Analog, Digital Signals; Computer Structure

David M. Auslander

Mechanical Engineering

University of California at Berkeley

Signal Impedance

- Relationship of current and voltage at a terminal (port)
- Recall, Power = current * voltage (properly defined)

Output, Input Impedance

- Output impedance (static)
  - Change in voltage associate with a load that draws current
- Input impedance (static)
  - Equivalent resistance looking into terminals of a device
- Maximize power by matching impedance
- Minimize power by mismatching impedance

Impedance in Measurement and Actuation

- Measurement
  - Minimize insertion loss
  - High output impedance to draw minimum power
  - Amplifier with very high input impedance provides isolation
- Actuation
  - Maximize power to load
  - Power amplifier with very low output impedance

Signal Variables

- Voltage or current
  - Voltage more common for labs because it is easier to measure and work with
  - Current more common indistrially because it rejects noise better
- Can be “raw” or modulated
  - Information associated with instantaneous value or with some property of the signal (eg, frequency)

Information Content

- "How many decisions can be made from this signal?"
- Information theory (Claude Shannon)
- Closely related to entropy, 2nd law of thermodynamics
- Common measure: bits
  - One bit → one decision
  - Number of decisions = 2^n; n is number of bits

Analog Signals

- Information is signal variable value
- Resolution depends only on system quality
  - Mainly noise, but also type of recording equipment, etc.
  - Information content is theoretically infinite
  - Actual information content is probabilistic

Digital Signals

- One wire, one bit
- Signal decision MUCH wider than noise
  - Nominal, *eg*, 0 volt and 5 volt
- Buffer zone to avoid ambiguity
- Device ("gate") output s are guaranteed to be far away from ambiguous range for device inputs
- Example …

TTL Signal Buffer Zones

- (Approximate) buffer zones so devices can never(!) confuse high and low (or 1 and 0)
  - Output for high > 3.5 volts
  - Input interprets high as > 2.5 volts => 1 volt buffer
  - Output for low < 1.5 volts
  - Input interprets low as < 0.7 volts => 0.8 volt buffer
- Between these values is undefined

(Nearly) Noise Free Operation

- These buffers are big enough so that signals can be propagated with no noise at all.
- Many wires are needed to get a signal with significant information content
- In addition to digital signal buffer, error detection and correction information can be added to a signal
- Example: parity bit

Synchronous Circuits

- Digital (binary) information gives noise-free values
- Large circuits have timing issues
  - Based on how long it takes signals to propagate from one place in the circuit to another
- Using a “clock” eliminates that error source as well
  - All subcircuits must settle before next tick of clock

Computers

- Based on digital logic and synchronous circuits
- Use error detection and correction internally
- Intrinsic error rate is so low we are willing to bet our lives on them!
- Software "bugs" way more common than operating errors

Computer Operating Principles
The Nickel Tour!

- Data – collections of binary signals
  - Data items coded according to a variety of conventions
  - Integers (signed or unsigned), floats, characters, etc.
  - Instructions – tell computer what to do
- Central Processing Unit (CPU)
  - Registers – hold data
  - Processors – operate on data in registers

Memory

- Technically, a separate memory is not needed
- A “register” (see CPU) is memory
- Register circuitry, however, is very expensive (and very fast)
- “Memory” is much slower, much bigger, and much cheaper
- Memory contains a mix of instructions and data

Address Space

- The memory uses “addresses” to identify specific data
- Addresses used in programs translate into electrical signals that operate the memory
- “Logical” addresses (in programs) can correspond directly to physical addresses (in memory) or can be mapped by access circuits

How It Works

- “Instruction” – unit activity
- CPU gets instruction from memory (stores it in a register) and figures out what to do
- Gets data from memory if necessary
- Operates on data in registers
- Returns result(s) to registers
- Writes data to memory if necessary

Sequential Device

- Computing is thus entirely sequential
- One instruction at a time
- Instructions are quite primitive
- High level languages retain sequential nature (Java, etc.) – line-at-a-time

Timing

- This activity is synchronized by the system "clock"
  - A clock is a circuit that provides a constant frequency square-wave output
- Instruction takes one or more ticks (cycles) to complete
- Typical desktop (2007) has ~1-3 GHz clock, so instruction completes in a modest number of nanoseconds

Speed of Embedded Processors

- Embedded applications have a much wider range
- Economics drives computing power (and how much can be accomplished)
- Speeds can be as much as 1000 Xs slower!

Connecting the Computer to the Outside World

- Peripheral devices – printers, disk drives, keyboard, display, control I/O, etc.
- These are MUCH slower than the computer
- Separate facilities (hardware) are used to connect them
- They use an address space – sometimes common with memory, sometimes separate

Real Computers

- This is a simplified model (corresponds to early computers)
- Computers use many tricks to speed up operation
  - Cache, pipelines, multiple data paths, etc.
- General principles remain the same (and have been for half a century!)

What Makes it Work?

- Almost error free, regardless of scale!
- This is accomplished by
  - Discrete data representation (bits)
  - Discrete time (clock synchronized)
  - Minimize size to reduce power, increase speed
  - Flexibility: intermix data of various codings and instructions in memory