Digital Logic

Transistors
Logic Gates
Boolean Algebra
Deriving circuits – Direct translation, Sum of Products
Simplifying circuits – Boolean Algebra, K-maps
What’s a Boolean?

Boolean value (**bit**): 0 or 1

Boolean functions (AND, OR, NOT, …)

Electronically:
- bit = high voltage vs. low voltage

![Graph showing voltage levels](image)

- Boolean functions = logic gates, built from transistors
Transistors (recap of lab)

If *Base* voltage is high:
Current may flow freely from *Collector* to *Emitter*.

If *Base* voltage is low:
Current may not flow from *Collector* to *Emitter*.

<table>
<thead>
<tr>
<th>$V_{in}$</th>
<th>$V_{out}$</th>
<th>in</th>
<th>out</th>
<th>$V_{in}$</th>
<th>$V_{out}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>high</td>
<td>0</td>
<td>1</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>1</td>
<td>0</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Digital Logic Gates

Tiny electronic devices that compute basic Boolean functions.

**NOT**

<table>
<thead>
<tr>
<th>$V_{in}$</th>
<th>$V_{out}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**NAND**

<table>
<thead>
<tr>
<th>$V_{1}$</th>
<th>$V_{2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
</tbody>
</table>

$V_{in}$

$V_{out}$

$+V_{cc}$

$V_{out}$

$V_{1}$

$V_{2}$

$+V_{cc}$
Five basic gates: define with truth tables

<table>
<thead>
<tr>
<th>NOT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAND</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOR</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Boolean Algebra

inputs = variables
wires = expressions
gates = operators/functions
circuits = functions

AND = Boolean product

\[
\begin{array}{c|cc}
\cdot & 0 & 1 \\
\hline
0 & 0 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

OR = Boolean sum

\[
\begin{array}{c|cc}
+ & 0 & 1 \\
\hline
0 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

NOT = inverse or complement

\[
\begin{array}{c|c}
0 & 1 \\
1 & 0 \\
\end{array}
\]

wire = identity

\[
\begin{array}{c|c}
0 & 0 \\
1 & 1 \\
\end{array}
\]
Connect inputs and outputs of gates with wires. Crossed wires touch *only if* there is a dot.

What is the output if $A=1$, $B=0$, $C=1$?
What is the truth table of this circuit?
What is an equivalent Boolean expression?
Direct Translation

Connect gates to implement these functions. Check with truth tables.

Use a direct translation -- it is straightforward and bidirectional.

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

\[ Z = \overline{W} + (X + \overline{WY}) \]
Note on notation: bubble = inverse/complement

Identity law, inverse law
Commutativity, Associativity

A + B = B + A

A(BC) = A((BC))
Idempotent law, Null/Zero law

\[ A + A = A \]

\[ A \quad 0A \quad = \quad A \quad 0 \]
Note on notation: bubble = inverse/complement

DeMorgan's Law
One law, Absorption law

Write truth tables. Do they correspond to simpler circuits?

\[ A + 1 = \]

\[ 1 \]

\[ A \]

\[ A + 1 \]

\[ = \]

\[ A + AB \]

\[ = \]

\[ AB \]
NAND is *universal*. 

All Boolean functions can be implemented using only NANDs. Build NOT, AND, OR, NOR, using only NAND gates.
XOR: Exclusive OR

Output = 1 if exactly one input = 1.

Truth table: Build from earlier gates:

Often used as a one-bit comparator.
Larger gates

- Build a 4-input AND gate using any number of 2-input gates.
Circuit derivation and simplification

Is there a simpler circuit that performs the same function?

Start with an equivalent Boolean expression, then simplify with algebra.

\[ F(A, B, C) = \]

Check the answer with a truth table.
Circuit derivation: *sum-of-products* form

- logical sum (OR)
- of products (AND)
- of inputs or their complements (NOT)

Draw the truth table and design a sum-of-products circuit for a 4-input code detector to accept two codes (ABCD=1001, ABCD=1111) and reject all others.

How are the truth table and the sum-of-products circuit related?
Circuit simplification – K-Maps

- Simplify circuits using Boolean Algebra… or Karnaugh Maps
- Gray Codes = reflected binary codes
  - Alternate binary encoding
  - designed for electromechanical switches and counting

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>000</td>
<td>001</td>
<td>011</td>
<td>010</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>110</td>
<td>111</td>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

How many bits change when incrementing?
1. Cover exactly the 1s by drawing a (minimum) number of maximally sized rectangles whose dimensions (in cells) are powers of 2. (They may overlap or wrap around!)

2. For each rectangle, make a product of the inputs (or complements) that are 1 for all cells in the rectangle. (minterms)

3. Take the sum of these products.
1. Cover exactly the 1s by drawing a (minimum) number of maximally sized rectangles whose dimensions (in cells) are powers of 2. (They may overlap or wrap around!)

2. For each rectangle, make a product of the inputs (or complements) that are 1 for all cells in the rectangle. (*minterms*)

3. Take the sum of these products.