“I believe we are now entering the Renaissance phase of the Information Age, where creativity and ideas are the new currency, and invention is a primary virtue, where technology truly has the power to transform lives, not just businesses, where technology can help us solve fundamental problems.”

Carly Fiorina, CEO, Hewlett Packard Corporation
eMerging Societal-Scale Systems

New System Architectures
New Enabled Applications
Diverse, Connected, Physical, Virtual, Fluid

Information Appliances
“Server”
“Client”

MEMS BioMonitoring

Scalable, Reliable, Secure Services
Gigabit Ethernet
Massive Cluster
Embedded Software Systems

• Computational
  – but not first-and-foremost a computer
• Integral with physical processes
  – sensors, actuators
• Reactive
  – at the speed of the environment
• Heterogeneous
  – hardware/software, mixed architectures
• Networked
  – shared, adaptive

Source: Edward A. Lee
Observations

• We are on the edge of a revolution in the way electronics products are designed

• System design is the key (also for IC design!)
  – Start with the highest possible level of abstraction (e.g. control algorithms)
  – Establish properties at the right level
  – Use formal models
  – Leverage multiple “scientific” disciplines
Course overview

Managing Complexity

Orthogonalizing concerns

Behavior Vs. Architecture

Computation Vs. Communication
Behavior Vs. Architecture

- Models of Computation
- Quantity estimation
- Synthesis: HW and SW
- Mapping
- Refinement
- Assign functionality to arch elements
  - HW/SW partitioning, Scheduling
- System behavior
- System Architecture
- Comm. and comp. resources

Synthesis:
- HW and SW

EE249Fall03
Behavior Vs. Communication

- Clear separation between functionality and interaction model
- Maximize reuse in different environments, change only interaction model

**ETROPOLIS**

**PIG**: Protocol interface generation

**PEARLS**: Latency insensitive protocols
Administration

• Office hours: Alberto’s : Tu-Th 12:30pm-2pm or (better) by appointment (2-4882)

• Teaching Assistant:
  – Alessandro PINTO, apinto@eecs.berkeley.edu
Grading

• Grading will be assigned on:
  – Homework (~30%)
  – Project (~50%)
  – Reading assignments (~20%)

• There will be approx. 7 homework (due 2 weeks after assignment) and 6 reading assignments
Discussion sections

- Lab section (Th. 4-6):
  - tool presentations

- Discussion Session (Tu. 5-6)
  - students’ presentation of selected papers
    - Each student will be required to fill in a questionnaire in class for each discussion session
    - Each student (in groups of 2-3 people) will have to make an oral presentation once during the class

- Auditors are OK but please register as P-NP
Links

- Class
  - http://www-cad.eecs.berkeley.edu/~polis/class/

- On the left frame, go to Start EE249!
  - Subscribe to the mailing list
  - Fill up the questionnaire

- There is a message board for communication
Outline of the course

• Part 2. Design Methodology (Platform-based Design)
• Part 3. Functional Design: Models of Computation
• Part 4. Architecture Design: Capture and Modeling
• Part 5. Exploration and Mapping
• Part 6. Implementation Verification and Synthesis, Hardware and Software
Outline for the Introduction

• Examples of Embedded Systems
• Their Impact on Society
• Design Challenges
• Embedded Software and Control
Electronics and the Car

- More than 30% of the cost of a car is now in Electronics
- 90% of all innovations will be based on electronic systems
FUNCTION OF CONTROLS

Typical minivan application

Configure
Sense
Actuate
Regulate
Display
Trend
Diagnose
Predict
Archive
CARRIER CONTROLS BUSINESS

Market segments

2001 ($ millions)

- Refrigeration: $87
- Residential HVAC: $212
- Commercial HVAC: $175

Total: $474
FUNCTION OF CONTROLS
Typical commercial HVAC application

Configure
Sense
Actuate
Regulate
Display
Trend
Diagnose
Predict
Archive
Elevators

1. EN: GeN2-Cx
2. ANSI: Gen2/GEM
3. JIS: GeN2-JIS
## Segments

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stops/Rise</strong></td>
<td>&lt; 20 stops</td>
<td>&lt; 64 stops</td>
<td>&lt; 128 stops</td>
</tr>
<tr>
<td></td>
<td>Opportunity: &lt; 6 stops (20m)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Group Size</strong></td>
<td>Simplex</td>
<td>1 – 8 cars</td>
<td>1 – 8 cars</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>&lt; 4 m/s</td>
<td>&lt; 4 m/s</td>
<td>&lt; 15 m/s</td>
</tr>
<tr>
<td></td>
<td>&lt;= .75 m/s (ANSI)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Op Features</strong></td>
<td>Basic</td>
<td>Advanced</td>
<td>Hi-End Dispatch</td>
</tr>
<tr>
<td><strong>Motion Features</strong></td>
<td>Basic Perf.</td>
<td>Limited Perf.</td>
<td>Advanced Perf.</td>
</tr>
<tr>
<td></td>
<td>Basic FM</td>
<td>Advanced FM</td>
<td>Advanced FM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Code</strong></td>
<td>EN, ANSI, JIS</td>
<td>EN, ANSI, JIS</td>
<td>EN, ANSI, JIS</td>
</tr>
<tr>
<td><strong>Remote Service</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Price Sensitivity</strong></td>
<td>High</td>
<td>High, Med</td>
<td>Med</td>
</tr>
<tr>
<td><strong>Market</strong></td>
<td>Utility</td>
<td>Utility, Design</td>
<td>Design</td>
</tr>
</tbody>
</table>
Embedded Systems

• Computational
  – but not first-and-foremost a computer

• Integral with physical processes
  – sensors, actuators

• Reactive
  – at the speed of the environment

• Heterogeneous
  – hardware/software, mixed architectures

• Networked
  – shared, adaptive

Source: Edward A. Lee
Common Situation in Industry

- Different hardware devices and architectures
- Increased complexity
- Non-standard tools and design processes
- Redundant development efforts
- Increased R&D and sustaining costs
- Lack of standardization results in greater quality risks
- Customer confusion
Outline for the Introduction

• Examples of Embedded Systems
• Their Impact on Society
• Design Challenges
• Embedded Software and Control
The Killer Applications?

- Energy Conservation
- Emergency Response and Homeland Defense
- Transportation Efficiency
- Monitoring Health Care
- Land and Environment
- Education
Smart Dust

Passive CCR comm.
MEMS/polysilicon

Active beam steering laser comm.
MEMS/optical quality polysilicon

Laser diode
III-V process

Sensor
MEMS/bulk, surface, ...

Analog I/O, DSP, Control
COTS CMOS

Power capacitor
Multi-layer ceramic

Solar cell
CMOS or III-V

Thick film battery
Sol/gel V_2O_5

Source: K. Pister, Berkeley
Energy Scavenging: Vibration

Source: P. Wright, Berkeley
Wireless Sensor Networks


Berkeley Dust Mote\textsuperscript{1}  
Berkeley Mote\textsuperscript{1}

\textsuperscript{1}From Pister et al., Berkeley Smart Dust Project
Applications

Distributed Bio-monitoring

- Wristband bio-monitors for chronic illness and the elderly
- Monitored remotely 24x7x365
- Emergency response and potential remote drug delivery
- Cardiac Arrest
  - Raise out-of-hospital survival rate from 6% to 20% => save 60K lives/year
Silicon-Processed Micro-needles

- Neural probe with fluid channel for bio-medical appl.
- Two micro-needles penetrating porterhouse (New-York)

Applications

Saving Energy

- Smart Buildings that adjust to inhabitants
- Make energy deregulation work via real-time metering and pricing
- Large potential savings in energy costs: for US commercial buildings
  - Turning down heat, lights saves up to $55B/year, 35M tons CO₂ emission/year
  - 30% of energy bill is from “broken systems”
Smart Buildings

Dense wireless network of sensor, monitor, and actuator nodes

• Disaster mitigation, traffic management and control
• Integrated patient monitoring, diagnostics, and drug administration
• Automated manufacturing and intelligent assembly
• Toys, Interactive Musea
• Other functions: security, identification and personalization, object tagging, seismic monitoring
Additional Applications

• Environmental Monitoring
  – Monitor air quality near highways to meet Federal Guidelines
  – Mutual impact of urban and agricultural areas
  – Monitor water shed response to climate events and land use changes
Applications

Disaster Mitigation (natural and otherwise)

- Monitor buildings, bridges, lifeline systems to assess damage after disaster
- Provide efficient, personalized responses
- Must function at maximum performance under very difficult circumstances
What is Disaster Response?

- Sensors installed near critical structural points
- Sensor measure motion, distinguish normal deterioration and serious damage
- Sensors report location, kinematics of damage during and after an extreme event
  - Guide emergency personnel
  - Assess structural safety without deconstructing building
Seismic Monitoring of Buildings: Before CITRIS

$8,000 each
Seismic Monitoring of Buildings: With CITRIS Wireless Motes

$70 each
Stability of Masada North Face:
The Foundations of King Herod’s Palace
Additional Applications

• Transportation Systems
  – Use SISs to improve the efficiency and utility of highways while reducing pollution
  – Improve carpooling efficiency using advanced scheduling
  – Improve freeway utilization by managing traffic flows
  – Large potential savings in commuter time, lost wages, fuel, pollution: for CA
    – 15 minutes/commuter/day => $15B/year in wages
    – $600M/year in trucking costs, 150K gallons of fuel/day

• Distributed Education
  – Smart Classrooms
  – Lifelong Learning Center for professional education
Discussion

• What are the most challenging aspects of these applications (and how does a company make money)?
  – Interaction mechanisms: sensors, actuators, wireless networks
  – Reliability and survivability
  – Infrastructure
  – Services
  – Legislation
  – ……
Picoradio Sensor Networks (BWRC)

- Control Environmental parameters (temperature, humidity…)
- Minimize Power consumption
- Cheap (<0.5$) and small ( < 1 cm³)
- Large numbers of nodes — between 0.05 and 1 nodes/m²
- Limited operation range of network — maximum 50-100 m
- Low data rates per node — 1-10 bits/sec average
- Low mobility (at least 90% of the nodes stationary)

• Key challenges
  - Satisfy tight performance and cost constraints (especially power consumption)
  - Identify Layers of Abstraction (Protocol Stack)
  - Develop distributed algorithms (e.g. locationing, routing) for ubiquitous computing applications
  - Design Embedded System Platform to implement Protocol Stack efficiently
Critical Infrastructures

- Government Operations
- Emergency Services
- Telecommunications
- Electrical Energy
- Transportation
- Gas & Oil Storage and Delivery
- Water Supply Systems
- Banking & Finance
Technology Generations of Information Assurance

1st Generation
(Prevent Intrusions)

Intrusions will Occur

2nd Generation
(Detect Intrusions, Limit Damage)

Some Attacks will Succeed

3rd Generation
(Operate Through Attacks)
Secure Network Embedded SystEms (SENSE)

- Networked embedded systems and distributed control creates a new generation of future applications: new infrastructures
- We need to think about how to prevent the introduction of vulnerabilities via this exciting technology
- Security, Networking, Embedded Systems
Outline for the Introduction

• Examples of Embedded Systems
• Their Impact on Society
• Design Challenges
• Embedded Software and Control
Opportunity: Electronic Systems Design Chain

Design Science

- System Design
- Implementation
- Manufacturing
- Interfaces
- IP
- Fabrics
Disaggregation:
Complex Design Chain Management

Supply Chain
• Movement of tangible goods from sources to end market
• Supply Chain Management is $3.8B market projected to be $20B in 2005

Design Chain
• Movement of technology (IP and knowledge) from sources to end market
• Design Chain Management is an untapped market
Supply Chain:
Design Roles -> Methodology -> Tools
Automotive Supply Chain: Car Manufacturers

- Product Specification & Architecture Definition (e.g., determination of Protocols and Communication standards)
- System Partitioning and Subsystem Specification
- Critical Software Development
- System Integration
Today, more than 80 Microprocessors and millions of lines of code
Automotive Supply Chain: Tier 1 Subsystem Providers

- Subsystem Partitioning
- Subsystem Integration
- Software Design: Control Algorithms, Data Processing
- Physical Implementation and Production

1. Transmission ECU
2. Actuation group
3. Engine ECU
4. DBW
5. Active shift display
6/7. Up/Down buttons
8. City mode button
9. Up/Down lever
10. Accelerator pedal position sensor
11. Brake switch
Automotive Supply Chain: Subsystem Providers

Application Platform layer
(\(\approx 10\%\) of total SW)

SW Platform layer
(> 60\% of total SW)

HW layer

Platform Integration
Software Design

“firmware” and “glue software”
“Application”
Automotive Supply Chain: Platform & IP Providers

- **Application Platform layer (≅ 10% of total SW)**
- **SW Platform layer (> 60% of total SW)**
- **HW layer**

- **“Software” platform**
  - RTOS and communication layer
  - Hardware and IO drivers

- **“Hardware” platform**
  - OSEK
  - RTOS
  - OSEK COM
  - OSEK
  - COM
  - KWP 2000
  - Transport

- **Application Programming Interface**
  - I/O drivers & handlers
  - 20 configurable modules

- **μControllers Library**
  - Nec78k
  - HC08
  - HC12
  - H8S26
  - MB90

- **Application Libraries**
  - Customer Libraries
  - Application Specific Software
  - Odometer
  - Tachometer
  - Speedometer
  - Water temp.

- **Application Specific Software**
  - Speedometer
  - Tachometer

- **Sys. Config. Boot Loader**

- **WindRiver**

- **WindRiver**

- **WindRiver**

- **WindRiver**
Outline for the Introduction

• Examples of Embedded Systems
• Their Impact on Society
• Design Challenges
• Embedded Software and Control
How Safe is Our Real-Time Software?
Computing for Embedded Systems
Mars, December 3, 1999
Crashed due to un-initialized variable
$4 billion development effort
40-50% system integration & validation cost
## Complexity, Quality, & Time To Market today

<table>
<thead>
<tr>
<th></th>
<th>PWT UNIT</th>
<th>BODY GATEWAY</th>
<th>INSTRUMENT CLUSTER</th>
<th>TELEMATIC UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>256 Kb</td>
<td>128 Kb</td>
<td>184 Kb</td>
<td>8 Mb</td>
</tr>
<tr>
<td>Lines Of Code</td>
<td>50,000</td>
<td>30,000</td>
<td>45,000</td>
<td>300,000</td>
</tr>
<tr>
<td>Productivity</td>
<td>6 Lines/Day</td>
<td>10 Lines/Day</td>
<td>6 Lines/Day</td>
<td>10 Lines/Day*</td>
</tr>
<tr>
<td>Residual Defect Rate @ End Of Dev</td>
<td>3000 Ppm</td>
<td>2500 ppm</td>
<td>2000 ppm</td>
<td>1000 ppm</td>
</tr>
<tr>
<td>Changing Rate</td>
<td>3 Years</td>
<td>2 Years</td>
<td>1 Year</td>
<td>&lt; 1 Year</td>
</tr>
<tr>
<td>Dev. Effort</td>
<td>40 Man-yr</td>
<td>12 Man-yr</td>
<td>30 Man-yr</td>
<td>200 Man-yr</td>
</tr>
<tr>
<td>Validation Time</td>
<td>5 Months</td>
<td>1 Month</td>
<td>2 Months</td>
<td>2 Months</td>
</tr>
<tr>
<td>Time To Market</td>
<td>24 Months</td>
<td>18 Months</td>
<td>12 Months</td>
<td>&lt; 12 Months</td>
</tr>
</tbody>
</table>

* C++ CODE

FABIO ROMEO, Magneti-Marelli
DAC, Las Vegas, June 20th, 2001
What About Real Time?

“Make it faster!”
Poor common infrastructure. Weak specialization of functions. Poor resource management. Poor planning.
Design “Practice”
Design Science:
Build upon Solid Foundations
Software Architecture Tomorrow?
The Goal (CHESS Project)

• To create a modern computational systems science and systems design practice with
  - Concurrency
  - Composability
  - Time
  - Hierarchy
  - Heterogeneity
  - Resource constraints
  - Verifiability
  - Understandability
A Traditional Systems Science – Feedback Control Systems

- Models of continuous-time dynamics
- Stability analysis
- But not accurate for software controllers
Discretized Model – A Step Towards Software

• Numerical integration techniques provided ways to get from the continuous idealizations to computable algorithms.
• Discrete-time signal processing techniques offer the same sophisticated stability analysis as continuous-time methods.
• But it’s still not accurate for software controllers

In general, $z$ is an $N$-tuple, $z = (z_1, \cdots, z_N)$, where $z_t: \text{Reals}_+ \rightarrow \text{Reals}$. The derivative of an $N$-tuple is simply the $N$-tuple of derivatives, $\dot{z} = (\dot{z}_1, \cdots, \dot{z}_N)$. We know from calculus that

$$\dot{z}(t) = \frac{dz}{dt} = \lim_{\delta \to 0} \frac{z(t+\delta) - z(t)}{\delta},$$

and so, if $\delta > 0$ is a small number, we can approximate this derivative by

$$\dot{z}(t) \approx \frac{z(t+\delta) - z(t)}{\delta}.$$

Using this for the derivative in the left-hand side of (5.50) we get

$$z(t+\delta) - z(t) = \delta g(z(t), v(t)). \quad (5.51)$$
Hybrid Systems – Reconciliation of Continuous & Discrete

- But it’s **still** not accurate for software controllers

---

This model gives two separate ordinary differential equations, one for each point mass attached to a spring. The ZeroCrossingDetector actor detects the collision of the point masses and emits the "touched" event.

V1 and V2 are velocities, and P1 and P2 are positions of the two masses.
Timing in Software is More Complex Than What the Theory Deals With

An example (Jie Liu) models two controllers sharing a CPU under an RTOS. Under preemptive multitasking, only one can be made stable (depending on the relative priorities). Under non-preemptive multitasking, both can be made stable.

Where is the theory for this?

This model shows two (independent) control loops whose controllers share the same CPU. The control loops are chosen such that it is unstable if the control signals are constantly delayed. By choosing different priority assignments and TM scheduling policies, different stability of the two loops may appear. For example, a nonpreemptive scheduling can stabilize both control loops, but none of the preemptive ones can.
The science of computation has systematically abstracted away the physical world. The science of physical systems has systematically ignored computational limitations. Embedded software systems, however, engage the physical world in a computational manner.

It is time to construct a Hybrid Systems Science that is simultaneously computational and physical. Time, concurrency, robustness, continuums, and resource management must be remarried to computation.