Lecture #1

OUTLINE

• Course overview
• Introduction: integrated circuits
• Analog vs. digital signals

Course Overview

EECS 40:

• One of five EECS core courses (with 20, 61A, 61B, and 61C)
  ▪ introduces “hardware” side of EECS
  ▪ prerequisite for EE105, EE130, EE141, EE150

• Prerequisites: Math 1B, Physics 7B

Course content:

• Electric circuits
• Integrated-circuit devices and technology
• CMOS digital integrated circuits
IC Technology Advancement

"Moore’s Law": # of transistors/chip doubles every 1.5-2 years
– achieved through miniaturization

Technology Scaling

Investment → Better Performance/Cost → Market Growth

Benefit of Transistor Scaling

<table>
<thead>
<tr>
<th>Generation:</th>
<th>1.5µ</th>
<th>1.0µ</th>
<th>0.8µ</th>
<th>0.6µ</th>
<th>0.35µ</th>
<th>0.25µ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel386™ DX Processor</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>Intel486™ DX Processor</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>Pentium® Processor</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
<tr>
<td>Pentium® II Processor</td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
</tr>
</tbody>
</table>

smaller chip area → lower cost

more functionality on a chip → better system performance
Analog-to-digital & digital-to-analog conversion is essential (and nothing new)

think of a piano keyboard

Most (but not all) observables are analog

think of analog vs. digital watches

but the most convenient way to represent & transmit information electronically is to use digital signals

think of telephony

→ Analog-to-digital & digital-to-analog conversion is essential (and nothing new)

think of a piano keyboard

Analog Signals

• may have direct relationship to information presented
• in simple cases, are waveforms of information vs. time
• in more complex cases, may have information modulated on a carrier, e.g. AM or FM radio

Amplitude Modulated Signal
Analog Signal Example: Microphone Voltage

Voltage with normal piano key stroke
50 microvolt 440 Hz signal

Voltage with soft pedal applied
25 microvolt 440 Hz signal

Digital Signal Representations

Binary numbers can be used to represent any quantity.

We generally have to agree on some sort of “code”, and the dynamic range of the signal in order to know the form and the number of binary digits (“bits”) required.

**Example 1**: Voltage signal with maximum value 2 Volts
- Binary two (10) could represent a 2 Volt signal.
- To encode the signal to an accuracy of 1 part in 64 (1.5% precision), 6 binary digits (“bits”) are needed.

**Example 2**: Sine wave signal of known frequency and maximum amplitude 50 μV; 1 μV “resolution” needed.
Example 2 (continued)

Possible digital representation for the sine wave signal:

<table>
<thead>
<tr>
<th>Analog representation: Amplitude in $\mu$V</th>
<th>Digital representation: Binary number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000001</td>
</tr>
<tr>
<td>2</td>
<td>000010</td>
</tr>
<tr>
<td>3</td>
<td>000011</td>
</tr>
<tr>
<td>4</td>
<td>000100</td>
</tr>
<tr>
<td>5</td>
<td>000101</td>
</tr>
<tr>
<td>8</td>
<td>001000</td>
</tr>
<tr>
<td>16</td>
<td>010000</td>
</tr>
<tr>
<td>32</td>
<td>100000</td>
</tr>
<tr>
<td>50</td>
<td>110010</td>
</tr>
<tr>
<td>63</td>
<td>111111</td>
</tr>
</tbody>
</table>

Why Digital?

(For example, why CDROM audio vs. vinyl recordings?)

- Digital signals can be transmitted, received, amplified, and re-transmitted with no degradation.

- Digital information is easily and inexpensively stored (in RAM, ROM, etc.), with arbitrary accuracy.

- Complex logical functions are easily expressed as binary functions (e.g. in control applications).

- Digital signals are easy to manipulate (as we shall see).
Digital signals offer an easy way to perform logical functions, using Boolean algebra.

- Variables have two possible values: “true” or “false”
  - usually represented by 1 and 0, respectively.

All modern control systems use this approach.

**Example:** Hot tub controller with the following algorithm

Turn on the heater if the temperature is less than desired ($T < T_{set}$) and the motor is on and the key switch to activate the hot tub is closed. Suppose there is also a “test switch” which can be used to activate the heater.

---

**Hot Tub Controller Example**

- Series-connected switches:
  - A = thermostatic switch
  - B = relay, closed if motor is on
  - C = key switch
- Test switch T used to bypass switches A, B, and C

**Simple Schematic Diagram of Possible Circuit**

![Simple Schematic Diagram](image-url)
“Truth Table” for Hot Tub Controller

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

Notation for Logical Expressions

**Basic logical functions:**

- **AND:** “dot” Example: \( X = A \cdot B \)
- **OR:** “+ sign” Example: \( Y = A + B \)
- **NOT:** “bar over symbol” Example: \( Z = \overline{A} \)

- Any logical expression can be constructed using these basic logical functions

**Additional logical functions:**

- **Inverted AND = NAND:** \( \overline{A \cdot B} \) (only 0 when \( A \) and \( B = 1 \))
- **Inverted OR = NOR:** \( \overline{A + B} \) (only 1 when \( A = B = 0 \))
- **Exclusive OR:** \( A \oplus B \) (only 1 when \( A, B \) differ)
  
  i.e., \( A + B \) except \( A \cdot B \)

The most frequently used logical functions are implemented as electronic building blocks called “gates” in integrated circuits.
First define logical values:

- closed switch = “true”, i.e. boolean 1
- open switch = “false”, i.e. boolean 0

**Logical Statement:**
Heater is on (H = 1) if A and B and C are 1, or if T is 1.

**Logical Expression:**
H = 1 if (A and B and C are 1) or (T is 1)

**Boolean Expression:**
H = (A · B · C ) + T

---

**Summary**

**Attributes of digital electronic systems:**

1. Ability to represent real quantities by coding information in digital form

2. Ability to control a system by manipulation and evaluation of binary variables using Boolean algebra