COMBINATIONAL LOGIC DESIGN

Chapter 2

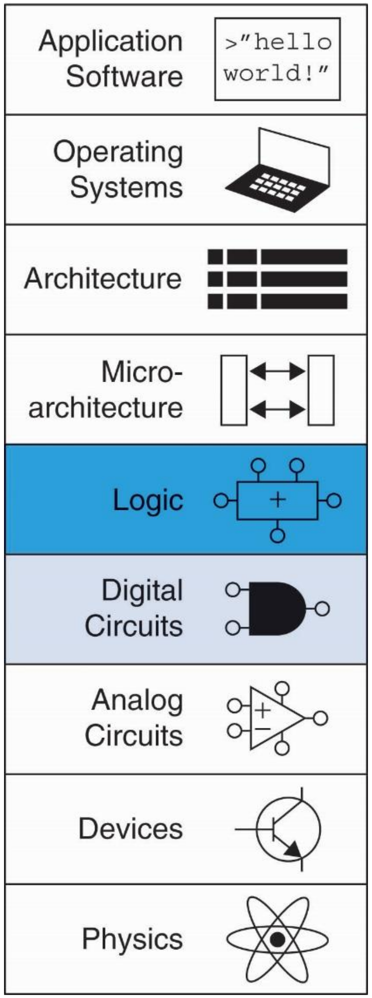
Professor Brendan Morris, SEB 3216, brendan.morris@unlv.edu
http://www.ee.unlv.edu/~b1morris/cpe100/

CPE100: Digital Logic Design I

Combinational Logic Design

Chapter 2 :: Topics

- Introduction
- Boolean Equations
- Boolean Algebra
- From Logic to Gates
- Multilevel Combinational Logic
- X's and Z's, Oh My
- Karnaugh Maps
- Combinational Building Blocks
- Timing



Chapter 2.1

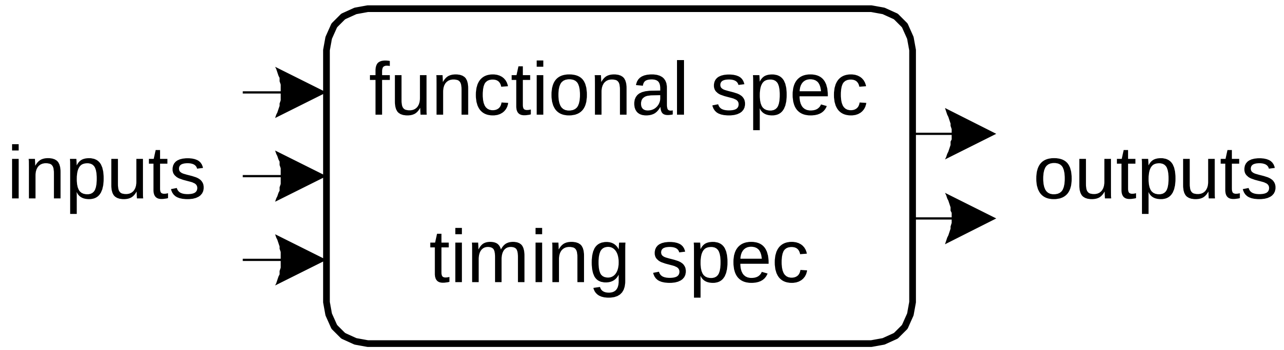
Introduction

ELSEVIER

Introduction

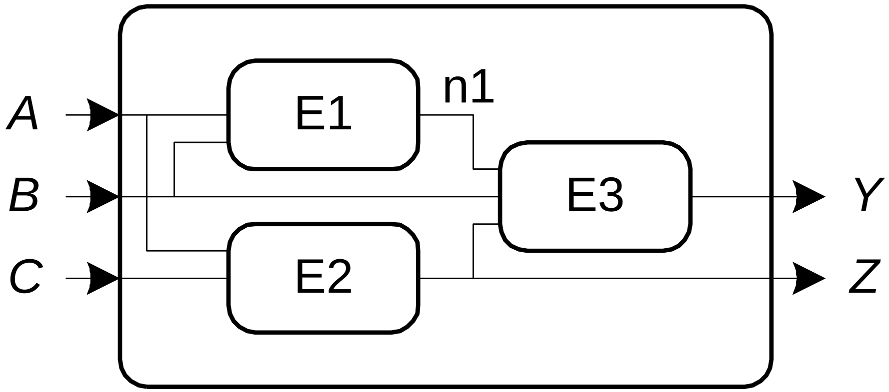
A logic circuit is composed of:

- Inputs
- Outputs
- Functional specification
- Timing specification



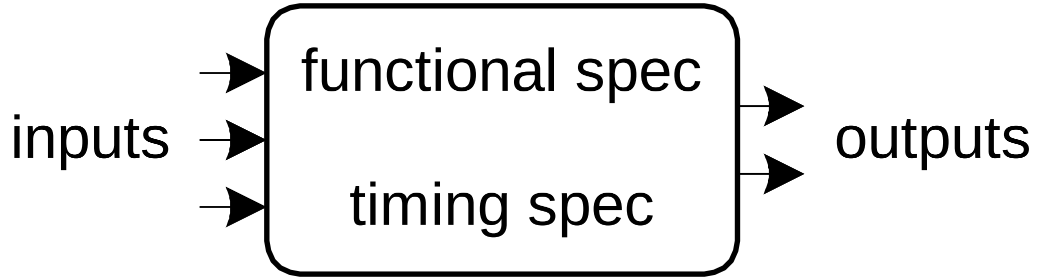
Circuits

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



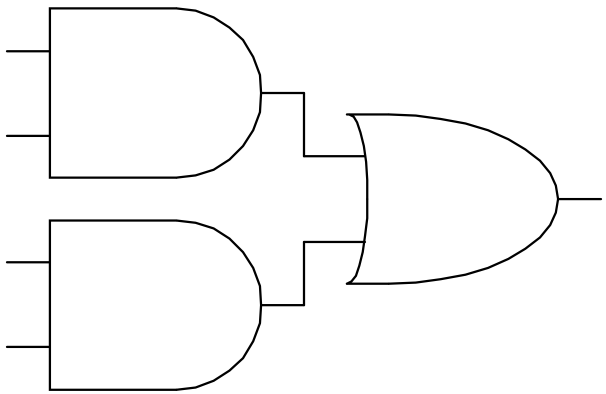
Types of Logic Circuits

- Combinational Logic (Ch 2)
  - Memoryless
  - Outputs determined by current values of inputs
- Sequential Logic (Ch 3)
  - Has memory
  - Outputs determined by previous and current values of inputs



Rules of Combinational Composition

- Every element is combinational
- Every node is either an input or connects to *exactly one* output
- The circuit contains no cyclic paths
  - E.g. no connection from output to internal node
- Example:



COMBINATIONAL LOGIC DESIGN

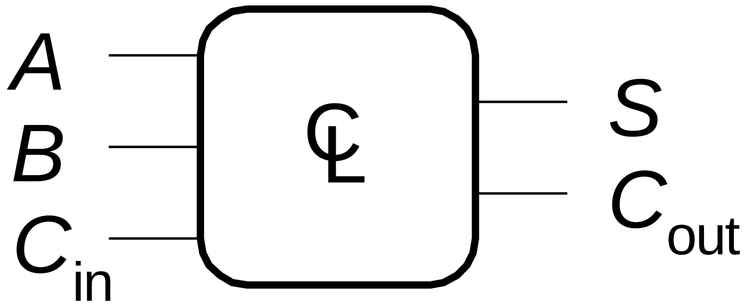
Chapter 2.2

Boolean Equations

Boolean Equations

- Functional specification of outputs in terms of inputs
- **Example:** $S = F(A, B, C_{in})$

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



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

A	B	C_{in}	S	C_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Functional specification

Goals:

- Systematically express logical functions using Boolean equations
- To simplify Boolean equations

Some Definitions

- Complement: variable with a bar over it

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

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

  $AB\overline{C}, \overline{A}C, BC$
- Minterm: product that includes all input variables

  $AB\overline{C}, A\overline{B}\overline{C}, ABC$
- Maxterm: sum that includes all input variables

  $(A+\overline{B}+C), (\overline{A}+B+\overline{C}), (\overline{A}+\overline{B}+C)$

Canonical Sum-of-Products (SOP) Form

- All equations can be written in SOP form
- Each row has a **minterm**
- A minterm is a product (AND) of literals
- Each minterm is TRUE for that row (and only that row)

A	B	Y	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

ELSEVIER

Canonical Sum-of-Products (SOP) Form

- All equations can be written in SOP form
- Each row has a **minterm**
- A minterm is a product (AND) of literals
- Each minterm is TRUE for that row (and only that row)
- Form function by ORing minterms where the output is TRUE

A	*B*	*Y*	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

$$Y = F(A, B) =$$

Canonical Sum-of-Products (SOP) Form

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

A	*B*	*Y*	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

$$Y = F(A, B) = \overline{A}B + AB = \Sigma(m_1, m_3)$$

SOP Example

- Steps:
- Find minterms that result in Y=1
- Sum “TRUE” minterms

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

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

SOP Example

- Steps:
- Find minterms that result in Y=1
- Sum "TRUE" minterms

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

minterm name	minterm
m_0	$\bar{A}\bar{B}$
m_1	$\bar{A}B$
m_2	
m_3	

$$\mathbf{Y = F(A, B) =} \quad m_0 + m_1 = \bar{A}\bar{B} + \bar{A}B$$

Aside: Precedence

- AND has precedence over OR
- In other words:
  - AND is performed **before** OR
- Example:
  - $Y = \bar{A} \cdot B + A \cdot B$
  - Equivalent to:
  - $Y = (\bar{A}B) + (AB)$

Canonical Product-of-Sums (POS) Form

- All Boolean equations can be written in POS form
- Each row has a **maxterm**
- A maxterm is a sum (OR) of literals
- Each maxterm is FALSE for that row (and only that row)

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

Canonical Product-of-Sums (POS) Form

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

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

$$Y = M_0 \cdot M_2 = (A + B) \cdot (\bar{A} + B)$$

SOP and POS Comparison

- Sum of Products (SOP)
  - Implement the “ones” of the output
  - Sum all “one” terms → OR results in “one”

- Product of Sums (POS)
  - Implement the “zeros” of the output
  - Multiply “zero” terms → AND results in “zero”

Boolean Equations Example

- You are going to the cafeteria for lunch
  - You will eat lunch (E=1)
  - If it's open (O=1) **and**
  - If they're not serving corndogs (C=0)
- Write a truth table for determining if you will eat lunch (E).

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

Boolean Equations Example

- You are going to the cafeteria for lunch
  - You will eat lunch (E=1)
  - If it's open (O=1) **and**
  - If they're not serving corndogs (C=0)
- 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		$\overline{O}\ \overline{C}$
0	1		$\overline{O}\ C$
1	0		$O\ \overline{C}$
1	1		$O\ C$

- POS – product-of-sums

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

SOP & POS Form

- SOP – sum-of-products

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

- POS – product-of-sums

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

SOP & POS Form

- SOP – sum-of-products

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

$$E = O\overline{C}$$
$$= \Sigma(m_2)$$

- POS – product-of-sums

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

SOP & POS Form

- SOP – sum-of-products

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

$$E = O\overline{C}$$
$$= \Sigma(m_2)$$

- POS – product-of-sums

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

$$E = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$
$$= \Pi(M0, M1, M3)$$

Chapter 2.3

Boolean Algebra

Boolean Algebra

- Axioms and theorems to **simplify** Boolean equations
- Like regular algebra, but simpler: variables have only two values (1 or 0)
- **Duality** in axioms and theorems:
  – ANDs and ORs, 0's and 1's interchanged

Boolean Axioms

	Axiom
A1	$B = 0$ if $B \neq 1$
A2	$\overline{0} = 1$
A3	$0 \bullet 0 = 0$
A4	$1 \bullet 1 = 1$
A5	$0 \bullet 1 = 1 \bullet 0 = 0$

Duality

Duality in Boolean axioms and theorems:

- ANDs and ORs, 0's and 1's interchanged

Boolean Axioms

	Axiom
A1	$B = 0$ if $B \neq 1$
A2	$\overline{0} = 1$
A3	$0 \bullet 0 = 0$
A4	$1 \bullet 1 = 1$
A5	$0 \bullet 1 = 1 \bullet 0 = 0$

Boolean Axioms

	Axiom		Dual
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$

Dual: Exchange: • and +

0 and 1

Boolean Axioms

	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

Dual: Exchange: • and +

0 and 1

COMBINATIONAL LOGIC DESIGN

Basic Boolean Theorems

	Theorem
T1	$B \bullet 1 = B$
T2	$B \bullet 0 = 0$
T3	$B \bullet B = B$
T4	$\overline{\overline{B}} = B$
T5	$B \bullet \overline{B} = 0$

Basic Boolean Theorems: Duals

	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

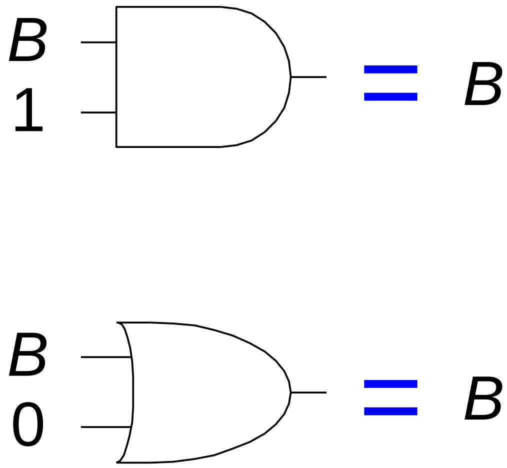
Dual: Exchange: • and +
0 and 1

T1: Identity Theorem

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

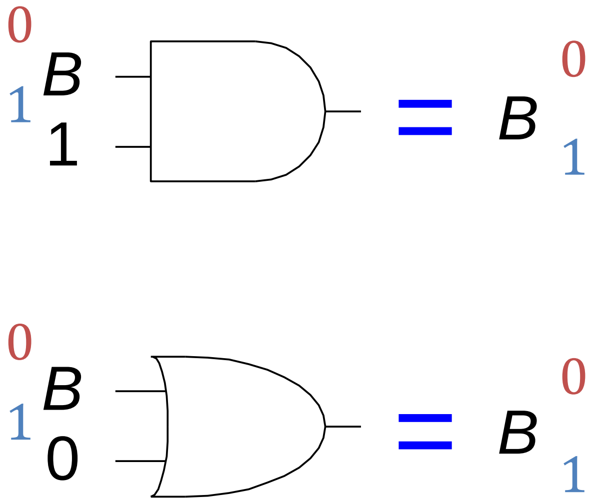
T1: Identity Theorem

- B • 1 = B
- B + 0 = B



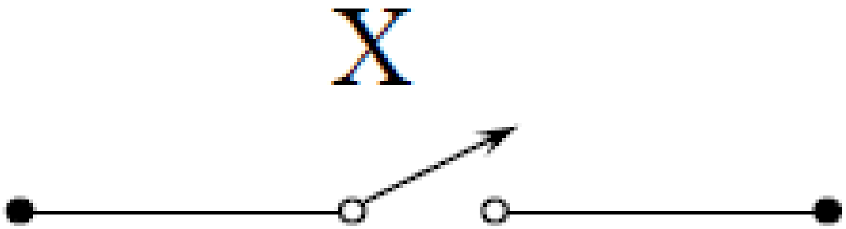
T1: Identity Theorem

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



Switching Algebra

- Simplification of digital logic → connecting wires with a on/off switch
- X = 0 (switch open)
- X = 1 (switch closed)



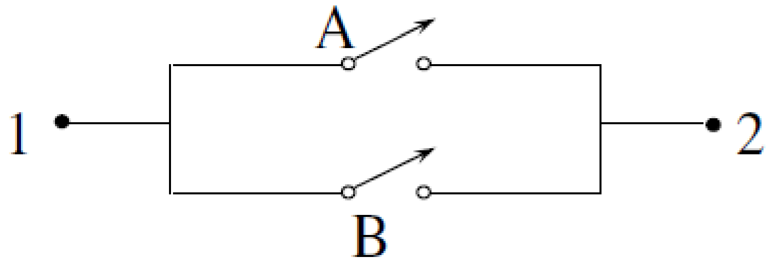
Series Switching Network: AND

- Switching circuit in series performs AND

- 1 is connected to 2 iff A **AND** B are 1

Parallel Switching Network: OR

- Switching circuit in parallel performs OR

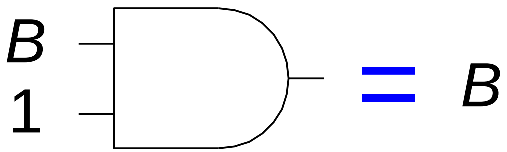


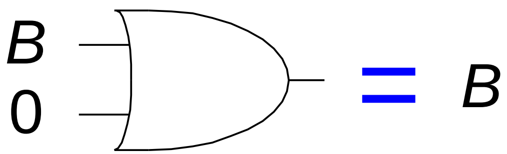
- 1 is connected to 2 if A **OR** B is 1

COMBINATIONAL LOGIC DESIGN

T1: Identity Theorem

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





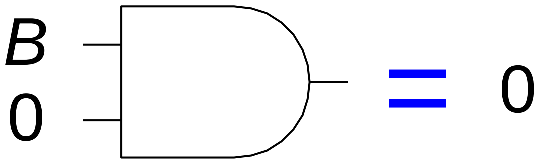
ELSEVIER

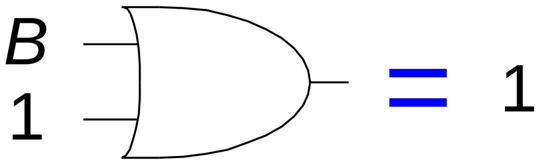
T2: Null Element Theorem

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

T2: Null Element Theorem

- B • 0 = 0
- B + 1 = 1



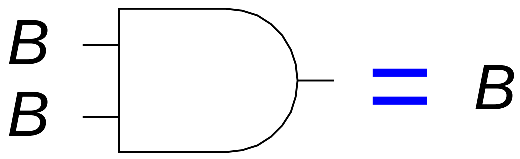


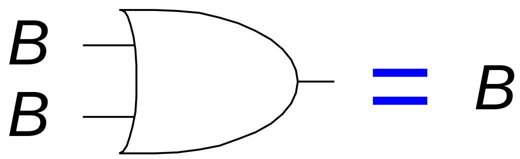
T3: Idempotency Theorem

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

T3: Idempotency Theorem

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





T4: Involution Theorem

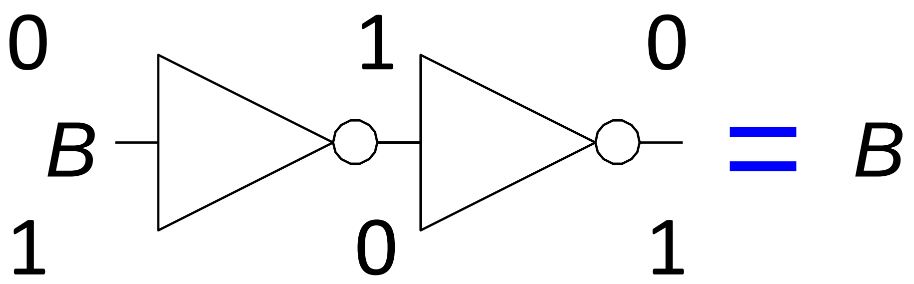
- $\overline{\overline{B}} = B$

ELSEVIER

COMBINATIONAL LOGIC DESIGN

T4: Involution Theorem

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

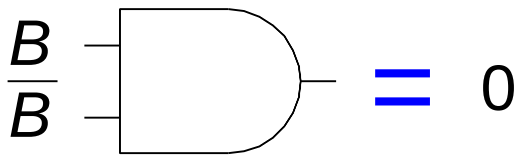


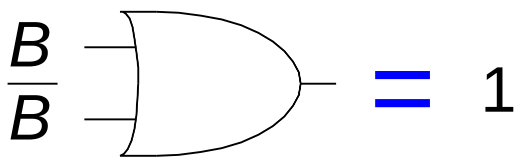
T5: Complements Theorem

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

T5: Complements Theorem

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





Recap: Basic Boolean Theorems

	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

COMBINATIONAL LOGIC DESIGN

Chapter 2.3.3

Theorems of Several Variables

ELSEVIER

Boolean Theorems of Several Vars

Number	Theorem	Name
T6	B•C = C•B	Commutativity
T7	(B•C) • D = B • (C • D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	Distributivity
T9	B• (B+C) = B	Covering
T10	(B•C) + (B•$\overline{C}$) = B	Combining
T11	B•C + ($\overline{B}$•D) + (C•D) = B•C + $\overline{B}$•D	Consensus

Boolean Theorems of Several Vars

Number	Theorem	Name
T6	$B \bullet C = C \bullet B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	Distributivity
T9	$B \bullet (B+C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	Combining
T11	$B \bullet C + (\overline{B} \bullet D) + (C \bullet D) = B \bullet C + \overline{B} \bullet D$	Consensus

How do we prove these are true?

How to Prove Boolean Relation

- **Method 1:** Perfect induction
- **Method 2:** Use other theorems and axioms to simplify the equation
  - Make one side of the equation look like the other

Proof by Perfect Induction

- Also called: proof by exhaustion
- Check every possible input value
- If two expressions produce the same value for every possible input combination, the expressions are equal

Example: Proof by Perfect Induction

Number	Theorem	Name
T6	B•C = C•B	Commutativity

B	*C*	*BC*	*CB*
0	0		
0	1		
1	0		
1	1		

Example: Proof by Perfect Induction

Number	Theorem	Name
T6	B•C = C•B	Commutativity

B	*C*	*BC*	*CB*
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Boolean Theorems of Several Vars

Number	Theorem	Name
T6	$B \bullet C = C \bullet B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	Distributivity
T9	$B \bullet (B+C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	Combining
T11	$B \bullet C + (\overline{B} \bullet D) + (C \bullet D) =$ $B \bullet C + \overline{B} \bullet D$	Consensus

T7: Associativity

Number	Theorem	Name
T7	(B•C) • D = B • (C • D)	Associativity

T8: Distributivity

Number	Theorem	Name
T8	B • (C + D) = (B•C) + (B•D)	Distributivity

T9: Covering

Number	Theorem	Name
T9	$B \bullet (B+C) = B$	Covering

T9: Covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

Prove true by:

- **Method 1:** Perfect induction
- **Method 2:** Using other theorems and axioms

T9: Covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

Method 1: Perfect Induction

B	*C*	*(B+C)*	*B(B+C)*
0	0		
0	1		
1	0		
1	1		

T9: Covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

Method 1: Perfect Induction

B	*C*	*(B+C)*	*B(B+C)*
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

T9: Covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

Method 2: Prove true using other axioms and theorems.

T9: Covering

Number	Theorem	Name
T9	$B \bullet (B+C) = B$	Covering

Method 2: Prove true using other axioms and theorems.

$B \bullet (B+C)$	$= B \bullet B + B \bullet C$	T8: Distributivity
	$= \mathbf{B} + B \bullet C$	T3: Idempotency
	$= B \bullet (1 + C)$	T8: Distributivity
	$= B \bullet (\mathbf{1})$	T2: Null element
	$= B$	T1: Identity

T10: Combining

Number	Theorem	Name
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	Combining

Prove true using other axioms and theorems:

T10: Combining

Number	Theorem	Name
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	Combining

Prove true using other axioms and theorems:

$$
\begin{aligned}
B \bullet C + B \bullet \overline{C} &= B \bullet (C + \overline{C}) && \text{T8: Distributivity} \\
&= B \bullet (\mathbf{1}) && \text{T5': Complements} \\
&= B && \text{T1: Identity}
\end{aligned}
$$

T11: Consensus

Number	Theorem	Name
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = (B \bullet C) + \overline{B} \bullet D$	Consensus

Prove true using (1) perfect induction or (2) other axioms and theorems.

Recap: Boolean Thms of Several Vars

Number	Theorem	Name
T6	B•C = C•B	Commutativity
T7	(B•C) • D = B • (C • D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	Distributivity
T9	B• (B+C) = B	Covering
T10	(B•C) + (B•$\overline{C}$) = B	Combining
T11	B•C + ($\overline{B}$•D) + (C•D) = B•C + $\overline{B}$•D	Consensus

Boolean Thms of Several Vars: Duals

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	(B•C) + (B•$\overline{C}$) = B	(B+C) • (B+$\overline{C}$) = B	Combining
T11	(B•C) + (B•$\overline{D}$) + (C•D) = (B•C) + (B•$\overline{D}$)	(B+C) • (B+$\overline{D}$) • (C+D) = (B+C) • (B+$\overline{D}$)	Consensus

Dual: Replace: • with +
0 with 1

COMBINATIONAL LOGIC DESIGN

ELSEVIER

Boolean Thms of Several Vars: Duals

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	(B•C) + (B•$\overline{C}$) = B	(B+C) • (B+$\overline{C}$) = B	Combining
T11	(B•C) + (B•$\overline{D}$) + (C•D) = (B•C) + (B•$\overline{D}$)	(B+C) • (B+$\overline{D}$) • (C+D) = (B+C) • (B+$\overline{D}$)	Consensus

Dual: Replace: • with +
0 with 1

Warning: T8' differs from traditional algebra: OR (+) distributes over AND (•)

Boolean Thms of Several Vars: Duals

#	Theorem	Dual	Name
T6	$B \bullet C = C \bullet B$	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	$B + (C \bullet D) = (B+C)\,(B+D)$	Distributivity
T9	$B \bullet (B+C) = B$	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	$(B+C) \bullet (B+\overline{C}) = B$	Combining
T11	$(B \bullet C) + (B \bullet \overline{D}) + (C \bullet D) =$ $(B \bullet C) + (B \bullet \overline{D})$	$(B+C) \bullet (B+\overline{D}) \bullet (C+D) =$ $(B+C) \bullet (B+\overline{D})$	Consensus

Axioms and theorems are useful for *simplifying* equations.

Chapter 2.3.5

Simplifying Equations

Simplifying an Equation

- Reducing an equation to the fewest number of implicants, where each implicant has the fewest literals

Simplifying an Equation

- Reducing an equation to the fewest number of implicants, where each implicant has the fewest literals

Recall:

- **Implicant:** product of literals
  $AB\overline{C}, AC, \overline{B}C$
- **Literal:** variable or its complement
  $A, \overline{A}, B, \overline{B}, C, \overline{C}$

Simplifying an Equation

- Reducing an equation to the fewest number of implicants, where each implicant has the fewest literals

Recall:

- **Implicant:** product of literals
  $AB\bar{C}$, AC, $\bar{B}C$
- **Literal:** variable or its complement
  A, $\bar{A}$, B, $\bar{B}$, C, $\bar{C}$

- Also called: **minimizing** the equation

Simplification methods

- **Distributivity (T8, T8')**

  B (C+D) = BC + BD

  B + CD = (B+ C)(B+D)

- **Covering (T9')**

  A + AP = A

- **Combining (T10)**

  $PA + P\overline{A} = P$

P	A	Minterm
0	0	$\bar{P}\bar{A}$
0	1	$\bar{P}A$
1	0	$P\bar{A}$
1	1	PA

Simplification methods

- **Distributivity (T8, T8')**

  B (C+D) = BC + BD

  B + CD = (B+ C)(B+D)

- **Covering (T9')**

  A + AP = A

- **Combining (T10)**

  $PA + P\bar{A} = P$

- **Expansion**

  $P = PA + P\bar{A}$

  A = A + AP

- **Duplication**

  A = A + A

P	A	Minterm
0	0	$\bar{P}\bar{A}$
0	1	$\bar{P}A$
1	0	$P\bar{A}$
1	1	PA

Simplification methods

- **Distributivity (T8, T8')**

  $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$

- **Covering (T9')**

  $A + AP = A$

- **Combining (T10)**

  $PA + P\overline{A} = P$

- **Expansion**

  $P = PA + P\overline{A}$

  $A = A + AP$

- **Duplication**

  $A = A + A$

- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

Simplification methods

- **A combination of Combining/Covering**

$$PA + \bar{A} = P + \bar{A}$$

Proof:

$PA + \bar{A}$	$= PA + (\bar{A} + \bar{A}P)$	**T9' Covering**
	$= PA + P\bar{A} + \bar{A}$	**T6 Commutativity**
	$= P(A + \bar{A}) + \bar{A}$	**T8 Distributivity**
	$= P(1) + \bar{A}$	**T5' Complements**
	$= P + \bar{A}$	**T1 Identity**

Recap: Boolean Thms of Several Vars

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	$(B+C) \bullet (B+\overline{C}) = B$	Combining
T11	$(B \bullet C) + (B \bullet \overline{D}) + (C \bullet D) =$ $(B \bullet C) + (B \bullet \overline{D})$	$(B+C) \bullet (B+\overline{D}) \bullet (C+D) =$ $(B+C) \bullet (B+\overline{D})$	Consensus

T11: Consensus

Number	Theorem	Name
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = (B \bullet C) + \overline{B} \bullet D$	Consensus

Prove using other theorems and axioms:

T11: Consensus

Number	Theorem	Name
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = (B \bullet C) + \overline{B} \bullet D$	Consensus

Prove using other theorems and axioms:

$B \bullet C + \overline{B} \bullet D + C \bullet D$

$= BC + \overline{B}D + (CDB + CD\overline{B})$	**T10: Combining**
$= BC + \overline{B}D + BCD + \overline{B}CD$	**T6: Commutativity**
$= BC + BCD + \overline{B}D + \overline{B}CD$	**T6: Commutativity**
$= (BC + BCD) + (\overline{B}D + \overline{B}CD)$	**T7: Associativity**
$= BC + \overline{B}D$	**T9': Covering**

Simplification methods

- **Distributivity (T8, T8')**

  $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$

- **Covering (T9')**

  $A + AP = A$

- **Combining (T10)**

  $PA + P\overline{A} = P$

- **Expansion**

  $P = PA + P\overline{A}$

  $A = A + AP$

- **Duplication**

  $A = A + A$

- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

Simplifying Boolean Equations

Example 1:

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

Simplifying Boolean Equations

Example 1:

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

Y = A	T10: Combining
or	
$= A(B + \overline{B})$	T8: Distributivity
$= A(1)$	T5': Complements
$= A$	T1: Identity

Simplification methods

- **Distributivity (T8, T8')**

  $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$

- **Covering (T9')**

  $A + AP = A$

- **Combining (T10)**

  $PA + P\overline{A} = P$

- **Expansion**

  $P = PA + P\overline{A}$

  $A = A + AP$

- **Duplication**

  $A = A + A$

- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

Simplifying Boolean Equations

Example 2:

$Y = A(AB + ABC)$

Simplifying Boolean Equations

Example 2:

$Y = A(AB + ABC)$

$= A(AB(1 + C))$	T8: Distributivity
$= A(AB(1))$	T2': Null Element
$= A(AB)$	T1: Identity
$= (AA)B$	T7: Associativity
$= AB$	T3: Idempotency

Simplification methods

- **Distributivity (T8, T8')**

  B (C+D) = BC + BD

  B + CD = (B+ C)(B+D)

- **Covering (T9')**

  A + AP = A

- **Combining (T10)**

  $PA + P\overline{A} = P$

- **Expansion**

  $P = PA + P\overline{A}$

  A = A + AP

- **Duplication**

  A = A + A

- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

COMBINATIONAL LOGIC DESIGN

Simplifying Boolean Equations

Example 3:

$Y = A'BC + A'$

Recall: $A' = \overline{A}$

Simplifying Boolean Equations

Example 3:

$Y = A'BC + A'$

Recall: $A' = \overline{A}$

$= A'$ — T9' Covering: $X + XY = X$

or

$= A'(BC + 1)$ — T8: Distributivity

$= A'(1)$ — T2': Null Element

$= A'$ — T1: Identity

COMBINATIONAL LOGIC DESIGN

Simplification methods

- **Distributivity (T8, T8')**

  $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$

- **Covering (T9')**

  $A + AP = A$

- **Combining (T10)**

  $PA + P\overline{A} = P$

- **Expansion**

  $P = PA + P\overline{A}$

  $A = A + AP$

- **Duplication**

  $A = A + A$

- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

Simplifying Boolean Equations

Example 4:

Y = AB'C + ABC + A'BC

Simplifying Boolean Equations

Example 4:

Y = AB'C + ABC + A'BC

= AB'C + **ABC + ABC** + A'BC	T3': Idempotency
= (AB'C+ABC) + (ABC+A'BC)	T7': Associativity
= AC + BC	T10: Combining

Simplification methods

- **Distributivity (T8, T8')**

  B (C+D) = BC + BD

  B + CD = (B+ C)(B+D)
- **Covering (T9')**

  A + AP = A
- **Combining (T10)**

  $PA + P\overline{A} = P$
- **Expansion**

  $P = PA + P\overline{A}$

  A = A + AP
- **Duplication**

  A = A + A
- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

Simplifying Boolean Equations

Example 5:

$Y = AB + BC + B'D' + AC'D'$

Simplifying Boolean Equations

Example 5:

Y = AB + BC +B'D' + AC'D'

Method 1:

Y = AB + BC + B'D' + (ABC'D' + AB'C'D')	T10: Combining
= (AB + ABC'D') + BC + (B'D' + AB'C'D')	T6: Commutativity
	T7: Associativity
= AB + BC + B'D'	T9: Covering

Simplifying Boolean Equations

Example 5:

$Y = AB + BC + B'D' + AC'D'$

Method 1:

$Y = AB + BC + B'D' + (ABC'D' + AB'C'D')$	T10: Combining
$= (AB + ABC'D') + BC + (B'D' + AB'C'D')$	T6: Commutativity
	T7: Associativity
$= AB + BC + B'D'$	T9: Covering

Method 2:

$Y = \mathbf{AB} + BC + \mathbf{B'D'} + AC'D' + \mathbf{AD'}$	T11: Consensus
$= AB + BC + B'D' + \mathbf{AD'}$	T9: Covering
$= AB + BC + B'D'$	T11: Consensus

ELSEVIER

Simplification methods

- **Distributivity (T8, T8')**

  $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$

- **Covering (T9')**

  $A + AP = A$

- **Combining (T10)**

  $PA + P\overline{A} = P$

- **Expansion**

  $P = PA + P\overline{A}$

  $A = A + AP$

- **Duplication**

  $A = A + A$

- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

Simplifying Boolean Equations

Example 6:

$Y = (A + BC)(A + DE)$

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Simplifying Boolean Equations

Example 6:

$Y = (A + BC)(A + DE)$

Apply T8' first when possible: $W+XZ = (W+X)(W+Z)$

Make: $X = BC$, $Z = DE$ and rewrite equation

Y	$= (A+X)(A+Z)$	substitution ($X=BC$, $Z=DE$)
	$= A + XZ$	T8': Distributivity
	$= A + BCDE$	substitution

or

Y	$= AA + ADE + ABC + BCDE$	T8: Distributivity
	$= A + ADE + ABC + BCDE$	T3: Idempotency
	$= \mathbf{A + ADE} + ABC + BCDE$	
	$= \mathbf{A} \quad + ABC + BCDE$	T9': Covering
	$= A + BCDE$	T9': Covering

Simplifying Boolean Equations

Example 6:

Y = (A + BC)(A + DE)

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Make: X = BC, Z = DE and rewrite equation

Y	= (A+X)(A+Z)	substitution (X=BC, Z=DE)
	= A + XZ	T8': Distributivity
	= A + BCDE	substitution

or

Y	= AA + ADE + ABC + BCDE	T8: Distributivity
	= A + ADE + ABC + BCDE	T3: Idempotency
	= **A + ADE** + ABC + BCDE	
	= **A** + ABC + BCDE	T9': Covering
	= A + BCDE	T9': Covering

This is called *multiplying out* an expression to get sum-of-products (SOP) form.

Multiplying Out: SOP Form

An expression is in simplified **sum-of-products (SOP)** form when all products contain literals only.

- SOP form: Y = AB + BC' + DE
- **NOT** SOP form: Y = DF + E(A'+B)
- SOP form: Z = A + BC + DE'F

Multiplying Out: SOP Form

Example:

$Y = (A + C + D + E)(A + B)$

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Multiplying Out: SOP Form

Example:

Y = (A + C + D + E)(A + B)

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Make: X = (C+D+E), Z = B and rewrite equation

Y	= (A+X)(A+Z)	substitution (X=(C+D+E), Z=B)
	= A + XZ	T8': Distributivity
	= A + (C+D+E)B	substitution
	= A + BC + BD + BE	T8: Distributivity

or

Y	= AA+AB+AC+BC+AD+BD+AE+BE	T8: Distributivity
	= **A**+AB+AC+AD+AE+BC+BD+BE	T3: Idempotency
	= **A** + BC + BD + BE	T9': Covering

Canonical SOP & POS Form

- SOP – sum-of-products

O	*C*	*E*	minterm
0	0	0	$\overline{O}\;\overline{C}$
0	1	0	$\overline{O}\;C$
1	0	1	$O\;\overline{C}$
1	1	0	$O\;C$

$$E = O\overline{C}$$
$$= \Sigma(m_2)$$

- POS – product-of-sums

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

$$E = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$
$$= \Pi(M0, M1, M3)$$

Factoring: POS Form

An expression is in simplified **product-of-sums (POS)** form when all sums contain literals only.

- POS form: Y = (A+B)(C+D)(E'+F)
- **NOT** POS form: Y = (D+E)(F'+GH)
- POS form: Z = A(B+C)(D+E')

Factoring: POS Form

Example 1:

Y = (A + B'CDE)

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Factoring: POS Form

Example 1:

Y = (A + B'CDE)

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Make: X = B'C, Z = DE and rewrite equation

Y	= (A+XZ)	substitution (X=B'C, Z=DE)
	= (A+B'C)(A+DE)	T8': Distributivity
	= (A+B')(A+C)(A+D)(A+E)	T8': Distributivity

Factoring: POS Form

Example 2:

Y = AB + C'DE + F

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Factoring: POS Form

Example 2:

Y = AB + C'DE + F

Apply T8' first when possible: W+XZ = (W+X)(W+Z)

Make: W = AB, X = C', Z = DE and rewrite equation

Y	= (W+XZ) + F	substitution W = AB, X = C', Z = DE
	= (W+X)(W+Z) + F	T8': Distributivity
	= (AB+C')(AB+DE)+F	substitution
	= (A+C')(B+C')(AB+D)(AB+E)+F	T8': Distributivity
	= (A+C')(B+C')(A+D)(B+D)(A+E)(B+E)+F	T8': Distributivity
	= (A+C'+F)(B+C'+F)(A+D+F)(B+D+F)(A+E+F)(B+E+F)	T8': Distributivity

Boolean Thms of Several Vars: Duals

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	$(B+C) \bullet (B+\overline{C}) = B$	Combining
T11	$(B \bullet C) + (B \bullet \overline{D}) + (C \bullet D) = (B \bullet C) + (B \bullet \overline{D})$	$(B+C) \bullet (B+\overline{D}) \bullet (C+D) = (B+C) \bullet (B+\overline{D})$	Consensus

Axioms and theorems are useful for *simplifying* equations.

Simplification methods

- **Distributivity (T8, T8')**

  $B(C+D) = BC + BD$

  $B + CD = (B+C)(B+D)$

- **Covering (T9')**

  $A + AP = A$

- **Combining (T10)**

  $PA + P\overline{A} = P$

- **Expansion**

  $P = PA + P\overline{A}$

  $A = A + AP$

- **Duplication**

  $A = A + A$

- **A combination of Combining/Covering**

  $PA + \overline{A} = P + \overline{A}$

DeMorgan's Theorem

Number	Theorem	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	DeMorgan's Theorem

COMBINATIONAL LOGIC DESIGN

DeMorgan's Theorem

Number	Theorem	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	DeMorgan's Theorem

The **complement of the product**
is the
sum of the complements

DeMorgan’s Theorem: Dual

#	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgan’s Theorem

DeMorgan's Theorem: Dual

#	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgan's Theorem

The complement of the product
is the
sum of the complements.

DeMorgan's Theorem: Dual

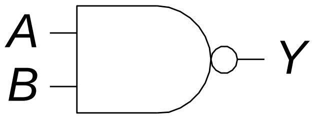
#	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgan's Theorem

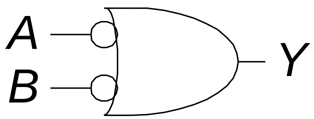
The complement of the product
is the
sum of the complements.

Dual: The **complement of the sum**
is the
product of the complements.

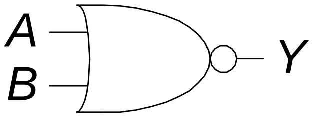
DeMorgan's Theorem

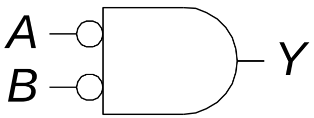
- $Y = \overline{AB} = \overline{A} + \overline{B}$





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





DeMorgan's Theorem Example 1

$$Y = \overline{(A+\overline{BD})\overline{C}}$$

DeMorgan's Theorem Example 1

COMBINATIONAL LOGIC DESIGN

$$Y = \overline{(A+\overline{BD})\overline{C}}$$

$$= \overline{(A+\overline{BD})} + \overline{\overline{C}}$$

$$= (\overline{A} \bullet (\overline{\overline{BD}})) + C$$

$$= (\overline{A} \bullet (BD)) + C$$

$$= \overline{A}BD + C$$

DeMorgan's Theorem Example 2

$$Y = \overline{(\overline{AC}E+\overline{D}) + B}$$

COMBINATIONAL LOGIC DESIGN

DeMorgan's Theorem Example 2

$$Y = \overline{(\overline{AC}E + \overline{D}) + B}$$

$$= \overline{(\overline{AC}E + \overline{D})} \bullet \overline{B}$$

$$= (\overline{\overline{AC}E} \bullet \overline{\overline{D}}) \bullet \overline{B}$$

$$= ((\overline{\overline{AC}} + \overline{E}) \bullet D) \bullet \overline{B}$$

$$= ((AC + \overline{E}) \bullet D) \bullet \overline{B}$$

$$= (ACD + D\overline{E}) \bullet \overline{B}$$

$$= A\overline{B}CD + \overline{B}D\overline{E}$$

Canonical SOP & POS Form Revisited

- SOP – sum-of-products

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

How do we implement this logic function with gates?

$$E = O\overline{C}$$
$$= \Sigma(m_2)$$

- POS – product-of-sums

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

$$E = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$
$$= \Pi(M0, M1, M3)$$

Canonical POS Expansion

- "Add" literal/complement terms to reverse simplification (→expand literal)
- Example
  - $Y = C$
  - $Y = C + A\bar{A}$
  - $Y = (C + A) \cdot (C + \bar{A})$
  - $Y = [(C + A) + B\bar{B}](C + \bar{A})$
  - $Y = [(C + A + B)(C + A + \bar{B})](C + \bar{A})$
  - …

COMBINATIONAL LOGIC DESIGN

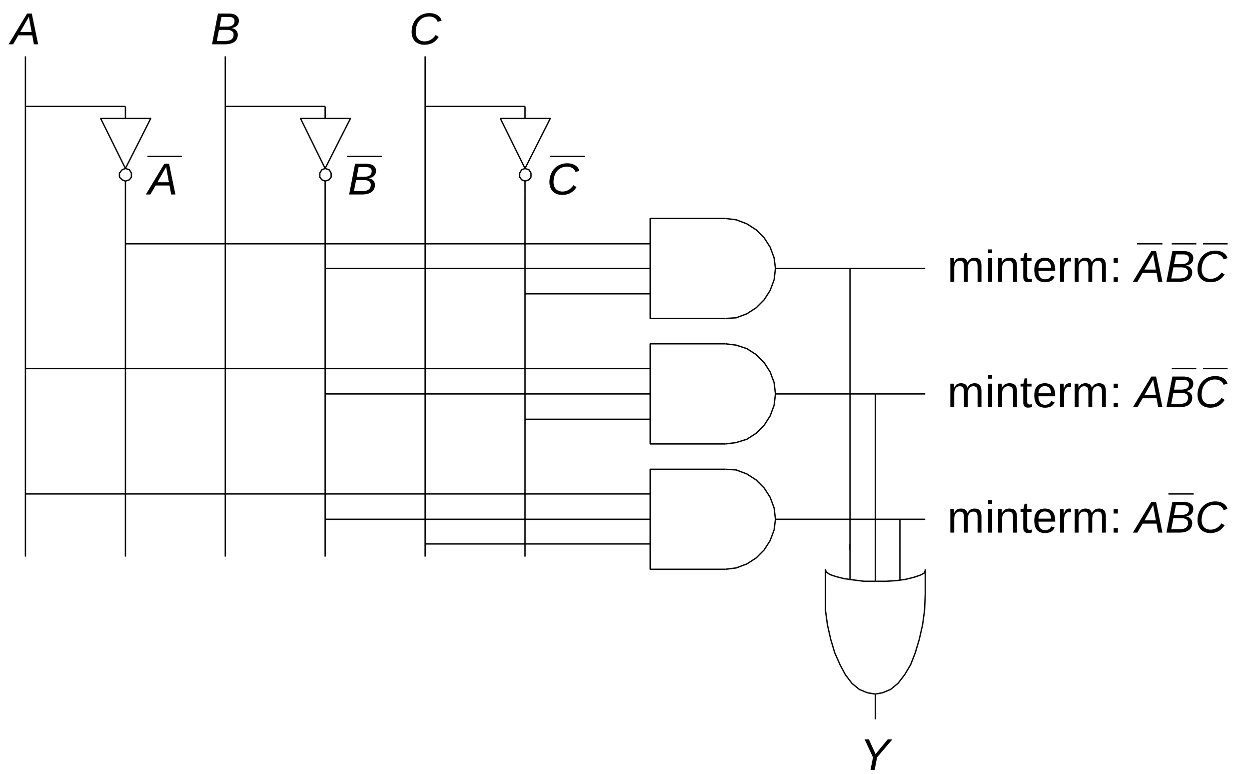
Chapter 2.4

From Logic to Gates

ELSEVIER

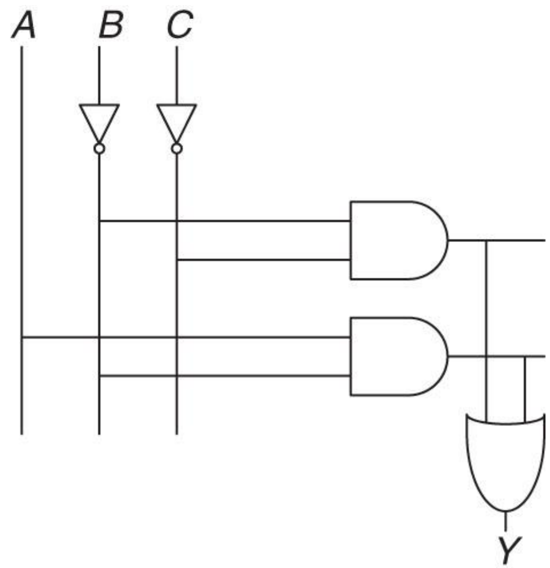
From Logic to Gates

- Two-level logic: ANDs followed by ORs
- Example: $Y = \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$



Circuit Schematics Rules

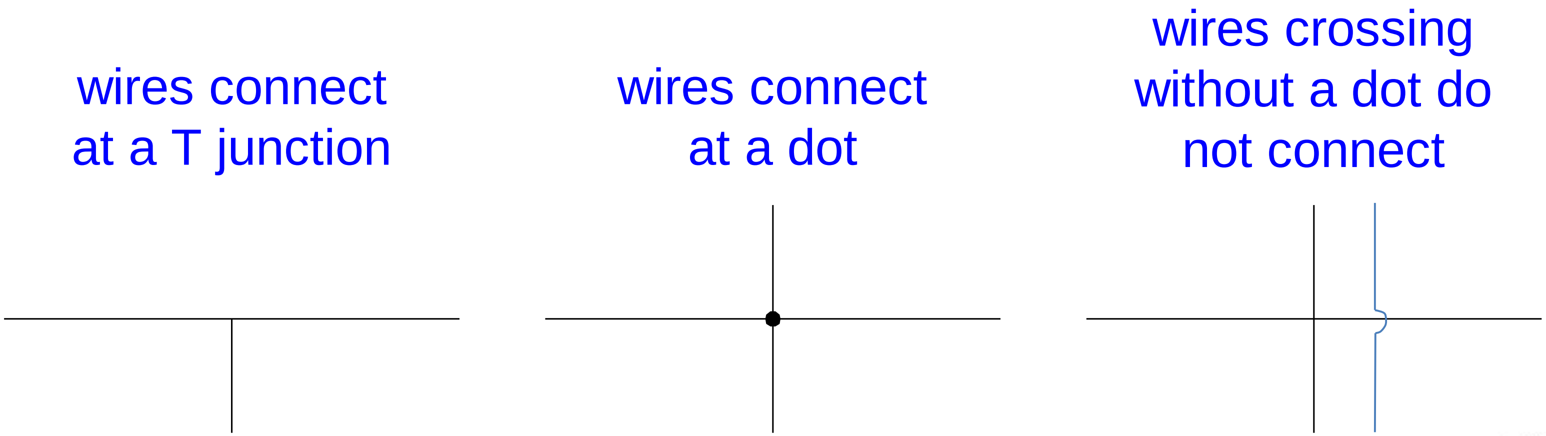
- Inputs on the left (or top)
- Outputs on right (or bottom)
- Gates flow from left to right
- Straight wires are best



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

Circuit Schematic Rules (cont.)

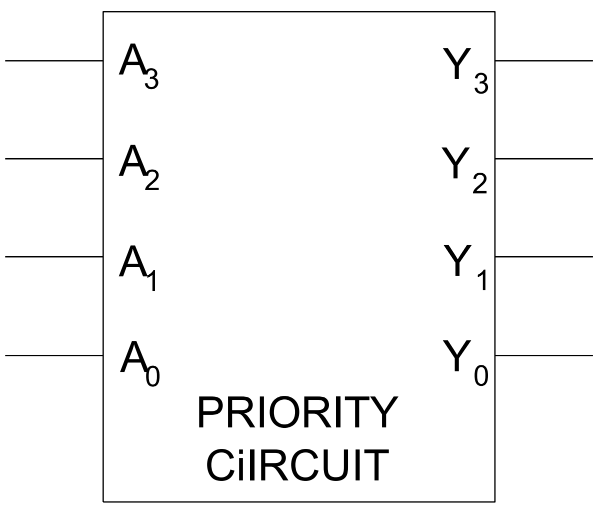
- Wires always connect at a T junction
- A dot where wires cross indicates a connection between the wires
- Wires crossing *without* a dot make no connection



Multiple-Output Circuits

- **Example: Priority Circuit**

  Output asserted corresponding to most significant TRUE input

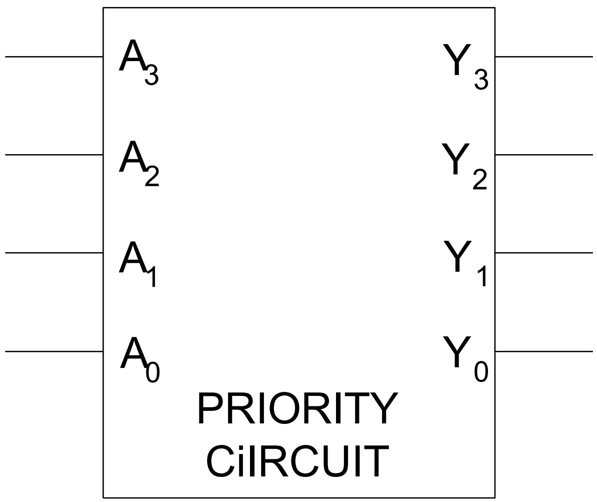


A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

Multiple-Output Circuits

- **Example: Priority Circuit**

  Output asserted corresponding to most significant TRUE input

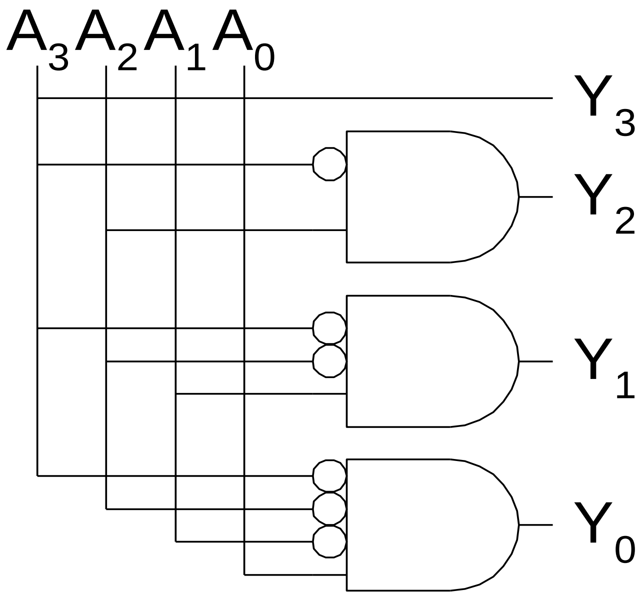


A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

COMBINATIONAL LOGIC DESIGN

Priority Circuit 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



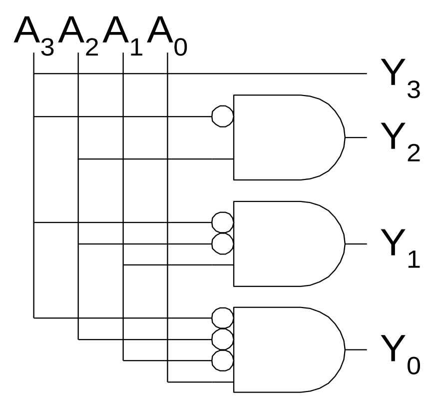
Don't Cares

- Simplify truth table by ignoring entries

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

Much easier to read off Boolean equations



$Y_3 = A_3$

$Y_2 = \overline{A_3} A_2$

$Y_1 = \overline{A_3}\, \overline{A_2}\, A_1$

$Y_0 = \overline{A_3}\, \overline{A_2}\, \overline{A_1}\, A_0$

COMBINATIONAL LOGIC DESIGN

Chapter 2.5

Multilevel Combinational Logic

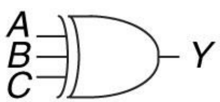
Multilevel Hardware Simplification

- Two-level logic: One level-ANDs, one-level OR
  - Can have high hardware requirements
  - Multiple inputs to AND and OR gates
- Multilevel design can simplify schematic

Multilevel Hardware Simplification

- 3-input XOR example (odd parity)
  - $Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

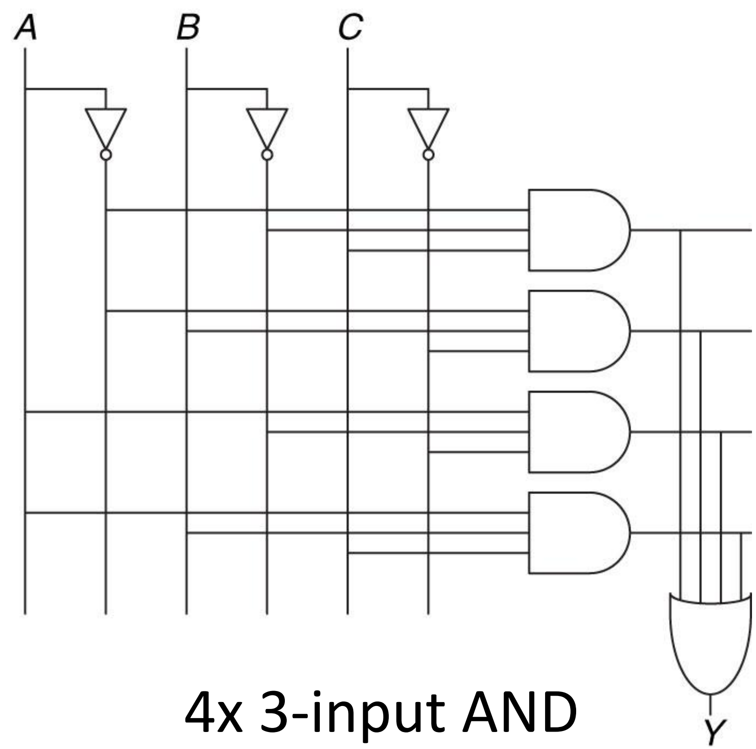
XOR3



$Y = A \oplus B \oplus C$

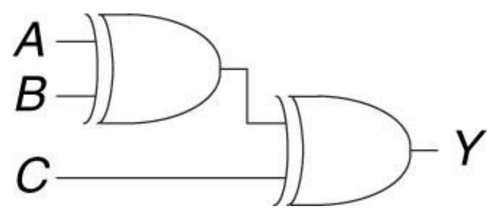
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(a)



(b)

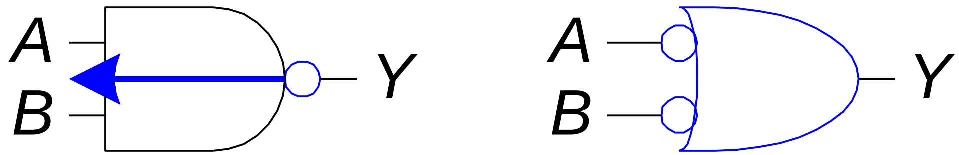
4x 3-input AND
1x 4-input OR



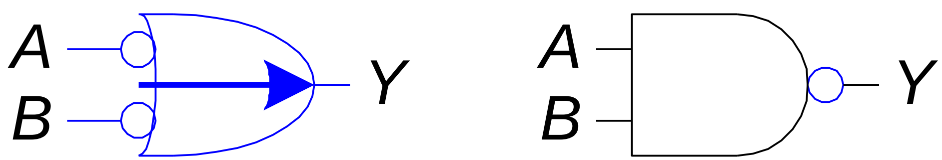
2x 2-input XOR

Bubble Pushing

- **Backward:**
  - Body changes
  - Adds bubbles to inputs

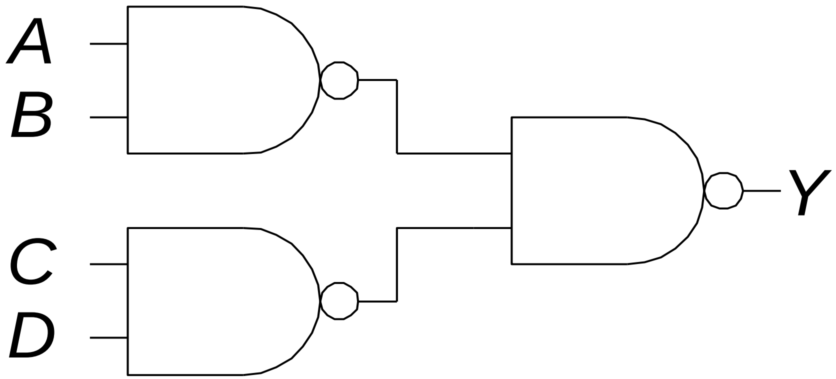


- **Forward:**
  - Body changes
  - Adds bubble to output



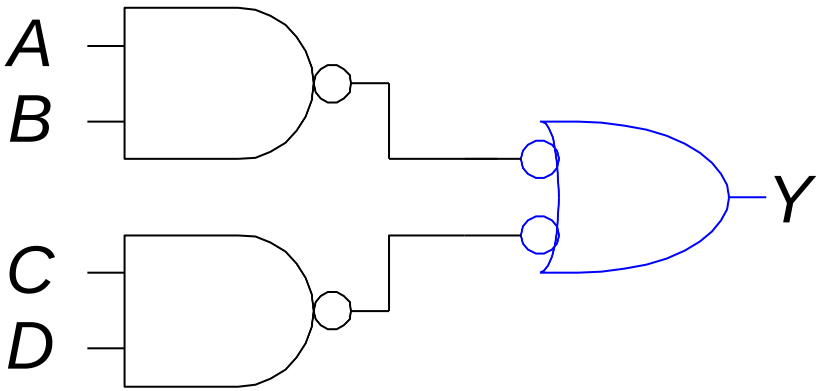
Bubble Pushing

- What is the Boolean expression for this circuit?



Bubble Pushing

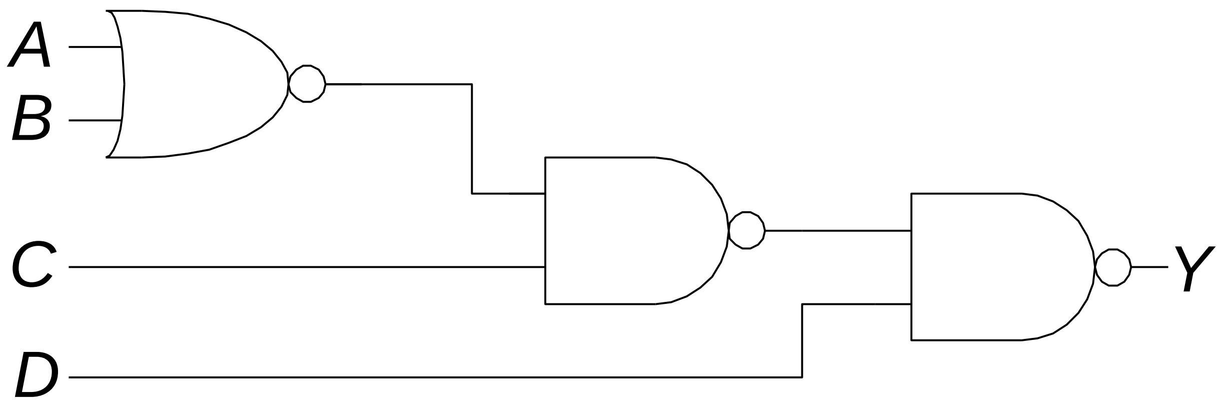
- What is the Boolean expression for this circuit?



$$Y = AB + CD$$

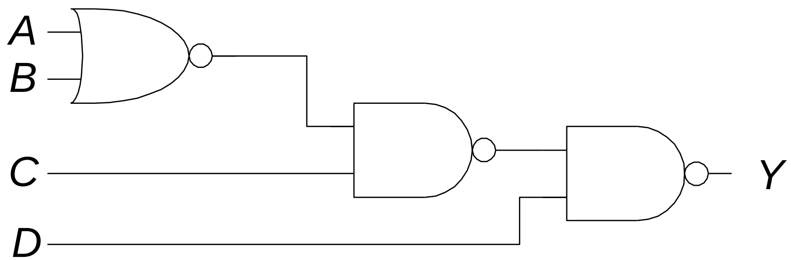
Bubble Pushing Rules

- Begin at output, then work toward inputs
- Push bubbles on final output back
- Draw gates in a form so bubbles cancel

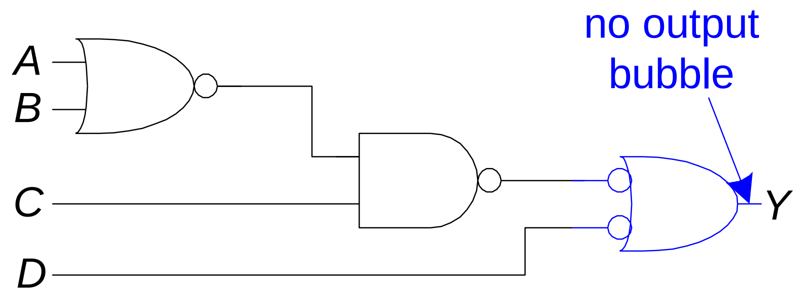


ELSEVIER

Bubble Pushing Example

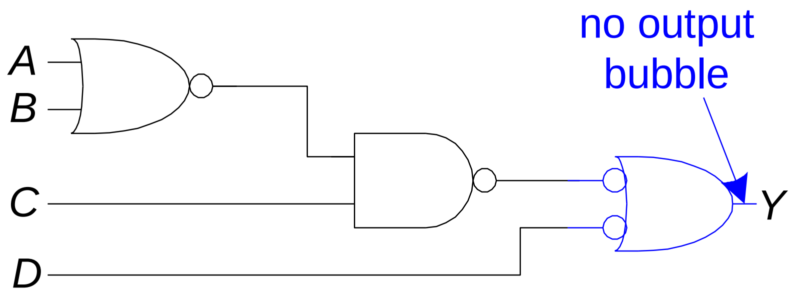


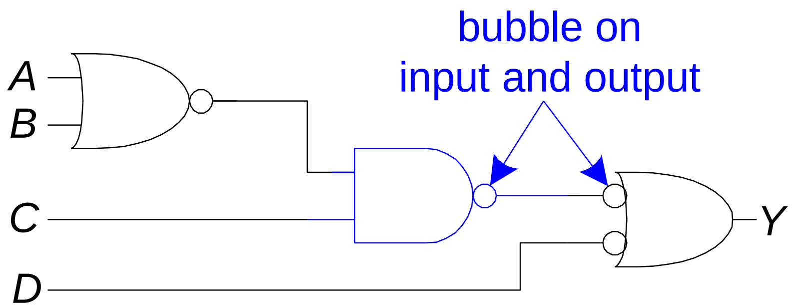
Bubble Pushing Example



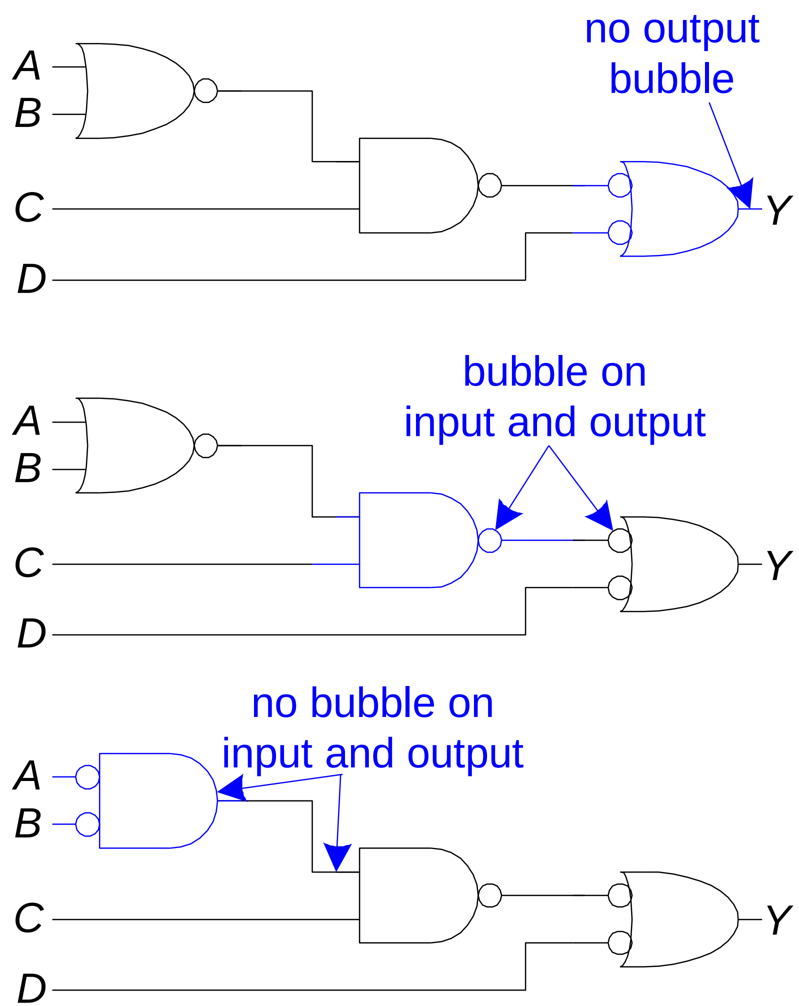
ELSEVIER

Bubble Pushing Example





Bubble Pushing Example



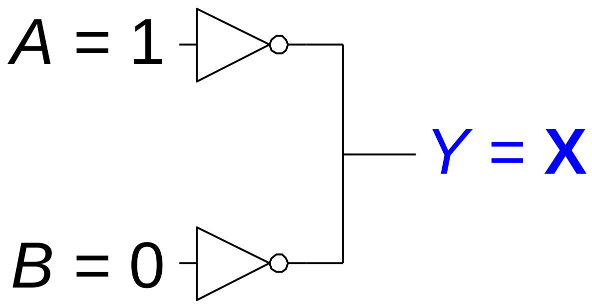
$$Y = \overline{A}\overline{B}C + \overline{D}$$

Chapter 2.6

X's and Z's, Oh My

Contention: X

- Contention: circuit tries to drive output to 1 **and** 0
  - Actual value somewhere in between
  - Could be 0, 1, or in forbidden zone
  - Might change with voltage, temperature, time, noise
  - Often causes excessive power dissipation

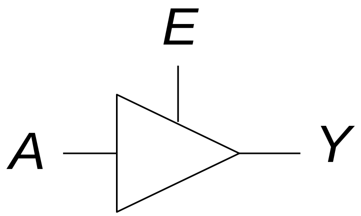


- **Warnings:**
  - Contention usually indicates a **bug**.
  - **X is used for "don't care" and contention** - look at the context to tell them apart

Floating: Z

- Floating, high impedance, open, high Z
- Floating output might be 0, 1, or somewhere in between
  - A voltmeter won't indicate whether a node is floating

Tristate Buffer

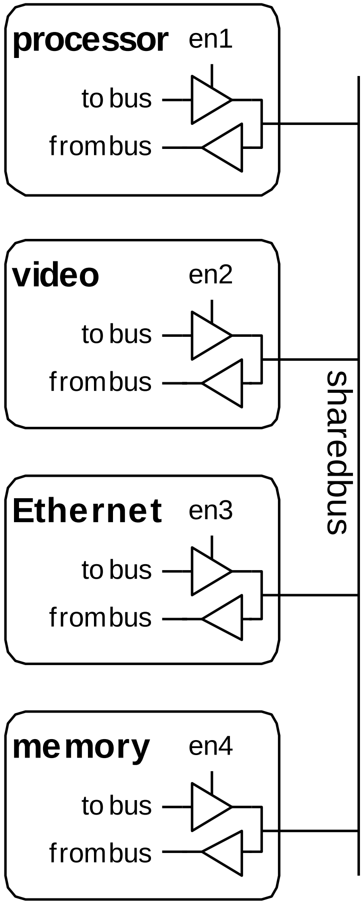


E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Note: tristate buffer has an enable bit (E) to turn on the gate

Tristate Busses

- Floating nodes are used in tristate busses
  - Many different drivers
  - Exactly one is active at once



COMBINATIONAL LOGIC DESIGN

Chapter 2.7

Karnaugh Maps

ELSEVIER

Karnaugh Maps (K-Maps)

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

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

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

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

K-Map

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

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 — *C* \ *AB*	00	01	11	10
0	1	0	0	0
1	1	0	0	0

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

C not included because both C and $\bar{C}$ included in circle

ELSEVIER

3-Input K-Map

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	0
1	1	0	0
1	1	1	1

K-Map

Y (C \ AB)	00	01	11	10
0				
1				

3-Input K-Map

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	0
1	1	0	0
1	1	1	1

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

K-Map

Y: C \ AB	00	01	11	10
0	0 (0)	1 (2)	0 (6)	0 (4)
1	0 (1)	1 (3)	1 (7)	0 (5)

$\bar{A}B$ BC

K-Map Definitions

- **Complement:** variable with a bar over it

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

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

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

K-Map Rules

- Every **1 must be circled** at least once
- 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**
- A “don't care” (**X**) is **circled only if it helps** minimize the equation

4-Input K-Map

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

Y

CD \ AB	00	01	11	10
00				
01				
11				
10				

4-Input K-Map

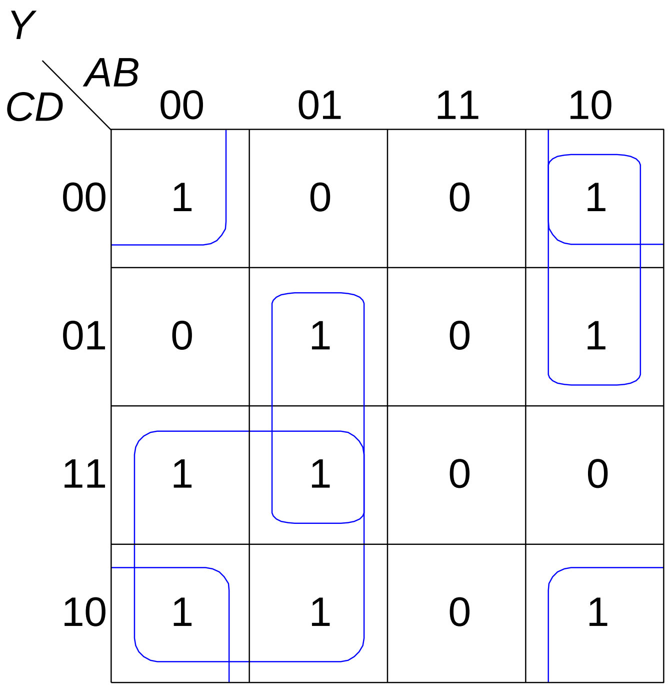
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

4-Input K-Map

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



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

K-Maps with Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

CD \ AB	00	01	11	10
00				
01				
11				
10				

K-Maps with Don't Cares

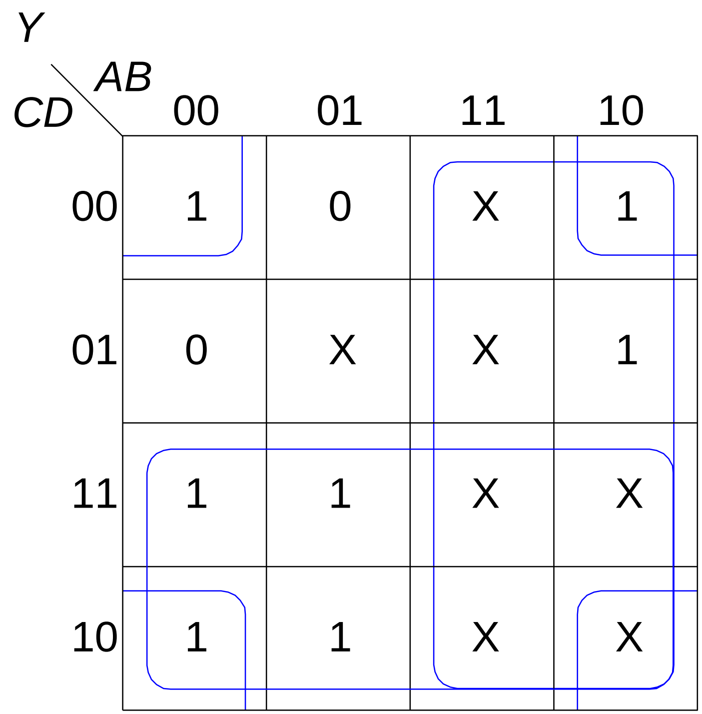
A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

CD \ *AB*	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

K-Maps with Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



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

ELSEVIER

4-Input K-Map: POS & SOP Form

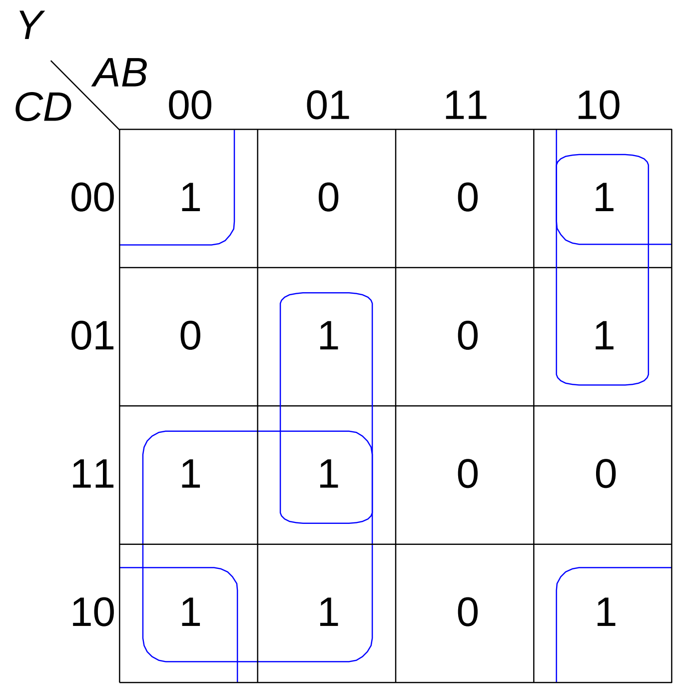
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

4-Input K-Map: SOP Form

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



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

4-Input K-Map: POS Form

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

Y

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

COMBINATIONAL LOGIC DESIGN

Chapter 2.8

Combinational Building Blocks

ELSEVIER

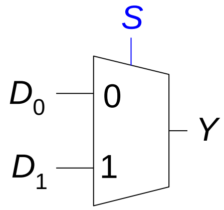
Combinational Building Blocks

- Multiplexers
- Decoders

Multiplexer (Mux)

- Selects between one of *N* inputs to connect to output
  - $\log_2 N$-bits required to select input – control input S
- Example:

  2:1 Mux (2 inputs to 1 output)
  - $N = 2$
  - $\log_2 2 = 1$ control bit required



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

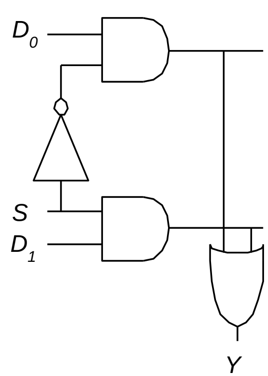
Multiplexer Implementations

- Logic gates
  - Sum-of-products form

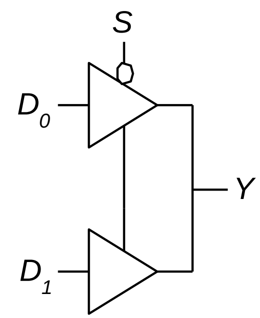
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Y (S \ $D_0 D_1$)	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$$Y = D_0\overline{S} + D_1S$$



- Tristates
  - For an N-input mux, use N tristates
  - Turn on exactly one to select the appropriate input



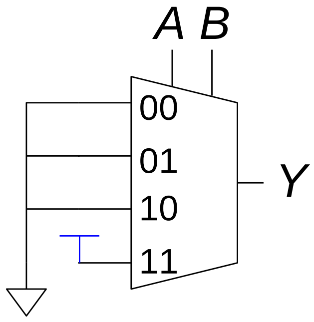
ELSEVIER

Logic using Multiplexers

- Using the mux as a lookup table
  - Zero outputs tied to GND
  - One output tied to VDD

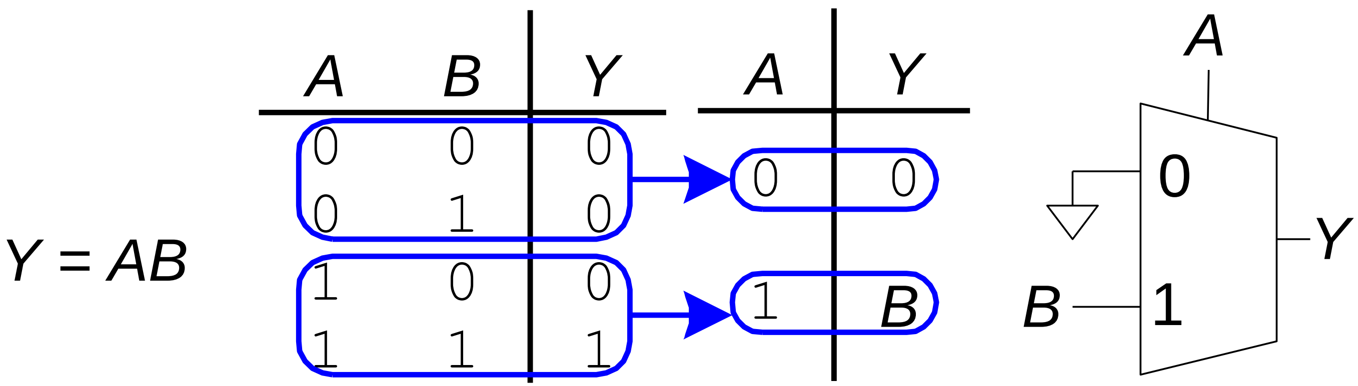
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



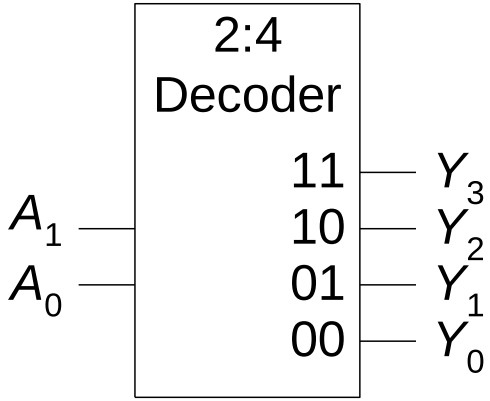
Logic using Multiplexers

- Reducing the size of the mux



Decoders

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



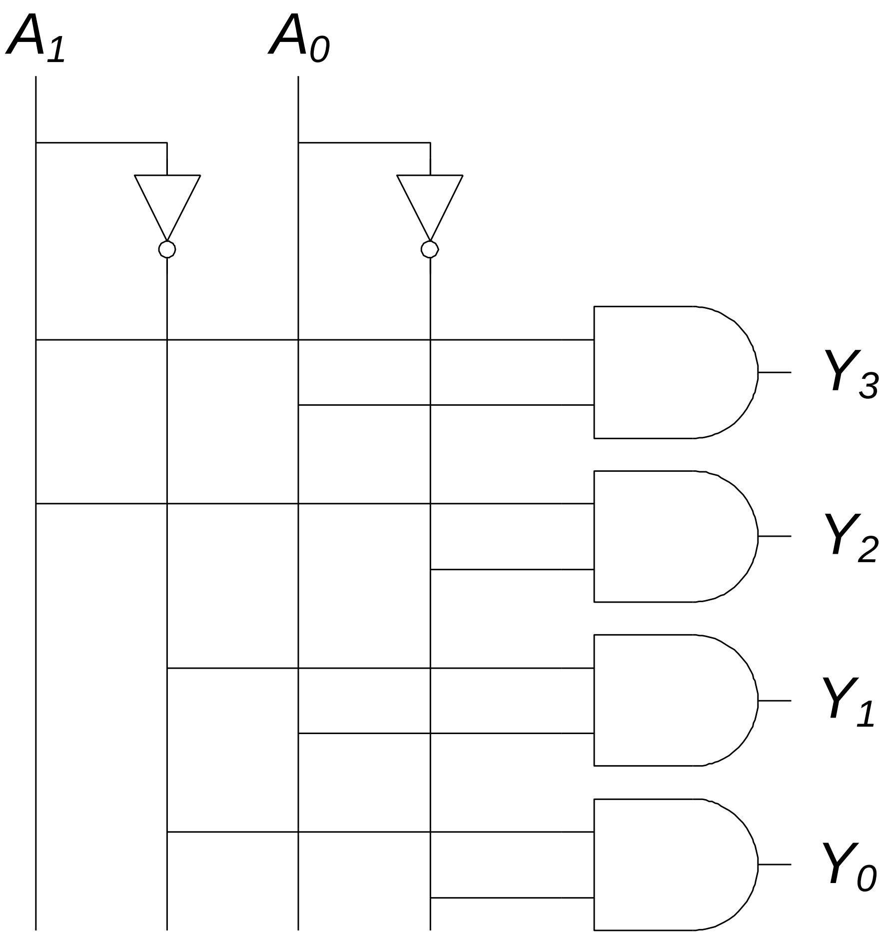
- Example

  2:4 Decoder (2 inputs to 4 outputs)

  - A_i decimal value selects the corresponding output

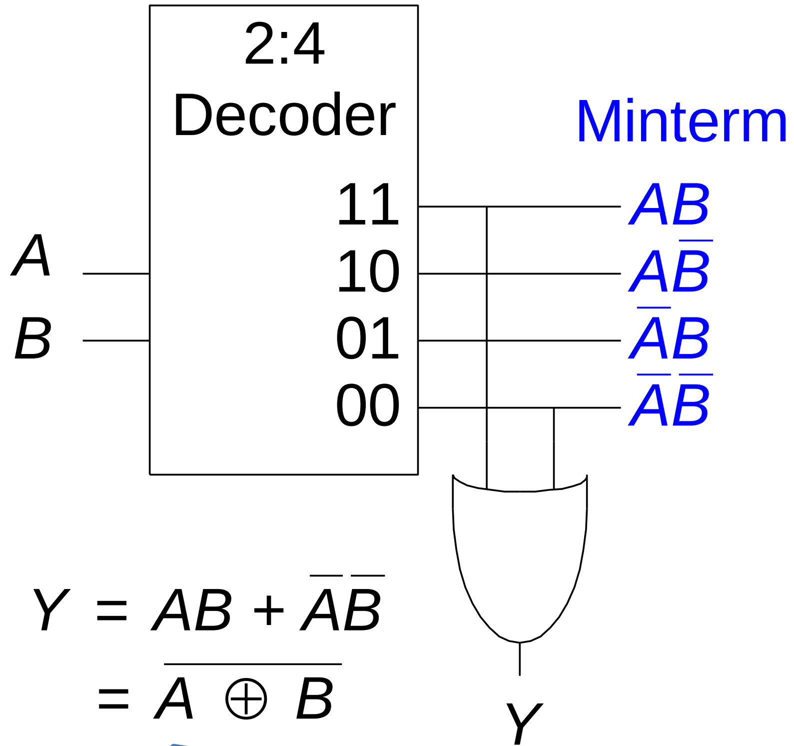
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Decoder Implementation



Logic Using Decoders

- OR minterms



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

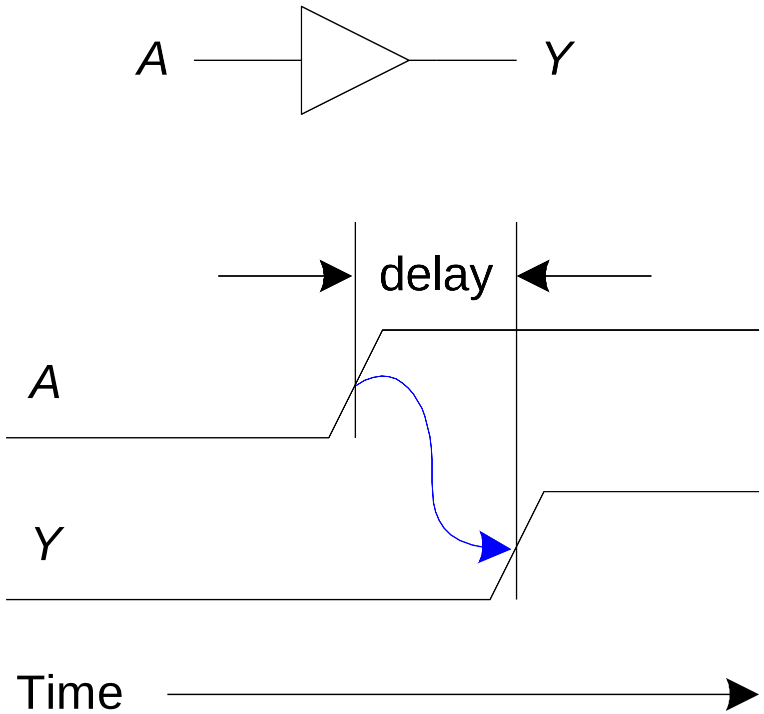
XNOR function

Chapter 2.9

Timing

Timing

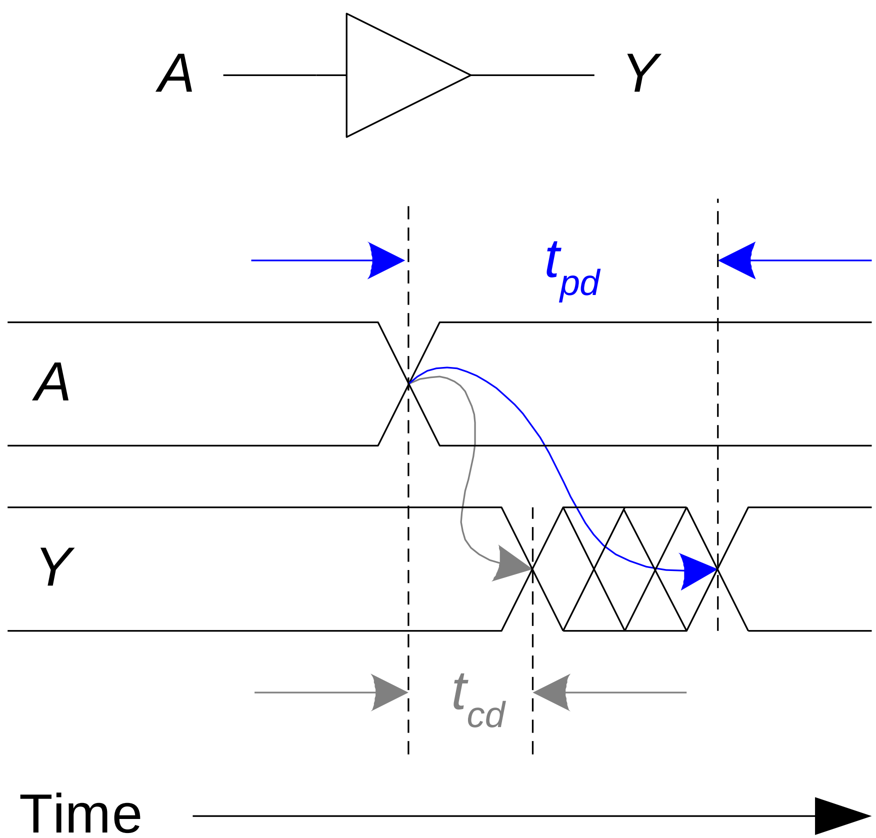
- Delay between input change and output changing



- How to build fast circuits?

Propagation & Contamination Delay

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



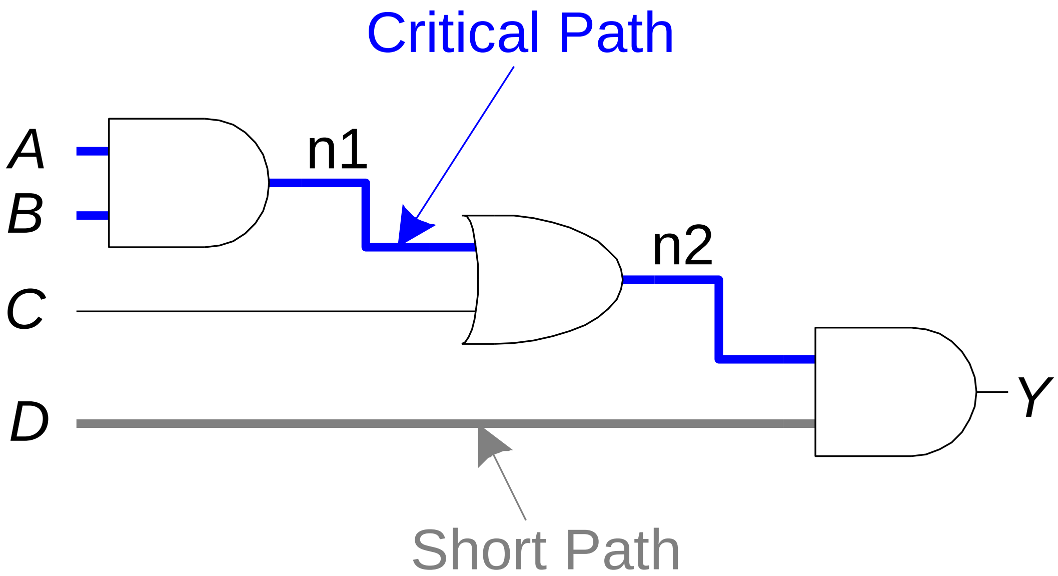
Note: Timing diagram shows a signal with a high and low and transition time as an 'X'.

Cross hatch indicates unknown/changing values

Propagation & Contamination Delay

- Delay is caused by
  - Capacitance and resistance in a circuit
  - Speed of light limitation
- Reasons why t_{pd} and t_{cd} may be different:
  - Different rising and falling delays
  - Multiple inputs and outputs, some of which are faster than others
  - Circuits slow down when hot and speed up when cold

Critical (Long) & Short Paths



Critical (Long) Path: $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$

Short Path: $t_{cd} = t_{cd\_AND}$

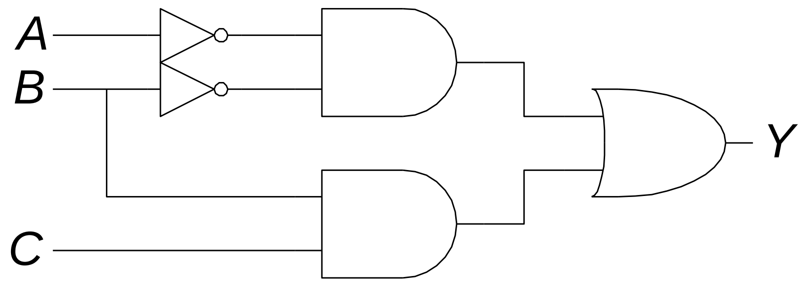
ELSEVIER

Glitches

- When a single input change causes an output to change multiple times

Glitch Example

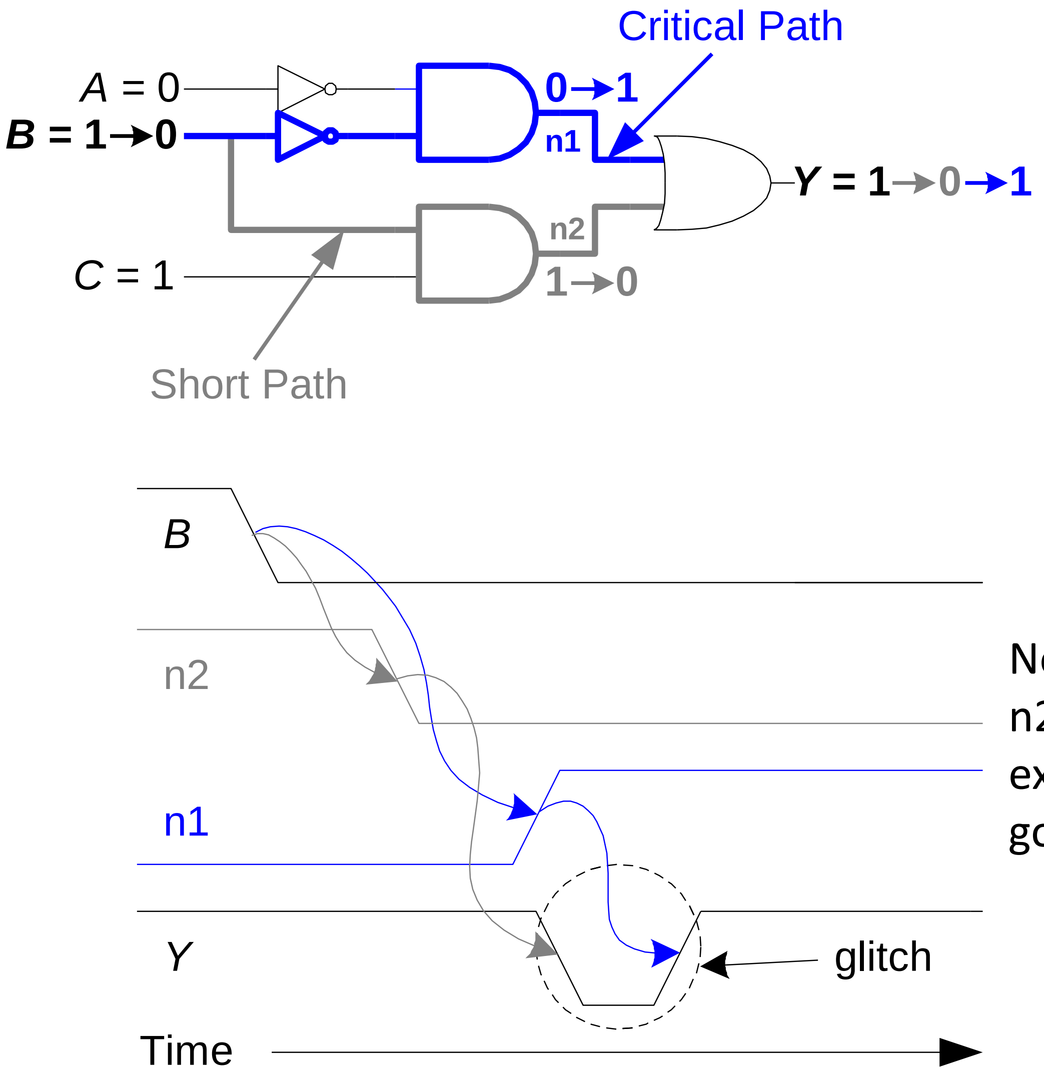
- What happens when A = 0, C = 1, B falls?



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

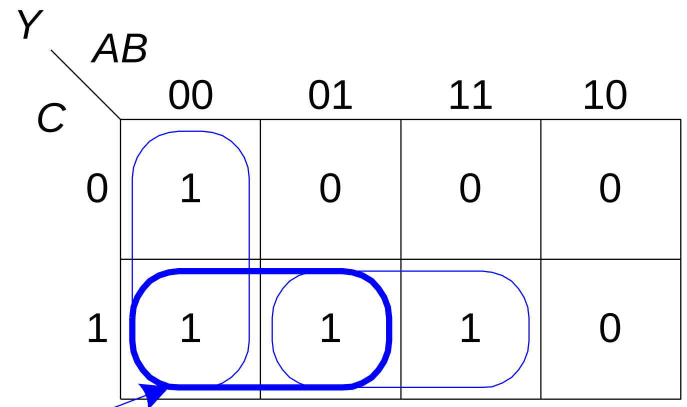
$$Y = \overline{A}\,\overline{B} + BC$$

Glitch Example (cont.)



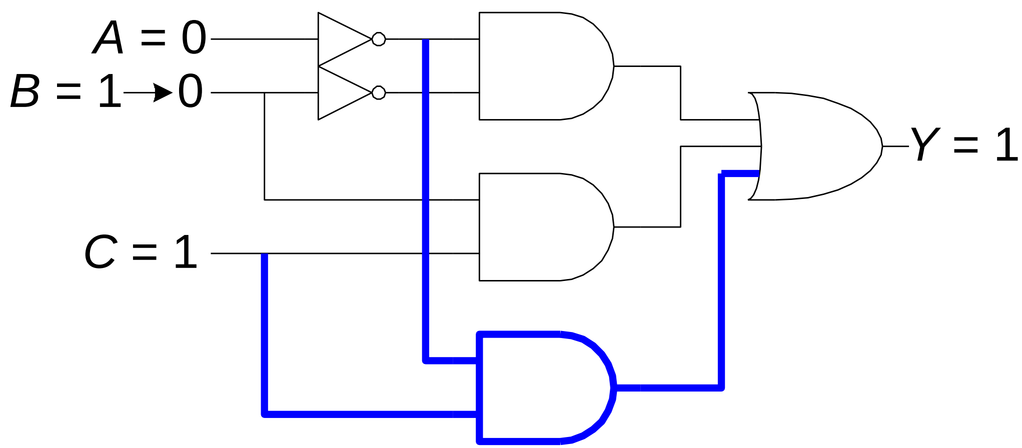
Note: n1 is slower than n2 because of the extra inverter for B to go through

Fixing the Glitch



Consensus term $\bar{A}C$

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



Why Understand Glitches?

- Glitches shouldn't cause problems because of **synchronous design** conventions (see Chapter 3)
- It's important to **recognize** a glitch: in simulations or on oscilloscope
- Can't get rid of all glitches – simultaneous transitions on multiple inputs can also cause glitches