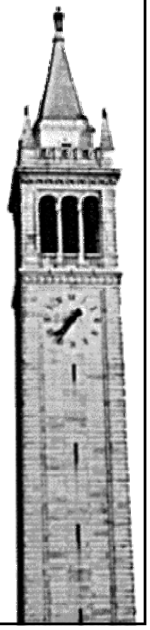
Course Development

Edited and Presented by
Edward A. Lee, Co-PI
UC Berkeley

Chess Review
November 18, 2004
Berkeley, CA



Recall the Goal

To create a integrated computational systems theory and systems design practice with

- Concurrency
- Composability
- Time
- Hierarchy
- Heterogeneity
- Resource constraints
- Verifiability
- Understandability

This systems theory must be at once computational and physical.

The Problem (1)

Models for the physical world and for computation diverge.

- physical: time continuum, ODEs, dynamics
- computational: a "procedural epistemology," logic

There is a huge cultural gap.

Physical system models must be viewed as semantic frameworks, and theories of computation must be viewed as alternative ways of talking about dynamics.

The Problem (2)

Students are taught to *use* modeling techniques, not to *evaluate* modeling techniques.

- "this is how computers work"
- "this equation describes that feedback circuit"

rather than

- "this is how VonNeumann proposed that we control automatic machines"
- "ignoring the intrinsic randomness and latency in this circuit, Black proposed that we could idealize its behavior in this way"

Students must be taught meta-modeling, not just modeling. They must learn to think critically about modeling methods, not just about models.

The Graduate Curriculum

We have developed courses in:

- Hybrid Systems
- Embedded Software and Systems
- Model Integrated Computing

A key challenge is to define the durable intellectual heart of hybrid and embedded systems.

Focus first on embedded software…

A Tempting Incremental Approach to Teaching Embedded Software

Orient courses around resource limitations:

- small memory
- small data word sizes
- relatively slow clocks

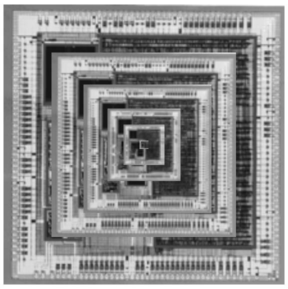
To deal with these problems:

- write software at a low level (in assembly code or C)
- teach microcomputer interfacing (DMA, interrupts)
- teach how to use RTOSs (priorities, preemption)
- discuss specialized computer architectures
  - programmable DSPs
  - network processors

This is not what we are doing (mostly)!

If Resource Limitations are the Key Defining Property of Embedded Systems,

Why hasn't Moore's law changed all this in 25 years?



Embedded Software Differs More Fundamentally from General Purpose SW

Hints that this is true:

- object-oriented techniques are rarely used
  - classes and inheritance
  - dynamic binding
- processors avoid memory hierarchy
  - virtual memory
  - dynamically managed caches
- memory management is avoided
  - allocation/de-allocation
  - garbage collection

To be fair, there are some applications: e.g. Java in cell phones, but mainly providing the services akin to general purpose software.

Fundamentally Different Techniques Are Being Applied to Embedded SW.

method-call models of computation

- nesC/TinyOS
  - developed for programming very small programmable sensor nodes called "motes"
- Click
  - created to support the design of software-based network routers

actor-oriented models of computation

- Simulink with Real-Time Workshop
  - created for embedded control software and widely used in the automotive industry
- Lustre/SCADE
  - created for safety-critical embedded software (e.g. avionics software)

Graduate Embedded Software Classes at Berkeley

- **EECS 249**
  Design of Embedded Systems: Models, Validation, and Synthesis
  This course is about the design of embedded real-time systems. It presents the principles of a methodology that favors design re-use, formal verification, software design and optimized architecture selection. The basic tenet of the methodology is separation of function and architecture, computation and communication. This methodology called platform-based design is presented as a paradigm that incorporates these principles and spans the entire design process, from system-level specification to detailed circuit implementation.

- **EECS 290O**
  Embedded Software Engineering
  Introduction to embedded software engineering: the first part covers real-time operating systems, real-time communication protocols, and scheduling theory; the second part focuses on real-time programming and code generation. Real-time communication protocols like the time-triggered protocol (TTP) and the event-triggered CAN protocol are discussed. Scheduling techniques like rate-monotonic and earliest deadline first are illustrated. The second half of the course emphasizes real-time programming and code generation for embedded systems focusing on the languages Esterel, Lustre, and Giotto.

- **EECS 290N**
  Concurrent Models of Computation for Embedded Software
  This course is about the foundations of specification and modeling of concurrent real-time systems. The course combines an experimental approach with a study of formal semantics. The objective is to develop a deep understanding of the wealth of alternative approaches to managing concurrency and time in software.

Complementary courses with slants towards practice & theory.

EECS 249: Design of Embedded Systems: Models, Validation, and Synthesis

Taught regularly at Berkeley, this project course introduces graduate students to modeling and design for embedded systems.

Lectures:
- Introduction
- Platform-Based Design
- Models of Computation
- MoCs: FSMs, CFSMs
- MoCs: KPN and Dataflow
- The Tagged Signal Model
- MoCs: Petri Nets
- MoCs: Synchronous Languages
- Heterogeneous Composition: Hybrid Systems
- SystemC
- Architecture Modeling
- Microprocessors Modeling
- Mapping of Function to Architecture
- Software Estimation
- Software Synthesis
- Scheduling and RTOS
- …..

EECS 290N: *Concurrent Models of Computation for Embedded Software*

Taught at Berkeley experimentally under the heading of "Advanced Topics in Systems Theory," this course is a starting point for a textbook and (we hope) a regular mainstream graduate class.

Lectures:
- Introduction
- Current Trends in Embedded Software
- Threads
- Actor-Oriented Models of Computation
- Process Networks
- Extending Ptolemy II
- Process Networks Semantics
- Continuous Functions and PN Compositions
- Execution of Process Networks
- Introduction to Synchronous Languages
- Synchronous Language Semantics
- Discrete-Event Systems
- Tags and Discrete Signals
- Metric Space Semantics
- Dataflow Process Networks
- Generalized Firing Rules
- The Cal Actor Language,
- Statically Schedulable Dataflow
- Extensions to SSDF
- Boolean Dataflow
- Scheduling Boolean Dataflow
- …

Outreach: Graduate Class in Civil and Environmental Engineering (CEE)

CE 290I … teaches students models of computation in the context of a suite of civil engineering case studies. The idea is to approach a control or information management problem by first choosing an appropriate model of computation, then choosing an appropriate high-level embedded programming tool, and finally using the tool to create software that is transparent, explainable, or verifiable.

Graduate students from structural engineering, construction, and transportation have taken the course, as have structural engineers from the NSF sponsored National Earthquake Engineering Simulation (NEES) program. Other students taking the class have been working on problems like deploying sensor networks to monitor the wind response of the Golden Gate Bridge, to measure seismic waves using down-hole sensor arrays, to control illumination in buildings, and to coordinate traffic signals for smoothing freeway traffic flow. The class has also drawn students from ME, EECS, and IEOR.

EE290o: Model-Integrated Computing[1.]

Adoption of the Vanderbilt course (see below) at Berkeley. This course teaches the purpose, design and implementation of domain-specific modeling environments through formal modeling and metamodeling.

Lectures:
- Introduction
- MIC Overview – Terms and Definitions
- MIC Overview – Concepts
- Language – Syntax and Semantics (2x)
- Language Representation
- Structural Modeling
- Structural/Behavioral Modeling
- Behavioral Modeling
- Abstraction and Encoding (2x)
- MGA Modeling Tool Suite & Resources
- MGA Modeling Resources
- Metamodeling
- Metamodeling Example
- Attaching and Defining Semantics
- Interpreter Writing – Class Library
- Interpreter Writing – Example
- Designing DSMEs (3x)
- Constraints
- Real World MIC Examples
- …

1. Presented as a poster by Jonathan Sprinkle

EECS 291E: Hybrid Systems

- Graduate Class in Hybrid Systems at UCB (also taught by Koo at Vanderbilt)
- Textbook "The Art of Hybrid Systems" Lygeros, Tomlin and Sastry, in preparation

Topics:
- Hybrid Systems Examples: automotive, aerospace,
- Hybrid Systems Definition, Existence of Solutions, Zeno Behavior
- Controller Design for Timed Automata
- Bisimulations, Decidability, Computability issues
- Controller Synthesis for Hybrid Systems
- Simulation Tools, Verification Tools
- Stochastic Hybrid Systems, Applications to Systems Biology, Networked Embedded Systems
- Projects: applications, conceptual.
- …

http://www.eecs.berkeley.edu/~imitchel/EE291E/

Next Steps at Berkeley

- Systematize the graduate curriculum in hybrid and embedded systems
  - Measure of success: Multiple faculty teaching the same course at different times.
- Integrate more broadly with other College of Engineering Courses
  - Hybrid and embedded software systems have natural applications in mechanical engineering, civil and environmental engineering, and bioengineering.
- Make hybrid and embedded systems a legitimate concentration, with its own prelim exam and graduate admissions category.
  - We have taken the first step: for 2005 admissions, it is an admissions category at Berkeley.

Vanderbilt Graduate Curriculum

- **CS375**
  Discrete Event Systems: Supervisory Control and Diagnosis
  - Algebraic structures, automata and formal language theory, process modeling with finite state automata, supervisory control theory, controllability and supervision, supervisory control under partial observation, modular and hierarchical supervisory control, supervisory control of real-time systems, fault diagnosis of discrete event systems, and modular diagnosis approaches.
- **CS376**
  Foundations of Hybrid and Embedded Systems
  - Modeling, analysis, and design of hybrid and embedded systems. Heterogeneous modeling and design of embedded systems using formal models of computation, modeling and simulation of hybrid systems, properties of hybrid systems, analysis methods based on abstractions, reachability, and verification of hybrid systems.
- **CS388**
  Model Integrated Computing
  - Model-Integrated Computing (MIC) addresses the problems of designing, creating, and evolving embedded systems by providing rich, domain-specific modeling environments including model analysis and model-based program synthesis tools. Students are required to give a class presentation and prepare a project.
- **CS379**
  Topics in Embedded Software and Systems
  - Specification and composition of domain-specific modeling languages. Design methodologies for embedded systems. Platforms for embedded system design and implementation. Analysis of embedded systems.

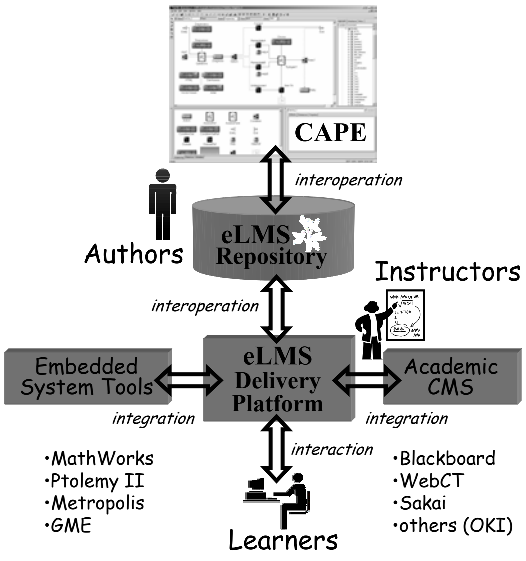
CS388 Model-Integrated Computing

Taught regularly at Vanderbilt as a general introduction course for modeling languages, model analysis and model based code generation.

Lectures:
- Introduction
- Design Approaches in Embedded Systems
- Modeling language examples
- Specification of DSMLs: Concreate Syntax, Abstract Syntax and Semantics
- Metamodeling approaches and languages
- Design of DSML-s for MoC-s: FSMs, HFSMs, and Dataflow
- Model transformation and code generation
- Writing model-based code generators using procedural languages
- Modeling code generators using graph transformation
- Projects

Integrating Learning into Technology-Based Engineering Environments

CAPE authoring technology (based on GME, same modeling infrastructure used for embedded system tools) plus eLMS content repository and delivery platform support collaborative development and use of adaptive web-based learning experiences.



Now support integrating learning resources into embedded systems tools and using embedded system tools within adaptive courseware.

eLMS supports integration with academic course management systems.

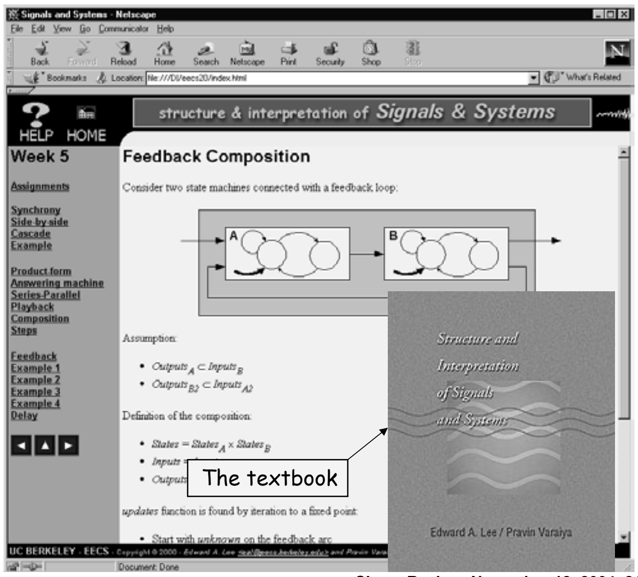
Undergraduate Curriculum at Berkeley

- Introductory course (EECS 20n) is mature and established.
- Department is hiring a lecturer to assume principal responsibility for this course.
- The course is starting to appear in outreach institutions (Cal State Hayward, for example).
- Next step is to develop the follow-up courses.

Mature Introductory Course on Computational Signals and Systems

Berkeley has a required sophomore course for EE and CS students that addresses mathematical modeling of signals and systems from a computational perspective.

A web page shows a broad view of feedback, where the behavior is a fixed point solution to a set of equations. This view covers feedback control systems, discrete-event systems, and hybrid systems.


Signals and Systems - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: file:///D|/eecs20/index.html
What's Related
HELP HOME
structure & interpretation of Signals & Systems
A
B
Structure and Interpretation of Signals and Systems
Edward A. Lee / Pravin Varaiya
The textbook
Document: Done

Week 5

Assignments

Synchrony
Side-by-side
Cascade
Example

Product form
Answering machine
Series-Parallel
Playback
Composition
Steps

Feedback
Example 1
Example 2
Example 3
Example 4
Delay

Feedback Composition

Consider two state machines connected with a feedback loop:

Assumption:

- $Outputs_A \subset Inputs_B$
- $Outputs_{B2} \subset Inputs_{A2}$

Definition of the composition:

- $States = States_A \times States_B$
- $Inputs$ =
- $Outputs$

updates function is found by iteration to a fixed point:

- Start with *unknown* on the feedback arc

UC BERKELEY - EECS - Copyright © 2000 - Edward A. Lee and Pravin Varaiya

Proposal for 12x courses in systems

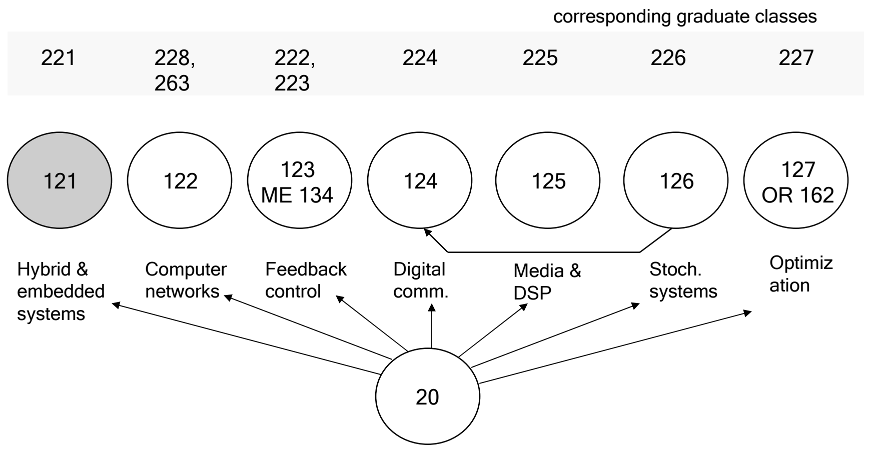
EECS 12x courses are designed to follow EECS 20:

- control systems
- communications
- embedded systems
- optimization
- signal processing
- stochastic systems

Needs to be developed

These can benefit from the hybrid physical/computational viewpoint.

Proposed prerequisite structure



Vanderbilt Undergraduate Curriculum

Development of an Embedded Systems concentration for EE, CS, CE, ME and CEE is in progress.

Courses that have been developed and offered:

- **EECE276**
  Microcontrollers 2 (to be renamed to Embedded Systems)
  - This course is about the design and implementation of embedded systems. The course covers basics of concurrent and real-time programming, real-time languages, the basics of model-based design of systems using functional and object-oriented techniques. Students are expected to participate in a team projects that constructs a small embedded system application.
- **CS274**
  Modeling and Simulation
  - General theory of modeling and simulation of a variety of systems: physical processes, computer systems, biological systems, and manufacturing processes. Principles of discrete-event, continuous, and hybrid system modeling, simulation algorithms for the different modeling paradigms, methodologies for constructing models of a number of realistic systems, and analysis of system behavior. Computational issues in modeling and analysis of systems. Stochastic simulations.
- **EECE296**
  Electrical and Computer Engineering Design
  - Based on product specifications typically supplied by industrial sponsors, teams of students responsible for the formulation, execution, qualification, and documentation of a culminating systems engineering design. The application of knowledge acquired from earlier coursework, both within and outside the major area, along with realistic technical, managerial, and budgetary constraints using standard systems engineering methodologies and practices.

Outreach

Tennessee Technological University

As part of the SIPHER Program, Profs. Ramaswamy and Mahmoud from TTU have developed a new course:

CSC/ECE 4990 Modeling and Simulation of Embedded Systems

Conclusion

- We still have a lot of work to do…