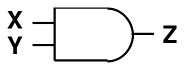
Logic gates

- Last lecture
  - Boolean algebra
    - Axioms
    - Useful laws and theorems
    - Simplifying Boolean expressions
- Today's lecture
  - Logic gates and truth tables
  - Implementing logic functions
  - CMOS switches

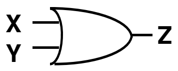
Logic gates and truth tables

- AND X•Y XY



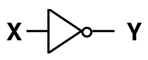
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

- OR X+Y



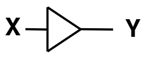
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

- NOT $\overline{X}$ X'



X	Y
0	1
1	0

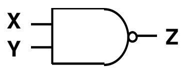
- Buffer X



X	Y
0	0
1	1

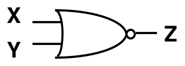
Logic gates and truth tables (con't)

- NAND $\overline{X \bullet Y}$ $\overline{XY}$



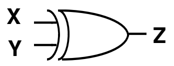
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

- NOR $\overline{X + Y}$



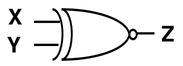
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

- XOR $X \oplus Y$



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

- XNOR $\overline{X \oplus Y}$



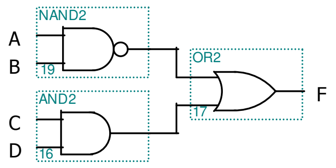
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

Definitions

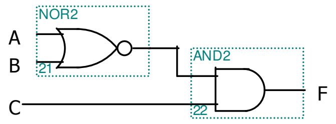
- Schematic: A drawing of interconnected gates
- Net: Wires at the same voltage (electrically connected)
- Netlist: A list of all the devices and connections in a schematic
- Fan-in: The # of inputs to a gate
- Fan-out: The # of loads the gate drives

Mapping Boolean expressions to logic gates

- Example: F = (A•B)′ + C•D

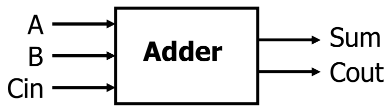


- Example: F = C•(A+B)′



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

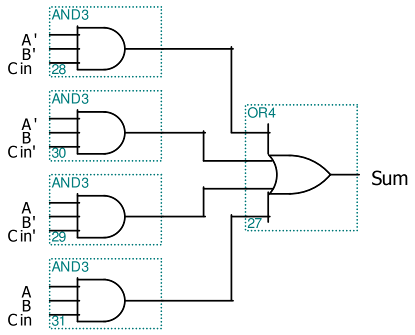
Sum = A'B'Cin + A'BCin' + AB'Cin' + ABCin

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

Full adder: Sum

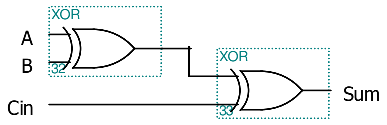
Before Boolean minimization

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



After Boolean minimization

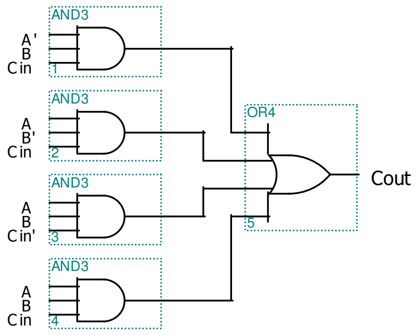
Sum = (A⊕B) ⊕ Cin



Full adder: Carry-out

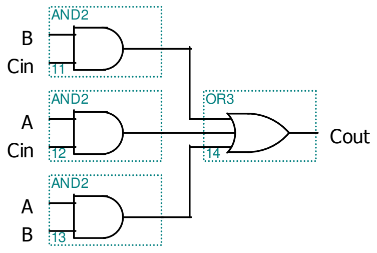
Before Boolean minimization

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

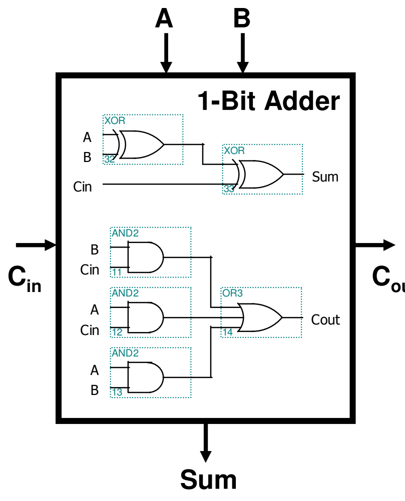


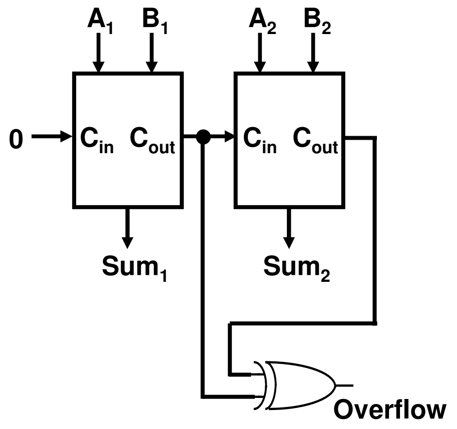
After Boolean minimization

Cout = BCin + ACin + AB



Preview: A 2-bit ripple-carry adder



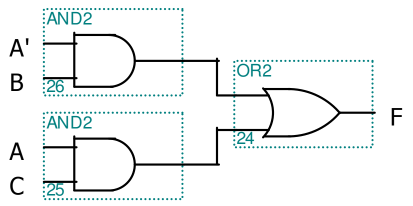


Mapping truth tables to logic gates

- Given a truth table
  - Write the Boolean expression
  - Minimize the Boolean expression
  - Draw as gates

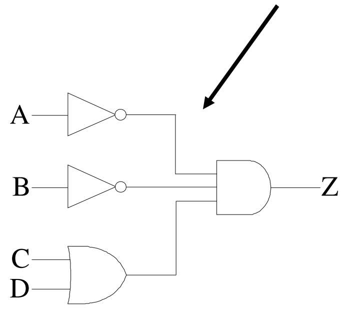
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

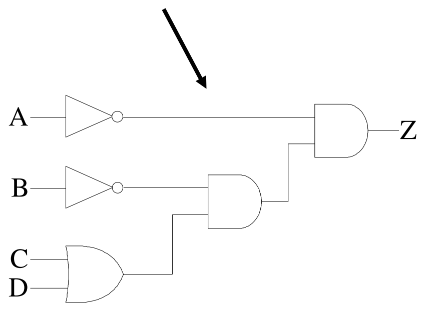
$$\begin{aligned} F &= A'BC'+A'BC+AB'C+ABC \\ &= A'B(C'+C)+AC(B'+B) \\ &= A'B+AC \end{aligned}$$



Many possible mappings

- Many ways to map expressions to gates
  - Example: $Z = A \bullet B \bullet (C + D) = A \bullet \; B \bullet (C + D)$





What is the optimal gate realization?

- 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

Minimal set

- We can implement any logic function from NOT, NOR, and NAND
  - Example: (X and Y) = not (X nand Y)
- In fact, we can do it with only NOR or only NAND
  - NOT is just NAND or NOR with two identical inputs

X	Y	X nor Y
0	0	1
1	1	0

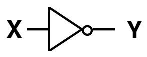
X	Y	X nand Y
0	0	1
1	1	0

  - NAND and NOR are duals: Can implement one from the other
    - X nand Y = not ((not X) nor (not Y))
    - X nor Y = not ((not X) nand (not Y))

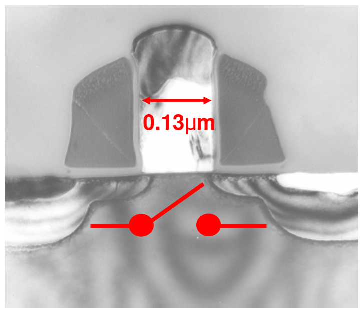
Most digital logic is CMOS

- CMOS technology
  - Complementary Metal-Oxide Semiconductor
  - Transistors act as voltage-controlled switches

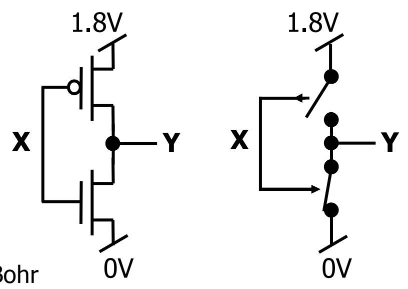
0V ≡ Logic 0
1.8V ≡ Logic 1



X	Y
0V	1.8V
1.8V	0V

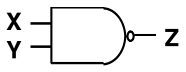


Mark Bohr
Intel

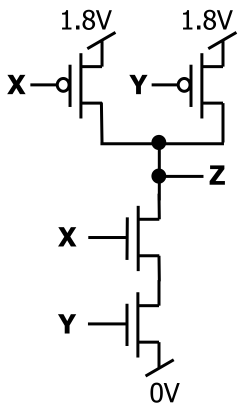


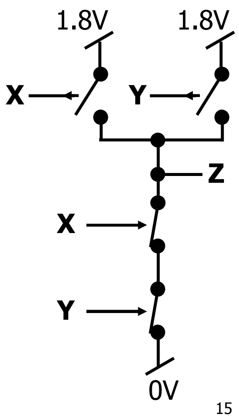
Multi-input logic gates

- 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


1.8V
1.8V
X
Y
Z
X
Y
0V


1.8V
1.8V
X
Y
Z
X
Y
0V