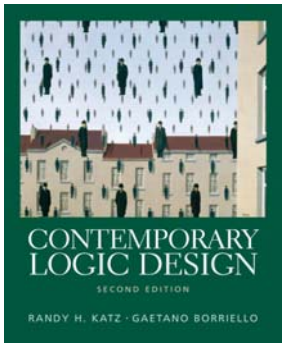
CSE 370 Spring 2006
Introduction to Digital Design

Lecture 4: Logic Gates



Last Lecture
- CMOS
- Basic Boolean Functions
- Boolean Algebra

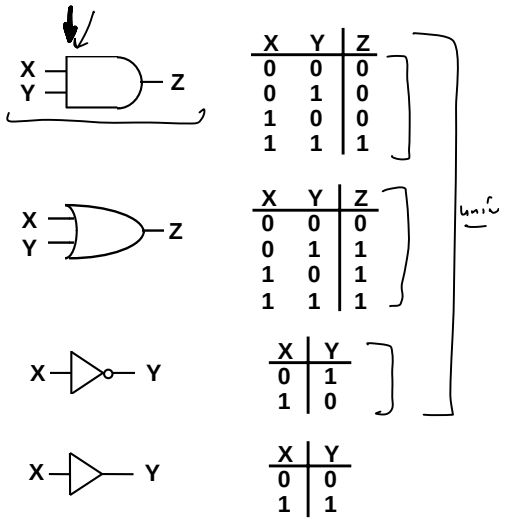
Today
- Logic Gates
- Different Implementations
- Bubbles

Administrivia

- Homework 2 on the web. Reading on the web (finish Chapter 2)
- Lab 2 is on the web, you might want to start the tutorial before you lab session this week.

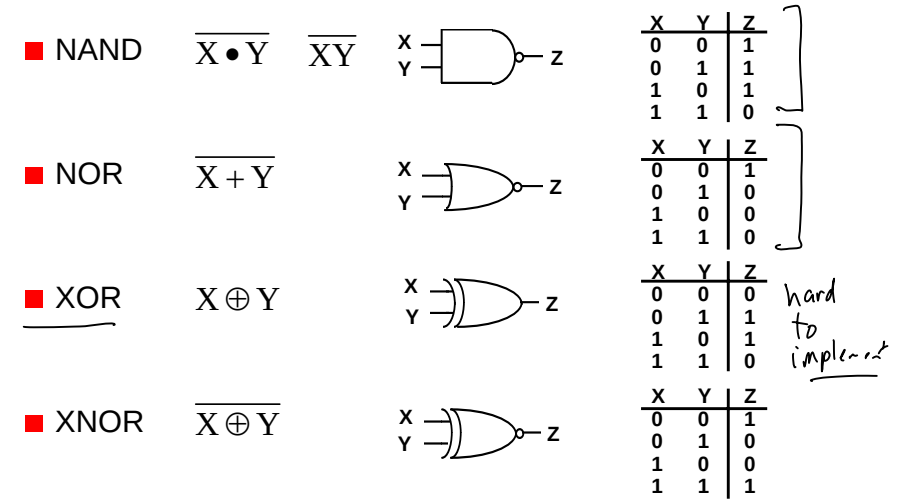
Logic Gates and Truth Tables

- AND X•Y XY
- OR X+Y
- NOT $\overline{X}$ X'
- Buffer X



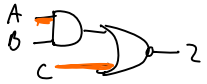
Logic Gates and Truth Tables

- NAND $\overline{X \bullet Y}$ $\overline{XY}$
- NOR $\overline{X + Y}$
- XOR $X \oplus Y$
- XNOR $\overline{X \oplus Y}$

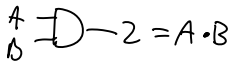


Useful Lingo

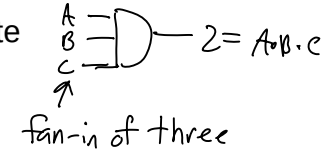
- Schematic: A drawing of interconnected logic gates
- Net: wires that are all at the same voltage

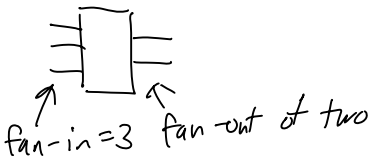


- Netlist: A list of all the devices and connections in a schematic
- Fan-in: The number of inputs to a gate



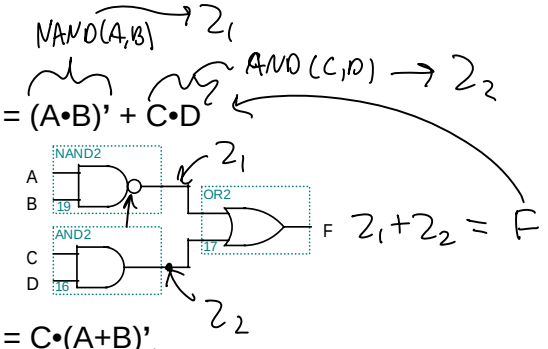
- Fan-out: The number of outputs of a gate



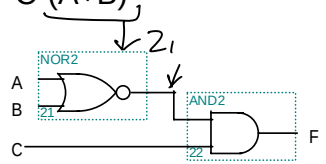


Boolean Functions to Gates

- Example: F = (A•B)' + C•D

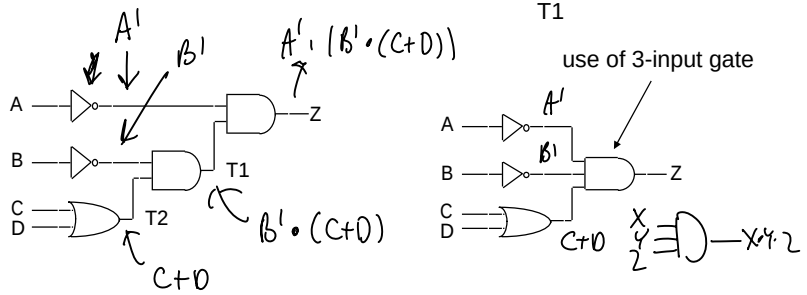


- Example: F = C•(A+B)'



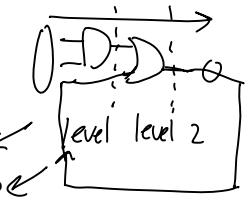
Boolean Functions to Gates

- More than one way to map expressions to gates
  - e.g., Z = A' • B' • (C + D) = (A' • (B' • (C + D)))



What is the Optimal Implementation?

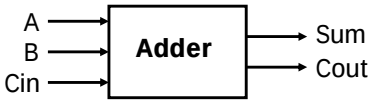
- We use the axioms and theorems of Boolean algebra to "optimize" our designs
- Design goals vary
  - Reduce the number of inputs?
  - Reduce the number of gates?
  - Reduce number of gate levels?



- 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

A' B' Cin

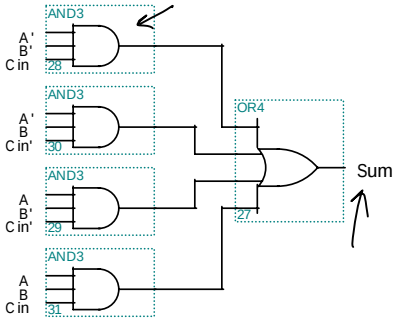
Sum = A'B'Cin + A'BCin' + AB'Cin' + ABCin

Cout = A'BCin + AB'Cin + ABCin' + ABCin

Full Adder: Sum

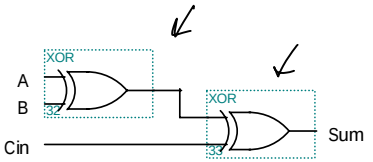
Before Boolean minimization

Sum = A'B'Cin + A'BCin'
+ AB'Cin' + ABCin



After Boolean minimization

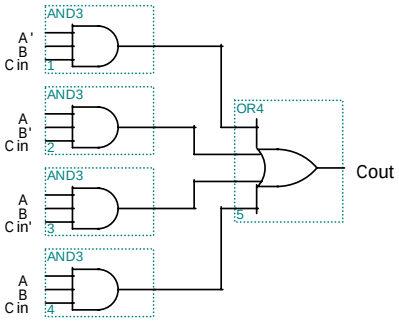
Sum = (A⊕B) ⊕ Cin



Full Adder: Carry

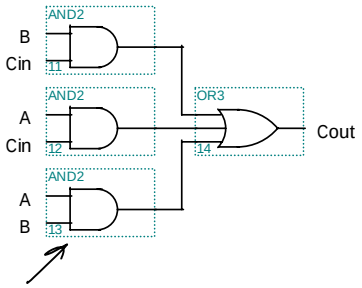
Before Boolean minimization

Cout = A'BCin + AB'Cin
+ ABCin' + ABCin

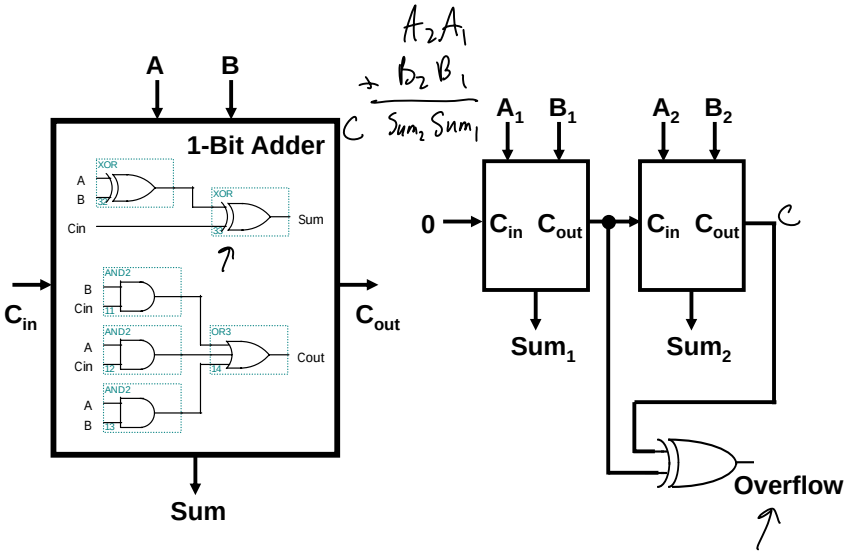


After Boolean minimization

Cout = BCin + ACin + AB

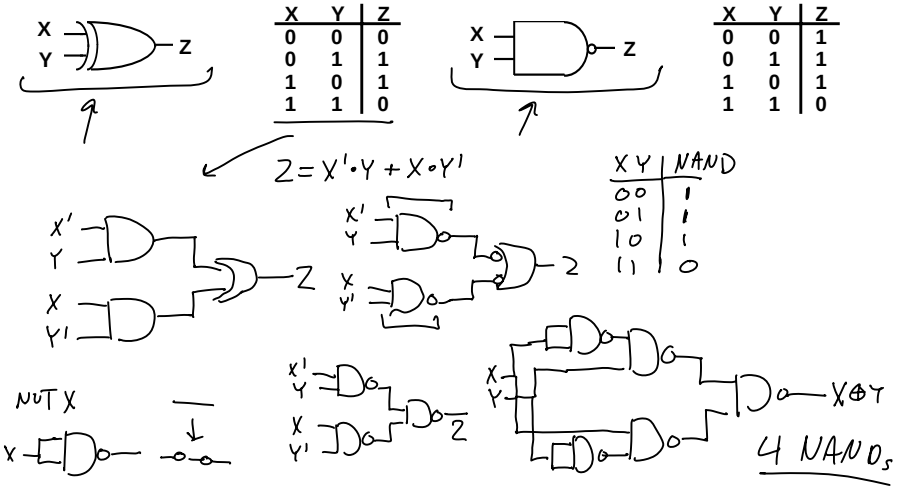


Preview: A 2-bit Ripple-Carry Adder



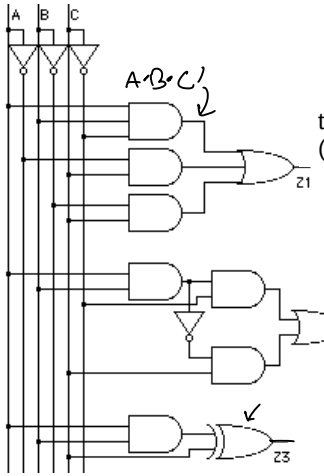
In Class Challenge

Implement XOR using NANDs (challenge: fewest)



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



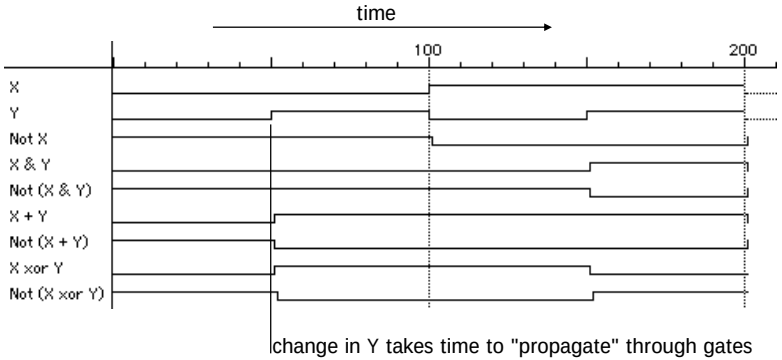
two-level realization
(we don't count NOT gates)

multi-level realization
(gates with fewer inputs)

XOR gate (easier to draw
but costlier to build)

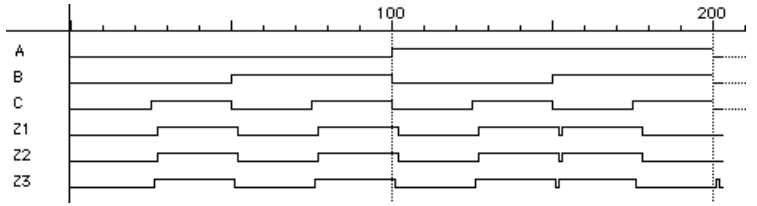
Waveform View of Logic Functions

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



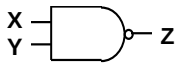
Are All Implementations Equivalent?

- 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

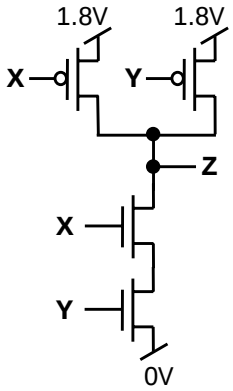


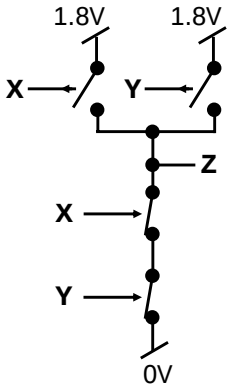
CMOS is Inverting

- 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

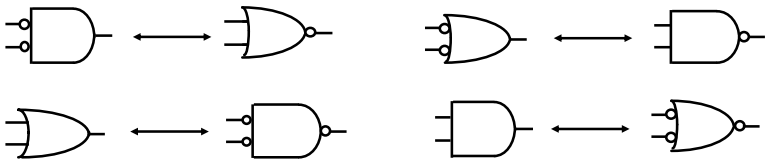
- Replace
  - • with +, + with •, 0 with 1, and 1 with 0
  - All variables with their complements
- Example 1: Z = A'B' + A'C'

  Z' = (A'B' + A'C')'
  = (A'B')' • (A'C')'
  = (A+B) • (A+C)
- Example 2: Z = A'B'C + A'BC + AB'C + ABC'

  Z' = (A'B'C + A'BC + AB'C + ABC')'
  = (A'B'C)' • (A'BC)' • (AB'C)' • (ABC')'
  = (A+B+C') • (A+B'+C') • (A'+B+C') • (A'+B'+C)

DeMorgan's, NAND, NOR

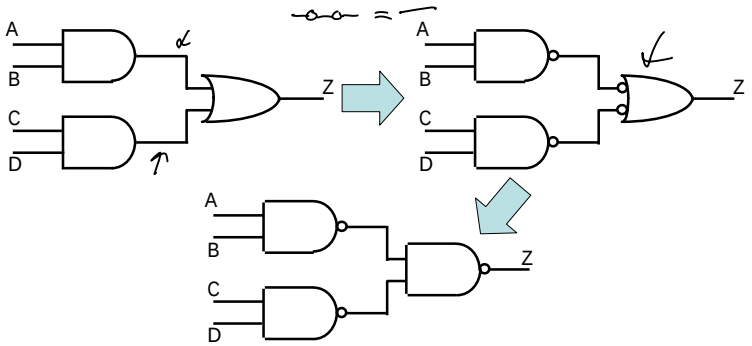
- DeMorgan's Theorem
  - Standard Form: (A+B)'=A'B' A'+B'=(AB)'
  - Inverted Form: (A+B)=(A'B')' (AB)=(A'+B')'
    - AND with complemented inputs ≡ NOR
    - OR with complemented inputs ≡ NAND
    - OR ≡ NAND with complemented inputs and outputs
    - AND ≡ NOR with complemented inputs and outputs



Double Bubble

- Introduce inversions ("bubbles")
  - Introduce bubbles in pairs
    - Conserve inversions
    - Do not alter logic function
- Example
  - AND/OR to NAND/NAND

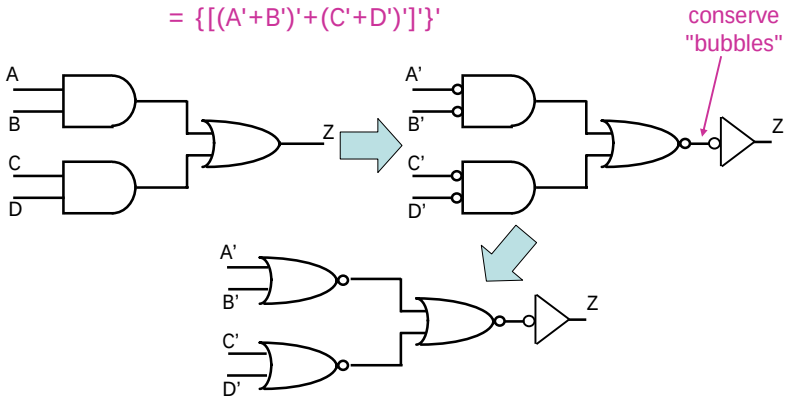
Z = AB + CD
= (A'+B')'+(C'+D')'
= [(A'+B')(C'+D')]'
= [(AB)'(CD)']'



Bubble Trouble Continued

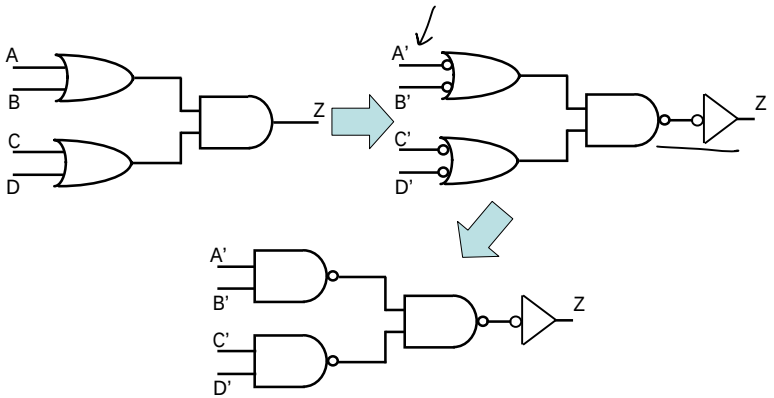
- Example: AND/OR network to NOR/NOR

Z = AB+CD
= (A'+B')'+(C'+D')'
= [(A'+B')'+(C'+D')']''
= {[(A'+B')'+(C'+D')']'}'



Bubble Trouble Continued

- Example: OR/AND to NAND/NAND



Bubble Trouble Continued

- Example: OR/AND to NOR/NOR