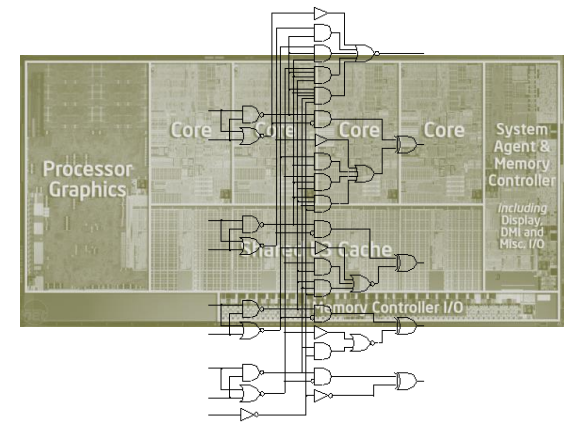




고려대학교
KOREA UNIVERSITY



COSE221, COMP211 Logic Design

Lecture 3. Combinational Logic #1

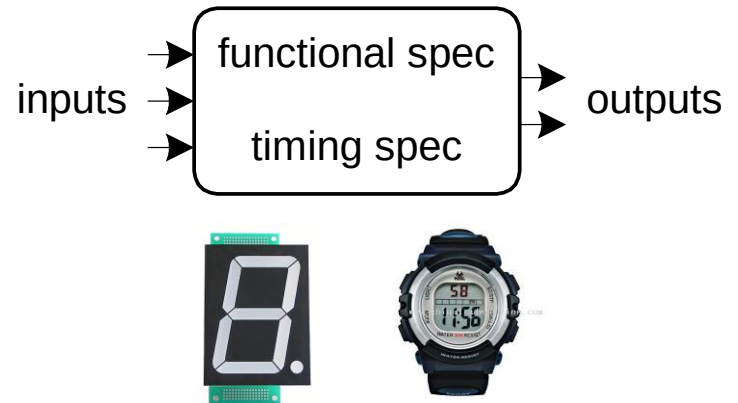
Prof. Taeweon Suh

Computer Science & Engineering

Korea University

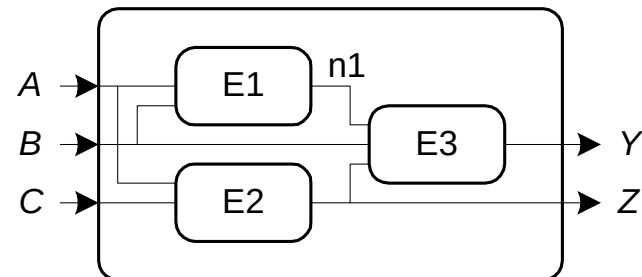
Logic Circuits

- A logic circuit is composed of
 - Inputs
 - Outputs
 - Functional specification
 - Relationship between inputs and outputs
 - Timing specification
 - Delay from inputs to outputs



- Nodes
 - Inputs: A, B, C
 - Outputs: Y, Z
 - Internal: $n1$

- Circuit elements
 - $E1, E2, E3$



Types of Logic Circuits

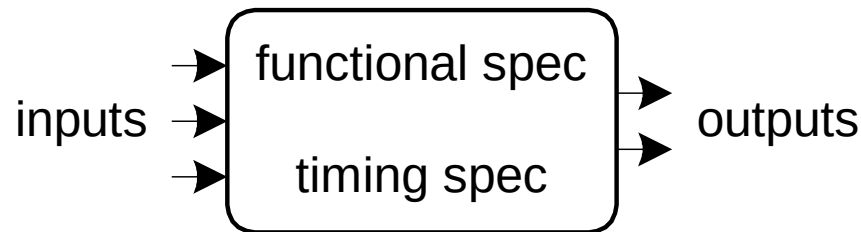
- **Combinational Logic**

- Outputs are determined by **current** values of inputs
- Thus, it is memoryless



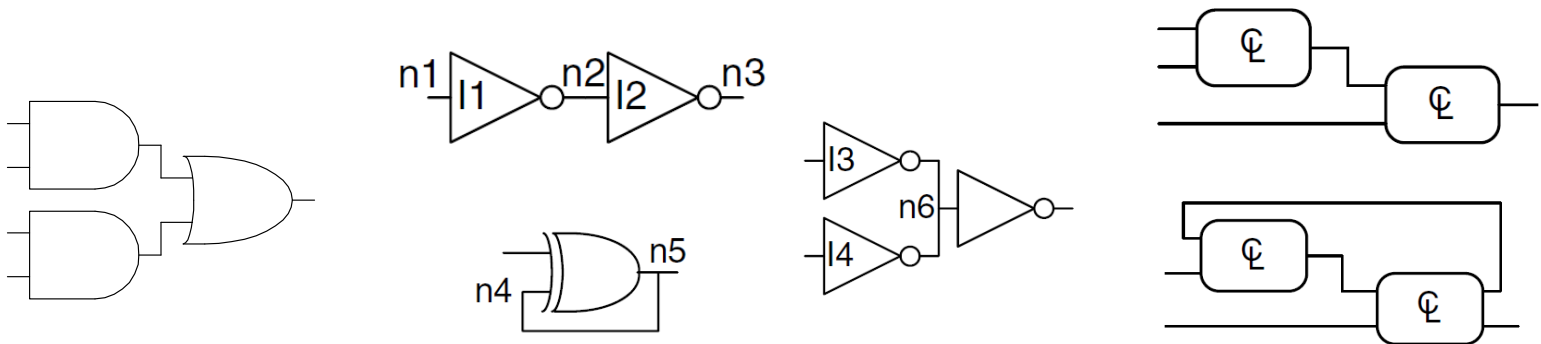
- **Sequential Logic**

- Outputs are determined by **previous** and **current** values of inputs
- Thus, it has memory



Rules of Combinational Composition

- A circuit is **combinational** if
 - 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
 - Every circuit element is itself combinational
- Select combinational logic?

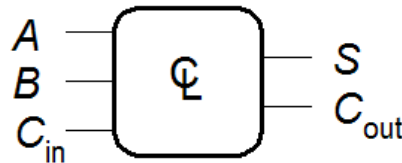


Boolean Equations

- The functional specification of a combination logic is usually expressed as a **truth table** or a **Boolean equation**
 - Truth table is in a tabular form
 - Boolean equation is in an algebraic form

Truth table

C _{in}	A	B	S(um)	C _{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$\begin{aligned} S &= F(A, B, C_{in}) \\ C_{out} &= F(A, B, C_{in}) \end{aligned}$$

Boolean equation

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

Terminology

- The **complement** of a variable A is $\overline{A}$
 - A variable or its complement is called **literal**
- **AND** of one or more literals is called a **product** or **implicant**
 - Example: $\overline{A}\overline{B}$, $A\overline{B}\overline{C}$, $\overline{B}$
- **OR** of one or more literals is called a **sum**
 - Example: $A + B$
- Order of operations
 - NOT has the highest precedence, followed by AND, then OR
 - Example: $Y = A + \overline{B}\overline{C}$

Minterms

- A **minterm** is a **product** (AND) of literals involving **all** of the inputs
- Each row in a truth table has **one minterm** that is **true only that row**

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

Sum-of-Products (SOP) Form

- The function is formed by **ORing the minterms** for which the output is true
 - Thus, a sum (OR) of products (AND terms)
- **All Boolean equations can be written in SOP form**

A	B	Y	Minterm
0	0	0	$\bar{A} \bar{B}$
0	1	1	$\bar{A} B$
1	0	0	$A \bar{B}$
1	1	1	$A B$

$$Y = F(A, B) = \bar{A}B + AB$$

Maxterms

- A **maxterm** is a **sum** (OR) of literals involving **all** of the inputs
- Each row in a truth table has **one maxterm** that is **false only for that row**

			M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
A	B	C								
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

Product-of-Sums (POS) Form

- The function is formed by **ANDing the maxterms** for which the output is FALSE
 - Thus, a product (AND) of sums (OR terms)
- **All Boolean equations can be written in POS form**

A	B	Y	Maxterm
0	0	0	$A + B$
0	1	1	$A + B$
1	0	0	$\bar{A} + B$
1	1	1	$\bar{A} + \bar{B}$

$$Y = F(A, B) = (A + B)(\bar{A} + \bar{B})$$

Boolean Equation Example

- You are going to the cafeteria for lunch
 - You won't eat lunch (E: eat)
 - If it's not open (O: open)
 - If they only serve corndogs (C: corndogs)
- Write a truth table and boolean equations (in SOP and POS) for determining if you will eat lunch (E)

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

1. SOP (sum-of-products)

$$E = O\bar{C}$$

2. POS (product-of-sums)

$$E = (O + C)(O + \bar{C})(\bar{O} + \bar{C})$$

O	C	E	minterm
0	0	0	$\bar{O}\bar{C}$
0	1	0	$\bar{O}C$
1	0	1	$O\bar{C}$
1	1	0	OC

O	C	E	maxterm
0	0	0	$O + C$
0	1	0	$O + \bar{C}$
1	0	1	$\bar{O} + C$
1	1	0	$\bar{O} + \bar{C}$

When to Use SOP and POS?

- SOP produces a shorter equation when the output is true on only a few rows of a truth table
- POS is simpler when the output is false on only a few rows of a truth table

Boolean Algebra

- We just learned how to write the boolean equation given a truth table
 - But, that expression does not necessarily lead to the simplest set of logic gates
- One way to simplify boolean equations is to use boolean algebra
 - Set of axioms and theorems
 - It is 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

- Axioms are not provable
- The prime (') symbol denotes the **dual** of a statement

Axiom		Dual		Name
A1	$B = 0 \text{ if } B \neq 1$	A1'	$B = 1 \text{ if } B \neq 0$	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

Boolean Theorems of One Variable

- The prime (') symbol denotes the **dual** of a statement

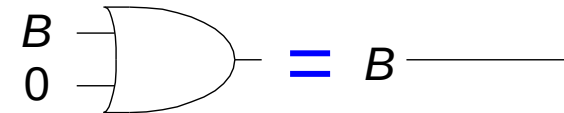
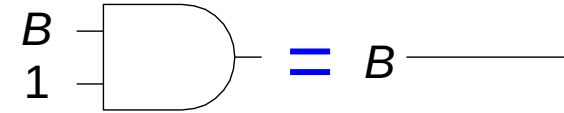
Theorem		Dual		Name
T1	$B \bullet 1 = B$	T1'	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2'	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3'	$B + B = B$	Idempotency
T4	$\overline{\overline{B}} = B$			Involution
T5	$B \bullet \overline{B} = 0$	T5'	$B + \overline{B} = 1$	Complements

Boolean Theorems of One Variable

- T1: Identity Theorem

- $B \cdot 1 = B$

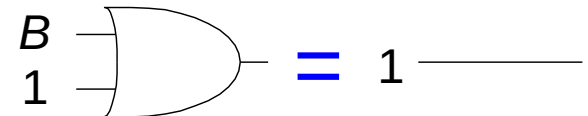
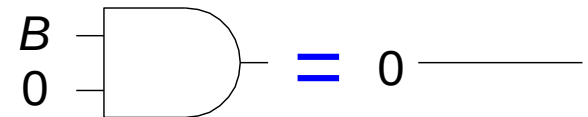
- $B + 0 = B$



- T2: Null Element Theorem

- $B \cdot 0 = 0$

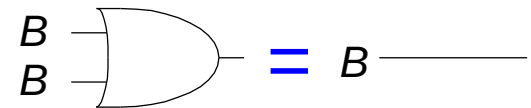
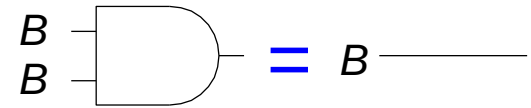
- $B + 1 = 1$



Boolean Theorems of One Variable

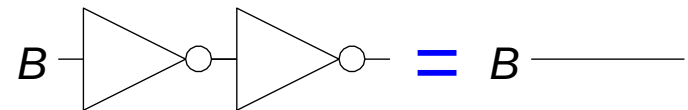
- Idempotency Theorem

- $B \cdot B = B$
- $B + B = B$



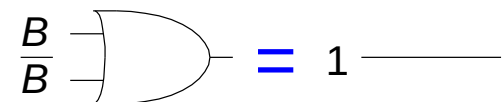
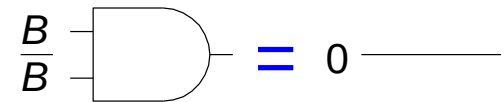
- T4: Involution

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



- T5: Complement Theorem

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



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	$\overline{B_0 \bullet B_1 \bullet B_2 \dots}$ $= (\overline{B_0} + \overline{B_1} + \overline{B_2} \dots)$	T12'	$\overline{B_0 + B_1 + B_2 \dots}$ $= (\overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2})$	De Morgan's Theorem

Super-important!

Proof of Consensus Theorem

- Prove the consensus theorem

Simplifying Boolean Expressions: Example 1

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

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

$$= B(1) \quad \text{T5'}$$

$$= B \quad \text{T1}$$

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

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

Simplifying Boolean Expressions: Example 2

- $$\begin{aligned}
 Y &= A(AB + ABC) \\
 &= A(AB(1 + C)) && \text{T8} \\
 &= A(AB(1)) && \text{T2'} \\
 &= A(AB) && \text{T1} \\
 &= (AA)B && \text{T7} \\
 &= AB && \text{T3}
 \end{aligned}$$

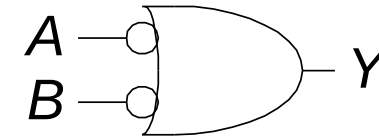
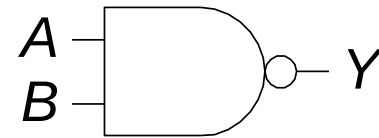
Theorem	Dual	Name
T1 $B \cdot 1 = B$	T1' $B + 0 = B$	Identity
T2 $B \cdot 0 = 0$	T2' $B + 1 = 1$	Null Element
T3 $B \cdot B = B$	T3' $B + B = B$	Idempotency
T4 $\overline{\overline{B}} = B$		Involution
T5 $B \cdot \overline{B} = 0$	T5' $B + \overline{B} = 1$	Complements

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

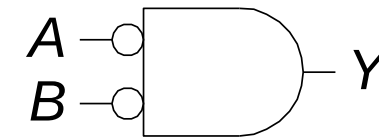
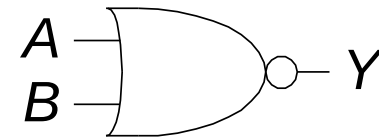
DeMorgan's Theorem

- Powerful theorem in digital design

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

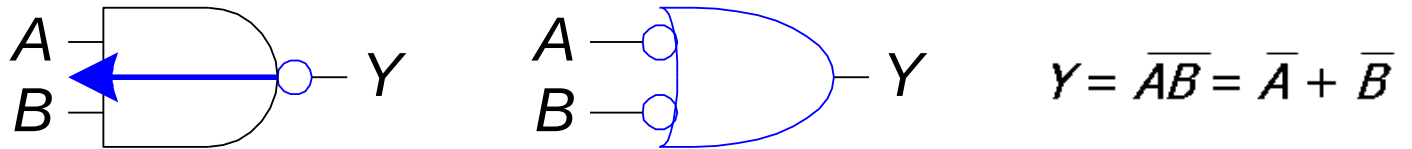


$$Y = \overline{\overline{A} + \overline{B}} = \overline{\overline{A}} \cdot \overline{\overline{B}}$$

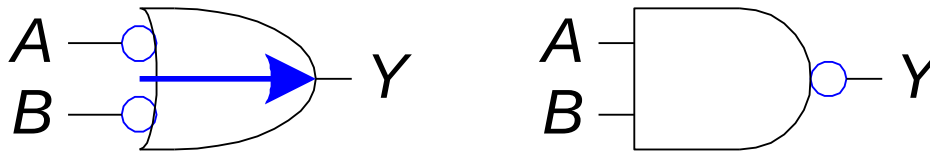


Bubble Pushing

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

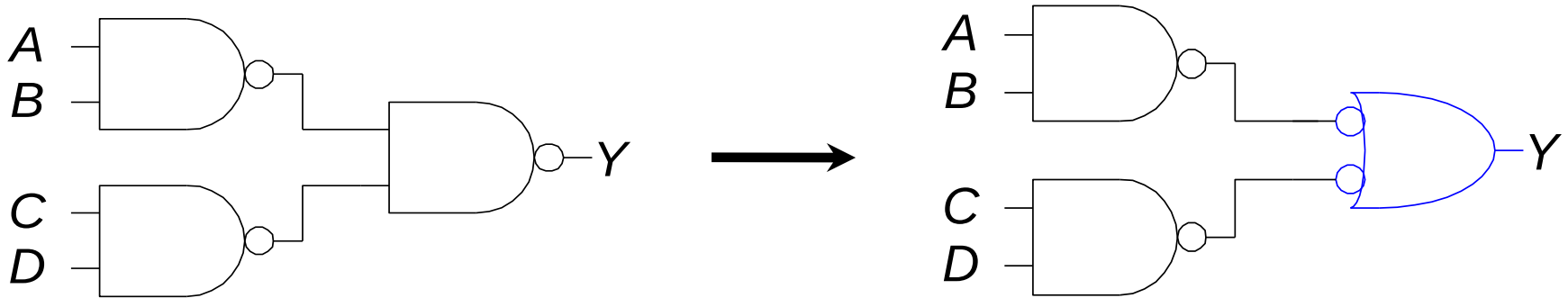


- Then, pushing bubbles on **all** gate inputs forward (toward the output) puts a bubble on the output and changes the gate body



Bubble Pushing

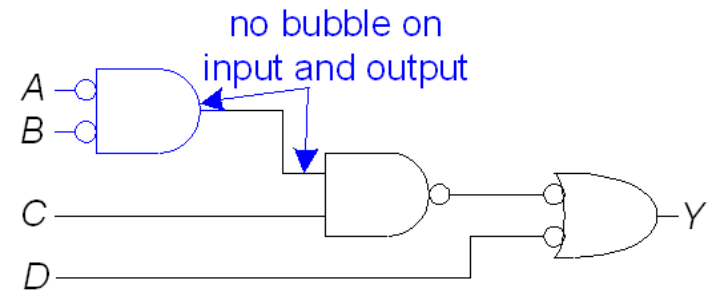
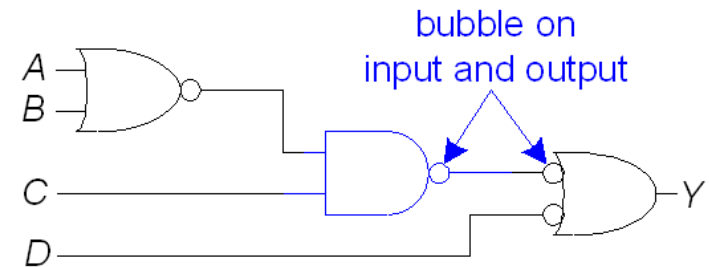
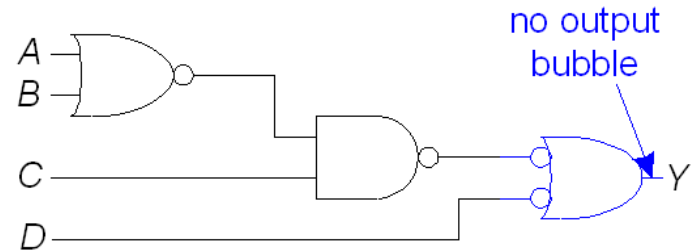
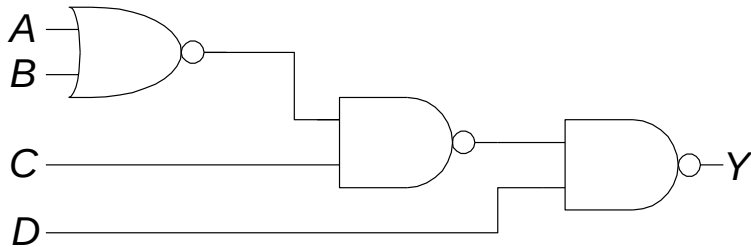
- What is the Boolean expression for this circuit?



$$Y = AB + CD$$

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

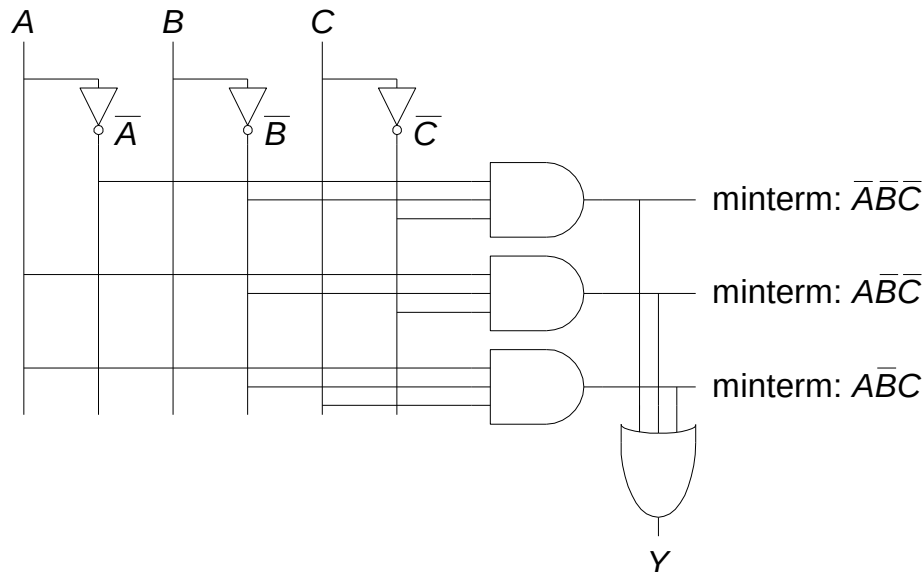


$$Y = \overline{A} \overline{B} C + \overline{D}$$

From Logic to Gates

- Schematic

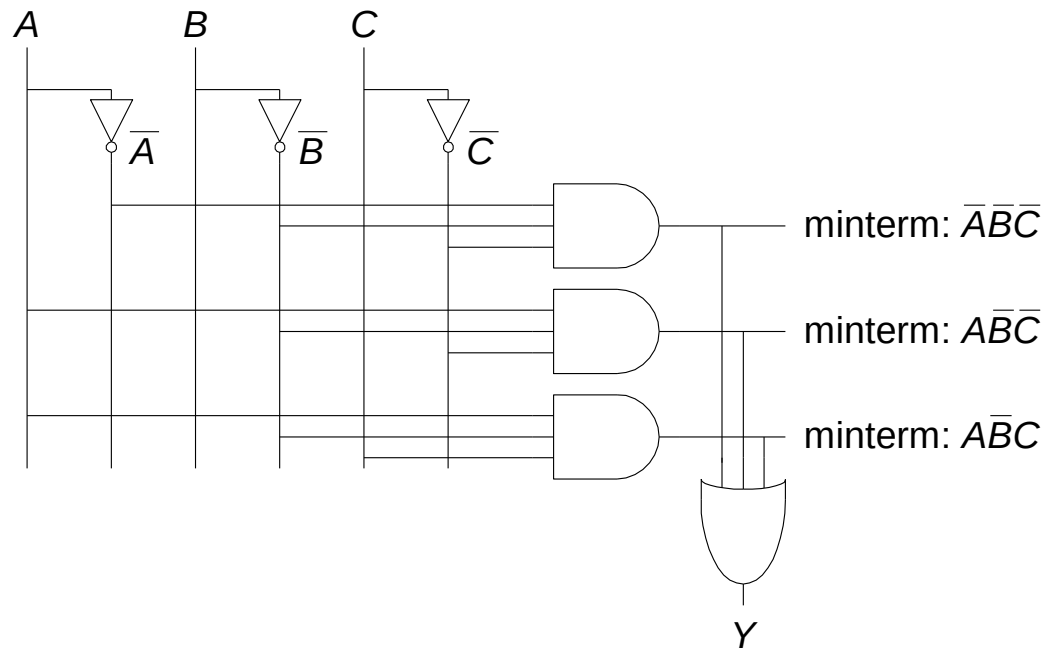
- A diagram of a digital circuit showing the elements and the wires that connect them together
- Example: $Y = \overline{A}\overline{B}\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$



Any Boolean equation in the SOP form can be drawn like above

Circuit Schematic Rules

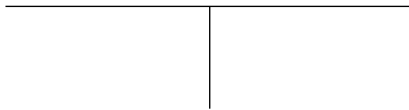
- Inputs are on the left (or top) side of a schematic
- Outputs are on the right (or bottom) side of a schematic
- Whenever possible, gates should flow from left to right
- Straight wires are better to use than wires with multiple corners



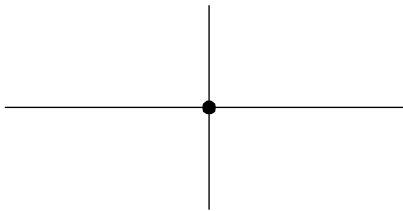
Circuit Schematic Rules (cont.)

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

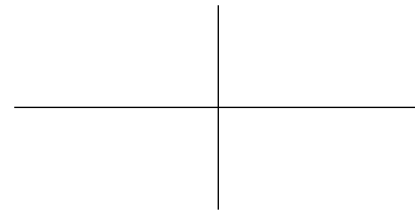
wires connect
at a T junction



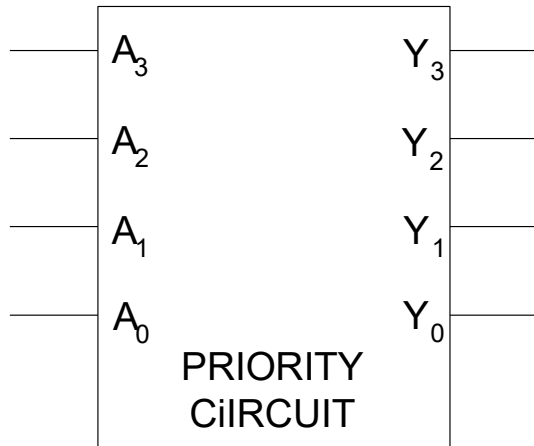
wires connect
at a dot



wires crossing
without a dot do
not connect



Multiple Output Circuits



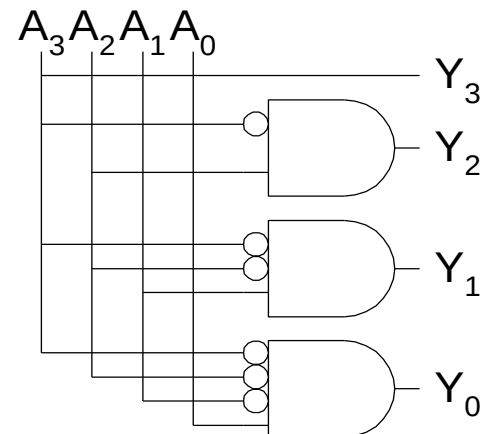
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Priority Circuit Logic

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

- Probably you want to write boolean equations for Y_3 , Y_2 , Y_1 , and Y_0 with SOP or POS, and minimize the logic
- But in this case it is not that difficult to come up with simplified boolean equations by inspection

$$\begin{aligned}
 Y_3 &= A_3 \\
 Y_2 &= \overline{A_3} A_2 \\
 Y_1 &= \overline{A_3} \overline{A_2} A_1 \\
 Y_0 &= \overline{A_3} \overline{A_2} \overline{A_1} A_0
 \end{aligned}$$



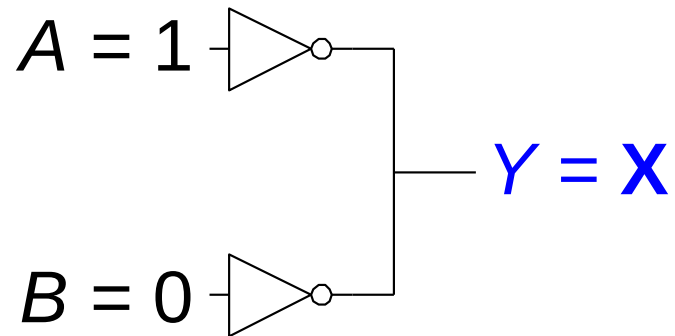
Don't Cares (X)

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_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	0	0	0	0	0
0	0	0	1	0	0	0	1		0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0		0	0	1	X	0	0	1	0
0	0	1	1	0	0	1	0		0	1	X	X	0	1	0	0
0	1	0	0	0	1	0	0		1	X	X	X	1	0	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	0	1	0	0	0									
1	1	1	1	1	0	0	0									
1	1	1	1	1	0	0	0									

$$\begin{aligned}
 Y_3 &= \overline{A_3} \\
 Y_2 &= \overline{A_3} \overline{A_2} \\
 Y_1 &= \overline{A_3} \overline{A_2} \overline{A_1} \\
 Y_0 &= \overline{A_3} \overline{A_2} \overline{A_1} A_0
 \end{aligned}$$

Contention: X

- Contention: circuit tries to drive the output to 1 and 0
 - So, you **should not** design a digital logic creating a contention!



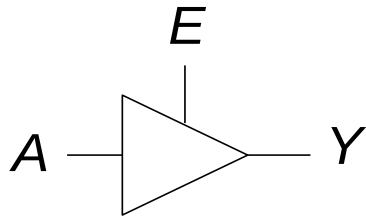
Note

- In **truth table**, the symbol X denotes 'don't care'
- In **circuit**, the same symbol X denotes 'unknown or illegal value'

Floating: Z

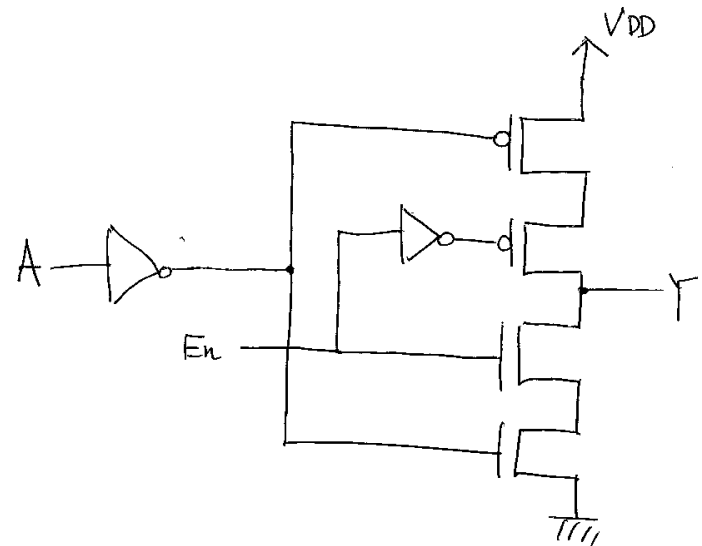
- Output is disconnected from the input if not enabled
 - We say output is floating, high impedance, open, or high Z

Tristate Buffer



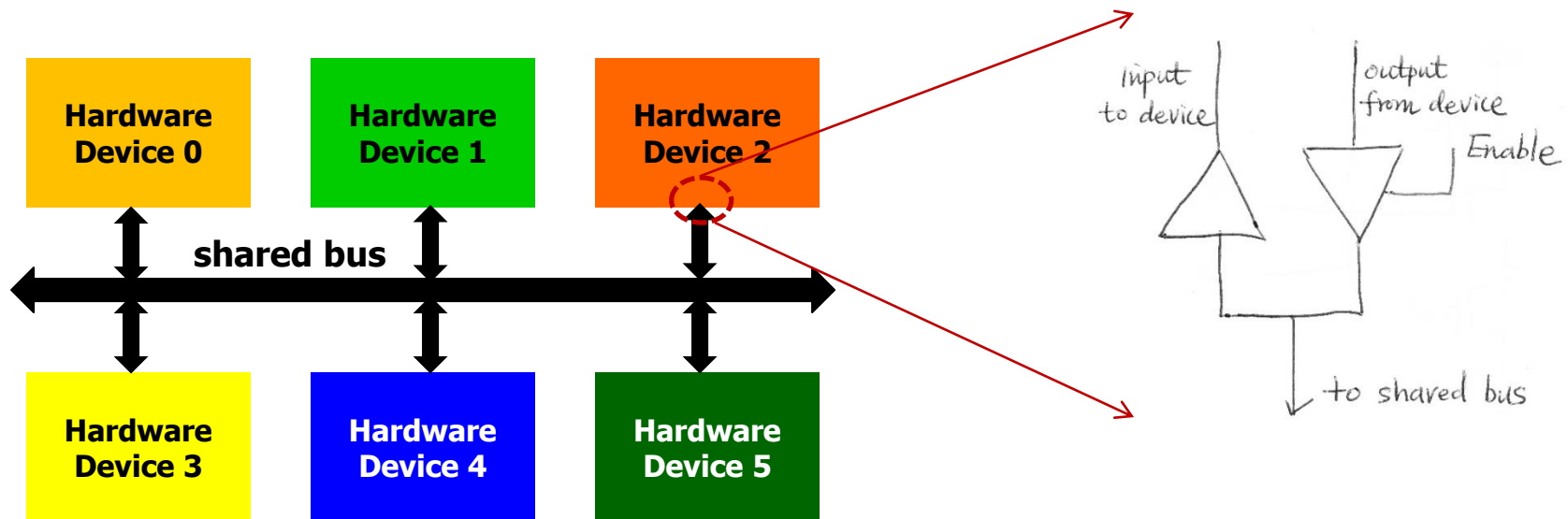
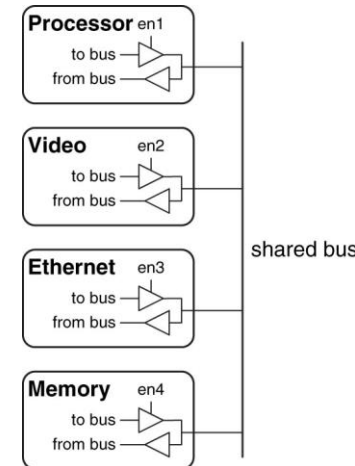
E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

An implementation Example



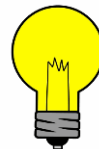
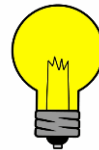
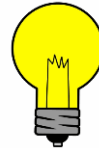
Where Is Tristate Buffer Used for?

- Tristate buffer is used when designing hardware components sharing a communication medium called **shared bus**
 - Many hardware components can be attached on a shared bus
 - Only one component is allowed to drive the bus at a time
 - The other components put their outputs to the floating
- What happens if you don't use the tristate buffer on shared bus?



Backup Slides

Priority Circuit Application Example



Customer Service

