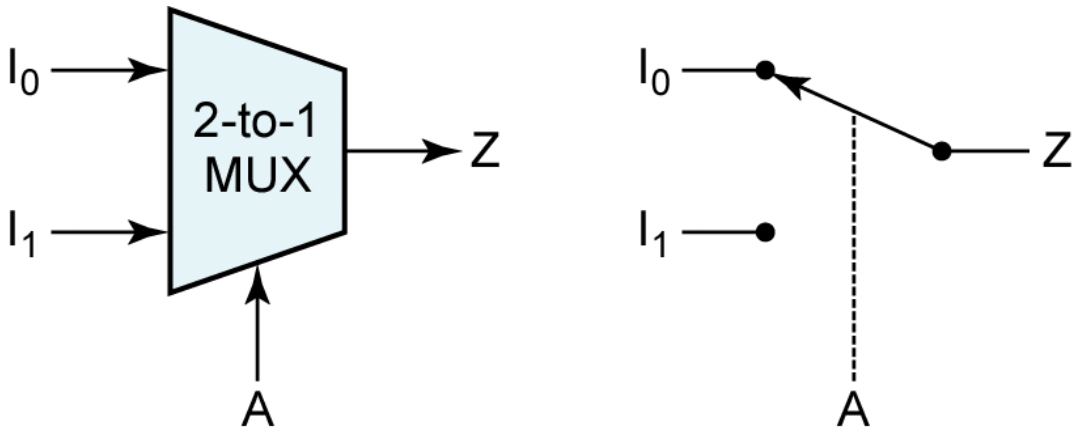
2012년 2학기

마이크로 프로세서 실습

CONTENTS

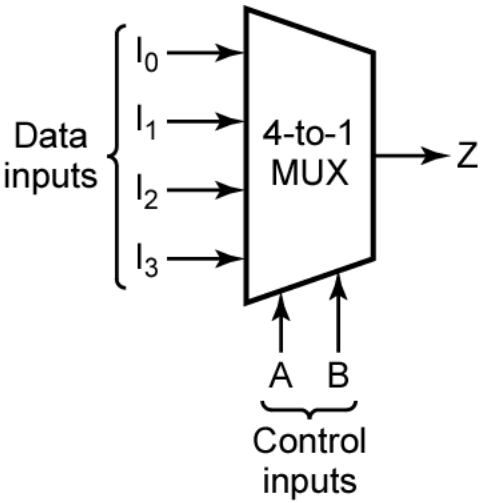
1 MULTIPLEXER

- Multiplexer(MUX)
  - 데이터 입력과 제어 입력으로 구성된 소자
  - 제어 입력은 데이타 입력 중에서 하나를 선택하여 출력단과 연결하는 역할
  - 2-to-1 MUX 및 등가 스위치

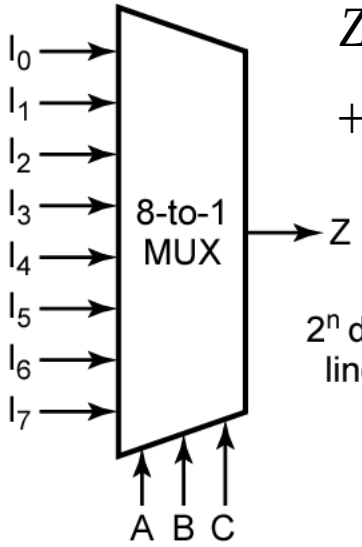


$$Z = A'I_0 + AI_1$$

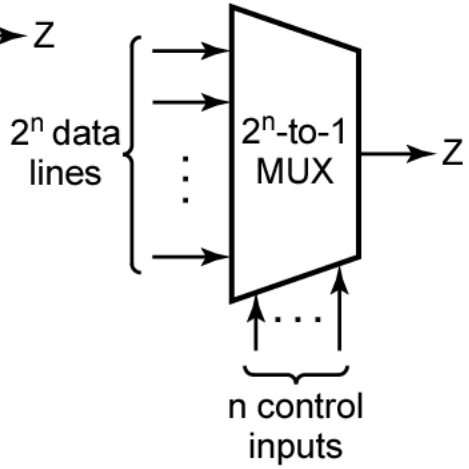
MUX INTRODUCTION [1 OF 6]



$$Z = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$$



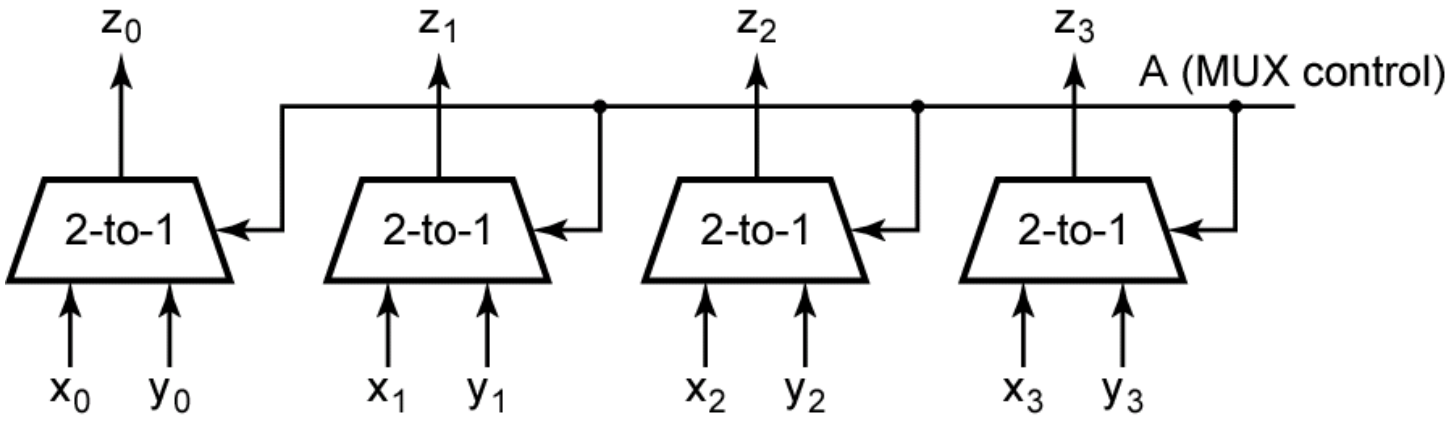
$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$



$$Z = \sum_{k=0}^{2^n-1} m_k I_k$$

MUX INTRODUCTION [2 OF 6]

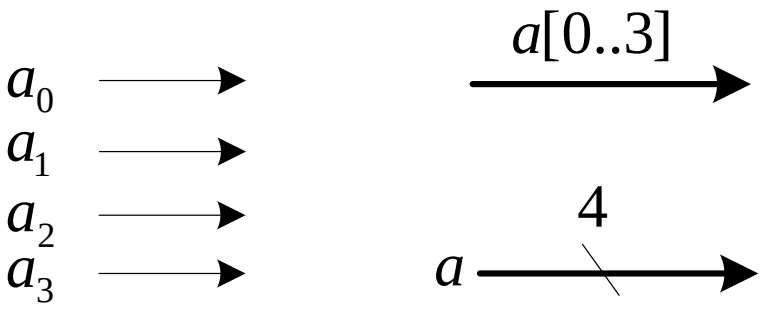
- MUX는 디지털 시스템에서 처리되거나 저장되는 데이터를 선택하는데 자주 사용



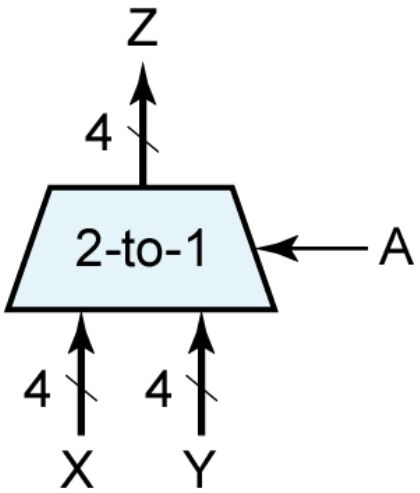
  - 4개의 2-to-1 멀티플렉서를 이용한 데이터 워드 선택 구현
  - 제어입력 A에 따라 다른 워드 선택
    - A = 0 : x0, x1, x2, x3
    - A = 1 : y0, y1, y2, y3

MUX INTRODUCTION [3 OF 6]

- Bus
  - 여러 개의 논리 신호가 공통된 기능을 수행하기 위해 그룹화 된 것
  - 여러 개의 선을 다 그리는 대신 하나의 굵은 선과 사선을 이용하여 표현



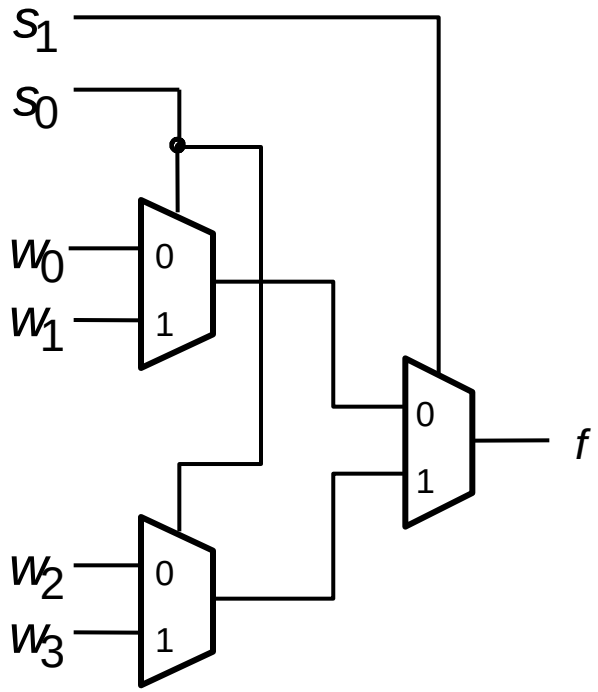
  - 4 bit 입력과 출력을 가지는 2-to-1 MUX



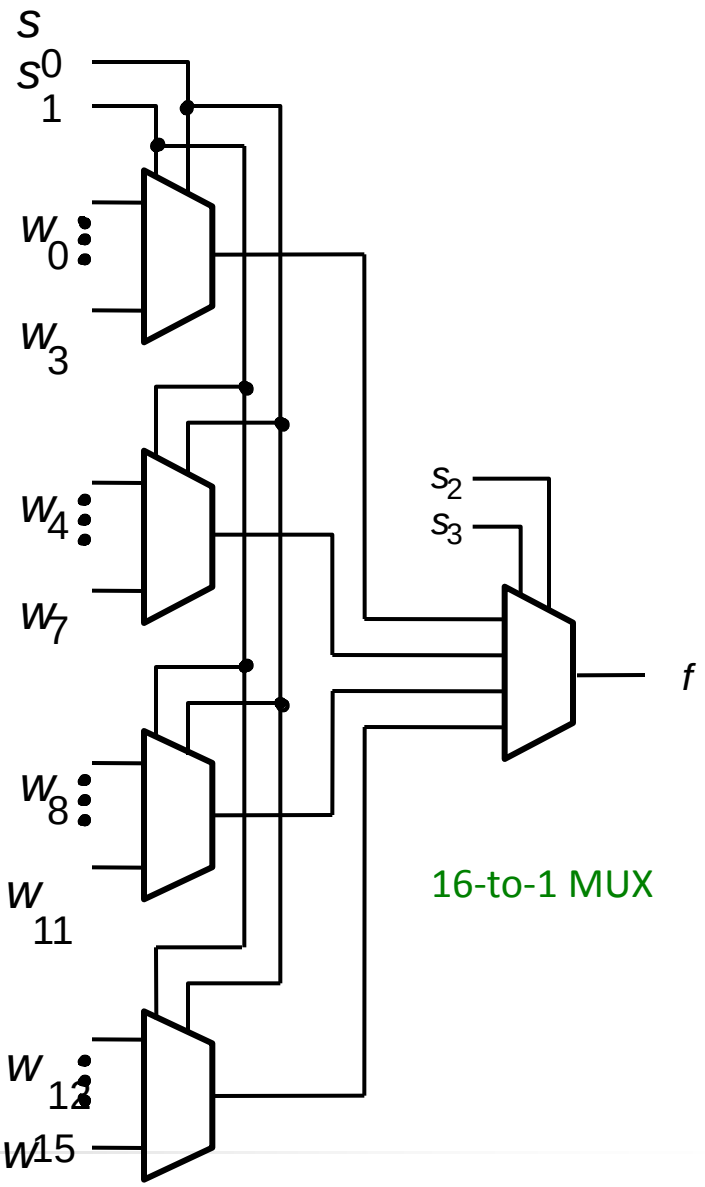
MUX INTRODUCTION [4 OF 6]

- 다양한 MUX
  - 선택된 입력 값을 반전시켜 출력하는 MUX
    - Active low 상태 출력을 가짐
    - 반전이 없는 경우는 Active high 상태 출력을 가진다고 함
  - Enable 단자를 가지는 MUX
    - MUX의 입력에 Enable 단자를 추가
    - E (Enable) = 0 인 경우 입력에 무관하게 출력은 0
    - E = 1 인 경우 멀티플렉서로 동작
    - Enable 단자 역시 Active high/low 존재
    - Enable 단자의 버블 유무로 구별 가능

MUX INTRODUCTION [5 OF 6]



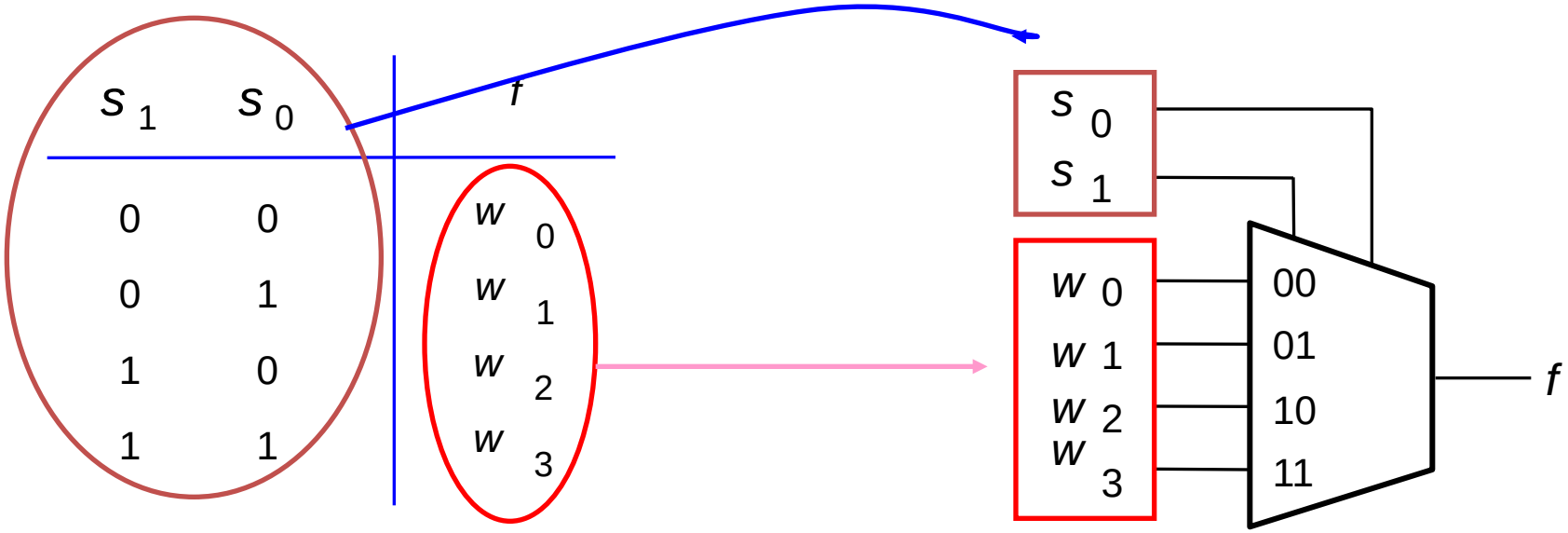
2-to-1 MUX를 이용한 4-to-1 MUX



16-to-1 MUX

MUX INTRODUCTION [6 OF 6]

- 진리표와 MUX와의 관계



Truth table

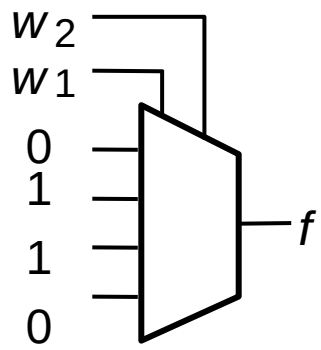
Graphic symbol

그렇다면, 입력이 4변수이면 어떤 MUX가 필요?

LOGICAL ANALYSIS OF MUX [1 OF 4]

- EXOR

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

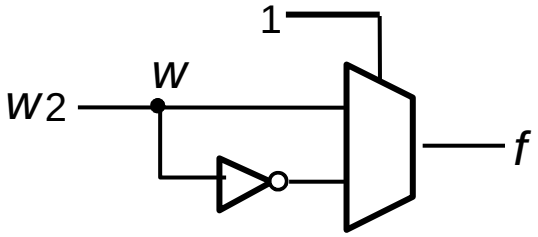


(a) Implementation using a 4-to-1 multiplexer

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

w_1	f
0	w_2
1	$\bar{w}_2$

(b) Modified truth table



(c) Circuit

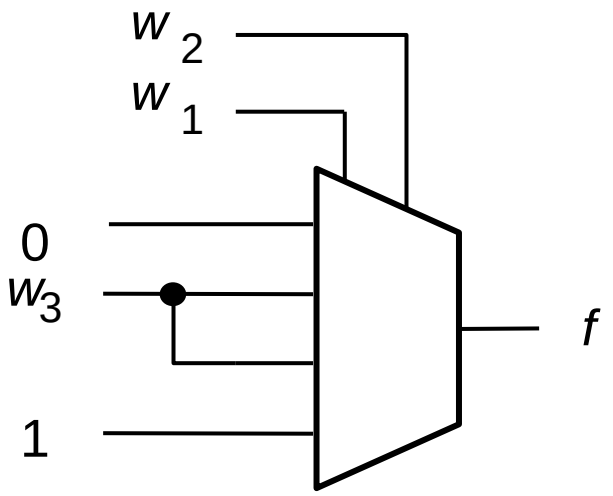
LOGICAL ANALYSIS OF MUX [2 OF 4]

- 3입력 중, (1의 개 수) > (0의 갯 수) 일 때 1이 되는 함수

w_1	w_2	w_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

w_1	w_2	f
0	0	0
0	1	w_3
1	0	w_3
1	1	1

(a) Modified truth table



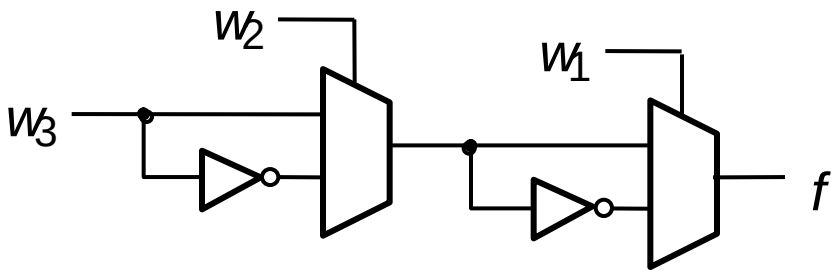
(b) Circuit

LOGICAL ANALYSIS OF MUX [3 OF 4]

- 3입력 XOR을 2 to 1 MUX로 구현하면?

w_1	w_2	w_3	f	
0	0	0	0	$w_2 \oplus w_3$
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	$\overline{w_2 \oplus w_3}$
1	0	1	0	
1	1	0	0	
1	1	1	1	

(a) Truth table



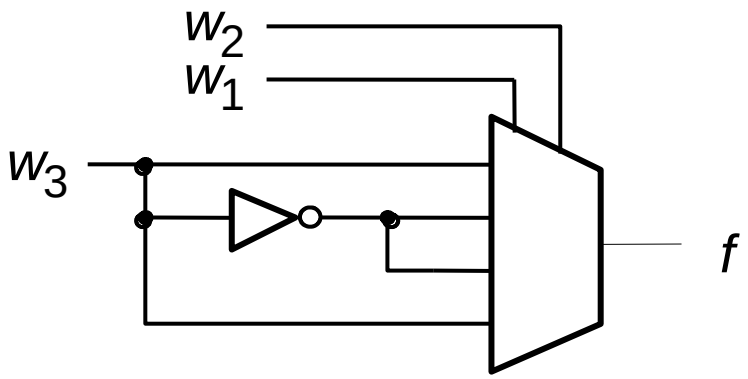
(b) Circuit

LOGICAL ANALYSIS OF MUX [4 OF 4]

- 3입력 XOR의 4-to-1 MUX를 이용한 합성

w_1	w_2	w_3	f	
0	0	0	0	w_3
0	0	1	1	
0	1	0	1	$\bar{w}_3$
0	1	1	0	
1	0	0	1	$\bar{w}_3$
1	0	1	0	
1	1	0	0	w_3
1	1	1	1	

(a) Truth table

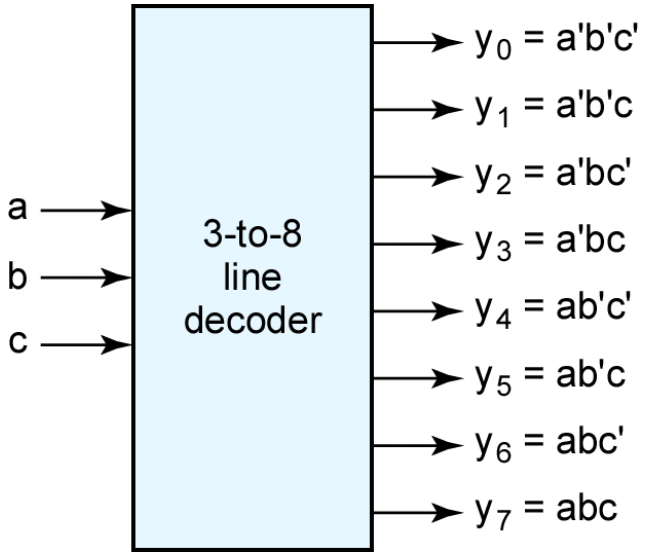


(b) Circuit

2 DECODER AND ENCODER

DECODER INTRODUCTION [1 OF 9]

- Decoder
  - N 개의 입력 변수를 가진 함수의 모든 최소항(2^n)을 생성한다.
  - 입력변수에 따라 출력 선 중 단 하나만이 1이 된다



3 to 9 line decoder

a	b	c	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

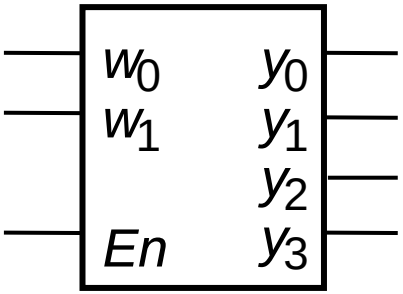
Truth table

DECODER INTRODUCTION [2 OF 9]

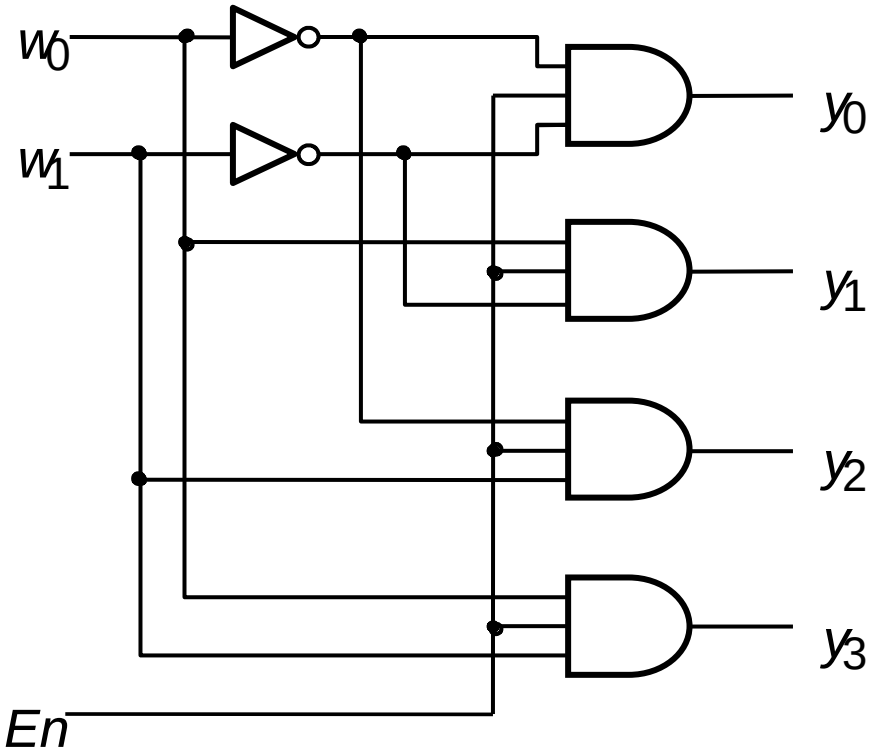
- 2 to 4 Decoder W/Enable

En	w_1	w_0	y_0	y_1	y_2	y_3
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
0	x	x	0	0	0	0

(a) Truth table



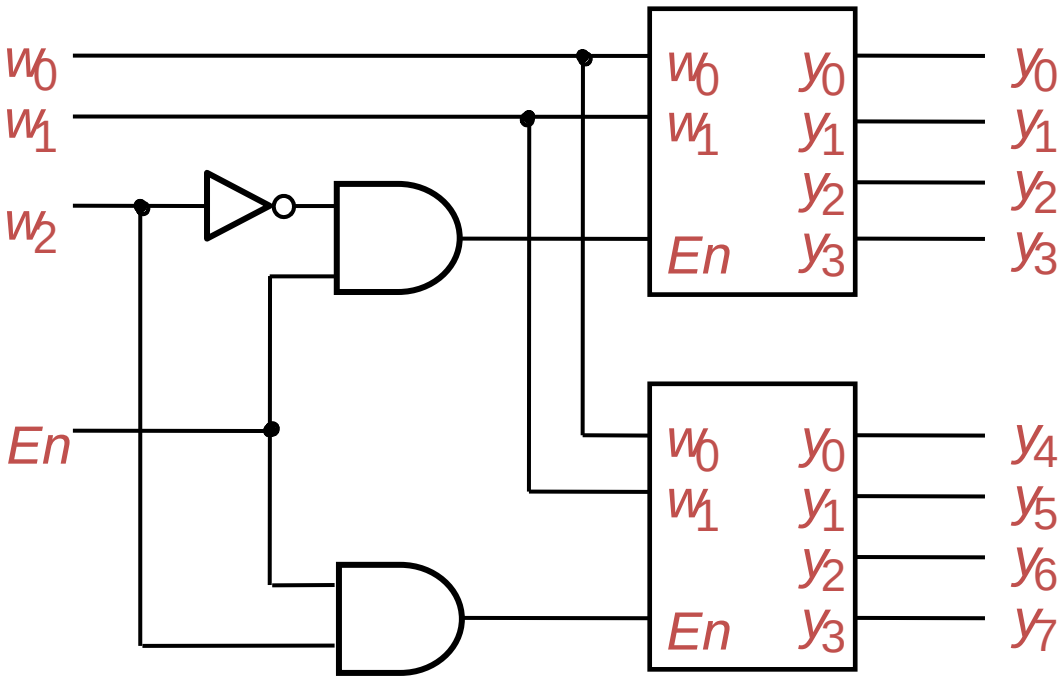
(b) Graphic symbol



(c) Logic circuit

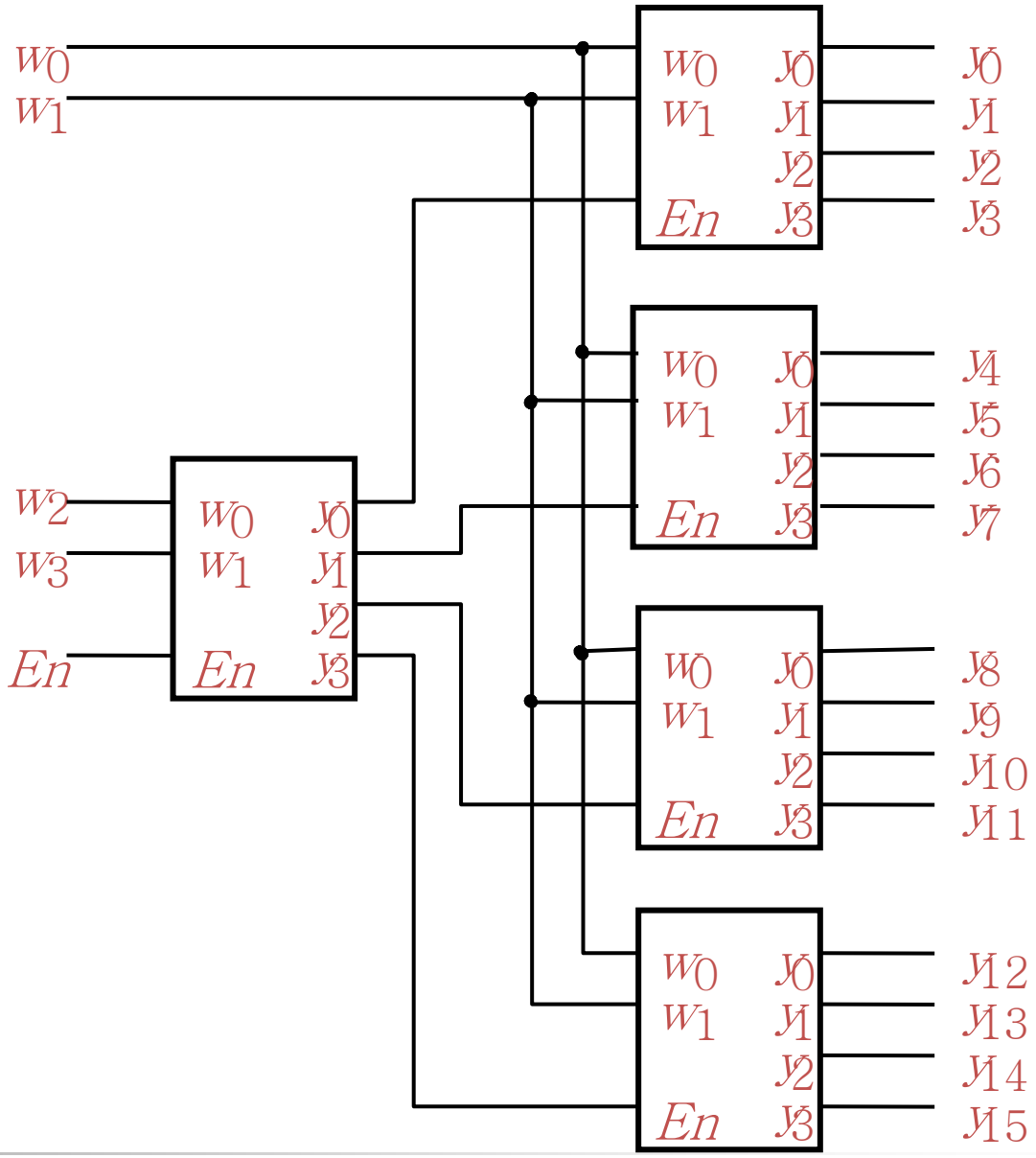
DECODER INTRODUCTION [3 OF 9]

- 3 to 8 Decoder using two 2 to 4 DECODERS



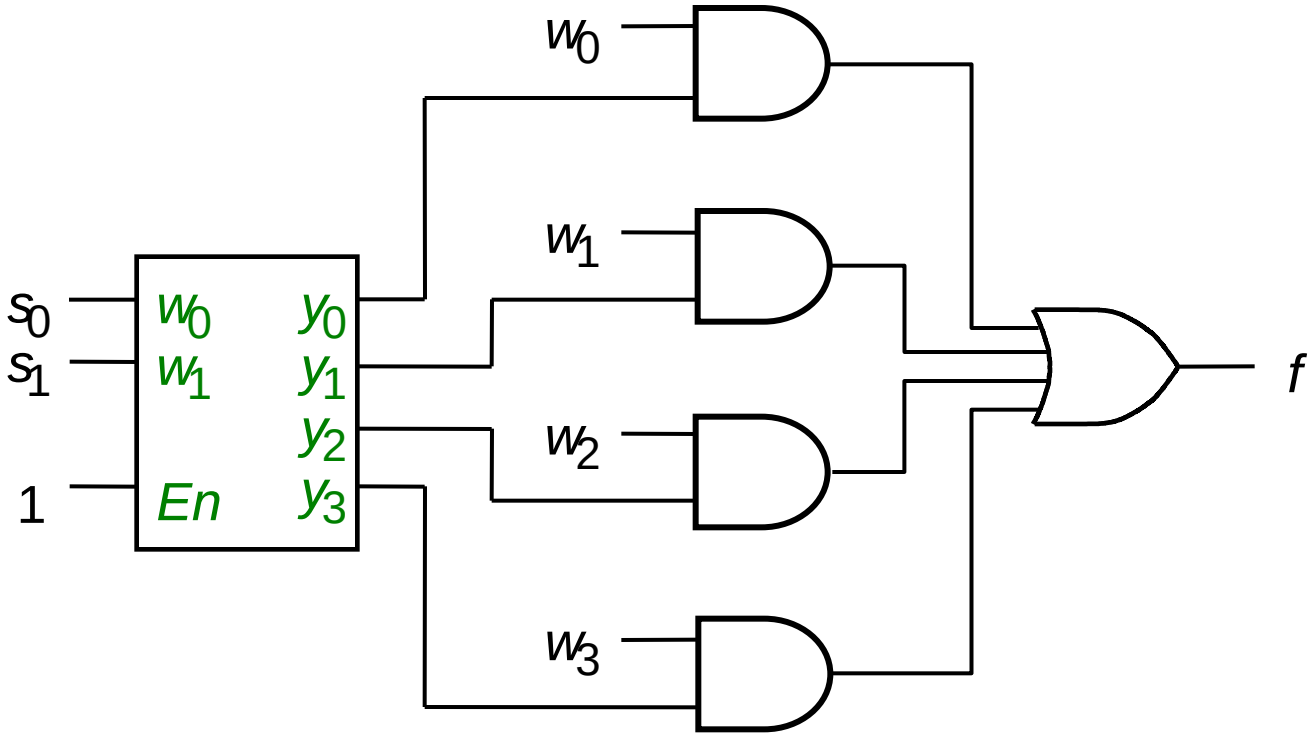
DECODER INTRODUCTION [4 OF 9]

- 4 to 16 DECODER



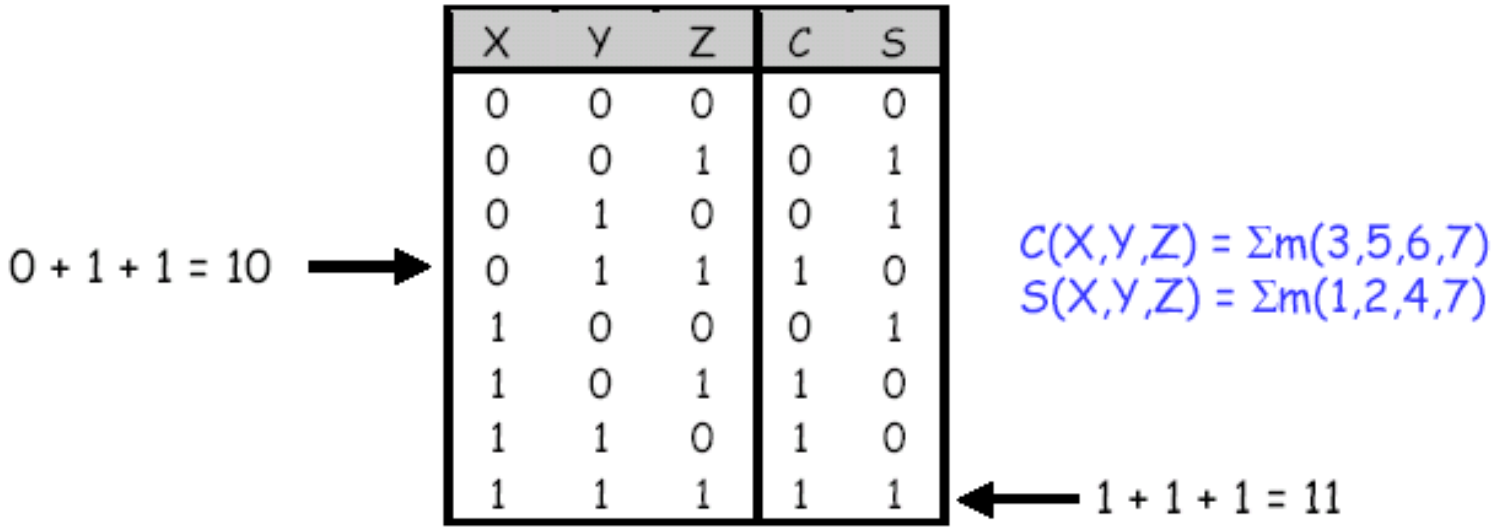
DECODER INTRODUCTION [5 OF 9]

- DECODER를 이용한 4 to 1 MUX 구현



DECODER INTRODUCTION [6 OF 9]

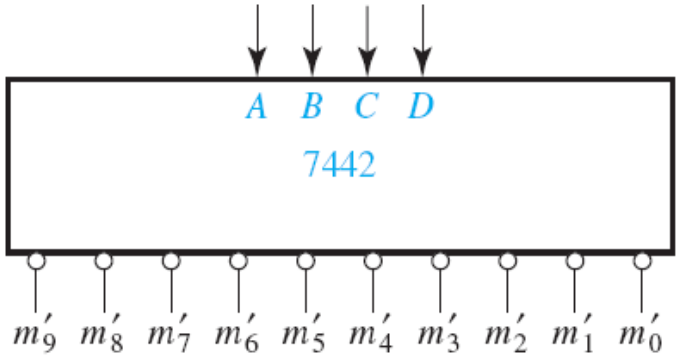
- DECODER를 이용한 일반 함수의 구현
  - decoder 만으로 어떤 함수도 구현할 수 있다. (decoder = minterm 생성기)

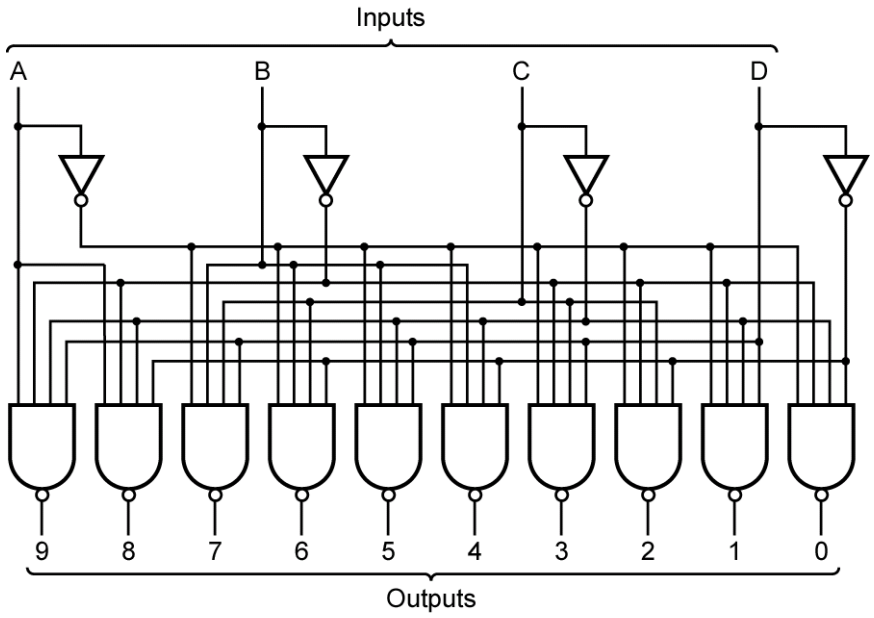


X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

DECODER INTRODUCTION [7 OF 9]

- 반전 출력을 가지는 디코더





(a) Logic diagram

- 4-to-10 **디코더**
- **나머지** ABCD **입력은 모두** 1
- **2진수 입력에 해당하는** 10**진수 출력**

BCD Input				Decimal Output									
A	*B*	*C*	*D*	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1

DECODER INTRODUCTION [8 OF 9]

- 일반적인 n-to-2^n 라인 디코더는 n개의 입력에 대해 아래와 같은 출력을 가진다.

$$y_i = m_i, \qquad i = 0 \quad \text{to} \quad 2^n - 1 \qquad \text{(noninverted outputs)}$$

$$or$$

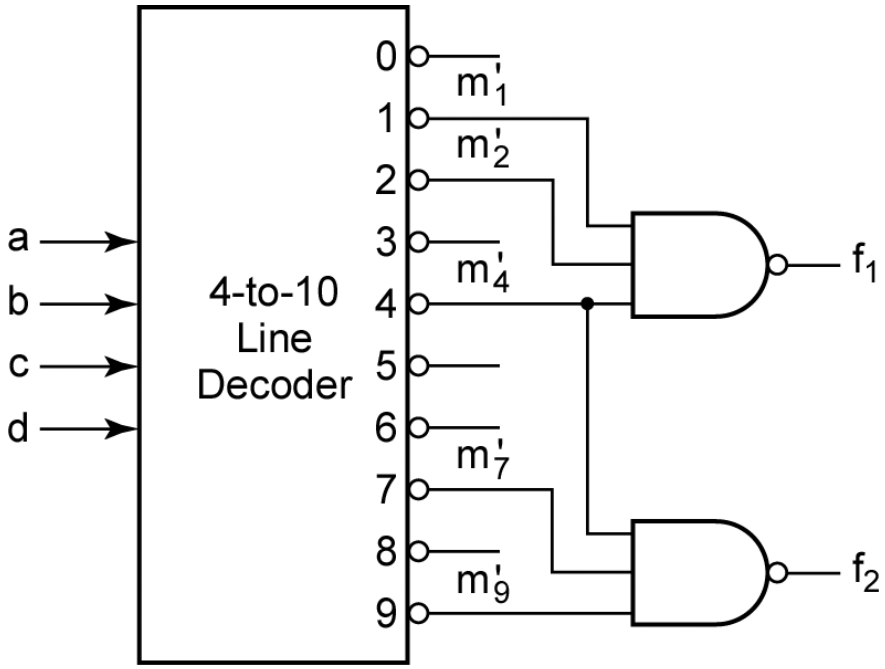
$$y_i = m_i' = M_i, \quad i = 0 \quad \text{to} \quad 2^n - 1 \qquad \text{(inverted outputs)}$$

  - m은 n개의 입력변수의 최소항
  - M은 n개의 입력변수의 최대항

- n입력 디코더를 이용하여 n-변수 함수를 최소항 출력들을 선택한 뒤 OR 함으로써 구현 가능
  - 디코더 출력을 반전하면 NAND 게이트로 구현 가능

DECODER INTRODUCTION [9 OF 9]

- DECODER 출력 반전 – NAND gate를 이용하여 함수 생성

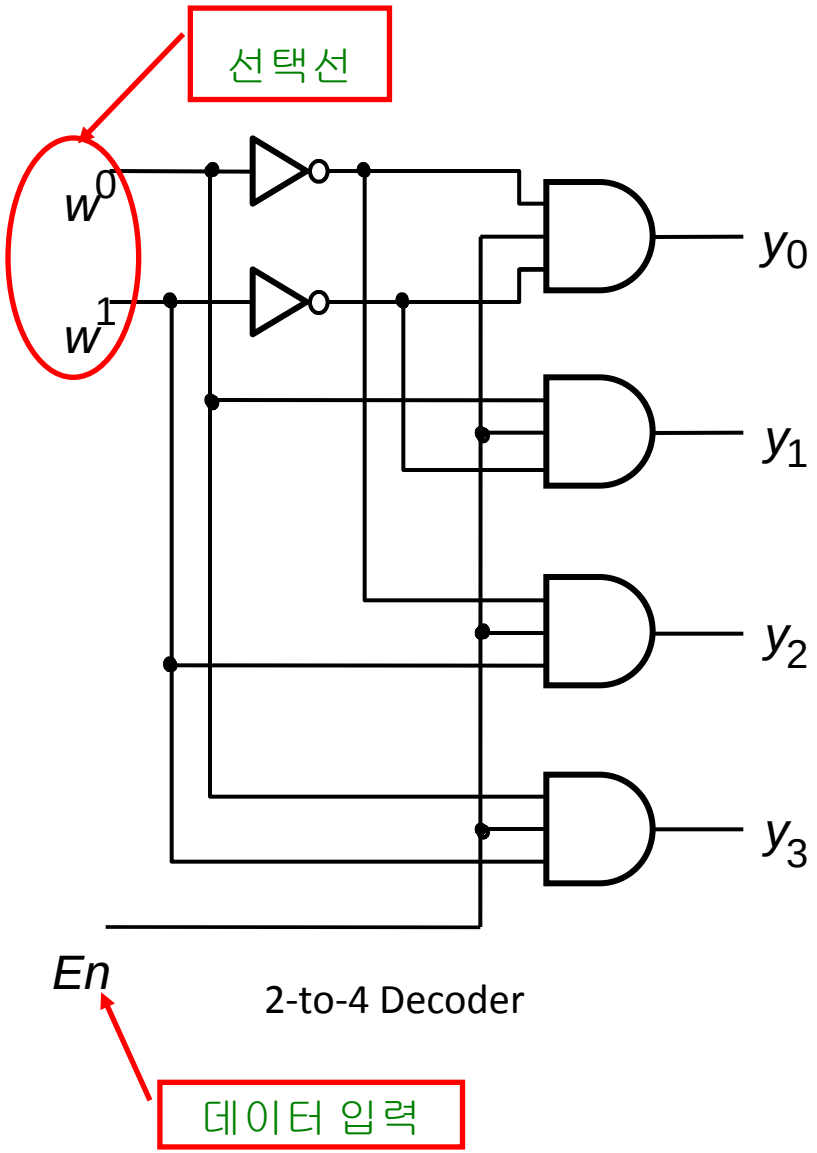


$$f_1(a,b,c,d) = m_1 + m_2 + m_4$$
$$= (m_1' m_2' m_4')'$$

$$f_2(a,b,c,d) = m_4 + m_7 + m_9$$
$$= (m_4' m_7' m_9')'$$

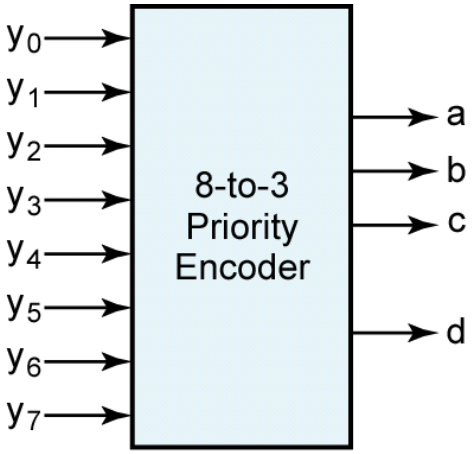
DEMULTIPLEXER (DEMUX)

- DEMUX란?
  - 하나의 데이터 입력에 있는 값을 다수의 데이터 출력에 연결하는 회로
  - decoder를 이용하여 구현 가능
  - 일반적으로 *n*-to-2^n decoder는 1-to-*n* demultiplexer로 사용할 수 있다.



ENCODER INTRODUCTION

- ENCODER
  - DECODER의 Inverse Function
  - 2^n-to-n 입출력



y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	a	b	c	d
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1

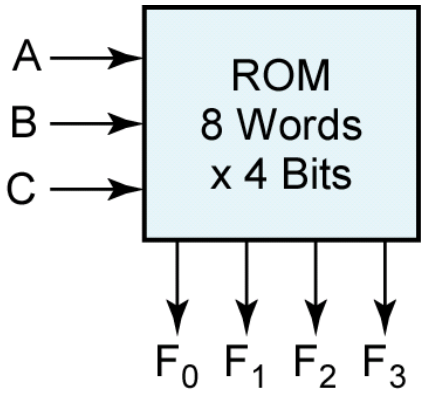
8 to 3 ENCODER

- 입력의 하나 이상이 '1'인 경우 우선순위에 의해 결정됨

3 ROM (READ ONLY MEMORY)

ROM INTRODUCTION [1 OF 7]

- ROM(Read-Only Memory)
  - 2진 데이터 배열을 저장하기 위해 상호 연결된 반도체 소자의 배열로 구성
  - 읽을 수는 있지만 변경할 수는 없음
  - 예시: 3개의 입력선과 4개의 출력선을 가지는 ROM



A B C	F_0 F_1 F_2 F_3
0 0 0	1 0 1 0
0 0 1	1 0 1 0
0 1 0	0 1 1 1
0 1 1	0 1 0 1
1 0 0	1 1 0 0
1 0 1	0 0 0 1
1 1 0	1 1 1 1
1 1 1	0 1 0 1

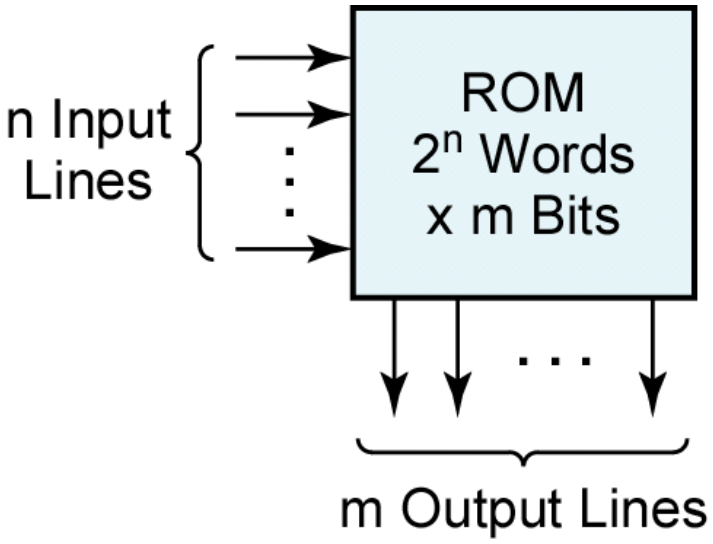
Typical Data Stored in ROM (2^3 words of 4 bits each)

(b) Truth table for ROM

  - ROM에 저장되는 각각의 출력 패턴을 워드(word)라 부른다.

ROM INTRODUCTION [2 OF 7]

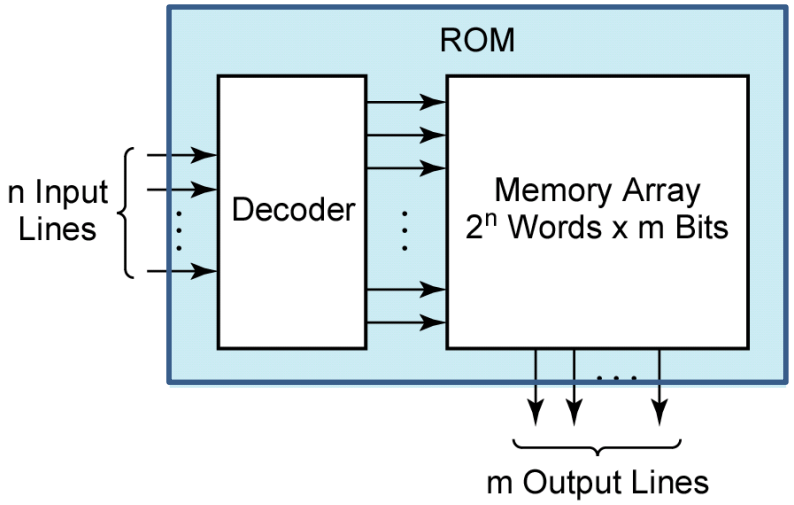
- n개의 입력을 가지는 경우 2^n가지의 다른 조합을 가짐
  - 입력 선은 2^n 개의 워드 중 하나의 주소의 역할
- m개의 출력을 가지면 워드의 길이는 m비트
- n개의 변수를 가지는 m개의 함수 구현 가능



n Input Variables	*m* Output Variables
00 · · · 00	100 · · · 110
00 · · · 01	010 · · · 111
00 · · · 10	101 · · · 101
00 · · · 11	110 · · · 010
⋮	⋮
11 · · · 00	001 · · · 011
11 · · · 01	110 · · · 110
11 · · · 10	011 · · · 000
11 · · · 11	111 · · · 101

ROM INTRODUCTION [3 OF 7]

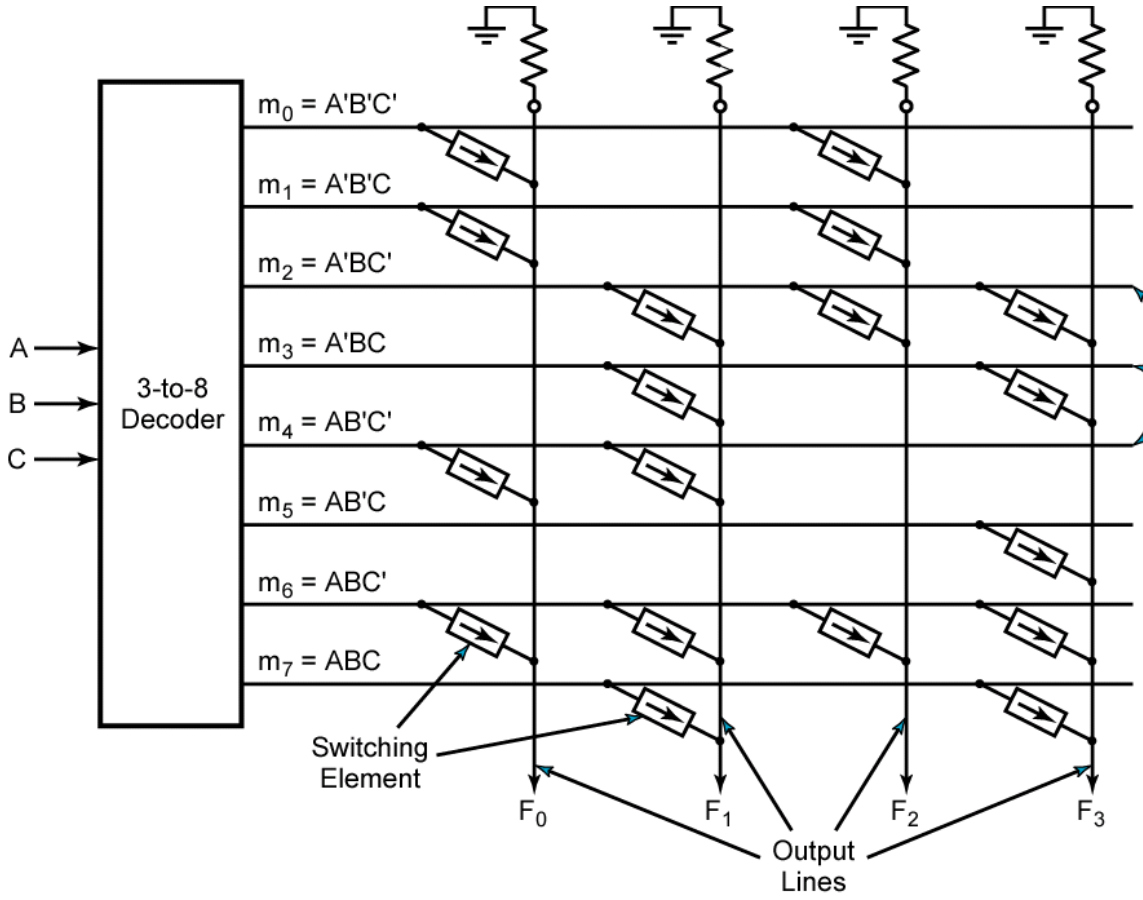
- 기본적인 ROM 구조



- n개의 패턴이 DECODER에 인가
- DECODER 출력 중 하나가 '1'이 됨
- 이 출력 선 하나가 메모리 배열의 하나인 워드를 선택하여 출력

ROM INTRODUCTION [4 OF 7]

- 8word 4bit ROM 내부 구조 예시

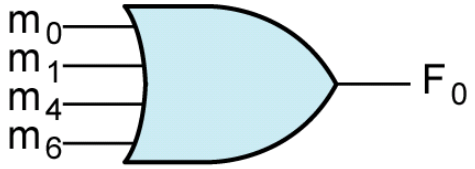


$$F_0 = \sum m(0,1,4,6) = A'B'+AC'$$

$$F_1 = \sum m(2,3,4,6,7) = B + AC'$$

$$F_2 = \sum m(0,1,2,6) = A'B'+BC'$$

$$F_3 = \sum m(2,3,5,6,7) = AC + B4$$

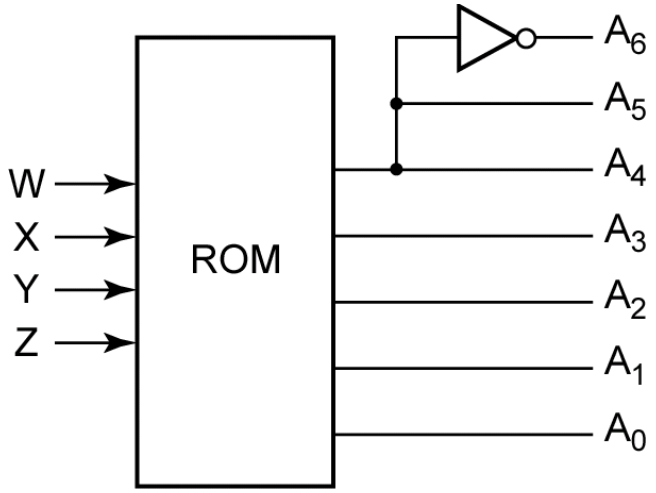


$$F_0 = \sum m(0,1,4,6) = A'B'+AC'$$

ROM INTRODUCTION [5 OF 7]

- ROM을 이용한 다중 출력 조합회로 구현
  - 예제: 16진수-ASCII Code 변환기

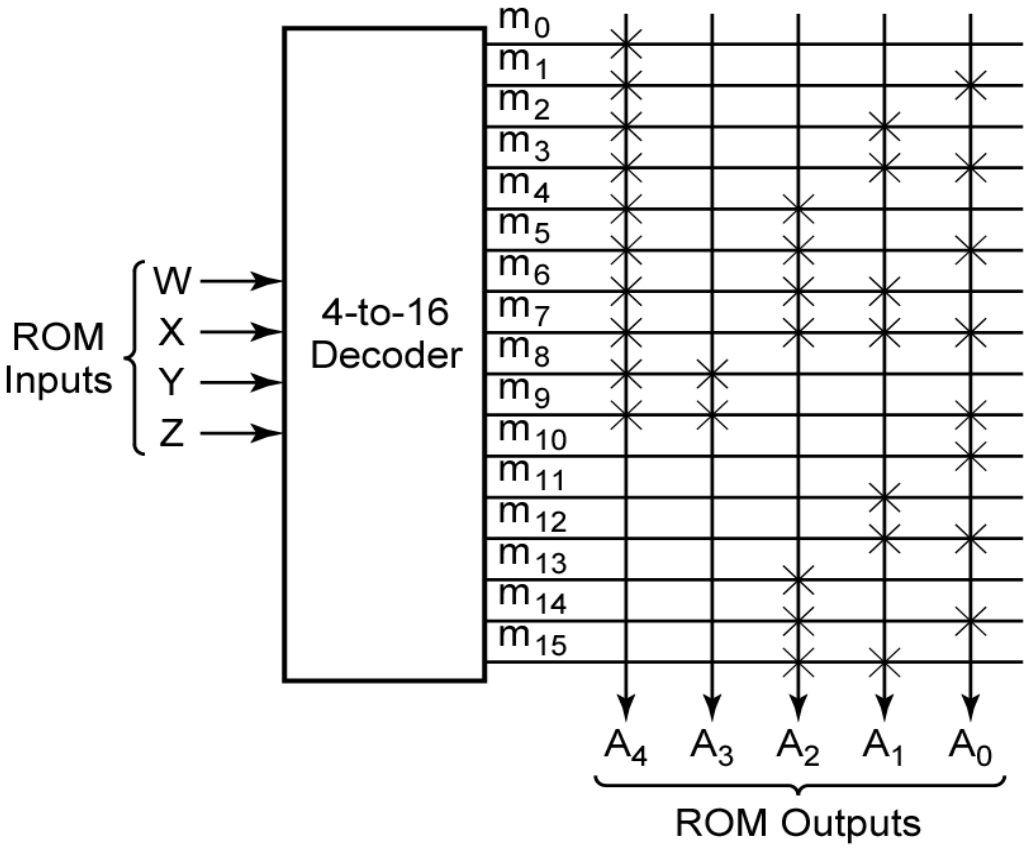
Input W X Y Z	Hex Digit	ASCII Code for Hex Digit A_6 A_5 A_4 A_3 A_2 A_1 A_0
0 0 0 0	0	0 1 1 0 0 0 0
0 0 0 1	1	0 1 1 0 0 0 1
0 0 1 0	2	0 1 1 0 0 1 0
0 0 1 1	3	0 1 1 0 0 1 1
0 1 0 0	4	0 1 1 0 1 0 0
0 1 0 1	5	0 1 1 0 1 0 1
0 1 1 0	6	0 1 1 0 1 1 0
0 1 1 1	7	0 1 1 0 1 1 1
1 0 0 0	8	0 1 1 1 0 0 0
1 0 0 1	9	0 1 1 1 0 0 1
1 0 1 0	A	1 0 0 0 0 0 1
1 0 1 1	B	1 0 0 0 0 1 0
1 1 0 0	C	1 0 0 0 0 1 1
1 1 0 1	D	1 0 0 0 1 0 0
1 1 1 0	E	1 0 0 0 1 0 1
1 1 1 1	F	1 0 0 0 1 1 0



- 4비트 2진수를 16비트로 변환
- 7비트 ASCII Code 출력
- 진리표의 $A_4A_3A_2A_1A_0$이 ROM에 저장

ROM INTRODUCTION [6 OF 7]

- 스위칭 소자는 x 표시로 나타낼 수 있음
- 위 예제의 내부 도표



ROM INTRODUCTION [7 OF 7]

- 대표적인 ROM의 종류
  - Mask-Programmable ROM
    - 생산공정시 데이터 배열이 영구적으로 저장
  - PROM(Programmable ROM)
    - 아무 내용이 들어있지 않은 빈 상태로 제조하여 공급되고 사용자가 PROM 라이터를 이용하여 내용을 써넣을 수 있음
  - EEPROM(Electrically Erasable Programmable ROM)
    - 디지털 시스템 개발 과정 중 이용
    - 전하 저장 장치를 사용하여 재 프로그램 가능
      - cf) 플래시는 프로그래밍 및 소거 기능을 내장하고 있음

THANK YOU