Digital Logic

*Gateway to computer science*

Boolean value (*bit*): 0 or 1
- basis of all digital data representations

Boolean functions (AND, OR, NOT, ...):
- basis of all digital computations

**Electronically:**
- bit = high voltage vs. low voltage

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3V</td>
<td>0</td>
</tr>
<tr>
<td>2.8V</td>
<td>1</td>
</tr>
<tr>
<td>0.5V</td>
<td></td>
</tr>
<tr>
<td>0.0V</td>
<td></td>
</tr>
</tbody>
</table>

Boolean functions = logic gates, built from transistors

**Transistors** (more in 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>Truth Table</th>
<th>NOT gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{IN}$</td>
<td>$V_{OUT}$</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>high</td>
<td>low</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

\[ V_{out} = \overline{V_1} \cdot \overline{V_2} \]

### NOR

\[ V_{out} = \overline{V_1} + \overline{V_2} \]

### AND

\[ V_{out} = V_1 \cdot V_2 \]

### OR

\[ V_{out} = V_1 + V_2 \]

Five basic gates: define with truth tables

Boolean Algebra

for combinational logic

<table>
<thead>
<tr>
<th>(A)</th>
<th>(B)</th>
<th>(A \cdot B)</th>
<th>(A + B)</th>
</tr>
</thead>
<tbody>
<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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Integrated Circuits (invented 1950s)

Gates are manufactured in units called integrated circuits. From SSI (tens) to VLSI (hundreds of thousands to billions)
Circuits

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?

Translation

Connect gates to implement these functions. Check with truth tables.
A one-to-one translation is straightforward and bidirectional.

\[
F = (AB + C)D
\]

\[
Z = W + (X + WY)
\]

Identity law, inverse law

Note on notation: bubble = inverse/complement

Commutativity, Associativity
Idempotent law, Null/Zero law

\[
A + A = A \\
0 + 0 = 0
\]

DeMorgan's Law

\[
\overline{A + B} = \overline{A} \overline{B} \\
A + B = A \cdot B
\]

Note on notation: bubble = inverse/complement

One law, Absorption law

Write truth tables. Do they correspond to simpler circuits?

\[
A \cdot 0 = 0 \\
A + 0 = A
\]

\[
A + AB = A \\
A + AB = A
\]

NAND is universal.

All Boolean functions can be implemented using only NANDs. Let's prove it! Build NOT, AND, OR, NOR, using only NAND gates.
**XOR: Exclusive OR**

Output = 1 if exactly one input = 1.

Truth table: 

Often used as a one-bit comparator.

*Video game designers, Halloween costume extraordinaire, sci-fi/fantasy screenwriters, I have an idea…*

**Larger gates**

Build a 4-input AND gate using any number of 2-input gates.

**Circuit simplification**

Can we find 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: code detectors

AND gate + NOT gates = code detector, recognizes exactly one input code.

Design a 4-input code detector to output 1 if ABCD = 1001, and 0 otherwise.

Design a 4-input code detector to accept two codes (ABCD=1001, ABCD=1111) and reject all others. (accept = 1, reject = 0)

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?

Voting machines

A majority circuit outputs 1 if and only if a majority of its inputs equal 1. Design a majority circuit for three inputs. Use a sum of products.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Majority</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Triply redundant computers in spacecraft

- Compute everything on three separate devices and compare answers. Majority wins.
- Cosmic rays can flip bits on wires, especially in space.
- Triple redundancy survives failure in one device.
- Space program also hastened integrated circuits.

Margaret Hamilton (speaking of space and reliability)