Model-Based Approaches to Embedded Software Design

Edward A. Lee
UC Berkeley & GSRC

SRC ETAB Summer Study
Colorado Springs, June 25-26, 2001

Why is Embedded Software an Issue for Semiconductor Manufacturers?

- Silicon without software is getting rarer.
- Time-to-volume is often dominated by SW development.
- Software requirements affect hardware design.
- Embedded SW design is getting harder (networking, complexity).
- Mainstream SW engineering is not addressing embedded SW well.

prime example today

Why is Embedded SW not just Software on Small Computers?

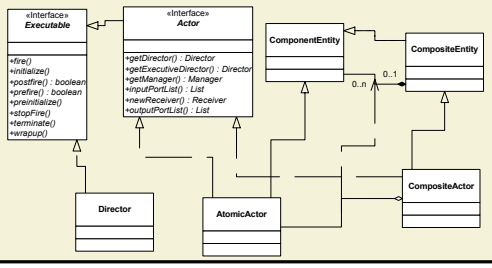
- Interaction with physical processes
  - sensors, actuators, processes
- Critical properties are not all functional
  - real-time, fault recovery, power, security, robustness
- Heterogeneous
  - hardware/software, mixed architectures
- Concurrent
  - interaction with multiple processes
- Reactive
  - operating at the speed of the environment

These feature look more like hardware!

Why not Leave This Problem to the Software Experts?

E.g. Object-Oriented Design

- Call/return imperative semantics
- Concurrency is via ad-hoc calling conventions
  - band-aids: futures, proxies, monitors
- Poorly models the environment
  - which does not have call/return semantics
- Little to say about time



Object modeling emphasizes inheritance and procedural interfaces.

We need to emphasize concurrency, communication, and temporal abstractions.

Why not Leave This Problem to the Software Experts (cont)?

E.g. Real-Time Corba

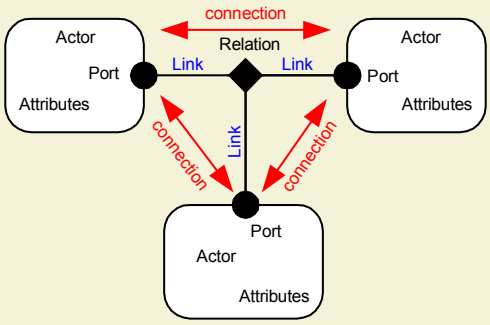
- Component specification includes:
  - worst case execution time
  - typical execution time
  - cached execution time
  - priority
  - frequency
  - importance

This is an elaborate prayer…

Hardware Experts Have Something to Teach to the Software World

- **Concurrency**
  - the synchrony abstraction
  - event-driven modeling
- **Reusability**
  - cell libraries
  - interface definition
- **Reliability**
  - leveraging limited abstractions
  - leveraging verification
- **Heterogeneity**
  - mixing synchronous and asynchronous designs
  - resource management

Alternative View of SW Architecture: *Actors* with *Ports* and *Attributes*



Model of Computation:

- Messaging schema
- Flow of control
- Concurrency

Examples:

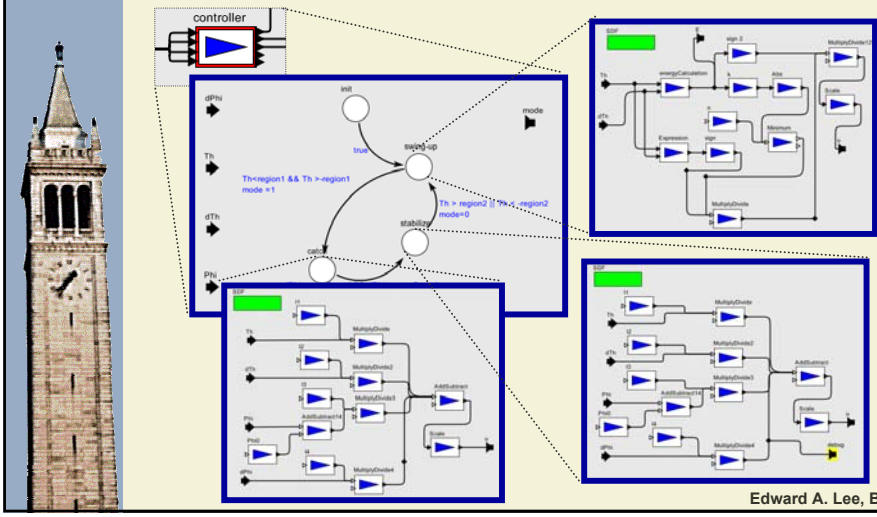
- Synchronous circuits
- Time triggered
- Process networks
- Discrete-event systems
- Dataflow systems
- Publish & subscribe

Key idea: The model of computation is part of the framework within which components are embedded rather than part of the components themselves.

Examples of Actors+Ports Software Architectures

- VHDL, Verilog, SystemC (Various)
- Simulink (The MathWorks)
- Labview (National Instruments)
- OCP, open control platform (Boeing)
- SPW, signal processing worksystem (Cadence)
- System studio (Synopsys)
- ROOM, real-time object-oriented modeling (Rational)
- Port-based objects (U of Maryland)
- I/O automata (MIT)
- Polis & Metropolis (UC Berkeley)
- Ptolemy & Ptolemy II (UC Berkeley)
- …

What an Embedded Program Might Look Like



Simple Example: Controlling an Inverted Pendulum with Embedded SW

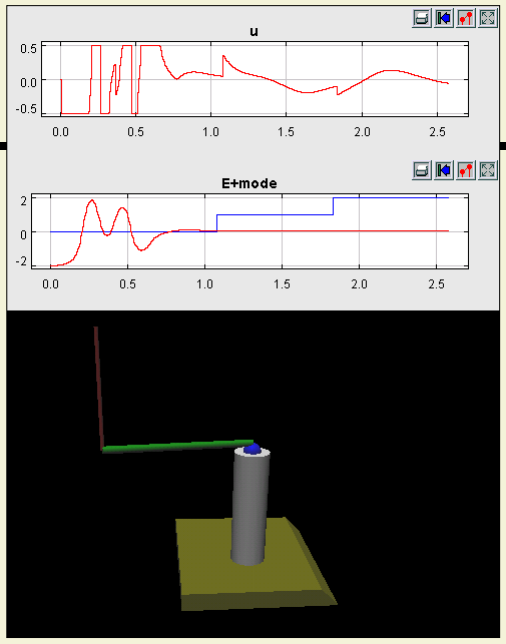
The Furuta pendulum has a motor controlling the angle of an arm, from which a free-swinging pendulum hangs. The objective is to swing the pendulum up and then balance it.

Metaphor for

- Disk drive controllers
- Manufacturing equipment
- Automotive:
  - Drive-by-wire devices
  - Engine control
  - Antilock braking systems, traction control
- Avionics
  - Fly-by-wire devices
  - Navigation
  - flight control
- Certain “software radio” functions
- Printing and paper handling
- Signal processing (audio, video, radio)
- …

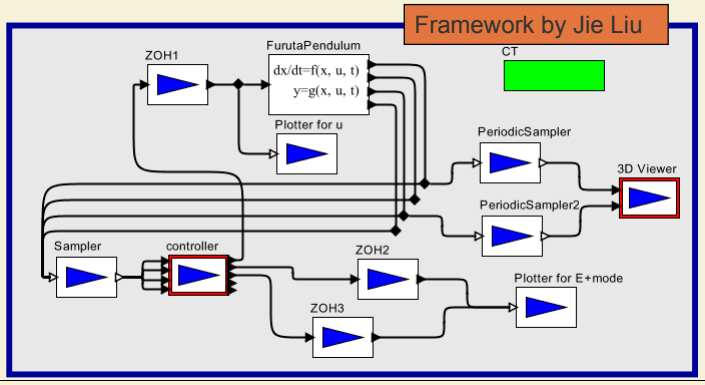
Execution

An execution of the model displays various signals and at the bottom produces a 3-D animation of the physical system.



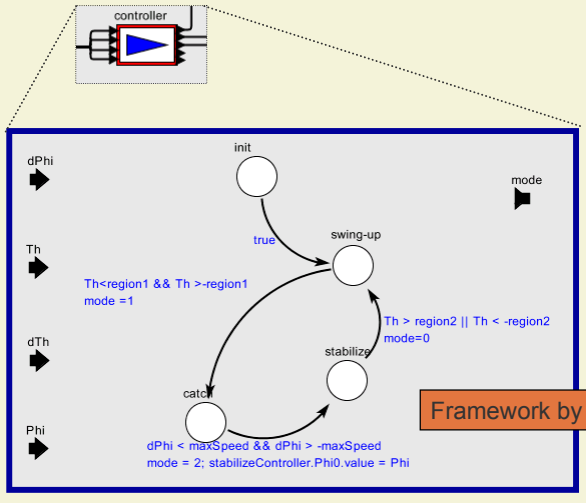
Model by Johan Eker

Top-Level Model



The top-level is a continuous-time model that specifies the dynamics of the physical system as a set of nonlinear ordinary differential equations, and encapsulates a closed loop controller.

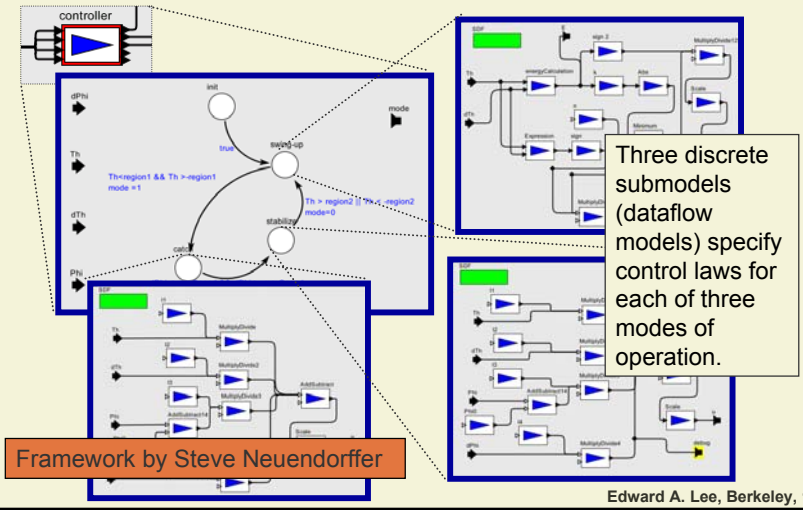
A Modal Controller



The controller itself is modal, with three modes of operation, where a different control law is specified for each mode.

Framework by Xiaojun Liu

The Discrete Controllers



Framework by Steve Neuendorffer

This is System-Level Modeling

- **SRC funding in system-level modeling, simulation, and design work 5-10 years ago has had demonstrable impact via:**
  - SystemC
  - VSIA standards efforts
  - Cadence SPW & VSS
  - Synopsys Cocentric Studio
  - Agilent ADS (RF + DSP)
  - …
- **Much of this work is now starting to address embedded software issues.**

The Key Idea

- Components are actors with ports
- Interaction is governed by a model of computation
  - flow of control
  - messaging protocols
  - non-functional properties (timing, resource management, …)

So what is a model of computation?

- It is the "laws of physics" governing the interaction between components
- It is the modeling paradigm

Model of Computation

- What is a component? (ontology)
  - States? Processes? Threads? Differential equations? Constraints? Objects (data + methods)?
- What knowledge do components share? (epistemology)
  - Time? Name spaces? Signals? State?
- How do components communicate? (protocols)
  - Rendezvous? Message passing? Continuous-time signals? Streams? Method calls? Events in time?
- What do components communicate? (lexicon)
  - Objects? Transfer of control? Data structures? ASCII text?

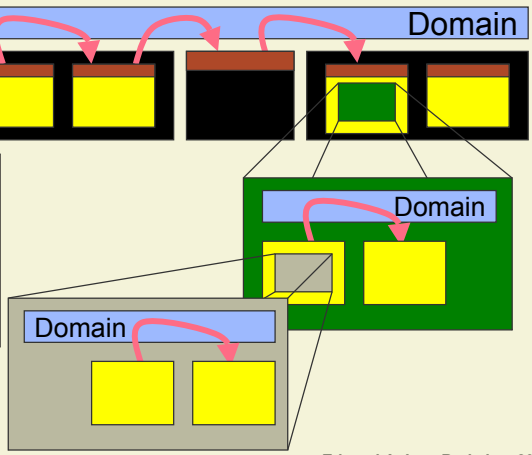
Domains – Realizations of Models of Computation

- CSP – concurrent threads with rendezvous
- CT – continuous-time modeling
- DE – discrete-event systems
- DDE – distributed discrete-event systems
- DT – discrete time (cycle driven)
- FSM – finite state machines
- Giotto – time driven cyclic models
- GR – graphics
- PN – process networks
- SDF – synchronous dataflow
- *x*DF – other dataflow

Each of these defines a component ontology and an interaction semantics between components. There are many more possibilities!

Hierarchical, Compositional Models

Actors with ports **are better than** ***objects with methods*** **for embedded system design.**



Heterogeneity – Hierarchical Mixtures of Models of Computation

- Modal Models
  - FSM + anything
- Hybrid systems
  - FSM + CT
- Mixed-signal systems
  - DE + CT
  - DT + CT
- Complex systems
  - Resource management
  - Signal processing
  - Real time

Key Advantages

- Domains are specialized
  - lean
  - targeted
  - optimizable
  - understandable
- Domains are mixable (hierarchically)
  - structured
  - disciplined interaction
  - understandable interaction

Model = Design

- We need modeling "languages" for humans to
  - realize complex functionality
  - understand the design
  - formulate the questions
  - predict the behavior

  The issue is "model" or "design" not "hardware" or "software"

- Invest in:
  - modeling "languages" for *systems*
  - finding the useful abstractions
  - computational systems theory
  - composable abstractions
  - expressing time, concurrency, power, etc.

Composing Systems

- We need systematic methods for composing systems
  - component frameworks
  - composition semantics
  - on-the-fly composition, admission control
  - legacy component integration

- Invest in:
  - methods and tools
  - reference implementations
  - semantic frameworks and theories
  - defining architectural frameworks
  - strategies for distribution, partitioning
  - strategies for controlling granularity and modularity

Transformations

- We need theory of transformations between abstractions
  - relationships between abstractions
  - generators (transformers, synthesis tools)
  - multi-view abstractions
  - model abstractors (create reduced-order models)
  - abstractions of physical environments
  - verifiable transformations
- Invest in:
  - open generator infrastructure (methods, libraries)
  - theories of generators
  - methods for correct by construction transformers
  - co-compilation

Conclusions

- Semiconductor manufacturers should not ignore embedded software.
- Software experts are unlikely to solve the embedded software problem on their own.
- *Actors* with *ports* are better than *objects* with *methods* for embedded system design.
- Well-founded models of computation matter a great deal, and specialization can help.