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 \( X \cdot Y \)
  - Truth table:
    - \( X \) \( Y \) \( Z \)
    - 0 0 0
    - 0 1 0
    - 1 0 0
    - 1 1 1

- OR \( X + Y \)
  - Truth table:
    - \( X \) \( Y \) \( Z \)
    - 0 0 0
    - 0 1 1
    - 1 0 1
    - 1 1 1

- NOT \( \overline{X} \)
  - Truth table:
    - \( X \) \( \overline{X} \)
    - 0 1
    - 1 0

- Buffer \( X \)
  - Truth table:
    - \( X \) \( Y \)
    - 0 1
    - 1 0

Logic gates and truth tables (con’t)

- NAND \( X \cdot \overline{Y} \)
  - Truth table:
    - \( X \) \( Y \) \( Z \)
    - 0 0 1
    - 0 1 1
    - 1 0 0
    - 1 1 0

- NOR \( X + \overline{Y} \)
  - Truth table:
    - \( X \) \( Y \) \( Z \)
    - 0 0 0
    - 0 1 0
    - 1 0 0
    - 1 1 0

- XOR \( X \oplus Y \)
  - Truth table:
    - \( X \) \( Y \) \( Z \)
    - 0 0 0
    - 0 1 1
    - 1 0 1
    - 1 1 0

- XNOR \( X \equiv Y \)
  - Truth table:
    - \( X \) \( Y \) \( Z \)
    - 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

Example: A binary full adder

- 1-bit binary adder
  - Inputs: A, B, Carry-in
  - Outputs: Sum, Carry-out
  - Truth table:
    - \( A \) \( B \) \( Cin \) \( S \) \( Cout \)
    - 0 0 0 0 0
    - 0 1 0 1 0
    - 0 1 0 1 1
    - 1 0 0 0 0
    - 1 0 0 0 1
    - 1 1 0 1 0
    - 1 1 0 1 1
    - 1 1 0 1 1
    - 1 1 0 1 1

  - Sum = \( A'B'Cin + AB'Cin' + AB'Cin + ABCin \)
  - Cout = \( A'B'Cin + AB'Cin' + ABCin + ABCin \)
Full adder: Sum

**Before Boolean minimization**

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

**After Boolean minimization**

\[ \text{Sum} = \left( A \oplus B \right) \oplus C_{\text{in}} \]

---

Full adder: Carry-out

**Before Boolean minimization**

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

**After Boolean minimization**

\[ \text{Cout} = BC_{\text{in}} + AC_{\text{in}} + AB \]

---

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

\[
\begin{array}{ccc|c}
A & B & C & F \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

- \[ F = A'B'C' + A'BC + AB'C + ABC \]
- \[ F = A'B(C' + C) + AC(B' + B) \]
- \[ F = A'B + AC \]

---

Many possible mappings

- Many ways to map expressions to gates
- Example: \[ Z = A \cdot B \cdot (C + D) = A \cdot B \cdot (C + D) \]

---

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: \((X \land Y) = \neg (X \text{ nand } Y)\)
- In fact, we can do it with only NOR or only NAND
  - NOT is just NAND or NOR with two identical inputs
- NAND and NOR are duals: Can implement one from the other
  - \(X \text{ nand } Y = \neg ((\neg X) \text{ nor } (\neg Y))\)
  - \(X \text{ nor } Y = \neg ((\neg X) \text{ nand } (\neg Y))\)

Most digital logic is CMOS

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

Multi-input logic gates

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