Outline

1. Quiz (ungraded)
2. Boolean algebra
3. Homework
What is the relationship between a truth table and a Boolean formula? Is it one/many-to-one/many?

What is the relationship between a Boolean formula and a combinational circuit? Is it one/many-to-one/many?

What quirks does the CMOS implementation technology have?

What is signal restoration or voltage regeneration?

What practices are unsafe in switch-based design? Why?
Outline

1. Quiz (ungraded)

2. Boolean algebra

3. Homework
Boolean algebra

- Set of elements, $B$
- Binary operators, $\{ [\text{AND, } \land, \ast, \cdot], [\text{OR, } \lor, +] \}$
  - We’ll prefer $\cdot$ and $+$
  - $\cdot$ frequently omitted
- Unary operator, $[\text{NOT, }', \overline{o}]$
Axioms of Boolean algebra

- **Closure**: \( \forall x, y \in B \), \( xy \in B \), \( x + y \in B \)
- **Commutative laws**: \( \forall x, y \in B \), \( xy = yx \), \( x + y = y + x \)
- **Identities**: \( 0, 1 \in B, \forall x \in B \), \( x1 = x \), \( x + 0 = x \)

\[ \exists x, y \in B \text{ s.t. } x \neq y \]
Axioms of Boolean algebra

∃x, y ∈ B s.t. x ≠ y

Distributive laws
∀x, y, z ∈ B
x + (yz) = (x + y)(x + z)
x(y + z) = xy + xz

Complement
x ∈ B
x ¯ = 0
x + ¯x = 1
De Morgan’s laws

\[(a + b) = \overline{a \cdot b}\]

\[ab = \overline{a} + \overline{b}\]

\[f(x_1, x_2, \ldots, x_n, \cdot, +) = f(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}, +, \cdot)\]

- Those \(x\)s could be functions
- Apply in stages
  - Top-down
De Morgan’s laws example

\[ a + bc \]
\[ \overline{a} \cdot (bc) \]
\[ \overline{a} \cdot (\overline{b} + \overline{c}) \]
Representations of Boolean functions

- Truth table
- Expression using only ·, +, and ’
- Symbolic
- Karnaugh map
  - More useful as visualization and optimization tool
Review: *AND*

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a·b</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>

\[ a \ AND \ b = a \cdot b \]

Will show Karnaugh map later.
**Review: OR**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a + b</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>

\[ a \ OR \ b = a + b \]
Review: \textit{NOT}

<table>
<thead>
<tr>
<th>(a)</th>
<th>(\overline{a})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\text{NOT } a = \overline{a}
\]
Different representations possible

\[ Z = ((C + D)\overline{B})\overline{A} \]

\[ Z = (C + D)\overline{A} \overline{B} \]
Simplifying logic functions

- Minimize literal count (related to gate count, delay)
- Minimize gate count
- Minimize levels (delay)
- Trade off delay for area
  - Sometimes no real cost
Proving theorems = simplification

Prove $XY + X\overline{Y} = X$

\[
XY + X\overline{Y} = X(Y + \overline{Y}) \quad \text{distributive law}
\]

\[
X(Y + \overline{Y}) = X(1) \quad \text{complementary law}
\]

\[
X(1) = X \quad \text{identity law}
\]
Proving theorems = simplification

Prove \( X + XY = X \)

\[
X + XY = X1 + XY & \quad \text{identity law} \\
X1 + XY = X(1 + Y) & \quad \text{distributive law} \\
X(1 + Y) = X1 & \quad \text{identity law} \\
X1 = X & \quad \text{identity law}
\]
Literals

- Each appearance of a variable (complement) in expression
- Fewer literals usually implies simpler to implement
- E.g.,  \( Z = \overline{A} \overline{B} C + \overline{A} B + \overline{A} B\overline{C} + \overline{B} C \)
  - Three variables, ten literals
NANDs and NORs

- Can be implemented in CMOS
  - More on this later
- $X \text{ NAND } Y = \overline{XY}$
- $X \text{ NOR } Y = \overline{X + Y}$
- Do we need inverters?
Summary

- Administrative details
- Finished CMOS and basic gates
- Boolean algebra
Outline

1. Quiz (ungraded)

2. Boolean algebra

3. Homework

Sections 2.4 and 2.5
Next lectures

- Karnaugh maps
- Visual minimization
- We’ll also learn the optimal Quine-McCluskey method
  - Optimal two-level minimization is fun!