

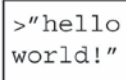


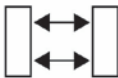
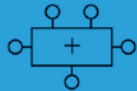

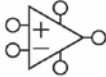


Chapter 2

Digital Design and Computer Architecture, 2nd Edition

David Money Harris and Sarah L. Harris

Chapter 2 :: Topics

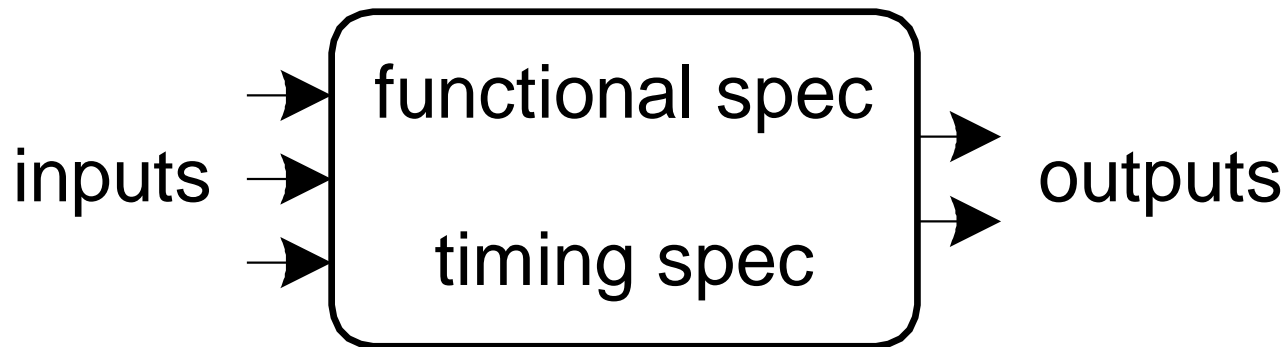
- Introduction
- Boolean Equations
- Boolean Algebra
- From Logic to Gates
- Multilevel Combinational Logic
- X's and Z's, Oh My
- Karnaugh Maps
- Combinational Building Blocks
- Timing

Application Software	
Operating Systems	
Architecture	
Micro-architecture	
Logic	
Digital Circuits	
Analog Circuits	
Devices	
Physics	

Introduction

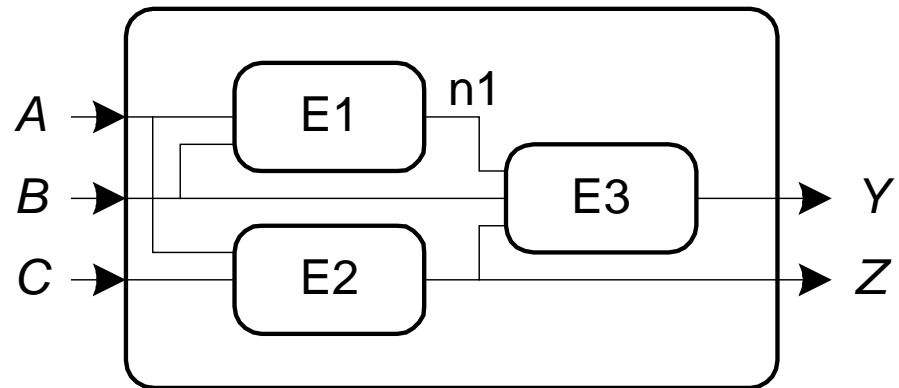
A logic circuit is composed of:

- 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



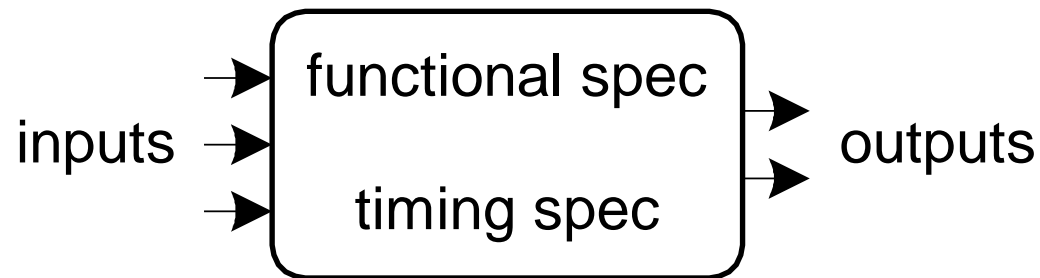
Types of Logic Circuits

- **Combinational Logic**

- Memoryless
- Outputs determined by current values of inputs

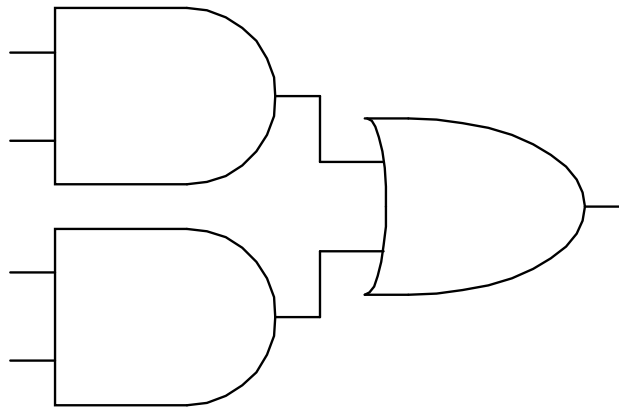
- **Sequential Logic**

- Has memory
- Outputs determined by previous and current values of inputs



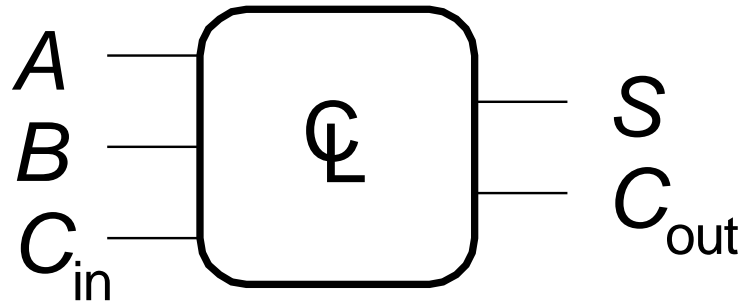
Rules of Combinational Composition

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



Boolean Equations

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



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

Some 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
 $ABC, \bar{A}C, BC$
- Minterm: product that includes all input variables
 $ABC, \bar{A}\bar{B}\bar{C}, ABC$
- Maxterm: sum that includes all input variables
 $(A+\bar{B}+C), (\bar{A}+B+\bar{C}), (\bar{A}+\bar{B}+C)$

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

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

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	$\bar{A} \bar{B}$	m_0
0	1	1	$\bar{A} B$	m_1
1	0	0	$A \bar{B}$	m_2
1	1	1	$A B$	m_3

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

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	$\bar{A} \bar{B}$	m_0
0	1	1	$\bar{A} B$	m_1
1	0	0	$A \bar{B}$	m_2
1	1	1	$A B$	m_3

$$Y = F(A, B) = \bar{A}B + AB = \Sigma(1, 3)$$

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

$$Y = F(A, B) = (A + B)(A + \overline{B}) = \Pi(0, 2)$$

Boolean Equations Example

- You are going to the cafeteria for lunch
 - You won't eat lunch (\bar{E})
 - If it's not open (\bar{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	1	
1	0	
1	1	

Boolean Equations Example

- You are going to the cafeteria for lunch
 - You won't eat lunch (\bar{E})
 - If it's not open (\bar{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

<i>O</i>	<i>C</i>	<i>E</i>	minterm	
0	0		\overline{O}	\overline{C}
0	1		\overline{O}	<i>C</i>
1	0		<i>O</i>	\overline{C}
1	1		<i>O</i>	<i>C</i>

- POS – product-of-sums

<i>O</i>	<i>C</i>	<i>E</i>	maxterm		
0	0		<i>O</i>	+	<i>C</i>
0	1		<i>O</i>	+	\overline{C}
1	0		\overline{O}	+	<i>C</i>
1	1		\overline{O}	+	\overline{C}



SOP & POS Form

- SOP – sum-of-products

<i>O</i>	<i>C</i>	<i>E</i>	minterm
0	0	0	$\overline{O} \overline{C}$
0	1	0	$\overline{O} C$
1	0	1	$O \overline{C}$
1	1	0	$O C$

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

$$= \Sigma(2)$$

- POS – product-of-sums

<i>O</i>	<i>C</i>	<i>E</i>	maxterm
0	0	0	$O + C$
0	1	0	$O + \overline{C}$
1	0	1	$\overline{O} + C$
1	1	0	$\overline{O} + \overline{C}$

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

$$= \Pi(0, 1, 3)$$



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		Dual		Name
A1	$B = 0 \text{ if } B \neq 1$	A1'	$B = 1 \text{ if } B \neq 0$	Binary field
A2	$\bar{0} = 1$	A2'	$\bar{1} = 0$	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		$\bar{\bar{B}} = B$		Involution
T5	$B \bullet \bar{B} = 0$	T5'	$B + \bar{B} = 1$	Complements

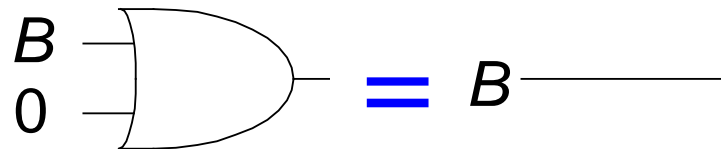
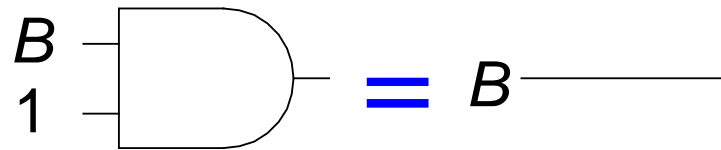


T1: Identity Theorem

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

T1: Identity Theorem

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

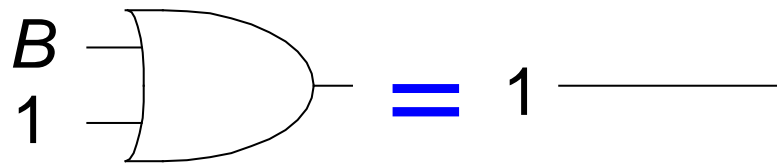
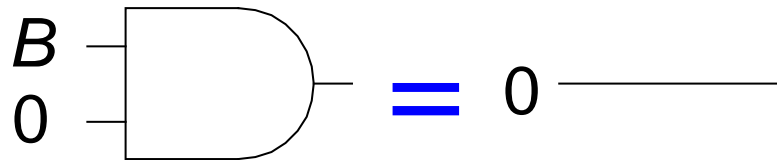


T2: Null Element Theorem

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

T2: Null Element Theorem

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

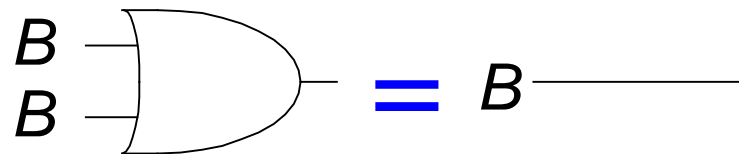
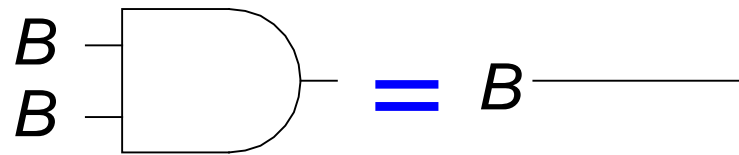


T3: Idempotency Theorem

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

T3: Idempotency Theorem

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

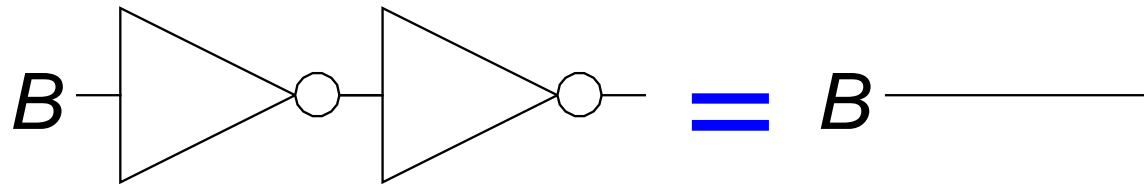


T4: Identity Theorem

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

T4: Identity Theorem

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

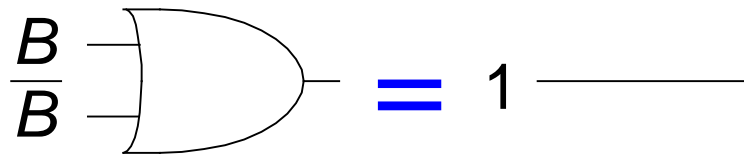
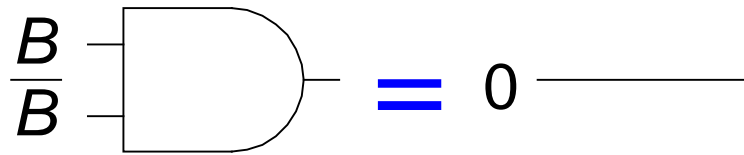


T5: Complement Theorem

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

T5: Complement Theorem

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



Boolean Theorems Summary

	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 Several Vars

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

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



Simplifying Boolean Equations

Example 1:

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

Simplifying Boolean Equations

Example 1:

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

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

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

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

Simplifying Boolean Equations

Example 2:

$$Y = A(AB + ABC)$$

Simplifying Boolean Equations

Example 2:

$$Y = A(AB + ABC)$$

$$= A(AB(1 + C)) \quad \text{T8}$$

$$= A(AB(1)) \quad \text{T2'}$$

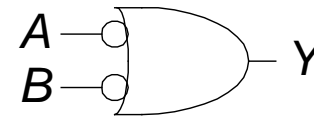
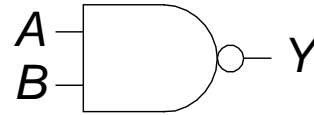
$$= A(AB) \quad \text{T1}$$

$$= (AA)B \quad \text{T7}$$

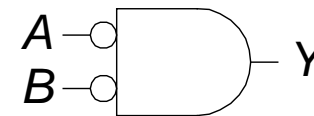
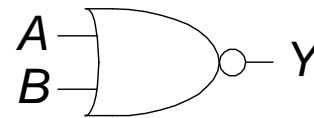
$$= AB \quad \text{T3}$$

DeMorgan's Theorem

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



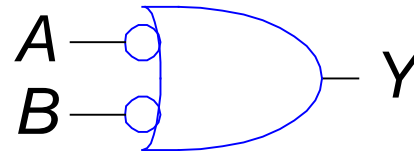
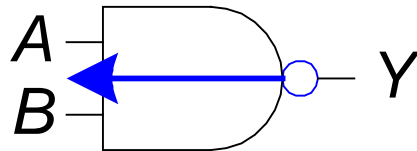
- $Y = \overline{A + B} = \overline{A} \cdot \overline{B}$



Bubble Pushing

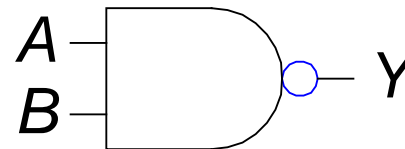
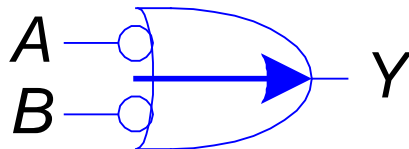
- **Backward:**

- Body changes
- Adds bubbles to inputs



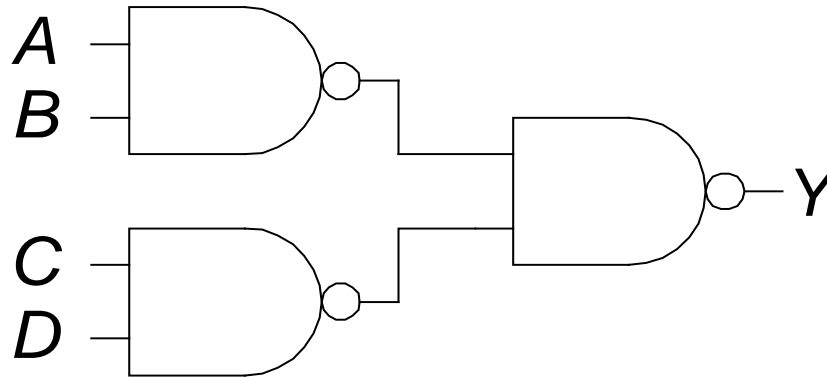
- **Forward:**

- Body changes
- Adds bubble to output



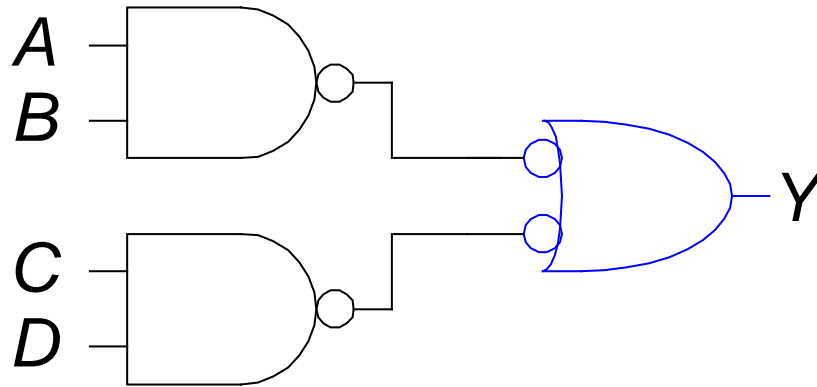
Bubble Pushing

- What is the Boolean expression for this circuit?



Bubble Pushing

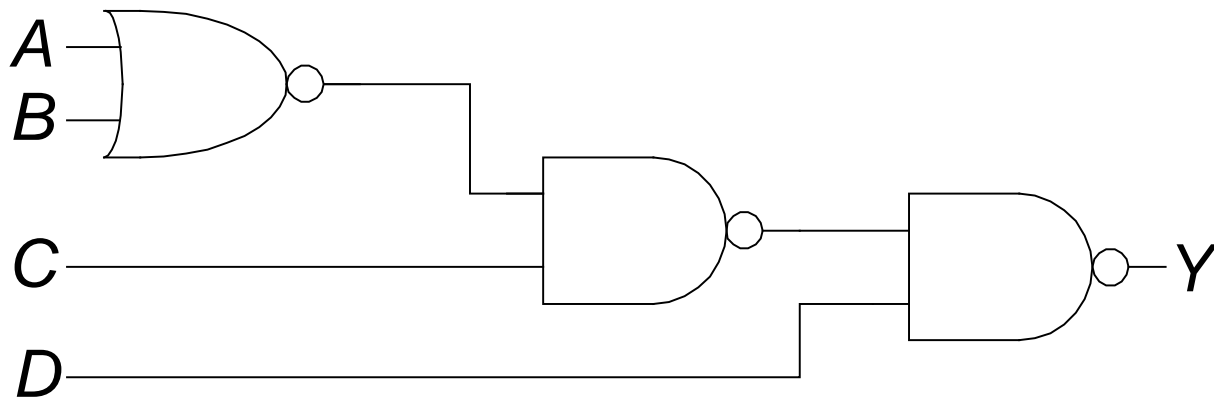
- What is the Boolean expression for this circuit?



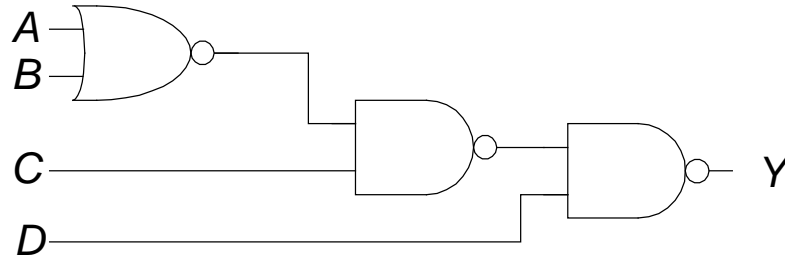
$$Y = AB + CD$$

Bubble Pushing Rules

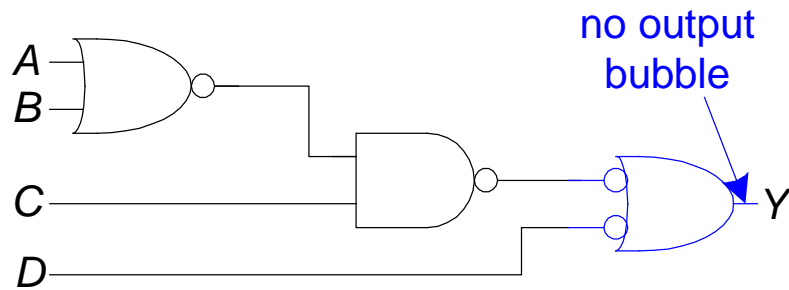
- Begin at output, then work toward inputs
- Push bubbles on final output back
- Draw gates in a form so bubbles cancel



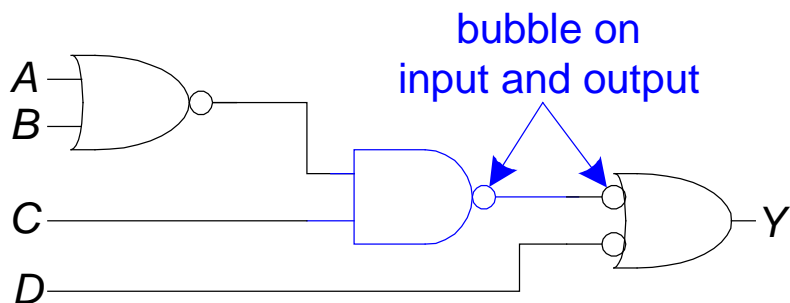
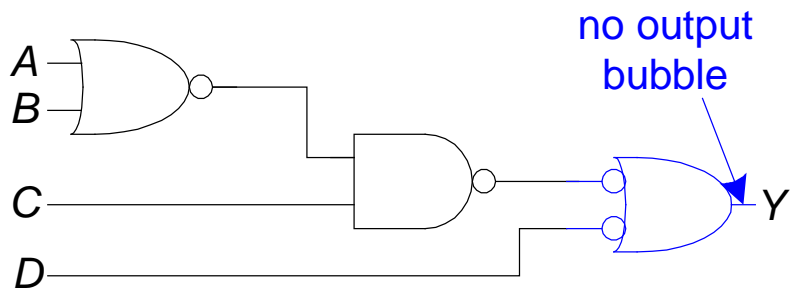
Bubble Pushing Example



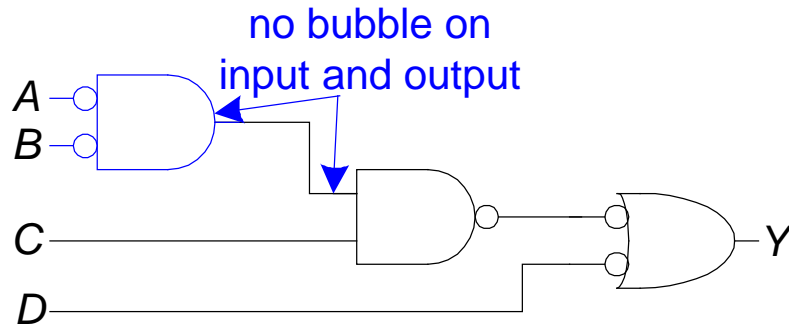
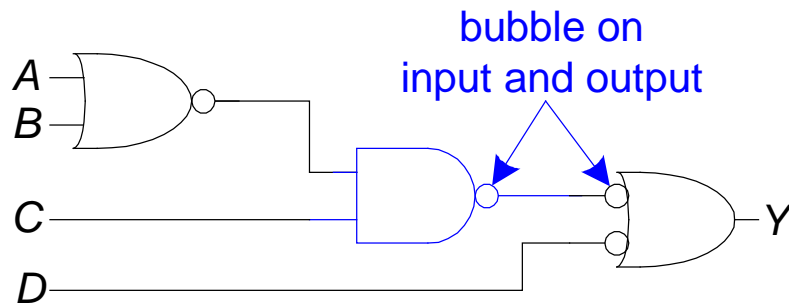
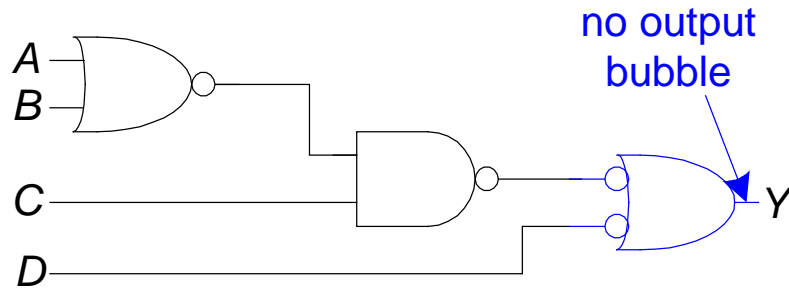
Bubble Pushing Example



Bubble Pushing Example



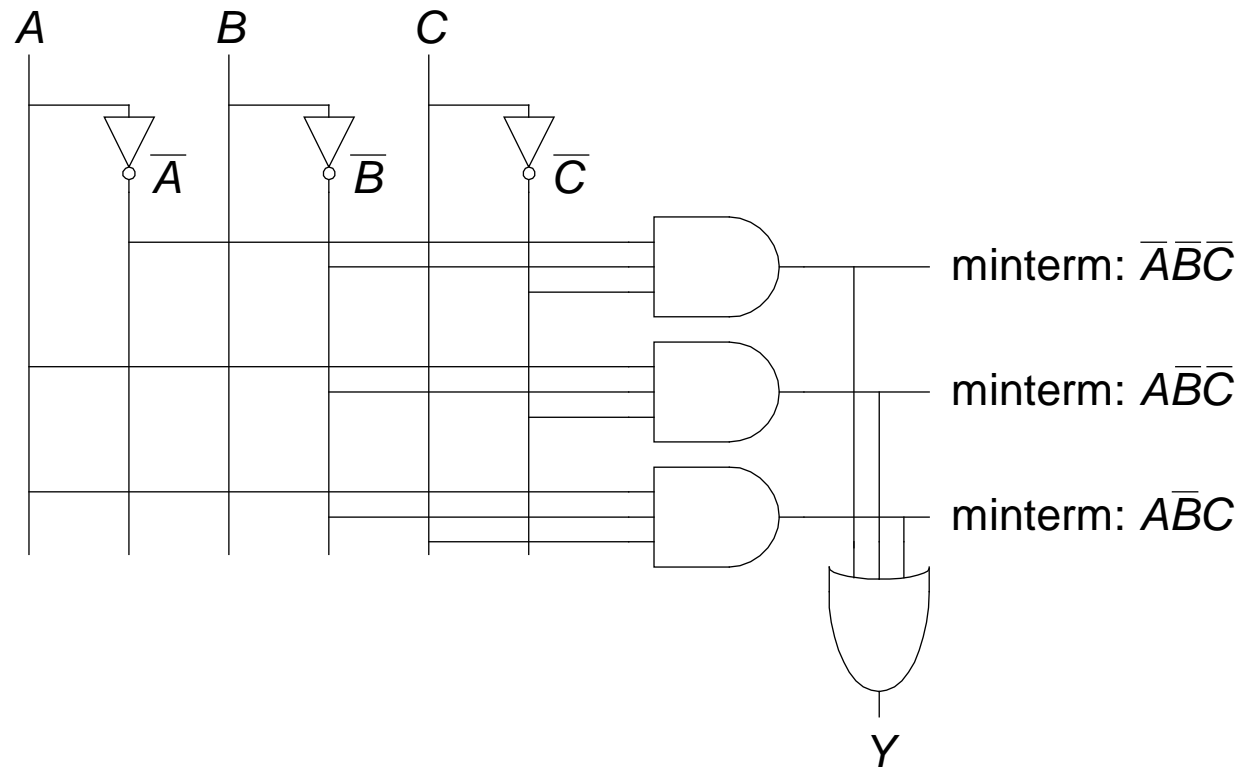
Bubble Pushing Example



$$Y = \overline{A}BC + \overline{D}$$

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 Schematics Rules

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

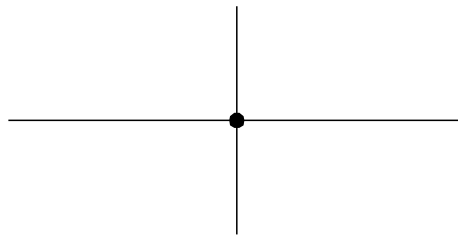
Circuit Schematic Rules (cont.)

- 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

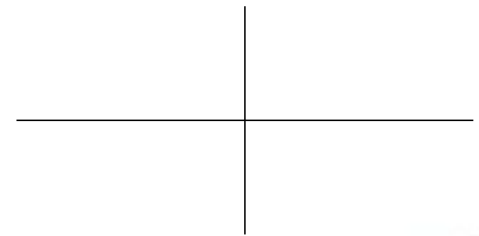
wires connect
at a T junction



wires connect
at a dot



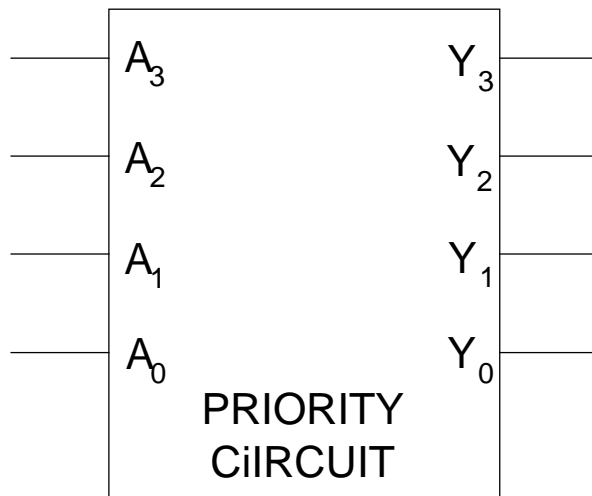
wires crossing
without a dot do
not connect



Multiple-Output Circuits

- Example: Priority Circuit**

Output asserted
corresponding to
most significant
TRUE input

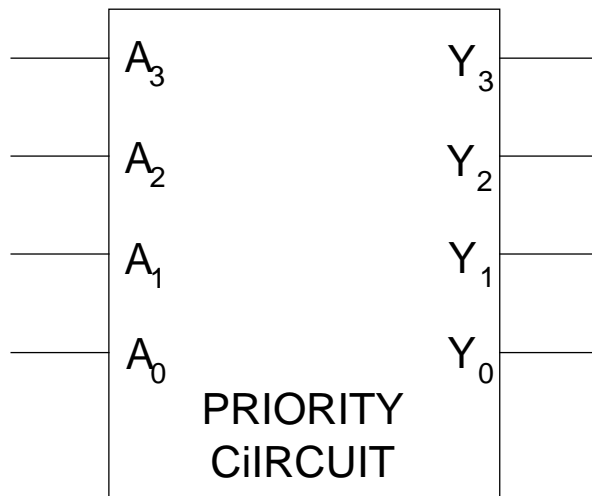


A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0				
0	0	0	1				1
0	0	1	0			1	
0	0	1	1			1	
0	1	0	0		1		
0	1	0	1		1		
0	1	1	0		1		
0	1	1	1		1		
1	0	0	0	1			
1	0	0	1	1			
1	0	1	0	1			
1	0	1	1	1			
1	1	0	0	1			
1	1	1	0	1			
1	1	1	1	0			
1	1	1	1	1			

Multiple-Output Circuits

- Example: Priority Circuit**

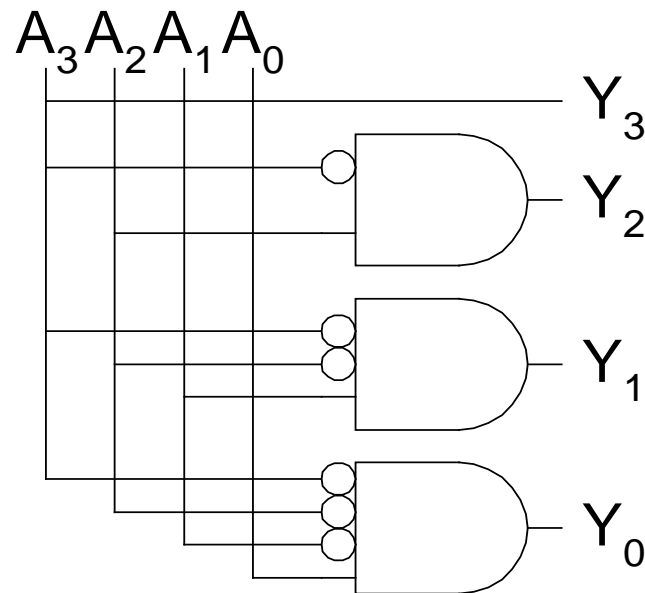
Output asserted
corresponding to
most significant
TRUE input



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
1	1	1	1	1	0	0	0

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

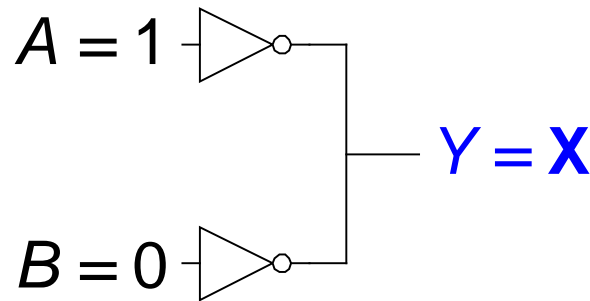
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

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 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

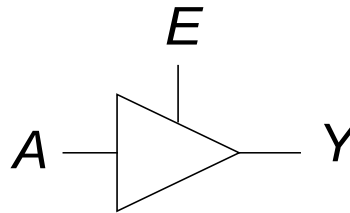


- **Warnings:**
 - Contention usually indicates a **bug**.
 - **X is used for “don’t care” and contention** - look at the context to tell them apart

Floating: Z

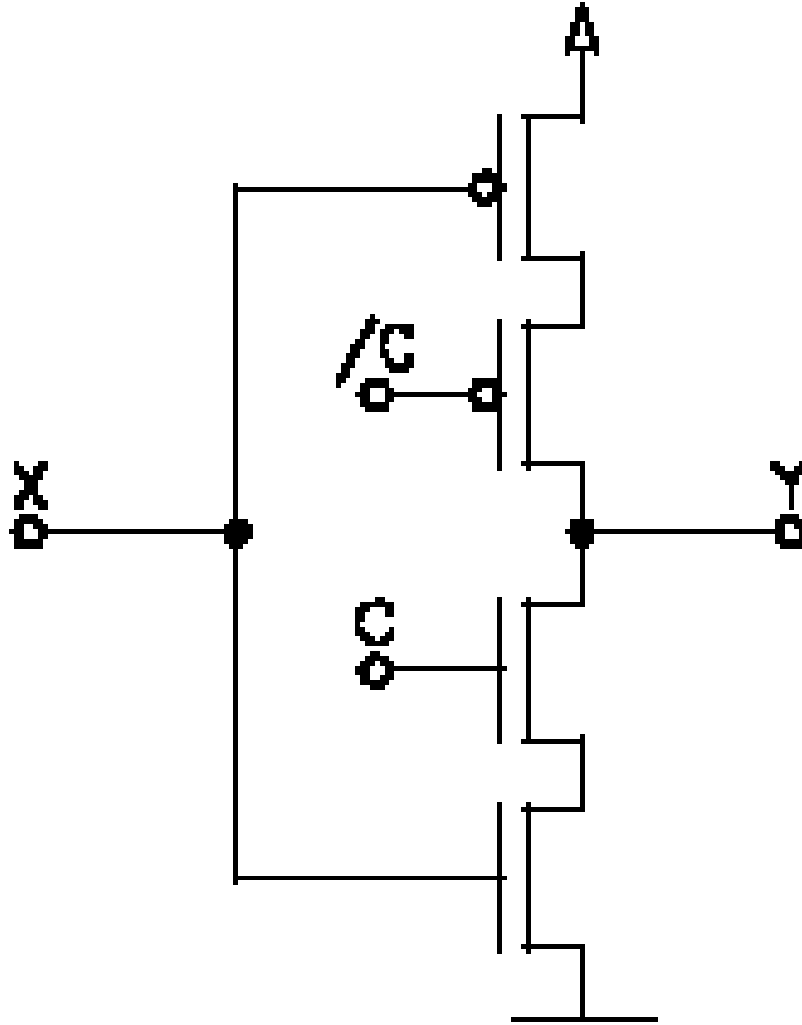
- Floating, high impedance, open, high Z
- Floating output might be 0, 1, or somewhere in between
 - A voltmeter won't indicate whether a node is floating

Tristate Buffer



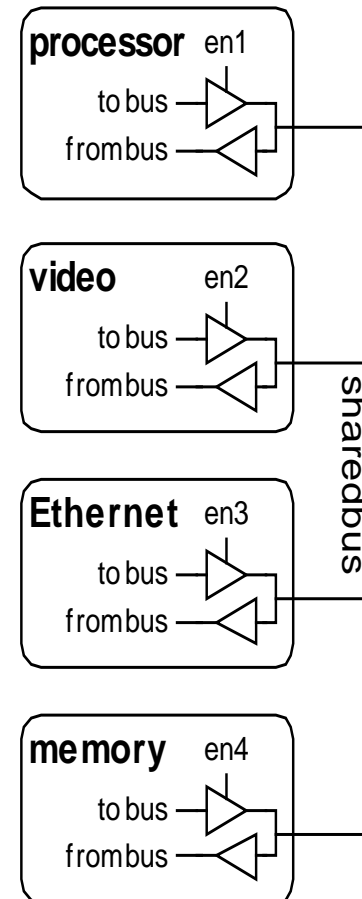
E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

How Floating: Z work ?



Tristate Busses

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



Karnaugh Maps (K-Maps)

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

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

		AB			
		00	01	11	10
C	0	1	0	0	0
	1	1	0	0	0

		AB			
		00	01	11	10
C	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 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

		AB			
		00	01	11	10
C	0	1	0	0	0
	1	1	0	0	0

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

3-Input K-Map

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

K-Map

		Y AB			
		00	01	11	10
C	0				
	1				

3-Input K-Map

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

K-Map

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

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

K-Map Definitions

- **Complement:** variable with a bar over it
 $\bar{A}, \bar{B}, \bar{C}$
- **Literal:** variable or its complement
 $\bar{A}, A, \bar{B}, B, C, \bar{C}$
- **Implicant:** product of literals
 $\bar{A}\bar{B}C, \bar{A}C, BC$
- **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		AB			
		00	01	11	10
CD	00				
	01				
	11				
	10				

4-Input K-Map

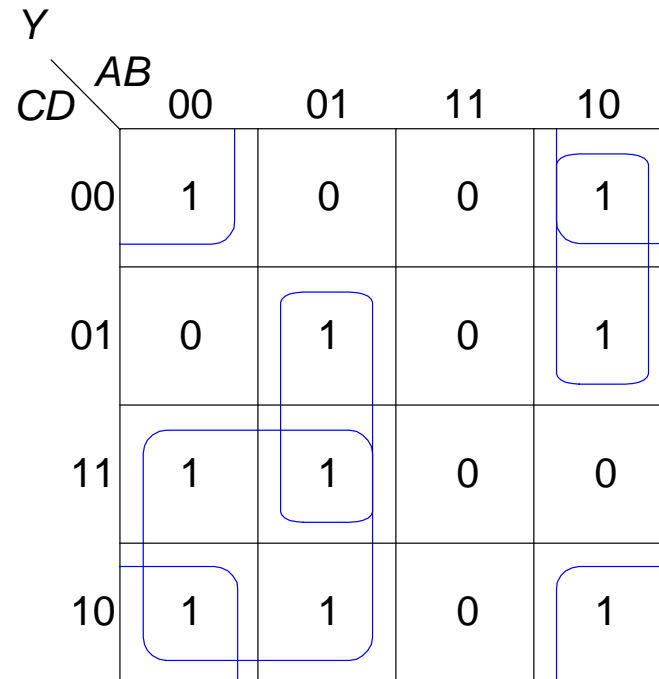
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

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



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

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		AB			
		00	01	11	10
CD	00				
	01				
	11				
	10				



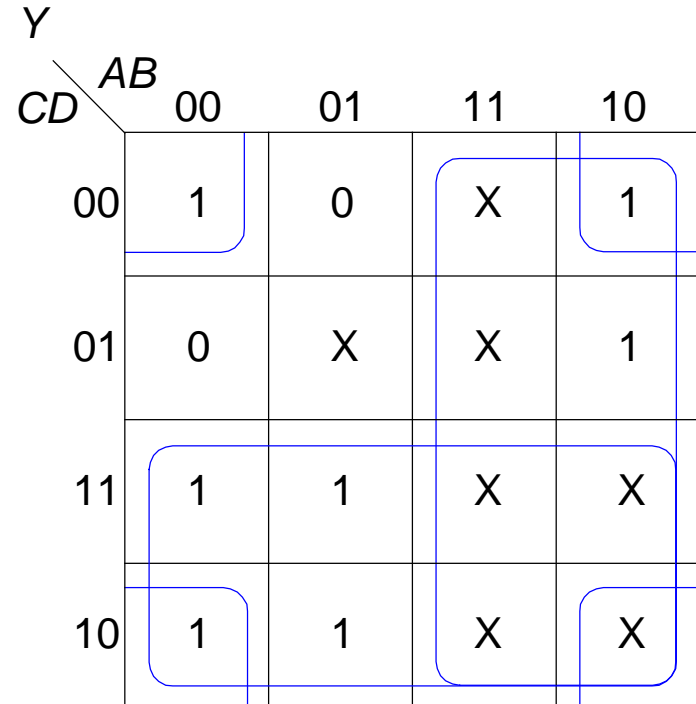
K-Maps with Don't Cares

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		AB			
		00	01	11	10
00	00	1	0	X	1
	01	0	X	X	1
11	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



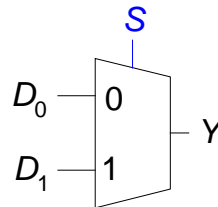
$$Y = A + \bar{B}\bar{D} + C$$

Combinational Building Blocks

- Multiplexers
- Decoders

Multiplexer (Mux)

- Selects between one of N inputs to connect to 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

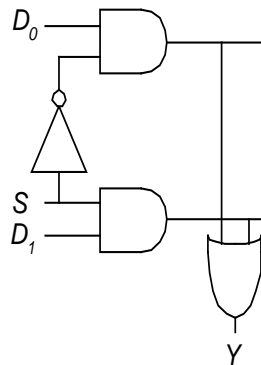
Multiplexer Implementations

- **Logic gates**

- Sum-of-products form

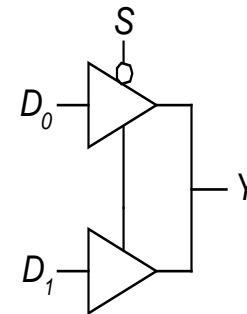
Y S	$D_0 D_1$			
	00	01	11	10
0	0	0	1	1
1	0	1	1	0

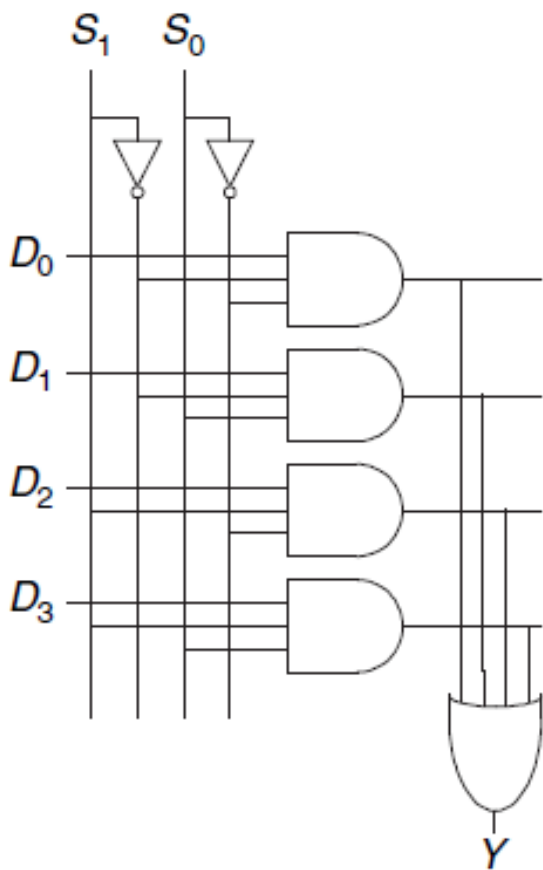
$$Y = D_0 \bar{S} + D_1 S$$



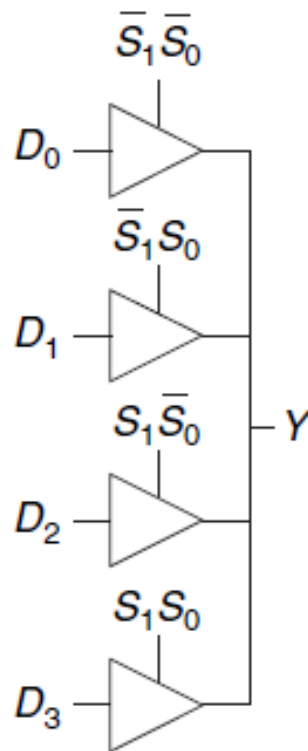
- **Tristates**

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

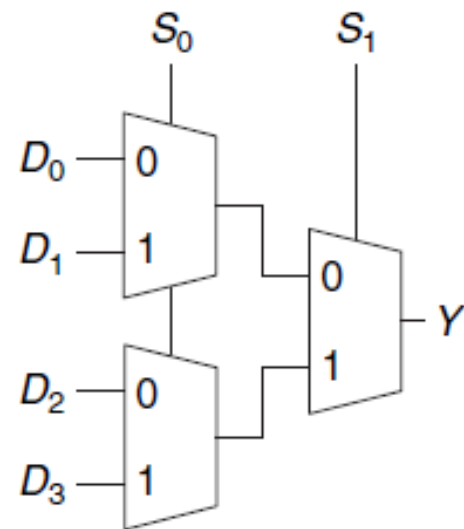




(a)



(b)



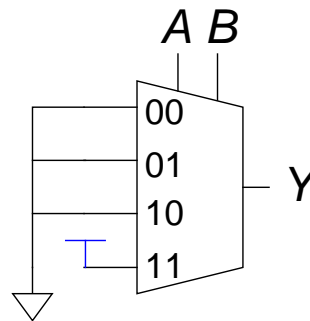
(c)

Logic using Multiplexers

- Using the mux as a lookup table

<i>A</i>	<i>B</i>	<i>Y</i>
0	0	0
0	1	0
1	0	0
1	1	1

$$Y = AB$$



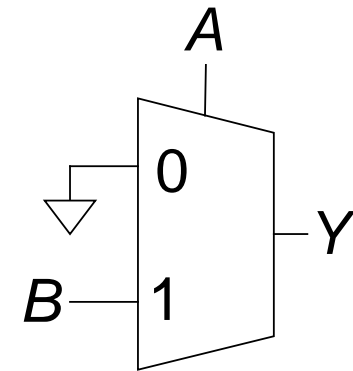
Logic using Multiplexers

- Reducing the size of the mux

$$Y = AB$$

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

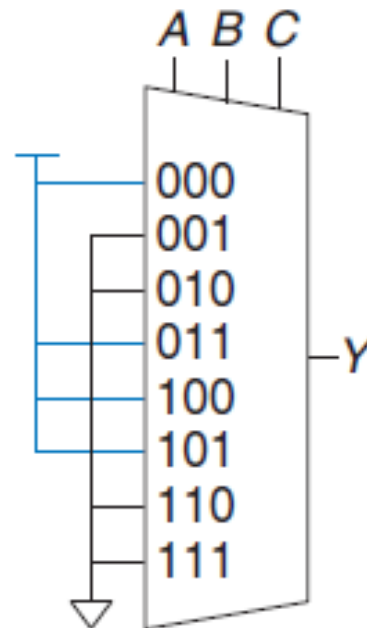
A	Y
0	0
1	B



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

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

(a)



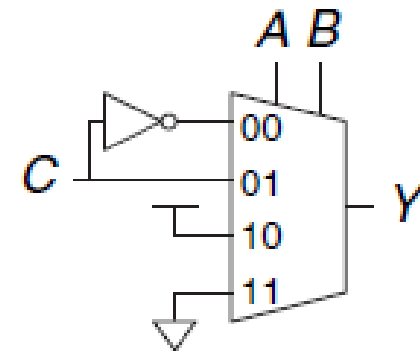
(b)

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

(a)

A	B	Y
0	0	\bar{C}
0	1	C
1	0	1
1	1	0

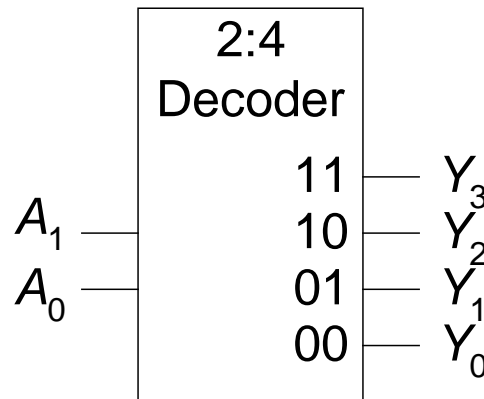
(b)



(c)

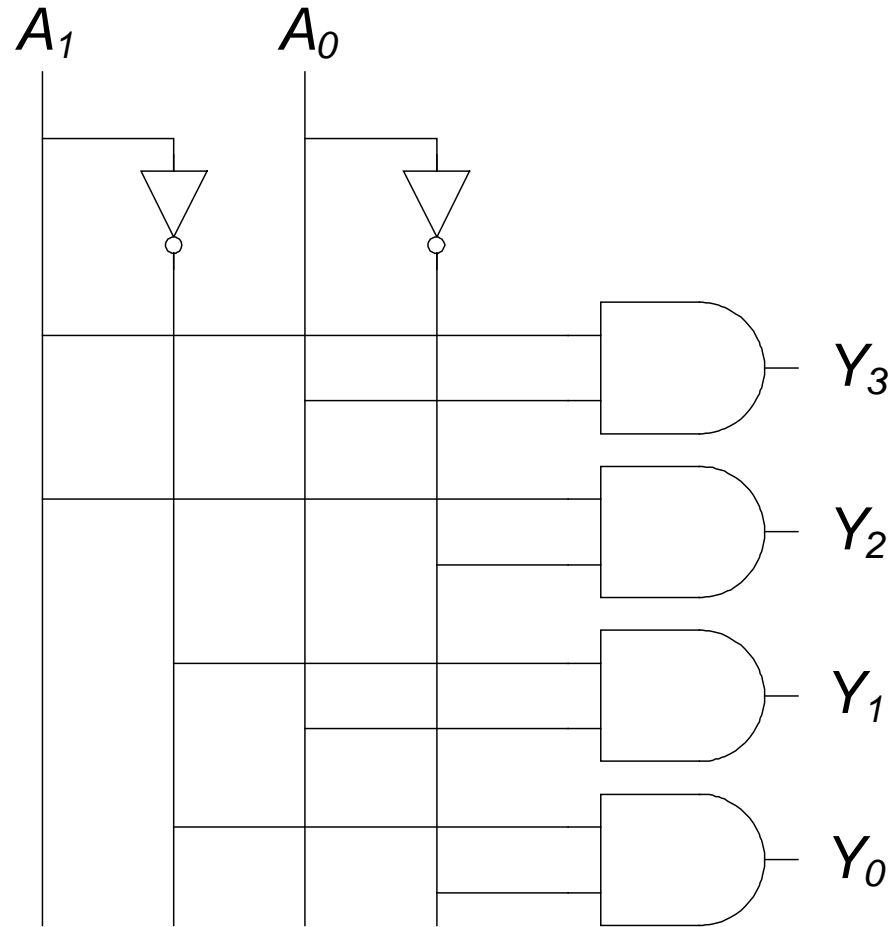
Decoders

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



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

