

Digital Basics

Digital vs Analog Signals

- real world vs digital world
- Zero/low and One/high

Digital logic

Switching algebra (Boolean)

Circuits to realize Boolean functions (gates)

Analog versus Digital Electronics

- Most (but not all) observables are analog
  think of analog versus digital watches

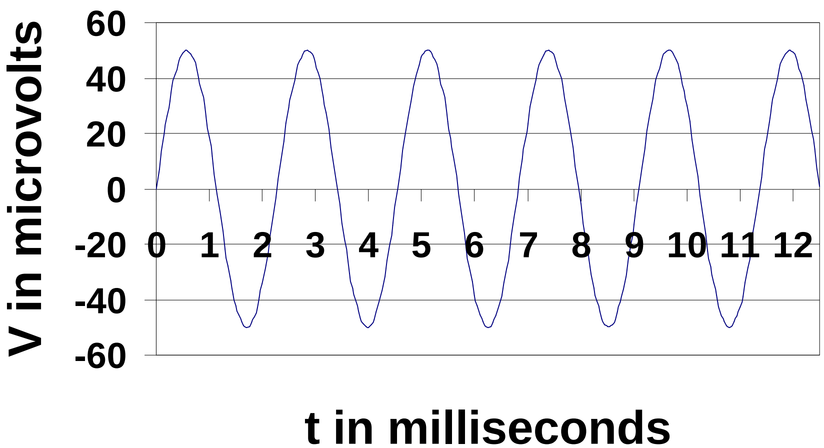
- But the most convenient way to represent and transmit information electronically is digital
  think of audio recordings vs original edison wax recordings

- Analog/Digital and Digital/Analog conversion is essential (and nothing new)
  think of a piano keyboard vs violin

Analog Example:

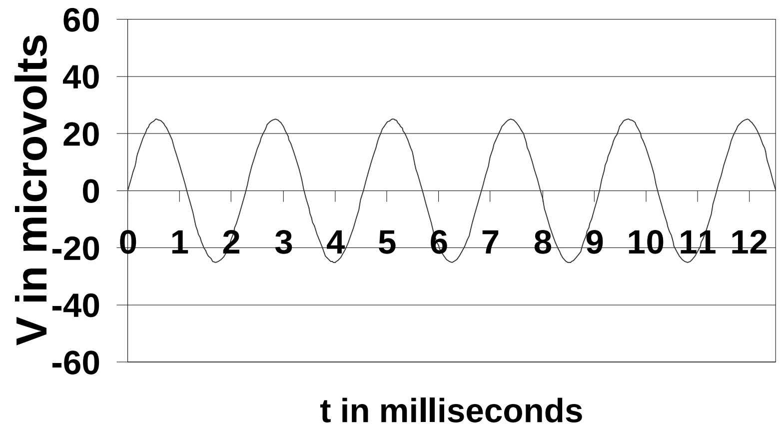
Analog signal (microphone voltage) representing piano middle A (440 Hz)..

50 microvolt 440 Hz signal



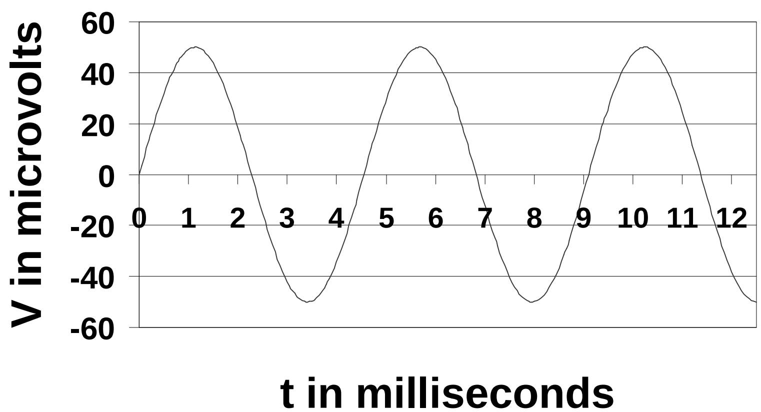
Microphone voltage with normal key stroke

25 microvolt 440 Hz signal



Microphone voltage with soft pedal

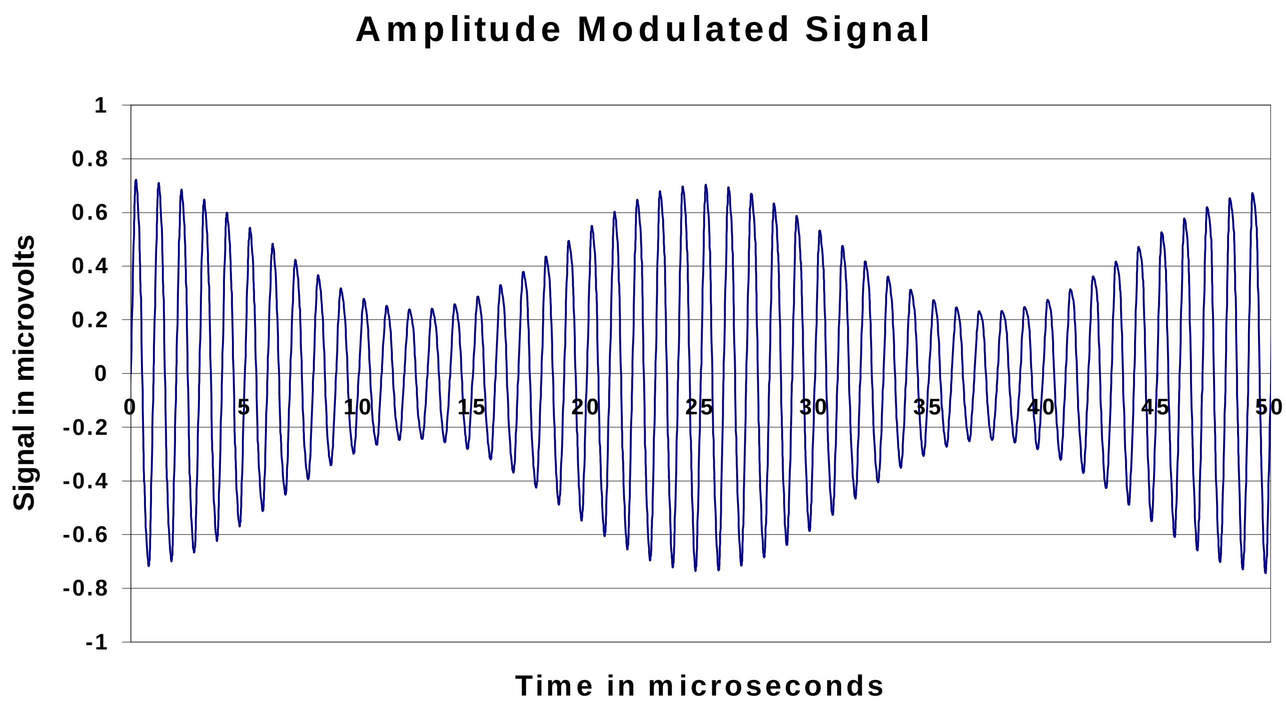
50 microvolt 220 Hz signal



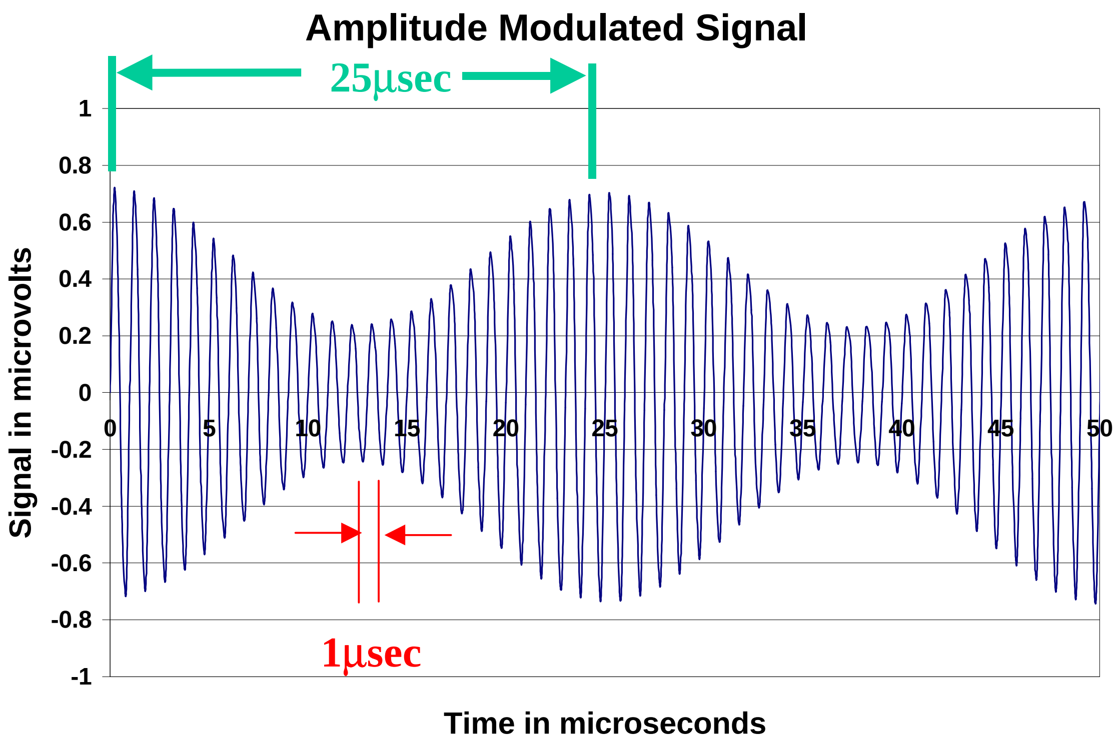
Analog signal representing piano A, but one octave below middle A (220 Hz)

Analog Signals

- May have very physical relationship to information presented
- In the simplest, are direct waveforms of information vs time
- In more complex cases, may have information modulated on a carrier as in AM or FM radio



1000 KHZ AM radio station signal (analog)



Parameters of sin waves:

Period (time to repeat)

Frequency (1/period)

Phase

Amplitude

Note: The period of the carrier is 1µsec * (that is, the frequency is 1MHz)
The period of the modulation is 25µsec (that is, the frequency is 40kHz)
The amplitude of the modulation is about 50% of the maximum possible

(Please forgive liberties with modulation frequency - AM stations cannot modulate above 5KHz - but 40KHz is easier to draw)

Digital Signal Representations

By using binary numbers we can represent any quantity. For example a binary two (**10**) could represent a 2 Volt signal. But 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 minimum number of bits.

Example: We want to encode to an accuracy of one part in 64 (i.e. 1.5% precision). It takes 6 binary digits (or "bits") to represent any number 0 to 63.

Example: Possible digital representation for a pure sine wave of known frequency. We must choose maximum value and "resolution" or "error," then we can encode the numbers. Suppose we want 1μV accuracy of amplitude with maximum amplitude of 50μV, We could use a simple pure binary code with 6 bits of information. (Why 6 bits…. What if we only use 5?)

Answer: with 5 binary digits we can represent only 32 values

Digital Signal Representations

Example: Possible digital representation for the sine wave signals, and highlighting our maximum possible 50μV sine wave

Amplitude in μV		Binary representation
1		000001
2		000010
3	?	000011
4	?	000100
5		000101
etc.		
8		001000
16		010000
32		100000
50 (= 32+16+2)		**110010**
63		111111

Digital Representations of Logical Functions

Digital signals also offer an effective way to execute logic. The formalism for performing logic with binary variables is called *Switching Algebra* or *Boolean Algebra.*

In switching algebra we have only **“true”** and **“false”** conditions, usually represented by binary **1** and **0**, respectively. Thus stating that “A is true” and “B is false” is equivalent to stating A=1 and B=0.

The utility of switching algebra is that we can perform elaborate logical operations with simple Boolean arithmetic. All modern control systems are digital, utilizing this approach.

Thus digital electronics combines two important properties: 1) The ability to represent real functions by coding the information in digital form, and 2) The ability to control a system by a process of manipulation and evaluation of digital variables using switching algebra.

So Why Digital?

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

- Digital signals can be transmitted, received, amplified, and re-transmitted with no degradation.
- Binary numbers are a natural method of expressing logical variables.
- Complex logical functions are easily expressed as binary functions (e.g., in control applications … see next page).
- Digital signals are easy to manipulate (as we shall see).
- With digital representation, we can achieve arbitrary levels of “dynamic range,” that is, the ratio of the largest possible signal to the smallest than can be distinguished above the background noise
- Digital information is easily and inexpensively stored (in RAM, ROM, EPROM, etc.), again with arbitrary accuracy.

Logic Functions

Logic Expression : To create logic values we will define **"True"** , as **boolean 1** and **"False"** , as **Boolean 0**.

Moreover we can associate a logic variable with a circuit node. Typically we associate logic 1 with a high voltage (e.g. 2V) and and logic 0 with a low voltage (e.g. 0V).

Example: The logic variable H is true (H=**1**) if (A **and** B **and** C are **1) or** T is **1,** where all of A,B,C and T are also logical variables**.**

- Logic Statement: H = 1 if A and B and C are 1 or T is 1.

- We use "dot" to designate logical "and" and "+" to designate logical or in switching algebra. So how can we express this as a Boolean Expression?

- Boolean Expression: $H = (A \cdot B \cdot C) + T$

Logical Expressions

Standard logic notation :

AND: "dot" Examples: $X = A \cdot B$; $Y = A \cdot B \cdot C$

OR : "+ sign" Examples: $W = A+B$; $Z = A+B+C$

NOT: "bar over symbol for complement" Example: $Z = \overline{A}$

With these basic operations we can construct any logical expression.

Logic Function Example

- Boolean Expression: $H = (A \cdot B \cdot C) + T$

This can be read H=**1** if (A **and** B **and** C are **1)** **or** T is **1,** or

H is true if all of A,B,and C are true, or T is true, or

The voltage at node H will be high if the input voltages at nodes A, B and C are high or the input voltage at node T is high

Evaluation of Logical Expressions with "Truth Tables"

Truth Table for Logic Expression $H = (A \cdot B \cdot C) + T$

A	B	C	T	H
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	**0**
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	1
0	0	0	1	1
0	0	1	1	1
0	1	0	**1**	1
0	1	1	**1**	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Evaluation of Logical Expressions with "Truth Tables"

The Truth Table completely describes a logic expression

In fact, we will use the Truth Table as the fundamental meaning of a logic expression.

Two logic expressions are equal if their truth tables are the same

The Important Logical Functions

The most frequent (i.e. important) logical functions are implemented as electronic “building blocks” or “gates”.

We already know about **AND** , **OR** and **NOT** What are some others:

Combination of above: inverted AND = **NAND,**
inverted OR **= NOR**

And one other basic function is often used: the “EXCLUSIVE OR” … which logically is “or except not and”

Some Important Logical Functions

- **“AND”** $A \cdot B \quad (\text{or } A \cdot B \cdot C)$
- **“OR”** $A + B \quad (\text{or } A + B + C + D \ldots)$
- **“INVERT” or “NOT”** not A or $\overline{A}$
- **“not AND” = NAND** $\overline{AB} \quad (\text{only } 0 \text{ when } A \text{ and } B = 1)$
- **“not OR” = NOR** $\overline{A + B} \quad (\text{only } 1 \text{ when } A = B = 0)$
- **exclusive OR = XOR** $A \oplus B \quad (\text{only } 1 \text{ when } A, B \text{ differ})$
  i.e., $A + B$ **except** $A \cdot B$

Logic Gates

These are circuits that accomplish a given logic function such as "OR". We will shortly see how such circuits are constructed. Each of the basic logic gates has a unique symbol, and there are several additional logic gates that are regarded as important enough to have their own symbol. The set is: AND, OR, NOT, NAND, NOR, and EXCLUSIVE OR.