Chapter 1

David Money Harris and Sarah L. Harris

Chapter 1 :: Topics

• Background
• The Game Plan
• The Art of Managing Complexity
• The Digital Abstraction
• Number Systems
• Logic Gates
• Logic Levels
• CMOS Transistors
• Power Consumption
Background

- Microprocessors have revolutionized our world
  - Cell phones, Internet, rapid advances in medicine, etc.
- The semiconductor industry has grown from $21 billion in 1985 to $300 billion in 2011

The Game Plan

- Purpose of course:
  - Understand what’s under the hood of a computer
  - Learn the principles of digital design
  - Learn to systematically debug increasingly complex designs
  - Design and build a microprocessor
The Art of Managing Complexity

- Abstraction
- Discipline
- The Three –Y’s
  - Hierarchy
  - Modularity
  - Regularity

Abstraction

- Hiding details when they aren’t important
Discipline

- Intentionally restrict design choices
- Example: Digital discipline
  - Discrete voltages instead of continuous
  - Simpler to design than analog circuits – can build more sophisticated systems
  - Digital systems replacing analog predecessors:
    - i.e., digital cameras, digital television, cell phones, CDs

The Three -Y’s

- **Hierarchy**
  - A system divided into modules and submodules

- **Modularity**
  - Having well-defined functions and interfaces

- **Regularity**
  - Encouraging uniformity, so modules can be easily reused
• **Hierarchy**
  
  - **Three main modules:**
    lock, stock, and barrel
  
  - **Submodules of lock:**
    hammer, flint, frizzen, etc.

  
  [Image of a Flintlock Rifle]

• **Modularity**
  
  - **Function of stock:** mount barrel and lock
  
  - **Interface of stock:** length and location of mounting pins

• **Regularity**
  
  - Interchangeable parts

  [Image of a Flintlock Rifle]
Most physical variables are **continuous**
- Voltage on a wire
- Frequency of an oscillation
- Position of a mass

Digital abstraction considers **discrete** subset of values

The Digital Abstraction

The Analytical Engine

- Designed by Charles Babbage from 1834 – 1871
- Considered to be the first digital computer
- Built from mechanical gears, where each gear represented a discrete value (0-9)
- Babbage died before it was finished
Digital Discipline: Binary Values

• Two discrete values:
  – 1’s and 0’s
  – 1, TRUE, HIGH
  – 0, FALSE, LOW
• 1 and 0: voltage levels, rotating gears, fluid levels, etc.
• Digital circuits use **voltage** levels to represent 1 and 0
• **Bit**: Binary digit

George Boole, 1815-1864

• Born to working class parents
• Taught himself mathematics and joined the faculty of Queen’s College in Ireland.
• Wrote *An Investigation of the Laws of Thought* (1854)
• Introduced binary variables
• Introduced the three fundamental logic operations: AND, OR, and NOT.
Number Systems

- Decimal numbers

\[ 5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0 \]

- Binary numbers

\[ 1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10} \]
### Powers of Two

<table>
<thead>
<tr>
<th>Power (n)</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^0$</td>
<td>1</td>
</tr>
<tr>
<td>$2^1$</td>
<td>2</td>
</tr>
<tr>
<td>$2^2$</td>
<td>4</td>
</tr>
<tr>
<td>$2^3$</td>
<td>8</td>
</tr>
<tr>
<td>$2^4$</td>
<td>16</td>
</tr>
<tr>
<td>$2^5$</td>
<td>32</td>
</tr>
<tr>
<td>$2^6$</td>
<td>64</td>
</tr>
<tr>
<td>$2^7$</td>
<td>128</td>
</tr>
<tr>
<td>$2^8$</td>
<td>256</td>
</tr>
<tr>
<td>$2^9$</td>
<td>512</td>
</tr>
<tr>
<td>$2^{10}$</td>
<td>1024</td>
</tr>
<tr>
<td>$2^{11}$</td>
<td>2048</td>
</tr>
<tr>
<td>$2^{12}$</td>
<td>4096</td>
</tr>
<tr>
<td>$2^{13}$</td>
<td>8192</td>
</tr>
<tr>
<td>$2^{14}$</td>
<td>16384</td>
</tr>
<tr>
<td>$2^{15}$</td>
<td>32768</td>
</tr>
</tbody>
</table>

Handy to memorize up to $2^9$
Number Conversion

• Decimal to binary conversion:
  – Convert \(10011_2\) to decimal
    \[16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 = 19_{10}\]

• Decimal to binary conversion:
  – Convert \(47_{10}\) to binary
    \[32 \times 1 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 1 + 1 \times 1 = 101111_2\]
Binary Values and Range

- **N-digit decimal number**
  - How many values? \(10^N\)
  - Range: \([0, 10^N - 1]\)
  - Example: 3-digit decimal number:
    - \(10^3 = 1000\) possible values
    - Range: \([0, 999]\)

- **N-bit binary number**
  - How many values? \(2^N\)
  - Range: \([0, 2^N - 1]\)
  - Example: 3-digit binary number:
    - \(2^3 = 8\) possible values
    - Range: \([0, 7] = [000_2 \text{ to } 111_2]\)
## Hexadecimal Numbers

<table>
<thead>
<tr>
<th>Hex Digit</th>
<th>Decimal Equivalent</th>
<th>Binary Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>F</td>
<td>15</td>
<td>1111</td>
</tr>
</tbody>
</table>
Hexadecimal Numbers

• Base 16
• Shorthand for binary

Hexadecimal to Binary Conversion

• Hexadecimal to binary conversion:
  – Convert $4AF_{16}$ (also written $0x4AF$) to binary

• Hexadecimal to decimal conversion:
  – Convert $0x4AF$ to decimal
Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert \(4AF_{16}\) (also written 0x4AF) to binary
  - 0100 1010 1111_2

- Hexadecimal to decimal conversion:
  - Convert \(4AF_{16}\) to decimal
  - \(16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}\)

Bits, Bytes, Nibbles...

- Bits
  - 10010110
    - Most significant bit
    - Least significant bit

- Bytes & Nibbles
  - byte
    - 10010110
      - Nibble

- Bytes
  - CEBF9AD7
    - Most significant byte
    - Least significant byte
Large Powers of Two

- \(2^{10} = 1 \text{kilo} \approx 1000 \ (1024)\)
- \(2^{20} = 1 \text{ mega} \approx 1 \text{ million} \ (1,048,576)\)
- \(2^{30} = 1 \text{ giga} \approx 1 \text{ billion} \ (1,073,741,824)\)

Estimating Powers of Two

- What is the value of \(2^{24}\)?
- How many values can a 32-bit variable represent?
Estimating Powers of Two

- What is the value of $2^{24}$?
  
  - $2^4 \times 2^{20} \approx 16\text{ million}$

- How many values can a 32-bit variable represent?
  
  - $2^2 \times 2^{30} \approx 4\text{ billion}$

Addition

- Decimal
  
  \[
  \begin{array}{c}
  3734 \\
  + 5168 \\
  \hline
  8902
  \end{array}
  \]
  
  \text{carries}

- Binary
  
  \[
  \begin{array}{c}
  1011 \\
  + 0011 \\
  \hline
  1110
  \end{array}
  \]
  
  \text{carries}
### Binary Addition Examples

1. Add the following 4-bit binary numbers:
   
   \[
   \begin{array}{c}
   1001 \\
   + 0101 \\
   \end{array}
   \]

2. Add the following 4-bit binary numbers:
   
   \[
   \begin{array}{c}
   1011 \\
   + 0110 \\
   \end{array}
   \]

3. Add the following 4-bit binary numbers:
   
   \[
   \begin{array}{c}
   1001 \\
   + 0101 \\
   \hline
   1110
   \end{array}
   \]

4. Add the following 4-bit binary numbers:
   
   \[
   \begin{array}{c}
   111 \\
   + 0110 \\
   \hline
   10001
   \end{array}
   \]

   **Overflow!**
Overflow

- Digital systems operate on a **fixed number of bits**
- Overflow: when result is too big to fit in the available number of bits
- See previous example of $11 + 6$

Signed Binary Numbers

- Sign/Magnitude Numbers
- Two’s Complement Numbers
Sign/Magnitude Numbers

- 1 sign bit, $N$-1 magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0 $A = \{a_{N-1}, a_{N-2}, \ldots, a_2, a_1, a_0\}$
  - Negative number: sign bit = 1 $A = (-1)^{a_{N-1}} \sum_{i=0}^{n-2} a_i 2^i$
- Example, 4-bit sign/mag representations of $\pm 6$:
  - $+6 = 0110$
  - $-6 = 1110$
- Range of an $N$-bit sign/magnitude number:
  $[-(2^{N-1} - 1), 2^{N-1}]$
Sign/Magnitude Numbers

- Problems:
  - Addition doesn’t work, for example -6 + 6:
    
    \[
    \begin{array}{c}
    1110 \\
    + 0110 \\
    \hline
    10100 \text{ (wrong!)}
    \end{array}
    \]
  
  - Two representations of 0 (± 0):
    
    1000
    0000

Two’s Complement Numbers

- Don’t have same problems as sign/magnitude numbers:
  - Addition works
  - Single representation for 0
Two’s Complement Numbers

- Msb has value of $-2^{N-1}$
  \[ A = a_{n-1}(-2^{n-1}) + \sum_{i=0}^{n-2} a_i 2^i \]

- Most positive 4-bit number: 0111
- Most negative 4-bit number: 1000
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an $N$-bit two’s comp number:
  \[ [-(2^{N-1}), 2^{N-1}-1] \]
“Taking the Two’s Complement”

- Flip the sign of a two’s complement number
- Method:
  1. Invert the bits
  2. Add 1
- Example: Flip the sign of $3_{10} = 0011_2$

1. $1100$
2. $+ 1$
   \[
   1101 = -3_{10}
   \]
Two’s Complement Examples

• Take the two’s complement of $6_{10} = 0110_2$

• What is the decimal value of $1001_2$?

Two’s Complement Examples

• Take the two’s complement of $6_{10} = 0110_2$

  1. $1001$
  2. $+ 1$

  $1010_2 = -6_{10}$

• What is the decimal value of the two’s complement number $1001_2$?

  1. $0110$
  2. $+ 1$

  $0111_2 = 7_{10}$, so $1001_2 = -7_{10}$
Two’s Complement Addition

- Add 6 + (-6) using two’s complement numbers
  
  \[
  \begin{array}{c}
  0110 \\
  + 1010 \\
  \hline
  1100 \\
  \end{array}
  \]

- Add -2 + 3 using two’s complement numbers
  
  \[
  \begin{array}{c}
  1110 \\
  + 0011 \\
  \hline
  1011 \text{ (with carry)} \\
  \end{array}
  \]
Increasing Bit Width

- Extend number from \( N \) to \( M \) bits \((M > N)\):  
  - Sign-extension  
  - Zero-extension

Sign-Extension

- Sign bit copied to msb’s  
- Number value is same

**Example 1:**  
- 4-bit representation of 3 = 0011  
- 8-bit sign-extended value: 00000011

**Example 2:**  
- 4-bit representation of -5 = 1011  
- 8-bit sign-extended value: 1111011
Zero-Extension

• Zeros copied to msb’s
• Value changes for negative numbers

Example 1:
- 4-bit value = 0011₂ = 3₁₀
- 8-bit zero-extended value: 00000011₂ = 3₁₀

Example 2:
- 4-bit value = 1011₂ = -5₁₀
- 8-bit zero-extended value: 00001011₂ = 1₁₀

Number System Comparison

<table>
<thead>
<tr>
<th>Number System</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned</td>
<td>[0, 2ᴺ-1]</td>
</tr>
<tr>
<td>Sign/Magnitude</td>
<td>[-2ᴺ-1, 2ᴺ-1-1]</td>
</tr>
<tr>
<td>Two’s Complement</td>
<td>[-2ᴺ-1, 2ᴺ-1]</td>
</tr>
</tbody>
</table>

For example, 4-bit representation:
Logic Gates

- **Perform logic functions:**
  - inversion (NOT), AND, OR, NAND, NOR, etc.

- **Single-input:**
  - NOT gate, buffer

- **Two-input:**
  - AND, OR, XOR, NAND, NOR, XNOR

- **Multiple-input**
Single-Input Logic Gates

**NOT**

\[ Y = \overline{A} \]

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**BUF**

\[ Y = A \]

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Two-Input Logic Gates

**AND**

\[ Y = AB \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</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>

**OR**

\[ Y = A + B \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</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>
Two-Input Logic Gates

AND

\[ Y = AB \]

\[
\begin{array}{ccc}
A & B & Y \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

OR

\[ Y = A + B \]

\[
\begin{array}{ccc}
A & B & Y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

XOR

\[ Y = A \oplus B \]

\[
\begin{array}{ccc}
A & B & Y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

NAND

\[ Y = \overline{A B} \]

\[
\begin{array}{ccc}
A & B & Y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

NOR

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

\[
\begin{array}{ccc}
A & B & Y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

XNOR

\[ Y = A \odot B \]

\[
\begin{array}{ccc}
A & B & Y \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

More Two-Input Logic Gates
More Two-Input Logic Gates

**XOR**

\[ Y = A \oplus B \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</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>0</td>
</tr>
</tbody>
</table>

**NAND**

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

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

**NOR**

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

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

**XNOR**

\[ Y = \overline{A} \oplus B \]

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

Multiple-Input Logic Gates

**NOR3**

\[ Y = \overline{A} + B + C \]

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

**AND4**

\[ Y = ABCD \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</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>0</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>0</td>
</tr>
<tr>
<td>1</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>
### Multiple-Input Logic Gates

#### NOR3

\[ Y = \overline{A + B + C} \]

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

#### AND4

\[ Y = ABCD \]

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

*Multi-input XOR: Odd parity*

### Logic Levels

- Discrete voltages represent 1 and 0
- For example:
  - \(0 = \text{ground} \text{ (GND)}\) or 0 volts
  - \(1 = V_{\text{DD}}\) or 5 volts
- What about 4.99 volts? Is that a 0 or a 1?
- What about 3.2 volts?
Logic Levels

- Range of voltages for 1 and 0
- Different ranges for inputs and outputs to allow for noise

What is Noise?
What is Noise?

- **Anything that degrades the signal**
  - E.g., resistance, power supply noise, coupling to neighboring wires, etc.

- **Example**: a gate (driver) outputs 5 V but, because of resistance in a long wire, receiver gets 4.5 V

```
Driver  Noise  Receiver
5 V     ↓         4.5 V
```

The Static Discipline

- With logically valid inputs, every circuit element must produce logically valid outputs

- Use limited ranges of voltages to represent discrete values
Chapter 1

Logic Levels

![Diagram of Logic Levels]

- **Driver**
  - Logic High Output Range: $V_{OH}$
  - Logic Low Output Range: $V_{OL}$

- **Receiver**
  - Logic High Input Range: $V_{IH}$
  - Logic Low Input Range: $V_{IL}$

- **Noise Margins**
  - **NM$_H$** = $V_{OH} - V_{IH}$
  - **NM$_L$** = $V_{IL} - V_{OL}$

Noise Margins

![Diagram of Noise Margins]
DC Transfer Characteristics

**Ideal Buffer:**

**Real Buffer:**

\[ NM_H = NM_L = \frac{V_{DD}}{2} \]

\[ NM_H, NM_L < \frac{V_{DD}}{2} \]
V_{DD} Scaling

- In 1970’s and 1980’s, V_{DD} = 5 V
- V_{DD} has dropped
  - Avoid frying tiny transistors
  - Save power
- 3.3 V, 2.5 V, 1.8 V, 1.5 V, 1.2 V, 1.0 V, ...
- Be careful connecting chips with different supply voltages

Chips operate because they contain magic smoke
Proof:
  - if the magic smoke is let out, the chip stops working

Logic Family Examples

<table>
<thead>
<tr>
<th>Logic Family</th>
<th>V_{DD}</th>
<th>V_{IL}</th>
<th>V_{IH}</th>
<th>V_{OL}</th>
<th>V_{OH}</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>5 (4.75 - 5.25)</td>
<td>0.8</td>
<td>2.0</td>
<td>0.4</td>
<td>2.4</td>
</tr>
<tr>
<td>CMOS</td>
<td>5 (4.5 - 6)</td>
<td>1.35</td>
<td>3.15</td>
<td>0.33</td>
<td>3.84</td>
</tr>
<tr>
<td>LVTTL</td>
<td>3.3 (3 - 3.6)</td>
<td>0.8</td>
<td>2.0</td>
<td>0.4</td>
<td>2.4</td>
</tr>
<tr>
<td>LVCMOS</td>
<td>3.3 (3 - 3.6)</td>
<td>0.9</td>
<td>1.8</td>
<td>0.36</td>
<td>2.7</td>
</tr>
</tbody>
</table>
Transistors

- Logic gates built from transistors
- 3-port ed voltage-controlled switch
  - 2 ports connected depending on voltage of 3rd
  - d and s are connected (ON) when g is 1

\[ \begin{array}{c|c}
  g = 0 & g = 1 \\
  \hline
  d & d \\
  \hline
  g & OFF \\
  s & ON
\end{array} \]

Robert Noyce, 1927-1990

- Nicknamed “Mayor of Silicon Valley”
- Cofounded Fairchild Semiconductor in 1957
- Cofounded Intel in 1968
- Co-invented the integrated circuit
**Silicon**

- Transistors built from silicon, a semiconductor
- Pure silicon is a poor conductor (no free charges)
- Doped silicon is a good conductor (free charges)
  - n-type (free negative charges, electrons)
  - p-type (free positive charges, holes)

**MOS Transistors**

- **Metal oxide silicon (MOS) transistors:**
  - Polysilicon (used to be metal) gate
  - Oxide (silicon dioxide) insulator
  - Doped silicon
Transistors: nMOS

Gate = 0
OFF (no connection between source and drain)

Gate = 1
ON (channel between source and drain)

Transistors: pMOS

• pMOS transistor is opposite
  – ON when Gate = 0
  – OFF when Gate = 1
Transistor Function

- **nMOS**: pass good 0’s, so connect source to GND
- **pMOS**: pass good 1’s, so connect source to $V_{DD}$
### CMOS Gates: NOT Gate

#### Logic Symbol

- **Input:** A
- **Output:** Y

- **Equation:** \( Y = \overline{A} \)

#### Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Circuit Diagram

- **Power Supply:** V\(_{DD}\)
- **Ground:** GND
- **Transistors:** P1, N1

<table>
<thead>
<tr>
<th>A</th>
<th>P1</th>
<th>N1</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ON</td>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>OFF</td>
<td>ON</td>
<td>0</td>
</tr>
</tbody>
</table>
CMOS Gates: NAND Gate

**NAND**

\[
Y = \overline{AB}
\]

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>P1</th>
<th>P2</th>
<th>N1</th>
<th>N2</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>0</td>
</tr>
</tbody>
</table>
CMOS Gate Structure

NOR Gate

How do you build a three-input NOR gate?
How do you build a two-input AND gate?
AND2 Gate

\[ A \rightarrow Y \]
\[ B \]

Transmission Gates

- nMOS pass 1’s poorly
- pMOS pass 0’s poorly
- Transmission gate is a better switch
  - passes both 0 and 1 well
- When \( EN = 1 \), the switch is ON:
  - \( EN = 0 \) and \( A \) is connected to \( B \)
- When \( EN = 0 \), the switch is OFF:
  - \( A \) is not connected to \( B \)
Pseudo-nMOS Gates

- Replace pull-up network with *weak* pMOS transistor that is always on
- pMOS transistor: pulls output HIGH *only* when nMOS network not pulling it LOW

Pseudo-nMOS Example

Pseudo-nMOS **NOR4**
Gordon Moore, 1929-

- Cofounded Intel in 1968 with Robert Noyce.
- **Moore’s Law:** number of transistors on a computer chip doubles every year (observed in 1965)
- Since 1975, transistor counts have doubled every two years.

---

Moore’s Law

- “If the automobile had followed the same development cycle as the computer, a Rolls-Royce would today cost $100, get one million miles to the gallon, and explode once a year . . .”
  
  – Robert Cringley
Power Consumption

- Power = Energy consumed per unit time
  - Dynamic power consumption
  - Static power consumption

Dynamic Power Consumption

- Power to charge transistor gate capacitances
  - Energy required to charge a capacitance, $C$, to $V_{DD}$ is $CV_{DD}^2$
  - Circuit running at frequency $f$: transistors switch (from 1 to 0 or vice versa) at that frequency
  - Capacitor is charged $f/2$ times per second (discharging from 1 to 0 is free)

- Dynamic power consumption:
  $$P_{\text{dynamic}} = \frac{1}{2}CV_{DD}^2f$$
Static Power Consumption

- Power consumed when no gates are switching
- Caused by the *quiescent supply current*, $I_{DD}$ (also called the *leakage current*)
- Static power consumption:

$$P_{static} = I_{DD}V_{DD}$$

Power Consumption Example

- Estimate the power consumption of a wireless handheld computer
  - $V_{DD} = 1.2$ V
  - $C = 20$ nF
  - $f = 1$ GHz
  - $I_{DD} = 20$ mA
• Estimate the power consumption of a wireless handheld computer
  – $V_{DD} = 1.2 \text{ V}$
  – $C = 20 \text{ nF}$
  – $f = 1 \text{ GHz}$
  – $I_{DD} = 20 \text{ mA}$

$$P = \frac{1}{2}CV_{DD}^2f + I_{DD}V_{DD}$$

$$= \frac{1}{2}(20 \text{ nF})(1.2 \text{ V})^2(1 \text{ GHz}) + (20 \text{ mA})(1.2 \text{ V})$$

$$= 14.4 \text{ W}$$