What We Will Learn?

- Boolean Equations
  - Logic operations with binary numbers
- Logic Gates
  - Basic blocks that are interconnected to form larger units that are needed to construct a computer
Boolean Equations and Logic Gates
Simple Equations: NOT / AND / OR

\(\overline{A} \) (reads “not \( A \)”) is 1 iff \( A \) is 0

<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>

\( A \cdot B \) (reads “\( A \) and \( B \)”) is 1 iff \( A \) and \( B \) are both 1

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( A \cdot 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>

\( A + B \) (reads “\( A \) or \( B \)”) is 1 iff either \( A \) or \( B \) is 1

<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>
Boolean Algebra: Big Picture

- An algebra on 1’s and 0’s
  - with AND, OR, NOT operations

- What you start with
  - **Axioms:** basic stuff about objects and operations you just assume to be true at the start

- What you derive first
  - **Laws and theorems:** allow you to manipulate Boolean expressions
  - ...also allow us to do some simplification on Boolean expressions

- What you derive later
  - More “sophisticated” properties useful for manipulating digital designs represented in the form of Boolean equations
## Common Logic Gates

<table>
<thead>
<tr>
<th>Buffer</th>
<th>AND</th>
<th>OR</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Buffer Diagram" /></td>
<td><img src="image" alt="AND Diagram" /></td>
<td><img src="image" alt="OR Diagram" /></td>
<td><img src="image" alt="XOR Diagram" /></td>
</tr>
<tr>
<td><strong>A</strong></td>
<td><strong>B</strong></td>
<td><strong>Z</strong></td>
<td><strong>A</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inverter</th>
<th>NAND</th>
<th>NOR</th>
<th>XNOR</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Inverter Diagram" /></td>
<td><img src="image" alt="NAND Diagram" /></td>
<td><img src="image" alt="NOR Diagram" /></td>
<td><img src="image" alt="XNOR Diagram" /></td>
</tr>
<tr>
<td><strong>A</strong></td>
<td><strong>Z</strong></td>
<td><strong>A</strong></td>
<td><strong>B</strong></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Boolean Algebra: Axioms

<table>
<thead>
<tr>
<th>Formal version</th>
<th>English version</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong> $B$ contains at least two elements, $0$ and $1$, such that $0 \neq 1$</td>
<td>Math formality...</td>
</tr>
</tbody>
</table>
| **2.** Closure $a, b \in B$,  
  (i) $a + b \in B$  
  (ii) $a \cdot b \in B$ | Result of AND, OR stays in set you start with |
| **3.** Commutative Laws: $a, b \in B$,  
  (i)  
  (ii) | For primitive AND, OR of 2 inputs, order doesn’t matter |
| **4.** Identities: $0, 1 \in B$  
  (i)  
  (ii) | There are identity elements for AND, OR, give you back what you started with |
| **5.** Distributive Laws:  
  (i)  
  (ii) | • distributes over $+$, just like algebra  
  …but $+$ distributes over $\cdot$, also (!!) |
| **6.** Complement:  
  (i)  
  (ii) | There is a complement element, ANDing, ORing give you an identity |
Boolean Algebra: Duality

- Interesting observation
  - All the axioms come in “dual” form
  - Anything true for an expression also true for its dual
  - So any derivation you could make that is true, can be flipped into dual form, and it stays true

- Duality -- More formally
  - A dual of a Boolean expression is derived by replacing
    - Every AND operation with... an OR operation
    - Every OR operation with... an AND
    - Every constant 1 with... a constant 0
    - Every constant 0 with... a constant 1
    - But don’t change any of the literals or play with the complements!

Example

\[
\begin{align*}
a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\
\Rightarrow a + (b \cdot c) &= (a + b) \cdot (a + c)
\end{align*}
\]
# Boolean Algebra: Useful Laws

**Operations with 0 and 1:**

1. \( X + 0 = X \)  
2. \( X + 1 = 1 \)

**Idempotent Law:**

1D. \( X \cdot 1 = X \)

**Involution Law:**

4. \( (\bar{X}) = X \)  

**Laws of Complementarity:**

5. \( X + \bar{X} = 1 \)
5D. \( X \cdot \bar{X} = 0 \)

**Commutative Law:**

6. \( X + Y = Y + X \)
6D. \( X \cdot Y = Y \cdot X \)

**Dual: AND, OR with identities:**

- gives you back the original variable or the identity

**Dual: AND, OR with self = self:**

- double complement = no complement

**Dual: AND, OR with complement:**

- gives you an identity

**Just an axiom…**
Useful Laws (cont)

**Associative Laws:**
7. \((X + Y) + Z = X + (Y + Z)\)  
   \[= X + Y + Z\]
7D. \((X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)\)  
   \[= X \cdot Y \cdot Z\]
   Parenthesis order doesn’t matter

**Distributive Laws:**
8. \(X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)\)
8D. \(X + (Y \cdot Z) = (X + Y) \cdot (X + Z)\)
   Axiom

**Simplification Theorems:**
9.  
9D.  
   Useful for simplifying expressions
10.  
10D.  
11.  
11D.  

Actually worth remembering — they show up a lot in real designs…
DeMorgan’s Law

*DeMorgan's Law:*

12. \( \overline{(X + Y + Z + \cdots)} = \overline{X} \cdot \overline{Y} \cdot \overline{Z} \cdot \ldots \)

12D. \( \overline{(X \cdot Y \cdot Z \cdot \ldots)} = \overline{X} + \overline{Y} + \overline{Z} + \ldots \)

- Think of this as a transformation
  - Let’s say we have:

\[
F = A + B + C
\]

- Applying DeMorgan’s Law (12), gives us:

\[
F = \overline{(A + B + C)} = \overline{A \cdot B \cdot C}
\]
DeMorgan’s Law (cont.)

Interesting — these are conversions between different types of logic
That’s useful given you don’t always have every type of gate

<table>
<thead>
<tr>
<th>( A = \overline{(X + Y)} = ZX \overline{Y} )</th>
<th>X</th>
<th>Y</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \overline{X} \overline{Y} )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( \overline{X} \overline{Y} )</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \overline{X} \overline{Y} )</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \overline{X} \overline{Y} )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

NOR is equivalent to AND with inputs complemented

<table>
<thead>
<tr>
<th>( B = \overline{(XY)} = \overline{X} + \overline{Y} )</th>
<th>X</th>
<th>Y</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \overline{X} + \overline{Y} )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( \overline{X} + \overline{Y} )</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( \overline{X} + \overline{Y} )</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( \overline{X} + \overline{Y} )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

NAND is equivalent to OR with inputs complemented