A Little History

- 1947 – Shockley, Brattain, and Bardeen invented the transistor at Bell Labs
- 1961 – First commercial IC by Fairchild/TI
- 1963 – CMOS invented
- 1965 – Moore’s law
- 1968 – State of the art: 64 transistor chip
Moore’s Law

- “Transistor capacity doubles every 18 months.”

Driving force behind
- Research and industry
- Embedded systems
- EDA
- PCs
- Internet
- PDAs
- Aerospace, Automobiles, Defense, …
Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)
Embedded Systems

- Devices other than desktop PCs, servers, and notebooks
  - Electricity running through
  - Perform something intelligent
- Hardware/software which form a component of a larger system, but are concealed from user
- Computers camouflaged as non-computers
- The future of computing!
Processors

- **What is a processor?**
  - Artifact that computes (runs algorithms)
  - Controller and data-path

- **General-purpose (GP) processors:**
  - Variety of computation tasks
  - Functional flexibility
  - Slow and power hungry

- **Single-purpose (SP) processors:**
  - One computation task
  - Functional inflexibility
  - Fast and power efficient
GP/SP Processor Architecture

Data-Path

Data Input

Data Output

Controller

Status

Control

Control
GP vs. SP Processors

**GP:**
- Programmable controller
  - Control logic is stored in memory
  - Fetch/decode overhead
- Highly general data-path
  - Typical bit-width (8, 16, 32, 64)
  - Complete set of arithmetic/logic units
  - Large set of registers

**SP:**
- Hardwired controller
  - No need for program memory and cache
  - No fetch/decode overhead
- Highly tuned data-path
  - Custom bit-width
  - Custom arithmetic/logic units
  - Custom set of registers
Storage

- What is a memory?
  - Artifact that stores bits
  - Storage fabric and access logic

- Write-ability
  - Manner and speed a memory can be written

- Storage-permanence
  - Ability of memory to hold stored bits after they are written

- Many different types of memories
  - Flash, SRAM, DRAM, etc.

- Common to compose memories
Write-ability

- Ranges of write-ability
  - High end
    - Processor writes to memory simply and quickly
    - E.g., RAM
  - Middle range
    - Processor writes to memory, but slower
    - E.g., FLASH, EEPROM
  - Lower range
    - Special equipment, “programmer”, must be used to write to memory
    - E.g., EPROM, OTP ROM
  - Low end
    - Bits stored only during fabrication
    - E.g., Mask-programmed ROM
Storage-permanence

- Range of storage-permanence
  - High end
    - Essentially never loses bits
    - E.g., mask-programmed ROM
  - Middle range
    - Holds bits days/months/years after memory’s power source turned off
    - E.g., NVRAM
  - Lower range
    - Holds bits as long as power supplied to memory
    - E.g., SRAM
  - Low end
    - Begins to lose bits almost immediately after written
    - E.g., DRAM
Memory Types

- Mask-programmed ROM
- OTP ROM
- EPROM
- EEPROM
- Flash
- NVRAM
- SRAM/DRAM

Storage-permanence

Write-ability

In-system programmable

Ideal
Communication

- What is a bus?
  - An artifact that transfers bits
  - Wires, air, or fiber and interface logic

- Associated with a bus, we have:
  - Connectivity scheme
    - Serial Communication
    - Parallel Communication
    - Wireless Communication
  - Protocol
    - Ports
    - Timing Diagrams
    - Read and write cycles
  - Arbitration scheme, error detection/correction, DMA, etc.
Serial Communication

- A single wire used for data transfer
- One or more additional wires used for control (but, some protocols may not use additional control wires)
- Higher throughput for long distance communication
  - Often across processing node
- Lower cost in terms of wires (cable)
- E.g., USB, Ethernet, PCIe, RS232, I²C, etc.
Parallel Communication

- Multiple wires used for data transfer
- One or more additional wires used for control
- Higher throughput for short distance communication
  - Data misalignment problem
  - Often used within a processing node
- Higher cost in terms of wires (cable)
- E.g., ISA, AMBA, PCI, etc.
Wireless Communication

- Infrared (IR)
  - Electronic wave frequencies just below visible light spectrum
  - Diode emits infrared light to generate signal
  - Infrared transistor detects signal, conducts when exposed to infrared light
  - Cheap to build
  - Need line of sight, limited range

- Radio frequency (RF)
  - Electromagnetic wave frequencies in radio spectrum
  - Analog circuitry and antenna needed on both sides of transmission
  - Line of sight not needed, transmitter power determines range
Peripherals

- Perform specific computation task

- *Custom* single-purpose processors
  - Designed by us for a unique task

- *Standard* single-purpose processors
  - “Off-the-shelf”
  - pre-designed for a common task
Timers/Counters

- **Timers:** measure time intervals
  - To generate timed output events
  - To measure input events
  - Top: max count reached

- **Range and resolution**

- **Counters:** like a timer, but count pulses on a general input signal rather than clock
  - e.g., count cars passing over a sensor
  - Can often configure physical device as either a timer or counter
Watchdog Timers

- Must reset timer every X time units, else timer generates a signal
- Common use: detect failure, self-reset
Pulse Width Modulators

- Generate pulses with specific high/low times
- Duty cycle: % time high
  - Square wave: 50% duty cycle
- Common use: control average voltage to electric device
  - Simpler than DC-DC converter or digital-analog converter
  - DC motor speed, dimmer lights
Displays & Keypads

- N rows by M columns
- Controller may be built into the LCD module or managed in software
- Display: challenge is to reverse polarity and scan the pixels at a high rate
- Keypad: challenge is to scan at a high rate
Stepper Motor Controller

- Stepper motor: rotates fixed number of degrees when given a “step” signal
  - In contrast, DC motor just rotates when power applied, coasts to stop

- Rotation achieved by applying specific voltage sequence to coils

- Controller greatly simplifies this
Analog-to-Digital Converter

\[ V_{\text{max}} = 7.5\text{V} \]

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0V</td>
<td>1110</td>
</tr>
<tr>
<td>6.5V</td>
<td>1101</td>
</tr>
<tr>
<td>6.0V</td>
<td>1100</td>
</tr>
<tr>
<td>5.5V</td>
<td>1011</td>
</tr>
<tr>
<td>5.0V</td>
<td>1010</td>
</tr>
<tr>
<td>4.5V</td>
<td>1001</td>
</tr>
<tr>
<td>4.0V</td>
<td>1000</td>
</tr>
<tr>
<td>3.5V</td>
<td>0111</td>
</tr>
<tr>
<td>3.0V</td>
<td>0110</td>
</tr>
<tr>
<td>2.5V</td>
<td>0101</td>
</tr>
<tr>
<td>2.0V</td>
<td>0100</td>
</tr>
<tr>
<td>1.5V</td>
<td>0011</td>
</tr>
<tr>
<td>1.0V</td>
<td>0010</td>
</tr>
<tr>
<td>0.5V</td>
<td>0001</td>
</tr>
<tr>
<td>0V</td>
<td>0000</td>
</tr>
</tbody>
</table>

Proportionality

Analog to digital

Digital output

0100 1000 0110 0101

Time

t1  t2  t3  t4

Analog output (V)

Digital input

0100 1000 0110 0101

Time

t1  t2  t3  t4

Digital to analog
Real-time Systems

- A real-time system has to produce correct result at the right time (deadline driven)

- A real-time system imposes stringent timing requirements in addition to correctness
  - Hard real-time
  - Firm real-time
  - Soft real-time
Hard Real-time

- System designed to meet all deadlines
- A missed deadline is a design flaw
- Examples:
  - Shuttle navigation system
  - Nuclear reactor monitoring system
- System hardware (over) designed for worst-case performance
- System software vigorously tested
- Formal proofs used to guaranty timing correctness
Firm Real-time

- System designed to meet all deadlines, but
  - “Occasional” missed deadline is allowed
    - Sometimes statistically quantified (e.g., 5% misses)
  - No need to compute further once a deadline is missed

- Examples:
  - Multimedia systems

- System hardware designed for average case performance

- System software tested under average (ideal) conditions
Soft Real-Time

- System designed to meet as many deadlines as possible
  - Best effort to complete within specified time, but may be late

- Examples:
  - Network switch or router

- System hardware designed for average case performance

- System software tested under average (ideal) conditions
Embedded Operating Systems

- Must provide means for dynamic task creation
  - Create, join, and cancel

- Must provide means for task synchronization and communication
  - Shared memory vs. message passing
  - Semaphore and condition variables vs. monitors

- Posix threads a common standard provides thread creation and synchronization
Fixed Point Arithmetic

- Using integer math to emulate floating point numbers and operations
- Determine range and precision (i.e., $m.n$)
- Define $+$, $-$, $\times$, and $/$
- Analyze for overflow
- Use tables for common math functions, e.g., sine, cosine, etc.
Digital Signal Processing

- Any interesting embedded system has to process some input signals and generate some output signals
  - We use the term *signal* in a general way
- Digital devices process signals in digital form
  - A uniformly sampled stream of data spread in time (e.g., audio) or space (e.g., image)
General DSP Architecture

![Diagram of DSP Architecture]

- **Environment**
  - Sensors
  - Actuators

- **Embedded System**
  - A/D
  - D/A
  - Microprocessor (μP)
  - Memory

Diagram arrows indicate:
- \( f(t) \) from Sensors to A/D
- \( f_n \) from A/D to D/A
- \( u(t) \) from Actuators to D/A
- \( u_n \) from D/A to Microprocessor

The system processes signals from the environment through sensors, converts them to digital form, processes them in the microprocessor, and converts them back to analog form for actuation.
Sensors and Actuators

- **Sensors:**
  - Capture physical stimulus (e.g., heat, light, sound, pressure, magnetism, or other mechanical motion)
  - Typical generate a proportional electrical current
  - May require analog interface

- **Actuators**
  - Convert a command to a physical stimulus (e.g., heat, light, sound, pressure, magnetism, or other mechanical motion)
  - May require analog interface

Sensors: solenoid, mic, speaker, laser diode/transistor, compass, dc motor, accelerometer
Analog / Digital Domain Conversion

- **Sampling**: how often is the signal converted?
  - Twice as high as the highest frequency signal present in the input
  - As much as 10 to 20 times for even better results

- **Quantization**: how many bits used to represent a sample?
  - Sufficient to provide required dynamic range (measured as dB)
    - E.g., 16-bit A/D $\Rightarrow 20 \times \log_{10}(2^{16}) = 96$ dB (human ear limit)
  - Under-loading: dynamic range not used properly
  - AC coupling: a DC offset renders some of dynamic range unusable
  - Clipping: input signal beyond the dynamic range

- **Aliasing**: erroneous signals, not present in analog domain, but present in digital domain
  - Use anti-aliasing filters
  - Sample at higher than necessary rate
  - Remember the spinning bicycle wheel
Signal Processing

- Digital signal $S_0, S_1, S_2 \ldots S_{n-1}$

- What can we do with it?
  - Transpose: e.g., $Z_i = S_i + K$
  - Amplify: e.g., $Z_i = S_i \times \alpha$
  - Compose: e.g., $Z_i = (S_{1i} \times \alpha^1 + K^1) + (S_{2i} \times \alpha^2 + K^2)$
  - Filter: e.g., $Z_i = (S_i + S_{i+1}) / 2$
  - Compress: e.g., using Huffman codes
  - Archive, match against database, etc.

- Or, process after converting to frequency domain
  - Spectral analysis
Frequency Domain

- Any continues time varying signal can be represented as the sum of cosine functions of different amplitude and frequency
  - E.g., input signal captured as the sum of 4 cosine functions

- Once in frequency domain, certain manipulations become trivial (e.g., filtering)
Control Systems

- Control systems are a common class of embedded systems
- Goal is to make a system’s output track a desired reference value
  - Cruise control, thermostat, VCR tape speed, etc.
- We’ll take a look at open-loop and closed-loop control systems
- We’ll take a look at PID control
Open-Loop Control Example

Controller

\[
\begin{align*}
\text{every 10ms do } & \{ \\
& u_k = \left(r_k / 1000.0\right) \times 5.0 \\
& u_k = c\left(u_k \times (2^8 - 1) / 5.0\right)
\}
\end{align*}
\]

\[r_k\]
\[u_k\]
\[0-5V \quad 8\text{-bit D/A}\]
\[\Delta T = .01 \text{ sec}\]
\[r_k = [0..1000]\]

What if a load is placed on the motor?

\[c(x) \text{ bound } x \text{ to } [0-2^8]\]

\[v(t)\]

DC Motor

Open load response:
\[0V = 0 \text{ RPM}\]
\[5V = 1000 \text{ RPM}\]
Closed-Loop Control

\[ r_k \text{ is the reference (to be controlled) value} \]
\[ k \text{ is a discrete time variable} \]
\[ t = k \times \Delta T \text{ is a continues time variable} \]
\[ e_k = r_k - v_k \text{ is plant error} \]
Proportional Integral Derivative (PID) Controllers

• Proportional control: A controller that multiplies the error by a constant
  \[ u_k = e_k \times P \]

• Integral control: A controller that considers the integral of error over time (using history)
  \[ u_k = (e_0 + e_1 + \ldots + e_k) \times I \]

• Derivative control: A controller that considers the differential of error over time (predict future)
  \[ u_k = (e_k - e_{k-1}) \times D \]
Conclusion

- Introduction to embedded systems
- Hardware
  - Processors
  - Memories
  - Communication
  - Peripherals
- Software
  - Real-time operating systems
  - Application domains (DSP, control)