



# Digital Combinational Circuit 1 (with K-map)

**Shinwoong Kim**

# 목표

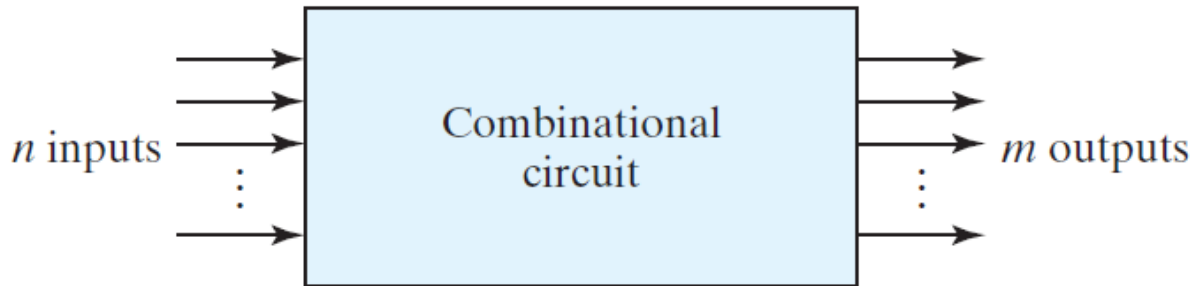
---

- **조합 논리 회로(Combinational logic circuit) 시스템을 설계할 수 있다.**
  - ✓ 진리표를 사용하여 주어진 문제를 나타내는 논리를 표현할 수 있다.
  - ✓ 진리표를 해석하여 필요한 조합회로를 구성할 수 있다.
- **Karnaugh Map (K-map)을 사용하여 로직을 단순화할 수 있다.**
  - ✓ 단순화 된 로직으로 조합 논리 회로를 구성하여 시스템 설계를 할 수 있다.

# Combinational Logic System

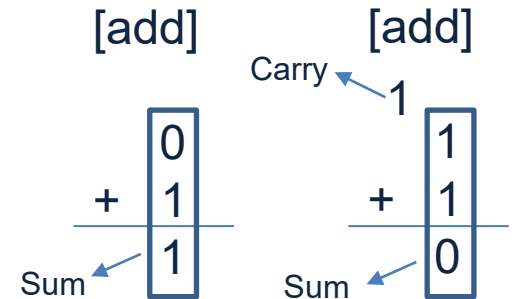
- **Combinational circuit**

- ✓ It consists of an interconnection of **logic gates**
- ✓ Outputs are determined **at any time** from **only the present combination of inputs**
- ✓ It performs an operation that can be specified **logically by a set of Boolean function**
- ✓ **No** feedback path or **memory** elements
- ✓ **No** operating **clock**



# Half Adder (1bit Adder)

- **Basic combinational system**
- **2-input and 2-output**
  - ✓ Two binary input, sum and carry

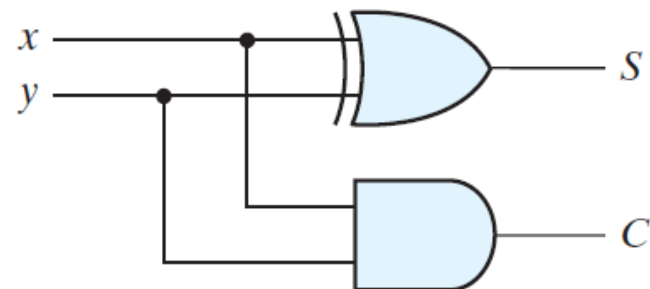
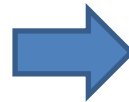
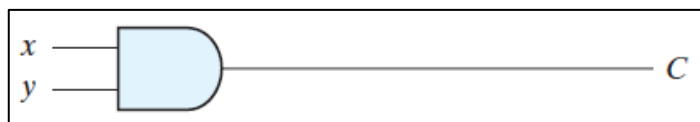
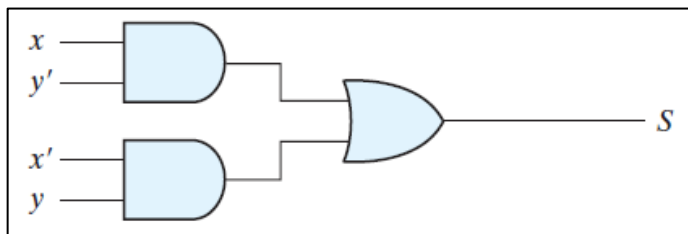
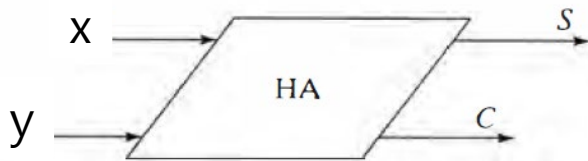


Half Adder

$x$	$y$	$C$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = x'y + xy'$$

$$C = xy$$



$$S = x \oplus y$$

$$C = xy$$

# Karnaugh Map

- Refer to as 'K-map'
  - ✓ Graphical approach to find minimum logic expression
  - ✓ 2-dimensional truth table
  - ✓ K-map consists of each square for each possible minterm in a function.

$x_1$	$x_2$	minterm
0	0	$m_0$
0	1	$m_1$
1	0	$m_2$
1	1	$m_3$

(a) Truth table

		$x_1$	
		0	1
$x_2$	0	$m_0$	$m_2$
	1	$m_1$	$m_3$

(b) Karnaugh map

# Karnaugh Map

- 2-variable K-map

$x_1$	$x_2$	minterm
0	0	$m_0$
0	1	$m_1$
1	0	$m_2$
1	1	$m_3$

(a) Truth table

e.g.,

$x_1$	$x_2$	$f$
0	0	0
0	1	1
1	0	0
1	1	0

		$x_1$	
		0	1
$x_2$	0	$m_0$	$m_2$
	1	$m_1$	$m_3$

(b) Karnaugh map

		$x_1$	
		0	1
$x_2$	0	0	0
	1	1	0

# Karnaugh Map

- 3-variable K-map

$x_1$	$x_2$	$x_3$	minterm
0	0	0	$m_0$
0	0	1	$m_1$
0	1	0	$m_2$
0	1	1	$m_3$
1	0	0	$m_4$
1	0	1	$m_5$
1	1	0	$m_6$
1	1	1	$m_7$

(a) Truth table

e.g.,

$x_1$	$x_2$	$x_3$	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

		$x_1 x_2$			
		00	01	11	10
$x_3$	0	$m_0$	$m_2$	$m_6$	$m_4$
	1	$m_1$	$m_3$	$m_7$	$m_5$

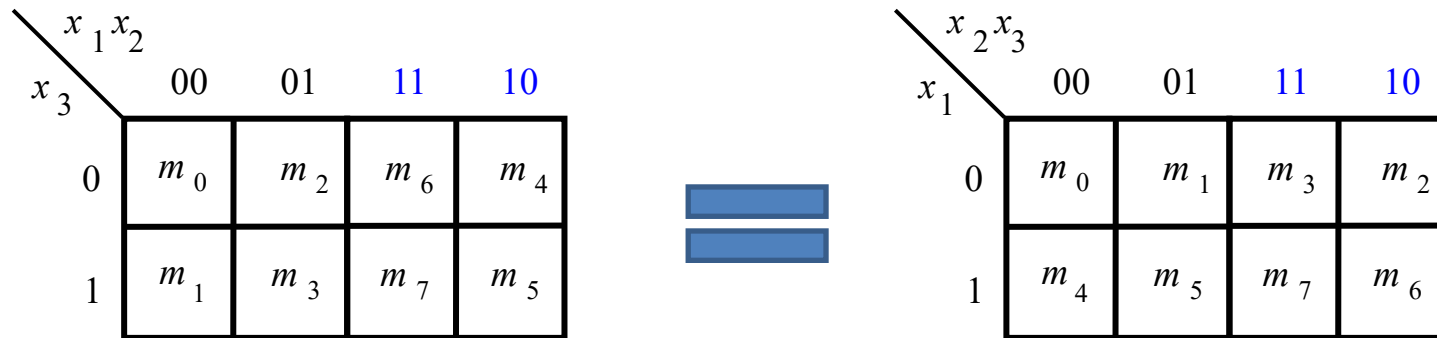
(b) Karnaugh map

		$x_1 x_2$			
		00	01	11	10
$x_3$	0	0	0	0	1
	1	1	0	1	1

# Karnaugh Map

- **3-variable K-map**

- ✓ Row of the map with first variable has the same results



- ✓ **Last two columns are not in numeric order**
  - This is the key idea that makes the map work
  - The minterms in adjacent square can always be combined using the adjacency property

**P9a.**  $ab + ab' = a$



# Karnaugh Map

- 4-variable K-map

$x_3x_4 \backslash x_1x_2$					
		00	01	11	10
$x_3x_4$	00	$m_0$	$m_4$	$m_{12}$	$m_8$
	01	$m_1$	$m_5$	$m_{13}$	$m_9$
	11	$m_3$	$m_7$	$m_{15}$	$m_{11}$
	10	$m_2$	$m_6$	$m_{14}$	$m_{10}$

Karnaugh map

e.g.

$x_1$	$x_2$	$x_3$	$x_4$	$f$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

$x_3x_4 \backslash x_1x_2$					
		00	01	11	10
00	0	1	0	0	
01	1	1	1	1	
11	0	1	1	0	
10	0	1	0	0	

# Minimum SOP using K-map

- **Square grouping in K-map**

- ✓ Adjacent squares differ only by one literal in minterm expression
- ✓ If the value of adjacent squares is '1', they can be combined and reduced
- ✓ Using the property **P9a.**  $ab + ab' = a$

b \ a	0	1
	0	1
0	$a'b'$ (m0)	$ab'$ (m2)
1	$a'b$ (m1)	$ab$ (m3)

K-map for  $f(a,b)$

e.g.

a b	f
0 0	0
0 1	1
1 0	1
1 1	1

$$\begin{aligned}
 f &= m1 + m2 + m3 \\
 &= \sum m(1,2,3) \\
 &= a'b + ab' + ab \text{ (SOP)} \\
 &\rightarrow a + b \text{ (min. SOP)}
 \end{aligned}$$

b \ a	0	1
	0	1
0	0	1
1	1	1

$\rightarrow a$   
 $\rightarrow b$

10

# How to Group?

---

- **Principles of K-map grouping**
  - ✓ Grouping only logic '1' squares
  - ✓ Grouping only rectangles
    - # of square in each rectangle: 1, 2, 4, 8, 16, or 32
  - ✓ Groups can be overlapped
  - ✓ Grouping around boundary is OK
- **To get an optimal grouping?**
  - ✓ Minimize the # of groups
  - ✓ Make each group as big as possible

# Example #1 : 2-variable K-map

e.g., 1

b \ a	0	1
	0	1
0	1	0
1	1	1

$$f(a,b) = a' + b$$

e.g., 2

b \ a	0	1
	0	1
0	1	0
1	1	0

$$f(a,b) = a'$$

e.g., 3

b \ a	0	1
	0	1
0	1	0
1	0	1

$$f(a,b) = a'b' + ab$$

# Example #2 : 3-variable K-map

e.g., 1

<b>c</b> \ a b	00	01	11	10
0	1	1	1	1
1	0	0	0	1

$$f(a,b,c) = \sum m(0,2,4,5,6)$$

$$= \mathbf{c'} + \mathbf{ab'}$$

<b>a</b> \ b c	00	01	11	10
0	1	0	0	1
1	1	1	0	1

$$f(a,b,c) = \sum m(0,2,4,5,6)$$

$$= \mathbf{c'} + \mathbf{ab'}$$

<b>a</b> \ b c	0	1
00	1	1
01	0	1
11	0	0
10	1	1

$$f(a,b,c) = \sum m(0,2,4,5,6)$$

$$= \mathbf{c'} + \mathbf{ab'}$$

## Example #2 : 3-variable K-map

e.g., 2

$\begin{array}{c} a \ b \\ \swarrow \searrow \\ c \end{array}$	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$f(a,b,c) = ac' + b'c$$

e.g., 3

$\begin{array}{c} a \ b \\ \swarrow \searrow \\ c \end{array}$	00	01	11	10
0	0	1	0	1
1	0	0	1	1

$$f(a,b,c) = a'bc' + ac + ab'$$

# Example #3 : 4-variable K-map

e.g., 1

c d \ a b	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	0	0	1
10	1	0	0	1

$$f(a,b,c,d) = ac'd + b'c$$

e.g., 2

c d \ a b	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	1	1	1
10	1	1	1	1

$$f(a,b,c,d) = ad + c$$

e.g., 3

c d \ a b	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	1	1	1	0
10	1	1	0	1

$$f(a,b,c,d) = b'd' + a'c + bcd$$

e.g., 4

c d \ a b	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	0	0	1	1
10	0	0	1	1

$$f(a,b,c,d) = a'c' + ab + ac$$

15

# 실습

## • 자판기 설계 (Vending machine)

### ✓ 문제정의

- 음료수 선택: 500원, 600원 음료수 2가지 종류가 있음 (1bit, S)
- 금액 입력: 500원 단위 (1bit,  $N_2$ ), 100원 단위 (2bits,  $N_1N_0$ ) 입력
- 출력: 음료수 출력 표시 ( $G_1, G_0$ ), 100원 단위 거스름돈 (2bits,  $C_1C_0$ )
- 즉, 입력은 총 4bits ( $S N_2 N_1 N_0$ ), 출력 총 4bits ( $G_1 G_0, C_1 C_0$ )

S: 제품선택

S	제품
0	500원
1	600원

$N_2N_1N_0$ : 투입금

$N_2$	금액	$N_1N_0$	금액
0	0원	00	0원
1	500원	01	100원
		10	200원
		11	300원



$G_1$ : 600원 제품 출력  
 $G_0$ : 500원 제품 출력

$C_1C_0$	잔돈 출력
00	0원
01	100원
10	200원
11	300원



# 실습

## • 자판기 설계 (Vending machine)

✓ 진리표

		입력				출력			
제품	금액	S	N2	N1	N0	G1	G0	C1	C0
500원 제품 선택	0원	0	0	0	0	0	0	0	0
	100원	0	0	0	1	0	0	0	0
	200원	0	0	1	0	0	0	0	0
	300원	0	0	1	1	0	0	0	0
	500원	0	1	0	0	0	1	0	0
	600원	0	1	0	1	0	1	0	1
	700원	0	1	1	0	0	1	1	0
	800원	0	1	1	1	0	1	1	1
600원 제품 선택	0원	1	0	0	0	0	0	0	0
	100원	1	0	0	1	0	0	0	0
	200원	1	0	1	0	0	0	0	0
	300원	1	0	1	1	0	0	0	0
	500원	1	1	0	0	0	0	0	0
	600원	1	1	0	1	1	0	0	0
	700원	1	1	1	0	1	0	0	1
	800원	1	1	1	1	1	0	1	0

- ✓ S: 제품 선택
  - 1'b0=500원 음료
  - 1'b1=600원 음료

- ✓ N2: 금액 입력 (500원 단위)
  - 1'b0=0원
  - 1'b1=500원

- ✓ N1N0: 금액 입력 (100원 단위)
  - 2'b00=0원
  - 2'b01=100원
  - 2'b10=200원
  - 2'b11=300원

- ✓ G1G0: 음료 출력 구분
  - If G1
    - 1'b0= 600원 음료 미출력
    - 1'b1= 600원 음료 출력
  - If G0
    - 1'b0= 500원 음료 미출력
    - 1'b1= 500원 음료 출력

- ✓ C1C0: 거스름돈 구분
  - 2'b00=0원
  - 2'b01=100원
  - 2'b10=200원
  - 2'b11=300원

# 실습

- (1) Boolean 수식 유도 (K-map 사용)

✓ G1, G0

➤ G1

S N2 \ N1 N0	00	01	11	10
00				
01				
11				
10				

G1 =

➤ G0

S N2 \ N1 N0	00	01	11	10
00				
01				
11				
10				

G0 =

# 실습

- (1) Boolean 수식 유도 (K-map 사용)

✓ C1, C0

➤ C1

S N2 \ N1 N0	00	01	11	10
00				
01				
11				
10				

C1 =

➤ C0

S N2 \ N1 N0	00	01	11	10
00				
01				
11				
10				

C0 =

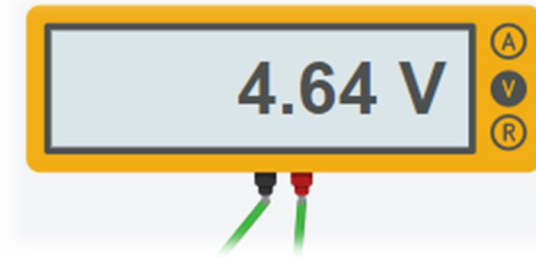
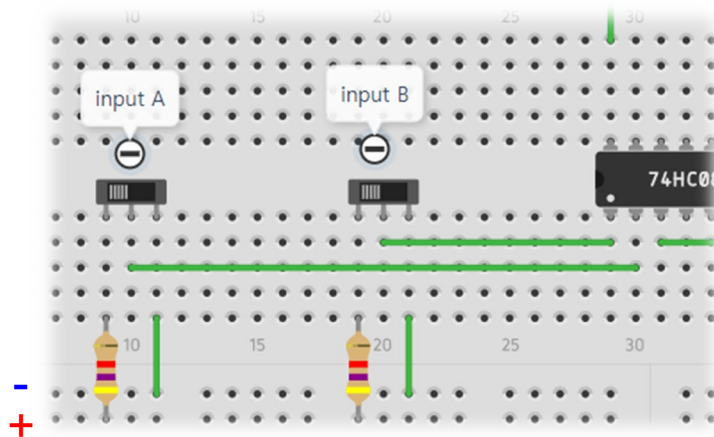
- (1) Boolean 수식 유도 (K-map 사용)

- ✓ **[참고]** K-map을 minterm을 이용하여 구하면, 즉 logic 1을 기준으로 묶어서 구하면 SOP type으로만 표현이 됨
- ✓ 즉,  $Y = ABC + ABD$ 의 형태를 가짐
  - 이 경우 실제 회로로 구현하는 경우, 현재 2-input 로직 게이트만 사용 가능하므로, AND gate 3개, OR gate 1개가 필요함
- ✓ 그러나 추가로 Distributive 성질을 이용하여 Boolean 수식을 줄이게 되면,
  - $Y = AB(C+D)$ 가 되므로, AND gate 2개, OR gate 1개만 사용하면 됨
  - 따라서, K-map 사용 이후, 수식의 형태를 조금 더 바꾸면 회로 구현하기 그나마 쉬워 짐

# 실습

## • (2) TinkerCAD 활용하여 Vending machine 구현

- ✓ TinkerCAD에서 Breadboard는 2개만 사용 (EEBoard 사이즈 고려)
  - 특별히 입력 제어로는 '슬라이드 스위치' 사용 → 조교 확인을 위해 TinkerCAD simulation 수행 중 입력 변경이 용이함
  - TinkerCAD에서 회로 출력은 LED 연결할 필요 없으며 멀티미터 출력만 연결해 놓으면 됨



- ✓ 사전에 해당 내용을 미리 수행해도 되며, 이 경우 실험 당일 날에는 조교 검사만 받고 바로 퇴실 가능 ☺