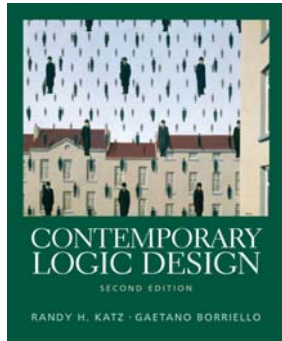


CSE 370 Spring 2006 Introduction to Digital Design

Lecture 4: Logic Gates



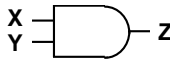



Last Lecture

- CMOS
- Basic Boolean Functions
- Boolean Algebra

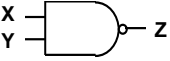

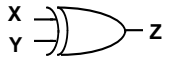
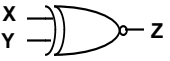
Today

- Logic Gates
- Different Implementations
- Bubbles

Logic Gates and Truth Tables

■ AND	$X \cdot Y$	XY		z	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th><th>Z</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	X	Y	Z	0	0	0	0	1	0	1	0	0	1	1	1
X	Y	Z																		
0	0	0																		
0	1	0																		
1	0	0																		
1	1	1																		
■ OR	$X + Y$			z	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th><th>Z</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	X	Y	Z	0	0	0	0	1	1	1	0	1	1	1	1
X	Y	Z																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	1																		
■ NOT	\bar{X}	X'		Y	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th></tr> </thead> <tbody> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </tbody> </table>	X	Y	0	1	1	0									
X	Y																			
0	1																			
1	0																			
■ Buffer	X			Y	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </tbody> </table>	X	Y	0	0	1	1									
X	Y																			
0	0																			
1	1																			

Logic Gates and Truth Tables

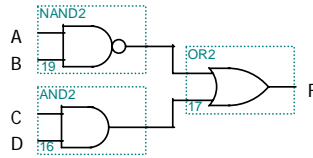
■ NAND	$\overline{X \cdot Y}$	\overline{XY}		z	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th><th>Z</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	X	Y	Z	0	0	1	0	1	1	1	0	1	1	1	0
X	Y	Z																		
0	0	1																		
0	1	1																		
1	0	1																		
1	1	0																		
■ NOR	$\overline{X + Y}$			z	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th><th>Z</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	X	Y	Z	0	0	1	0	1	0	1	0	0	1	1	0
X	Y	Z																		
0	0	1																		
0	1	0																		
1	0	0																		
1	1	0																		
■ XOR	$X \oplus Y$			z	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th><th>Z</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	X	Y	Z	0	0	0	0	1	1	1	0	1	1	1	0
X	Y	Z																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	0																		
■ XNOR	$\overline{X \oplus Y}$			z	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr><th>X</th><th>Y</th><th>Z</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	X	Y	Z	0	0	1	0	1	0	1	0	0	1	1	1
X	Y	Z																		
0	0	1																		
0	1	0																		
1	0	0																		
1	1	1																		

Useful Lingo

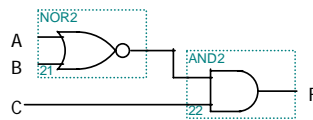
- Schematic: A drawing of interconnected logic gates
- Net: wires that are all at the same voltage
- Netlist: A list of all the devices and connections in a schematic
- Fan-in: The number of inputs to a gate
- Fan-out: The number of outputs of a gate

Boolean Functions to Gates

- Example: $F = (A \cdot B)' + C \cdot D$



- Example: $F = C \cdot (A + B)'$

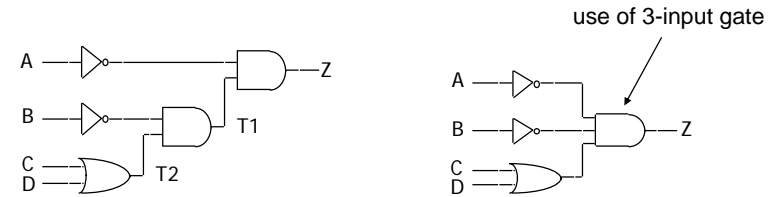


Boolean Functions to Gates

- More than one way to map expressions to gates

■ e.g., $Z = A' \cdot B' \cdot (C + D) = (A' \cdot (B' \cdot (C + D)))$

$$\begin{array}{c} \overline{\overline{C + D}} \\ \hline T2 \\ \hline T1 \end{array}$$



What is the Optimal Implementation?

- We use the axioms and theorems of Boolean algebra to “optimize” our designs
- Design goals vary
 - Reduce the number of inputs?
 - Reduce the number of gates?
 - Reduce number of gate levels?
- How do we explore the tradeoffs?
 - CAD tools
 - Logic minimization: Reduce number of gates and complexity
 - Logic optimization: Maximize speed and/or minimize power

Example: A Binary Full Adder

- 1-bit binary adder

■ Inputs: A, B, Carry-in

■ Outputs: Sum, Carry-out



A	B	Cin	S	Cout
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

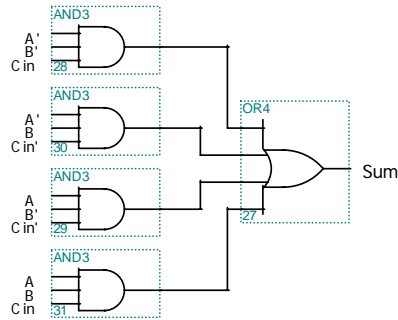
$$\text{Sum} = A'B'Cin + A'BCin' + AB'Cin' + ABCin$$

$$\text{Cout} = A'BCin + AB'Cin + ABCin' + ABCin$$

Full Adder: Sum

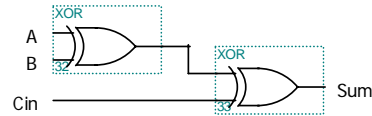
Before Boolean minimization

$$\text{Sum} = A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in}$$



After Boolean minimization

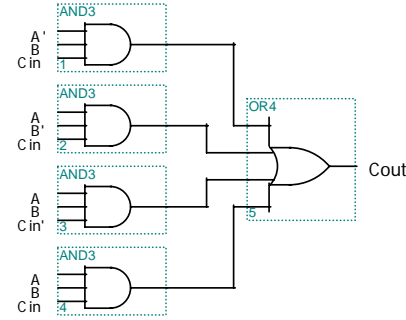
$$\text{Sum} = (A \oplus B) \oplus C_{in}$$



Full Adder: Carry

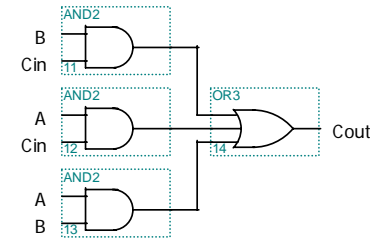
Before Boolean minimization

$$\text{Cout} = A'BC_{in} + AB'C_{in}' + ABC_{in}' + ABC_{in}$$

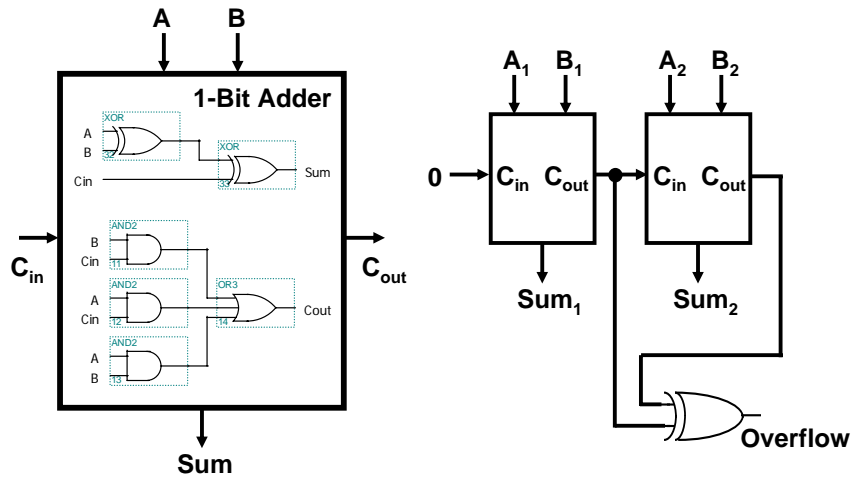


After Boolean minimization

$$\text{Cout} = BC_{in} + AC_{in} + AB$$



Preview: A 2-bit Ripple-Carry Adder

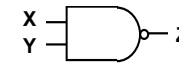


In Class Challenge

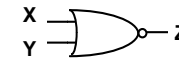
Implement XOR using NANDs and NORs (challenge: fewest)



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0



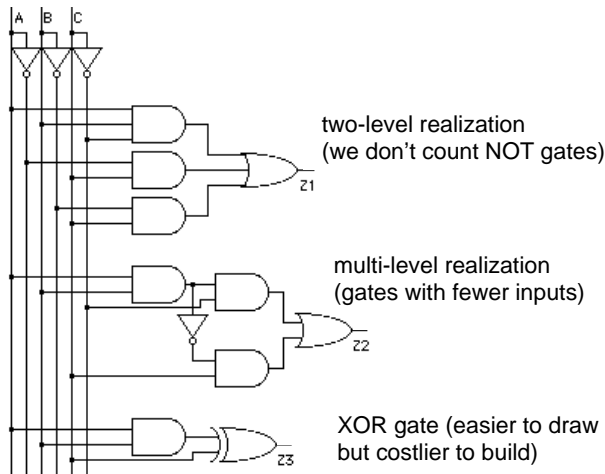
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

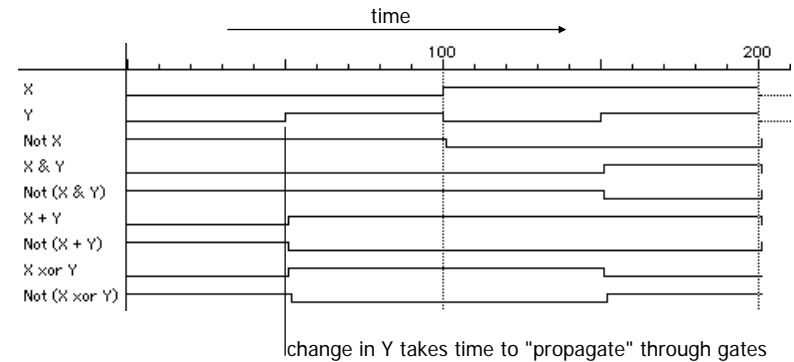
Different Realizations

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



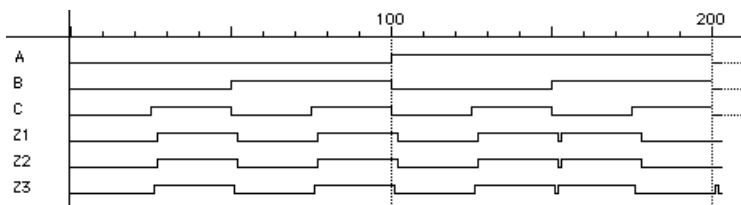
Waveform View of Logic Functions

- Just a sideways truth table
 - but note how edges don't line up exactly
 - it takes time for a gate to switch its output!



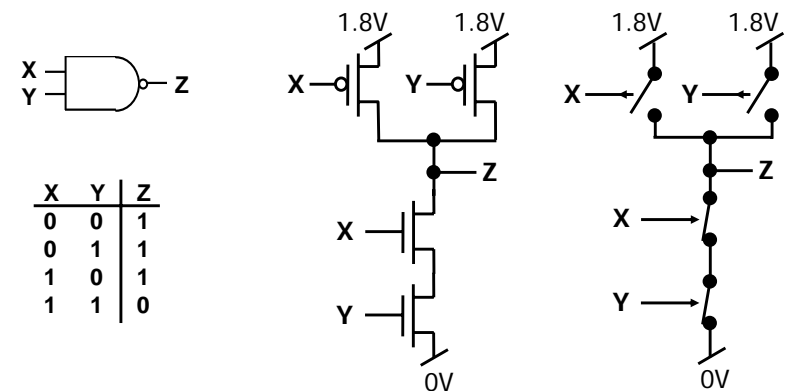
Are All Implementations Equivalent?

- Under the same input stimuli, the three alternative implementations have almost the same waveform behavior
 - delays are different
 - glitches (hazards) may arise – these could be bad, it depends
 - variations due to differences in number of gate levels and structure
- The three implementations are functionally equivalent



CMOS is Inverting

- CMOS logic gates are inverting
 - Get NAND, NOR, NOT
 - Don't get AND, OR, Buffer



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

DeMorgan's Theorem

■ Replace

- • with +, + with •, 0 with 1, and 1 with 0
- All variables with their complements

■ Example 1: $Z = A'B' + A'C'$

$$\begin{aligned} Z' &= (A'B' + A'C')' \\ &= (A'B')' \cdot (A'C')' \\ &= (A+B) \cdot (A+C) \end{aligned}$$

• Example 2: $Z = A'B'C + A'BC + AB'C + ABC'$

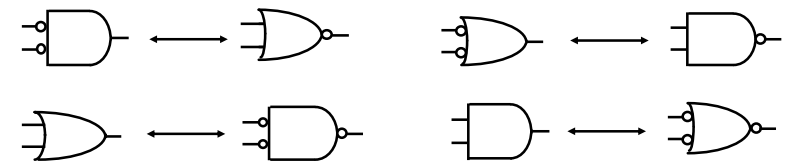
$$\begin{aligned} Z' &= (A'B'C + A'BC + AB'C + ABC')' \\ &= (A'B'C)' \cdot (A'BC)' \cdot (AB'C)' \cdot (ABC')' \\ &= (A+B+C)' \cdot (A+B'+C)' \cdot (A'+B+C)' \cdot (A'+B'+C) \end{aligned}$$

DeMorgan's, NAND, NOR

■ DeMorgan's Theorem

- Standard Form: $(A+B)' = A'B'$ $A'+B' = (AB)'$
- Inverted Form: $(A+B) = (A'B')'$ $(AB) = (A'+B')'$

- AND with complemented inputs \equiv NOR
- OR with complemented inputs \equiv NAND
- OR \equiv NAND with complemented inputs and outputs
- AND \equiv NOR with complemented inputs and outputs



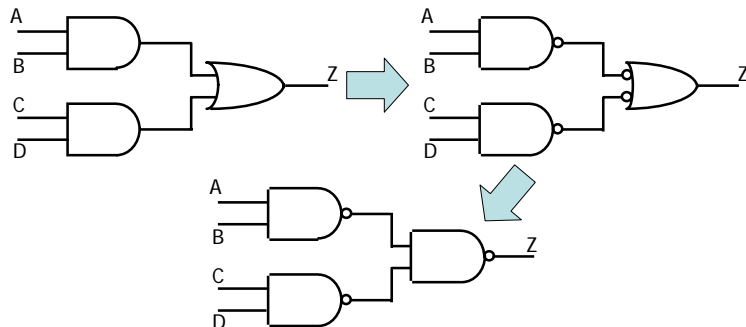
Double Bubble

■ Introduce inversions ("bubbles")

- Introduce bubbles in pairs
- Conserve inversions
- Do not alter logic function

■ Example

■ AND/OR to NAND/NAND

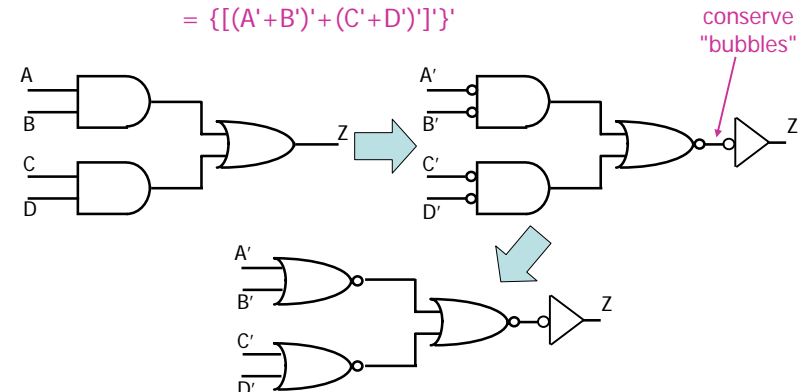


$$\begin{aligned} Z &= AB + CD \\ &= (A'+B')' + (C'+D')' \\ &= [(A'+B')(C'+D')]'' \\ &= [(AB)'(CD)']' \end{aligned}$$

Bubble Trouble Continued

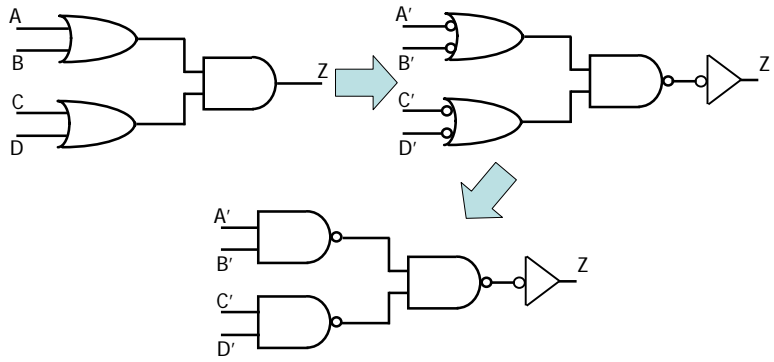
■ Example: AND/OR network to NOR/NOR

$$\begin{aligned} Z &= AB + CD \\ &= (A'+B')' + (C'+D')' \\ &= [(A'+B') + (C'+D')]'' \\ &= \{[(A'+B') + (C'+D')]\}' \end{aligned}$$



Bubble Trouble Continued

■ Example: OR/AND to NAND/NAND



Bubble Trouble Continued

■ Example: OR/AND to NOR/NOR

