Chapter 2 :: Combinational Logic Design (조합 논리 설계)

Digital Design and Computer Architecture

David Money Harris and Sarah L. Harris

Lectured by Jeong-Gun Lee @ Hallym

Chapter 2 :: Topics

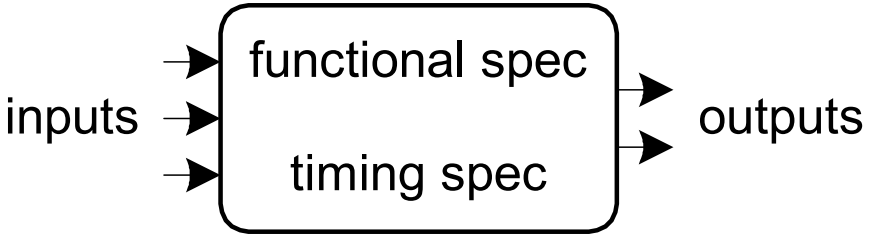
- **Introduction (소개)**
- **Boolean Equations (부울 식, 이진논리식)**
- **Boolean Algebra (부울대수)**
- **From Logic to Gates (논리에서 회로로…)**
- **Multilevel Combinational Logic (다층 조합회로)**
- **X's and Z's, Oh My (X와 Z?)**
- **Karnaugh Maps (카르노 맵)**
- **Combinational Building Blocks (조합회로 블록)**
- **Timing (타이밍)**

Introduction

A logic circuit is composed of:

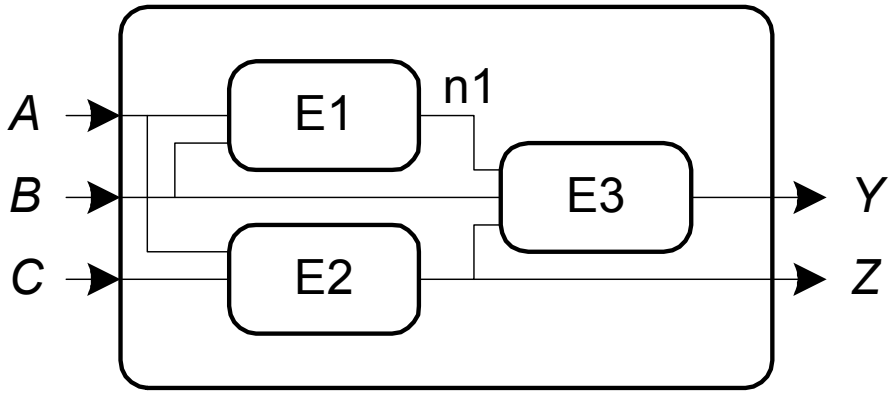
(논리 회로는 다음 등으로 이루어진다)

- Inputs (입력)
- Outputs (출력)
- Functional specification (기능)
- Timing specification (타이밍)



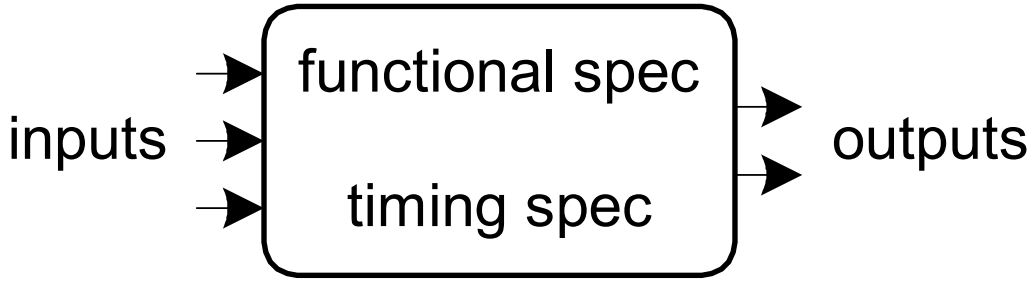
Circuits (회로)

- Nodes (노드)
  - Inputs: *A*, *B*, *C*
  - Outputs: *Y*, *Z*
  - Internal: n1
- Circuit elements (회로 구성요소)
  - E1, E2, E3
  - Each a circuit



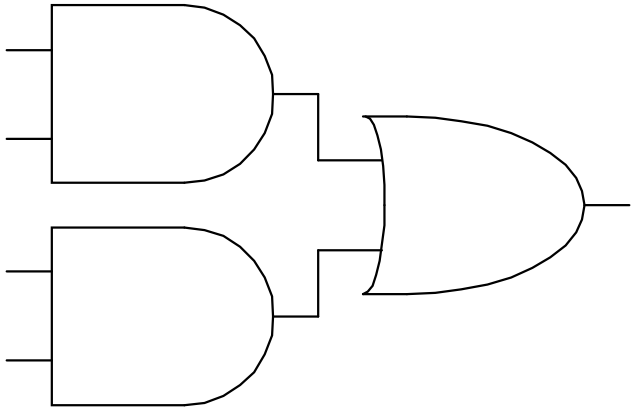
Types of Logic Circuits

- Combinational Logic (조합회로)
  - Memoryless (메모리 없음)
  - Outputs determined by current values of inputs
    (출력이 현재의 입력 값에 의해 결정됨)
- Sequential Logic (조합회로)
  - Has memory (메모리 가짐)
  - Outputs determined by previous and current values of inputs
    (출력이 현재와 이전 입력에 의존하여 결정됨)



Rules of Combinational Composition

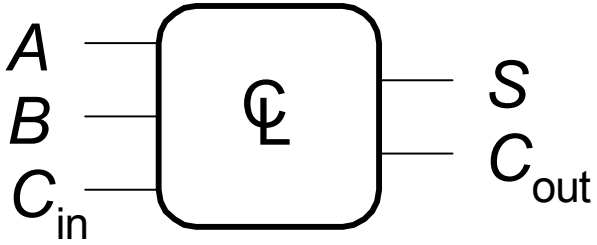
- Every circuit element is itself combinational
- Every node of the circuit is either designated as an input to the circuit or connects to exactly one output terminal of a circuit element
- The circuit contains ***no cyclic paths***: every path through the circuit visits each circuit node at most once
- Example:



Boolean Equations (부울식)

- Functional specification of outputs in terms of inputs
- Example:

$$S = \mathbf{F}(A, B, C_{in})$$
$$C_{out} = \mathbf{F}(A, B, C_{in})$$



$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

Sum-of-Products (SOP) Form

- All Boolean equations can be written in SOP form
- Each row in a truth table has a **minterm**
- A minterm is a product (AND) of literals
- Each minterm is TRUE for that row (and only that row)
- The function is formed by ***ORing the minterms*** for which the output is **TRUE**
- Thus, a sum (OR) of products (AND terms)

A	B	Y	minterm
0	0	0	$\overline{A}\,\overline{B}$
0	1	1	$\overline{A}\,B$
1	0	0	$A\,\overline{B}$
1	1	1	$A\,B$

$$Y = \mathrm{F}(A, B, C) = \overline{A}B + AB$$

Product-of-Sums (POS) Form

- All Boolean equations can be written in POS form
- Each row in a truth table has a **maxterm**
- A maxterm is a sum (OR) of literals
- Each maxterm is FALSE for that row (and only that row)
- The function is formed by ***ANDing the maxterms*** for which the output is **FALSE**
- Thus, a product (AND) of sums (OR terms)

A	B	Y	maxterm
0	0	0	$A + B$
0	1	1	$A + \overline{B}$
1	0	0	$\overline{A} + B$
1	1	1	$\overline{A} + \overline{B}$

$$Y = \mathrm{F}(A, B, C) = (A + B)(\overline{A} + B)$$

SOP = POS ?

$$Y = \mathrm{F}(A, B, C) = \overline{A}B + AB$$
$$= (\overline{A} + A)B = 1B = \boldsymbol{B}$$
$$= (1 + A + \overline{A})B = B + AB + \overline{A}B$$

$$Y = \mathrm{F}(A, B, C) = (A + B)(\overline{A} + B)$$
$$= A\overline{A} + AB + \overline{A}B + BB$$
$$= AB + \overline{A}B + B$$
$$= (A + \overline{A} + 1)B = 1B = \boldsymbol{B}$$

Boolean Equations Example

- You are going to the cafeteria for lunch
  - You won't eat lunch ($\overline{E}$)
  - If it's not open ($\overline{O}$) or
  - If they only serve corndogs (C)

- Write a truth table for determining if you will eat lunch (E).

O	*C*	*E*
0	0	0
0	1	0
1	0	1
1	1	0

SOP & POS Form

- SOP – sum-of-products

O	C	E	minterm
0	0	0	$\overline{A}\ \overline{B}$
0	1	0	$\overline{A}\ B$
1	0	1	$A\ \overline{B}$
1	1	0	$A\ B$

$$Y = A\overline{B}$$

- POS – product-of-sums

O	C	E	maxterm
0	0	0	$A + B$
0	1	0	$A + \overline{B}$
1	0	1	$\overline{A} + B$
1	1	0	$\overline{A} + \overline{B}$

$$Y = (A + B)(A + \overline{B})(\overline{A} + \overline{B})$$

Boolean Algebra

- Set of axioms and theorems to **simplify Boolean equations**
- Like regular algebra, but in some cases simpler because variables can have only two values (1 or 0)
- Axioms and theorems obey the **principles of duality**:
  - ANDs and ORs interchanged, 0's and 1's interchanged

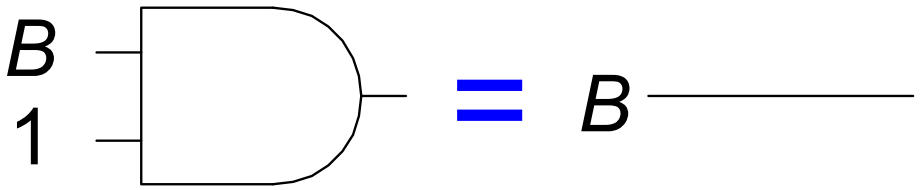
Boolean Axioms and Theorems

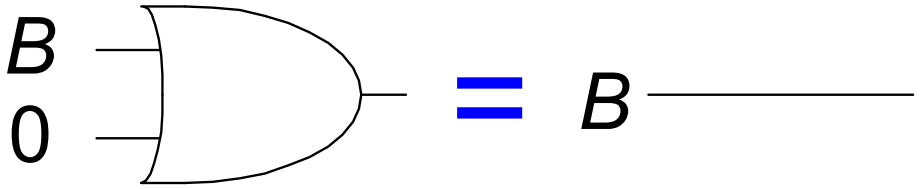
	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

T1: Identity Theorem

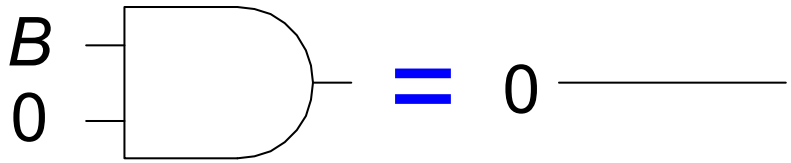
- $B \bullet 1 = B$
- $B + 0 = B$

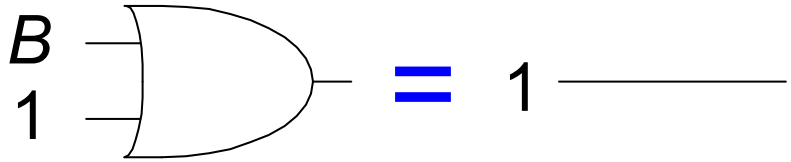




T2: Null Element Theorem

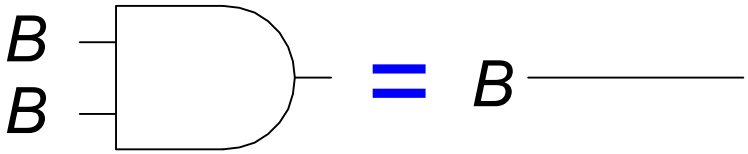
- $B \bullet 0 = 0$
- $B + 1 = 1$

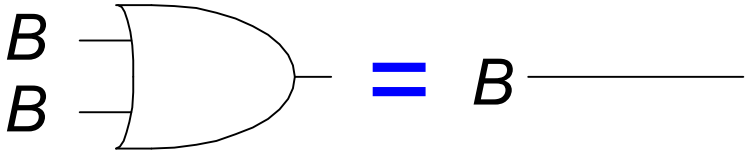




T3: Idempotency Theorem

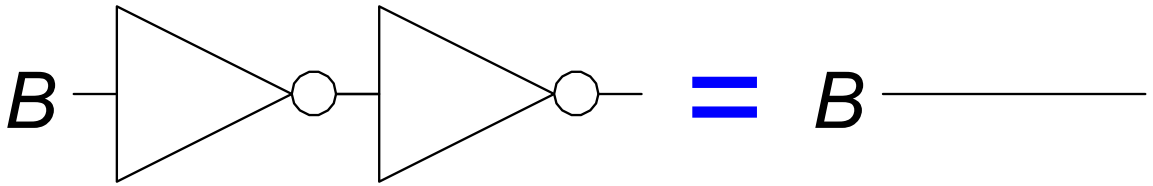
- $B \bullet B = B$
- $B + B = B$





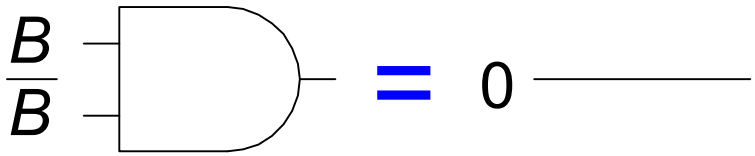
T4: Identity Theorem

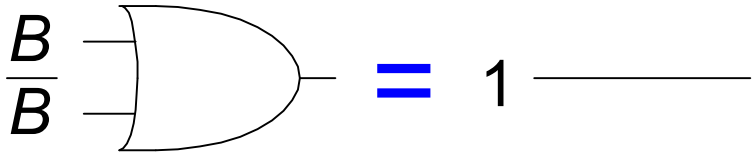
- $\overline{\overline{B}} = B$



T5: Complement Theorem

- $B \bullet \overline{B} = 0$
- $B + \overline{B} = 1$





Boolean Theorems of Several Variables

	Theorem		Dual	Name
T6	$B \bullet C = C \bullet B$	T6′	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	T7′	$(B + C) + D = B + (C + D)$	Associativity
T8	$(B \bullet C) + B \bullet D = B \bullet (C + D)$	T8′	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Distributivity
T9	$B \bullet (B + C) = B$	T9′	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	T10′	$(B + C) \bullet (B + \overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D)$ $= B \bullet C + \overline{B} \bullet D$	T11′	$(B + C) \bullet (\overline{B} + D) \bullet (C + D)$ $= (B + C) \bullet (\overline{B} + D)$	Consensus
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots}$ $= (\overline{B_0} + \overline{B_1} + \overline{B_2} \ldots)$	T12′	$\overline{B_0 + B_1 + B_2 \ldots}$ $= (\overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2})$	De Morgan's Theorem

Simplifying Boolean Expressions: Example 1

- $Y = \overline{A}B + AB$

$= B(\overline{A} + A)$ T8

$= B(1)$ T5’

$= B$ T1

Simplifying Boolean Expressions: Example 2

- $Y = B(AB + ABC)$

$= A(AB(1 + C))$ T8

$= A(AB(1))$ T2’

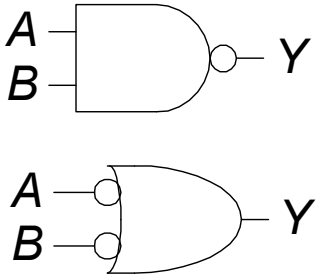
$= A(AB)$ T1

$= (AA)B$ T7

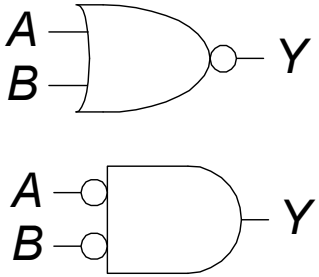
$= AB$ T3

DeMorgan's Theorem

- $Y = \overline{AB} = \overline{A} + \overline{B}$

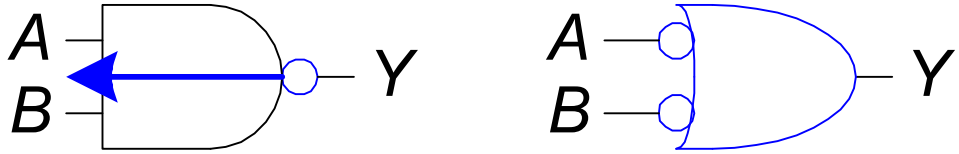


- $Y = \overline{A + B} = \overline{A} \bullet \overline{B}$

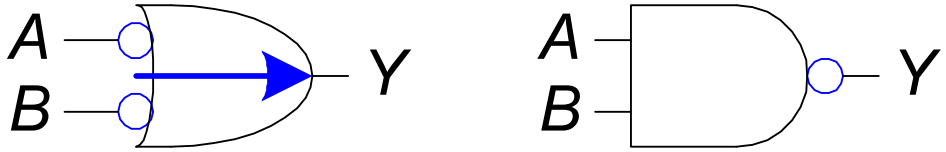


Bubble Pushing

- **Pushing bubbles backward (from the output) or forward (from the inputs) changes the body of the gate from AND to OR or vice versa**.
- Pushing a bubble from the output back to the inputs puts bubbles on all gate inputs.

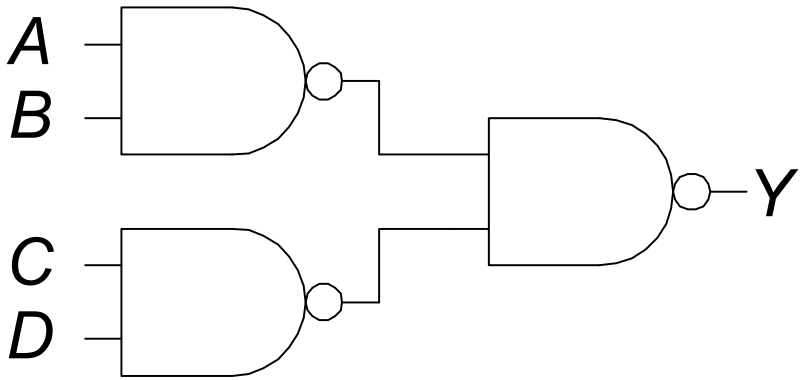


- Pushing bubbles on *all* gate inputs forward toward the output puts a bubble on the output and changes the gate body.



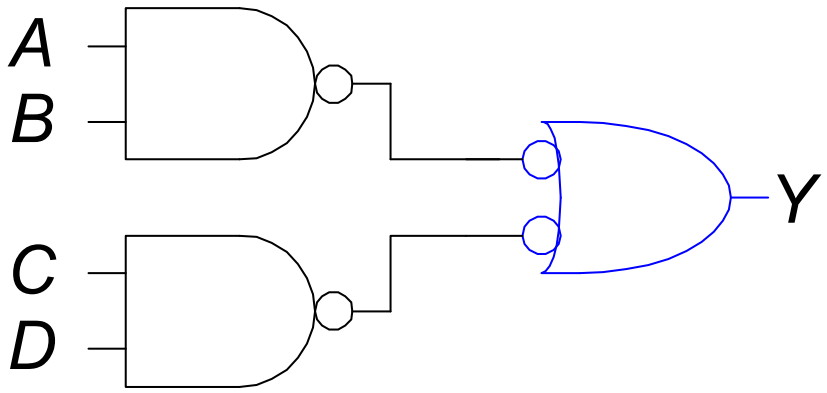
Bubble Pushing

- What is the Boolean expression for this circuit?



Bubble Pushing

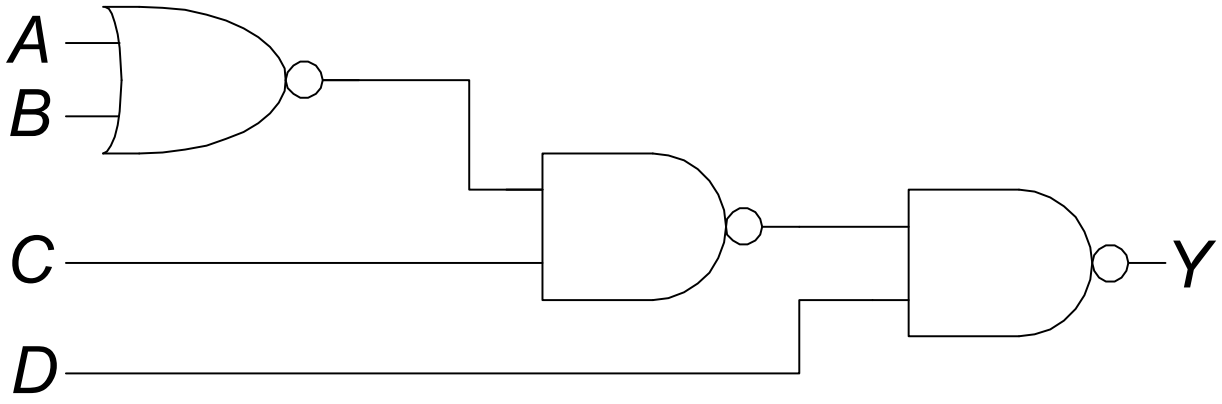
- What is the Boolean expression for this circuit?



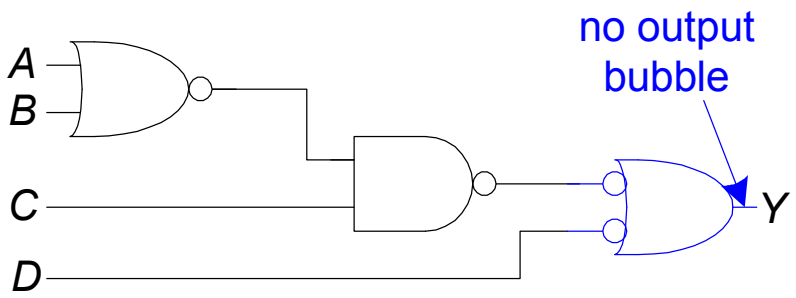
$$Y = AB + CD$$

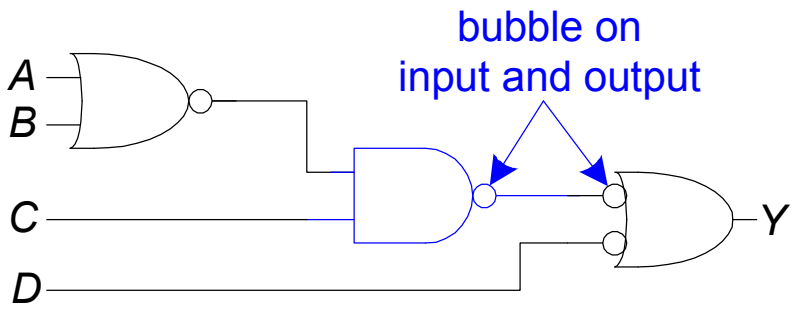
Bubble Pushing Rules

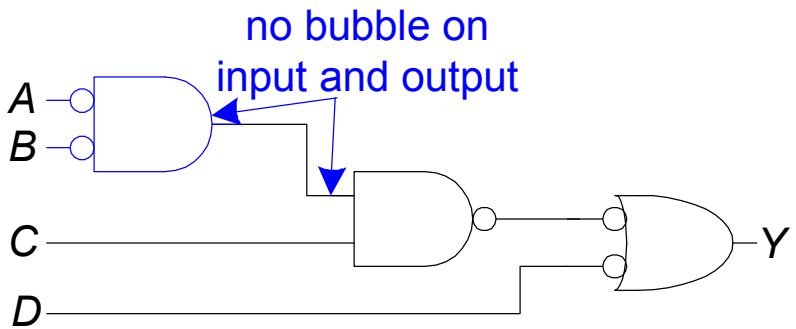
- Begin at the output of the circuit and work toward the inputs.
- Push any bubbles on the final output back toward the inputs.
- Working backward, draw each gate in a form so that bubbles cancel.



Bubble Pushing Rules



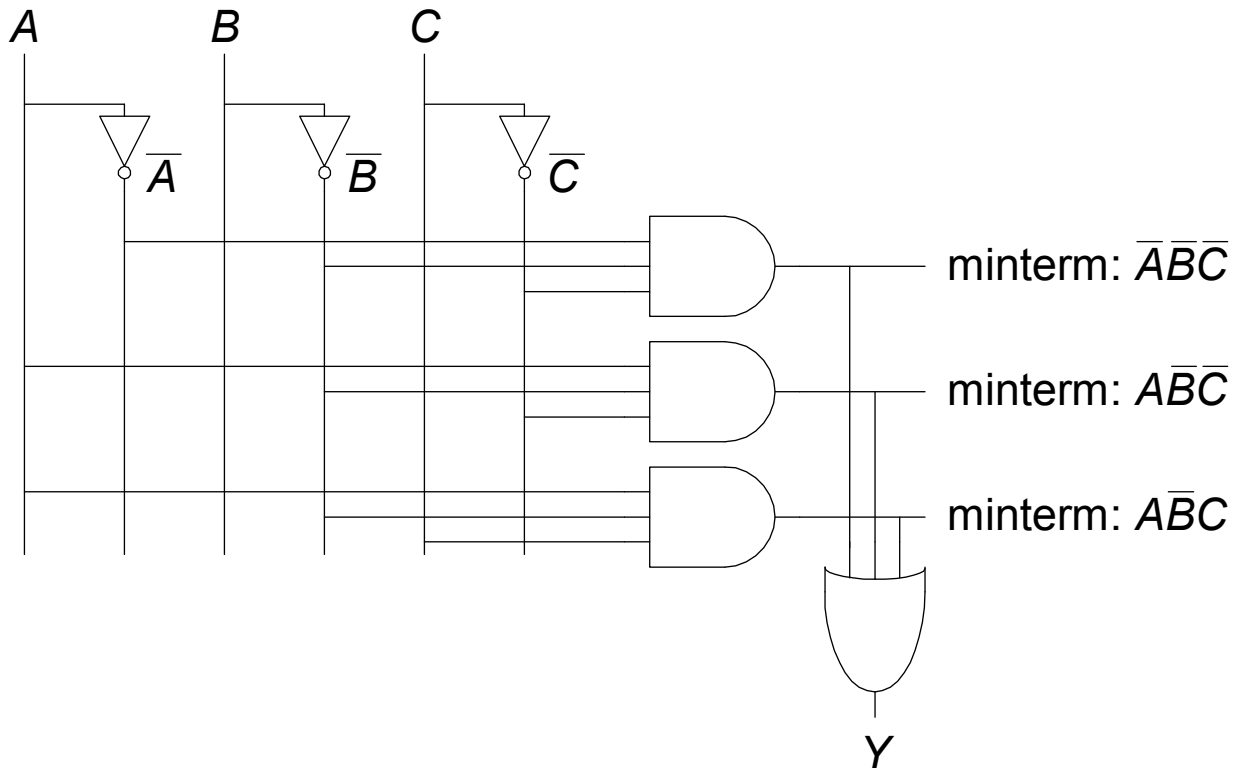




$$Y = \overline{A}\overline{B}C + \overline{D}$$

From Logic to Gates

- Two-level logic: ANDs followed by Ors
- Example: $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$



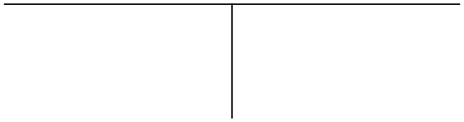
Circuit Schematic Rules
(회로 그림 규칙)

- Inputs are on the left (or top) side of a schematic
  입력은 왼쪽 또는 위쪽에…
- Outputs are on the right (or bottom) side of a schematic
  출력은 오른쪽 또는 아래쪽에…
- Whenever possible, gates should flow from left to right
  가능한 회로는 왼쪽에서 오른쪽으로 데이터가 흐르게…
- Straight wires are better to use than wires with multiple corners
  가능하면 선은 일직선으로…

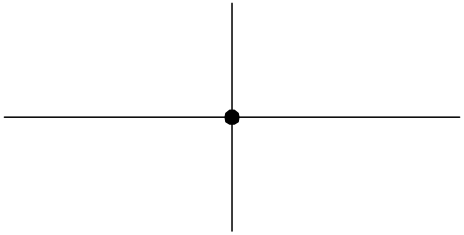
Circuit Schematic Rules (cont.)

- Wires always connect (연결) at a T junction
- A dot where wires cross indicates a connection between the wires
- Wires crossing *without* a dot make no connection

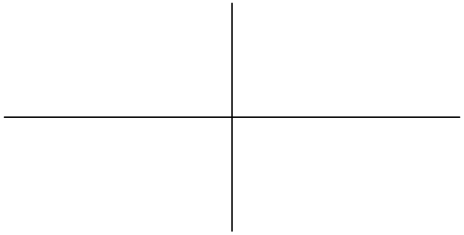
wires connect at a T junction



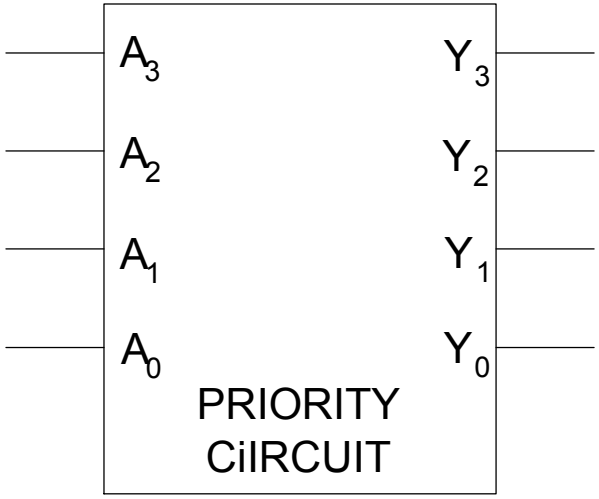
wires connect at a dot



wires crossing without a dot do not connect



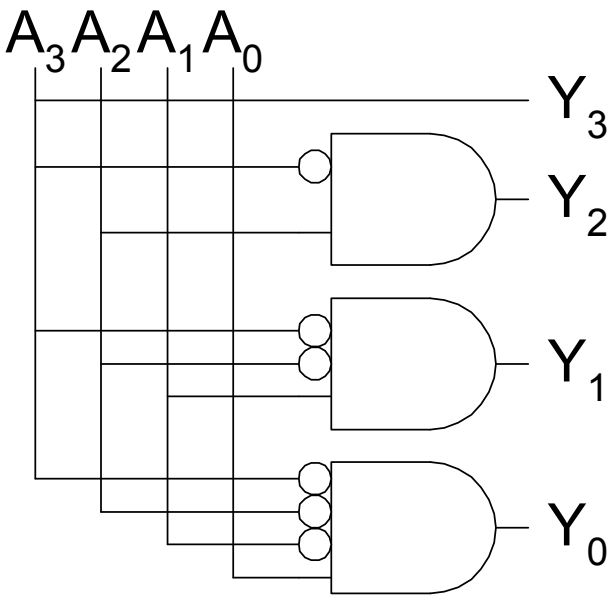
Multiple Output Circuits



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Priority Encoder Hardware

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0



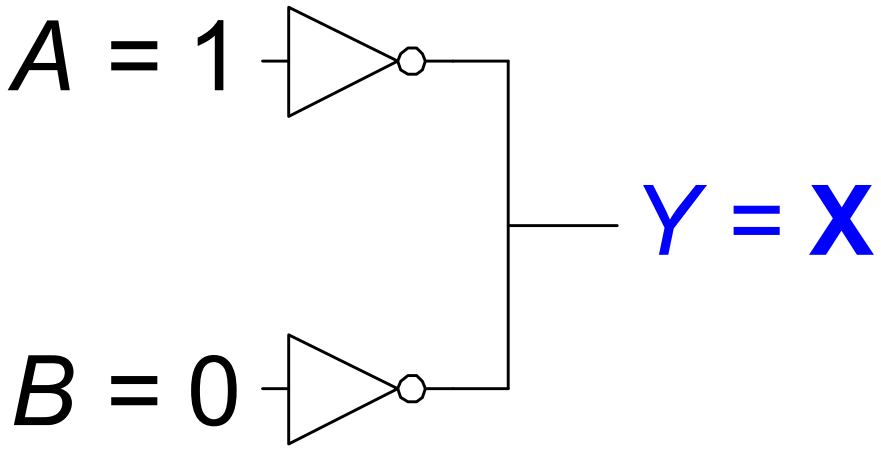
Don’t Cares

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	x	0	0	1	0
0	1	x	x	0	1	0	0
1	x	x	x	1	0	0	0

Contention: X

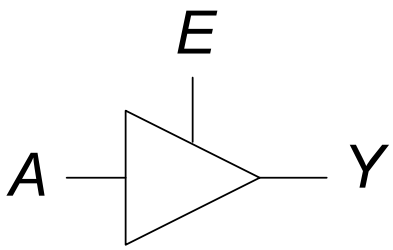
- Contention: circuit tries to drive the output to 1 and 0



Floating: Z

- Floating, high impedance, open, high Z
- Output is not connected to the input

Tristate Buffer



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Karnaugh Maps (K-Maps)

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations graphically
- $PA + P\overline{A} = P$

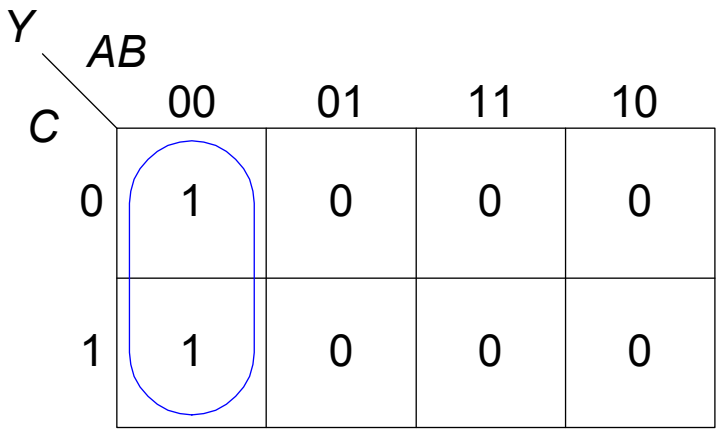
A	B	C	Y
0	0	0	**1**
0	0	1	**1**
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Y \ AB / C	00	01	11	10
0	**1**	0	0	0
1	**1**	0	0	0

Y \ AB / C	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

K-map

- Circle 1's in adjacent squares
- In the Boolean expression, include only the literals whose true and complement form are *not* in the circle



$$Y = \overline{A}\overline{B}$$

K-map Example

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Truth Table

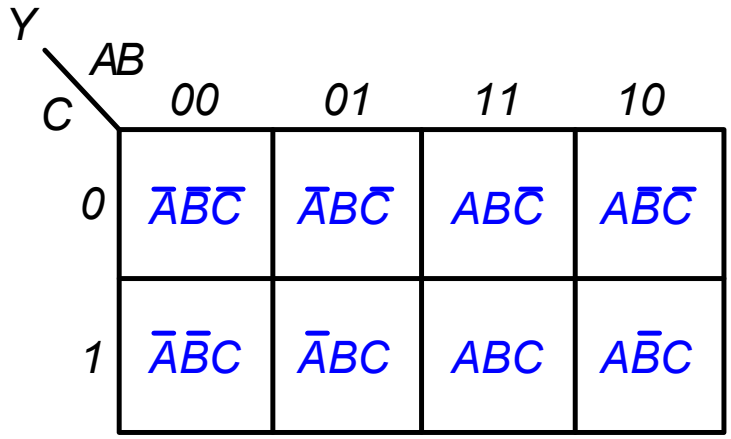
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	**1**
1	0	0	0
1	0	1	**1**
1	1	0	0
1	1	1	0

K-Map

Y: C \ AB	00	01	11	10
0	0	1	0	0
1	0	1	0	1

$$Y = \bar{A}B + A\bar{B}C$$

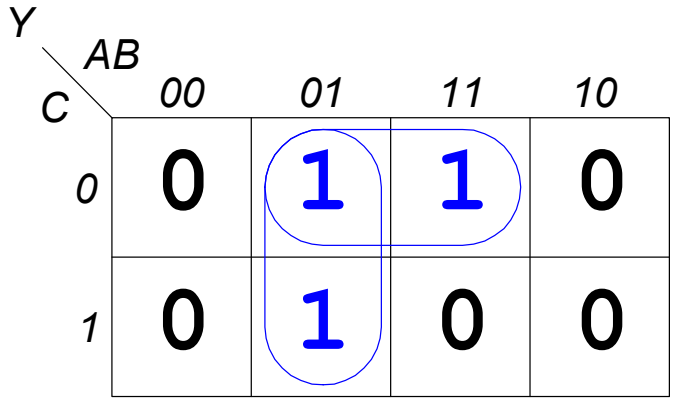
3-input K-map



Truth Table

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

K-Map



$$Y = \bar{A}B + B\bar{C}$$

K-map Definitions

- Complement: variable with a bar over it

  $\bar{A}$, $\bar{B}$, $\bar{C}$
- Literal: variable or its complement

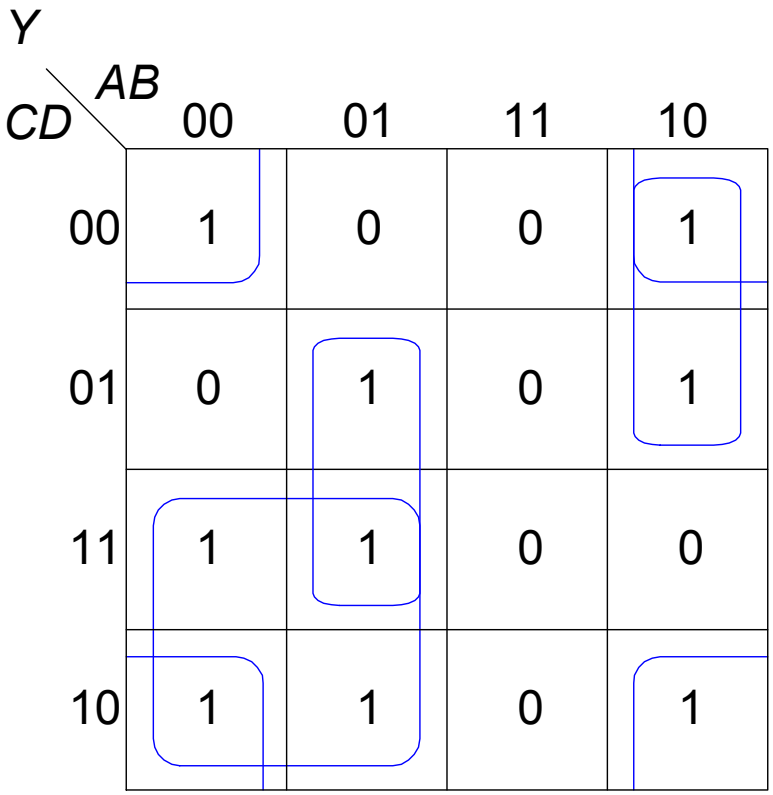
  A, $\bar{A}$, B, $\bar{B}$, C, $\bar{C}$
- Implicant: product of literals

  $AB\bar{C}$, $\bar{A}C$, BC
- Prime implicant: implicant corresponding to the **largest circle** in a K-map

K-map Rules

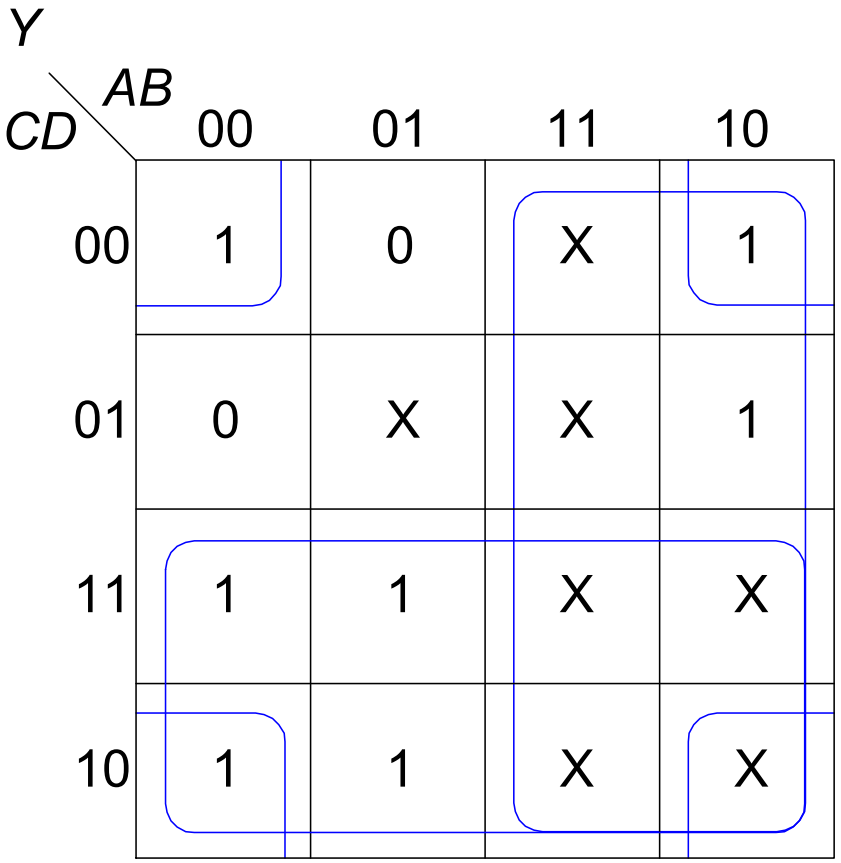
- **Each circle must span a power of 2** (i.e. 1, 2, 4) squares in each direction
- Each circle must be **as large as possible**
- A circle may **wrap around the edges** of the K-map
- A one in a K-map may be **circled multiple times**
- A “don't care” (X) is circled only if it helps minimize the equation

4-input K-map



$$Y = \bar{A}C + \bar{A}BD + A\bar{B}\bar{C} + \bar{B}\bar{D}$$

K-Maps with Don't Cares



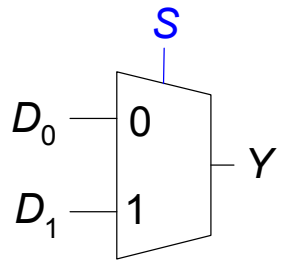
$$Y = A + \overline{B}\overline{D} + C$$

Combinational Building Blocks

- Multiplexers (멀티플렉서)
- Decoders (디코더)

Multiplexer (Mux:먹스)

- Selects between **one of *N* inputs** to connect to the output.
- $\log_2 N$-bit select input – control input
- Example: 2:1 Mux



S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

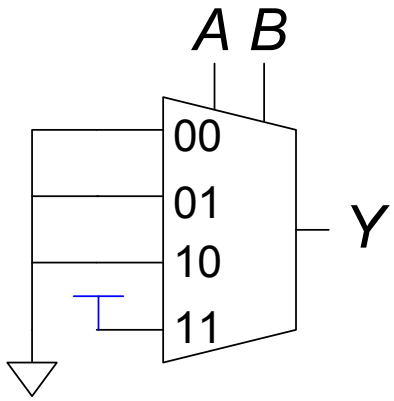
S	Y
0	D_0
1	D_1

Logic using Multiplexers

- Using the mux as a lookup table – AND-gate
  (먹스를 참조테이블로 사용하여 논리게이트 만들기)

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



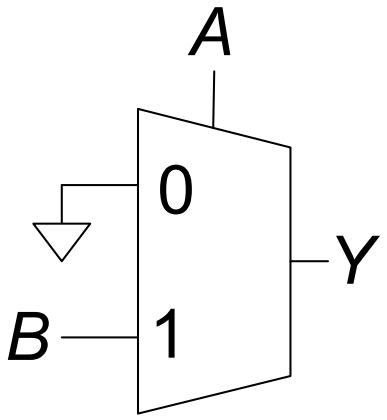
Logic using Multiplexers

- Reducing the size of the mux (먹스의 사이즈 줄이기)

$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

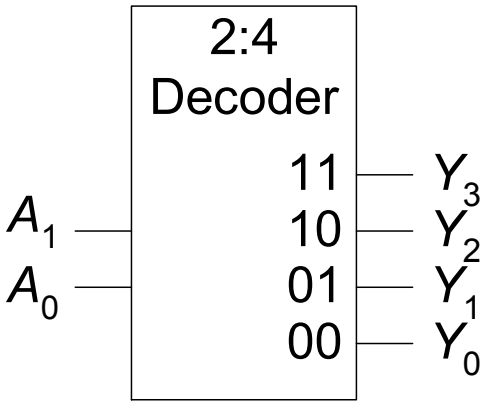
A	Y
0	0
1	B



Decoders

- N inputs, 2^N outputs
- **One-hot outputs**: only one output HIGH at once

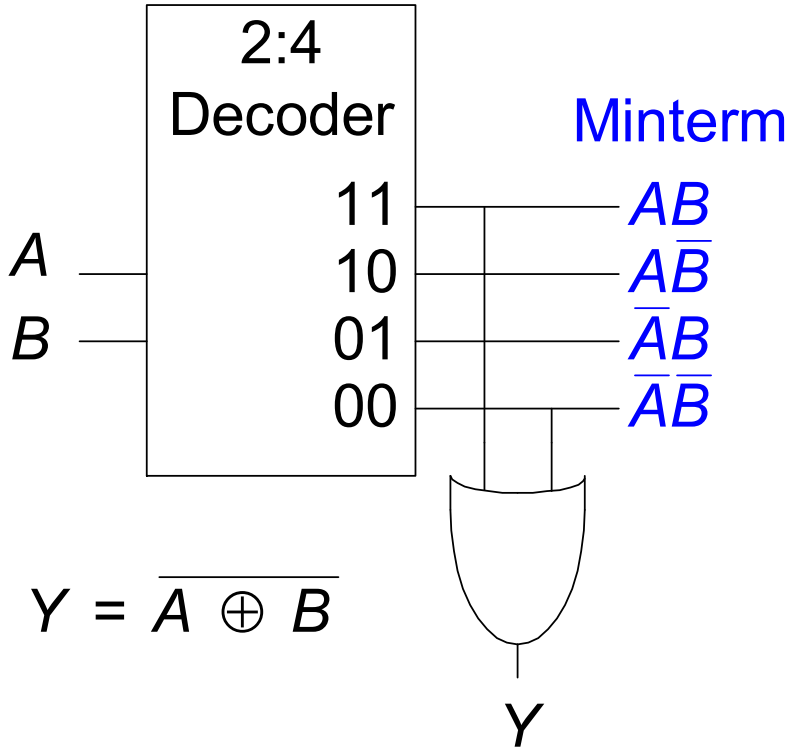
  출력 신호들 중 단 하나의 값만 '1'을 갖는다.



A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

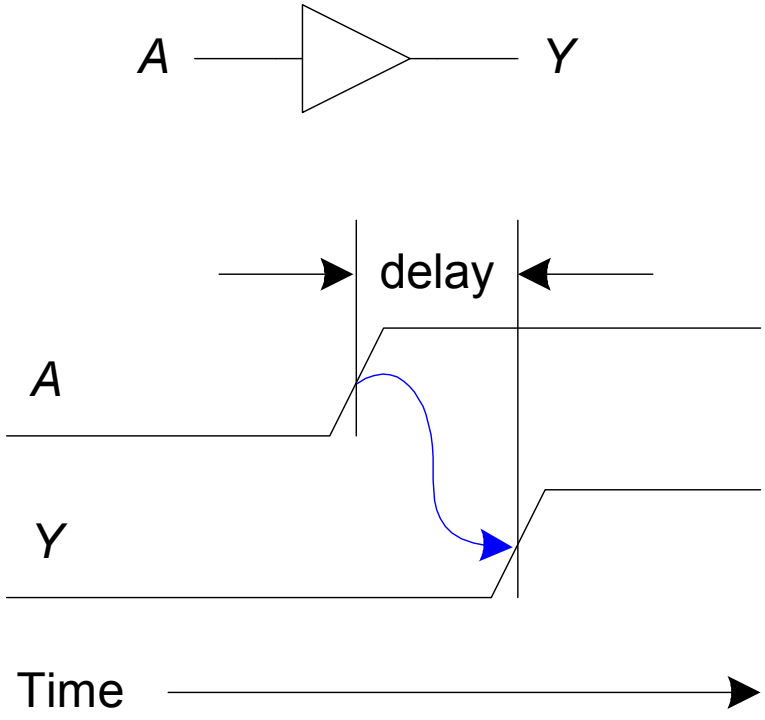
Logic using Decoders
(디코더를 이용한 회로)

- OR minterms



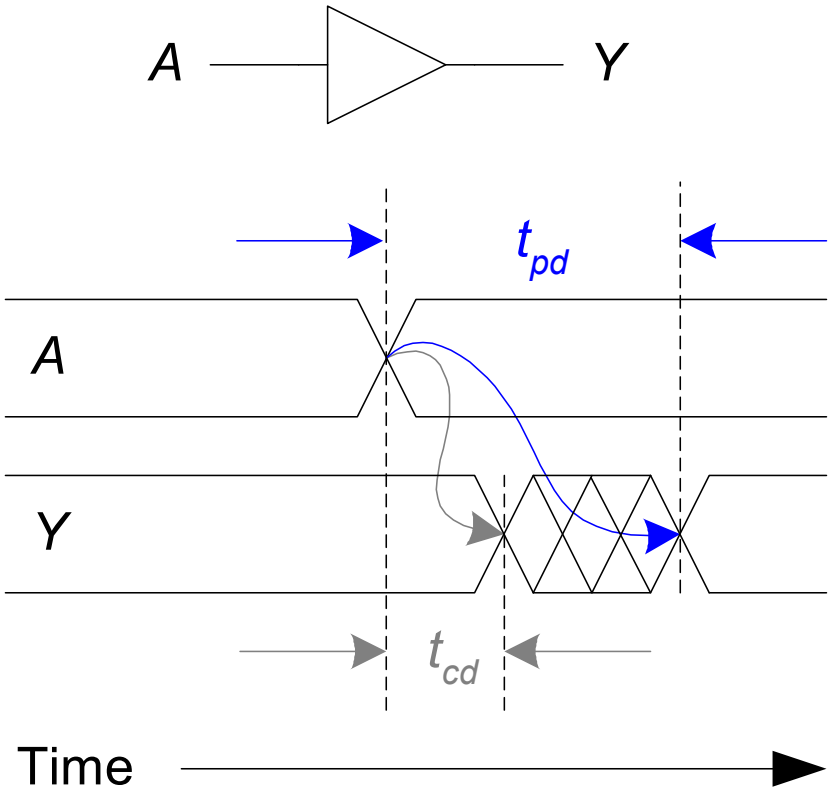
Timing (타이밍)

- **Delay between input change and output changing**
- One of the biggest challenges in circuit design: **making the circuit fast**



Propagation & Contamination Delay (전파 및 혼합 지연)

- Propagation delay: t_{pd} = max delay from input to output
- Contamination delay: t_{cd} = min delay from input to output

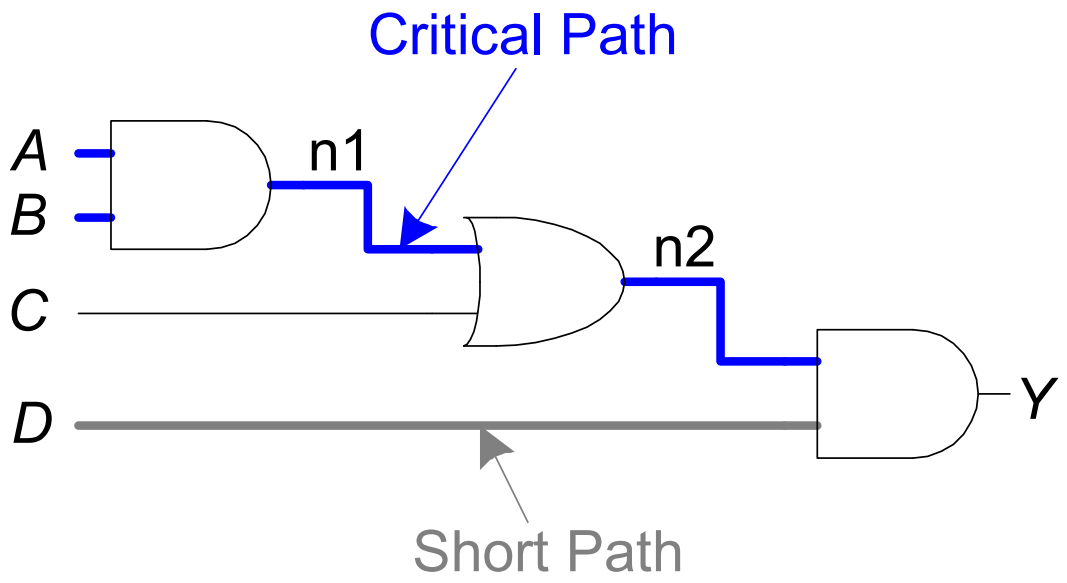


Propagation & Contamination Delay

- Delay is caused by
  - Capacitance and resistance in a circuit
  - Speed of light limitation

- Reasons why *tpd* and *tcd* may be different:
  - Different rising and falling delays
    상승지연과 하강지연이 다르다.
  - Multiple inputs and outputs, some of which are faster than others
    입력이나 출력이 여러 개 있을 경우, 서로간의 속도 차가 있다.
  - Circuits slow down when hot and speed up when cold
    회로는 뜨거울 때 속도가 느리고, 차가울 때 속도가 빨라진다.

Critical and Short Paths
(임계 경로 및 짧은 경로)



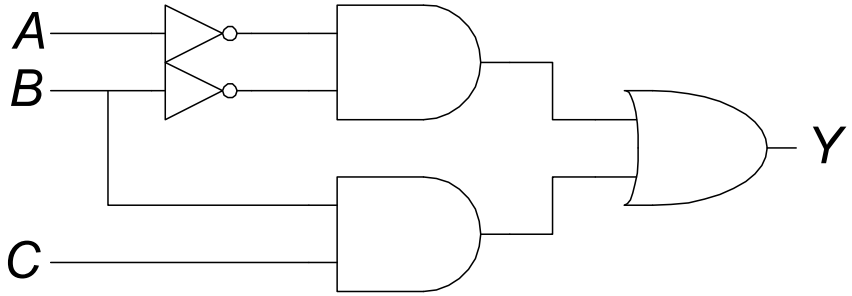
Critical (Long) Path: $t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$

Short Path: $t_{cd} = t_{cd\_\mathrm{AND}}$

Glitches (글리치)

- A glitch occurs when a single input change causes multiple output changes
  (단일 입력의 변화가 다수의 출력의 유도할때)
- Glitches don't cause problems because of synchronous design conventions (which we'll talk about in Chapter 3)
  (동기식 회로에서는 문제를 야기하지 않음)
- But it's important to recognize a glitch when you see one in simulations or on an oscilloscope
  (하지만 존재에 대해서 이해하고 해석할 능력은 필요함)

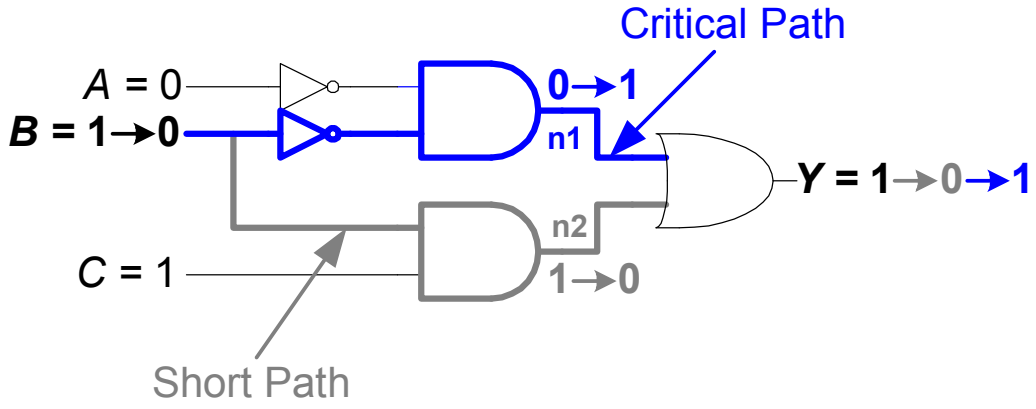
Glitch Example

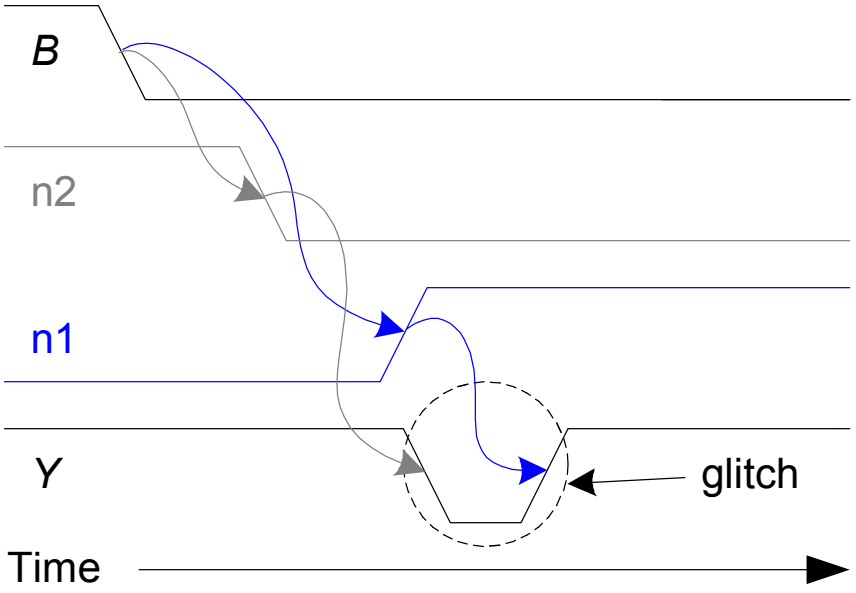


Y \ AB	00	01	11	10
C = 0	1	0	0	0
C = 1	1	1	1	0

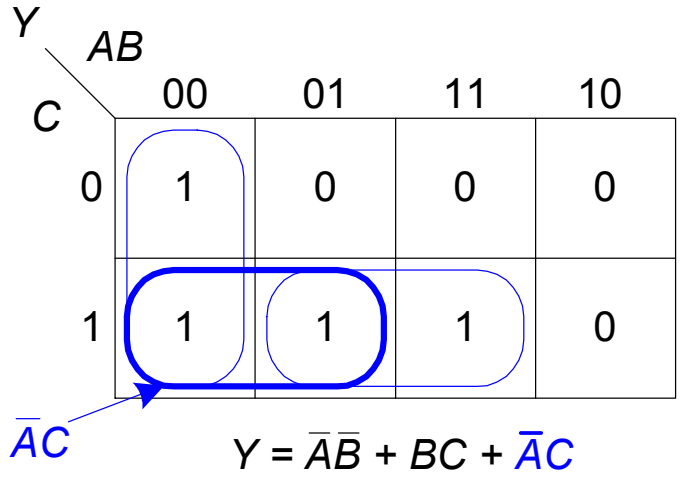
$$Y = \bar{A}\bar{B} + BC$$

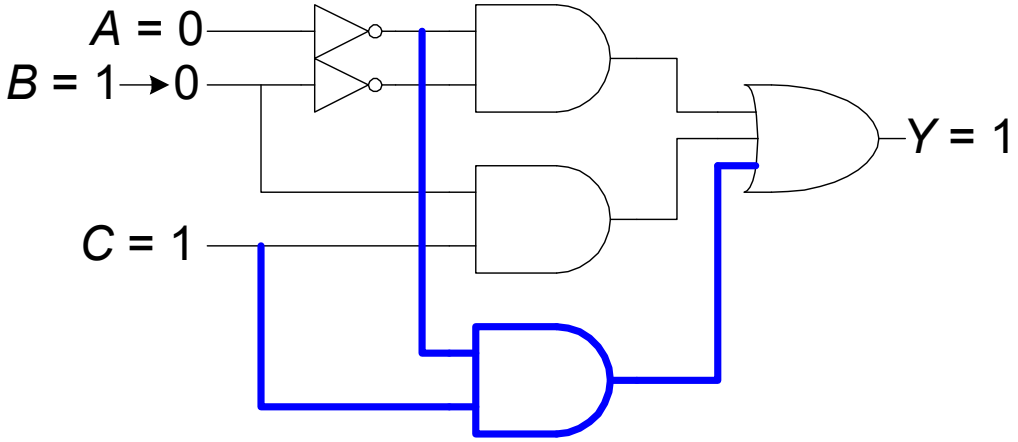
Glitch Example (cont.)





Glitch Example (cont.)





Why Understand Glitches?

- Recognize them when look at timing diagrams in (simulations or on an oscilloscope)
- Can't get rid of all glitches – simultaneous transitions on multiple inputs can also cause glitches