The Ptolemy Project

**Modeling and Design of Reactive Systems**

**Presenter:**
Praveen Murthy, PhD
Postdoc and Presenter

Edward A. Lee
Professor and PI
UC Berkeley
Dept. of EECS

---

**Abstract**

Ptolemy is a research project and software environment focused on the design and modeling of reactive systems, providing high-level support for signal processing, communication, and real-time control. The key underlying principle in the project is the use of multiple models of computation in a hierarchical heterogeneous design and modeling environment. This talk gives an overview of some of the models of computation of interest, with a focus on their concurrency, their ability to model and specify real-time systems, and their ability to mix control logic with signal processing.

---

**Organizational**

**Staff**
- Diane Chang, administrative assistant
  - Kevin Chang, programmer
  - Christopher Hylands, programmer analyst
  - Edward A. Lee, professor and PI
  - Mary Stewart, programmer analyst

**Postdocs**
- Praveen Murthy
- Seehyun Kim
- John Reekie
- Dick Stevens (on leave from NRL)

**Students**
- Cliff Cordeiro
- John Davis
- Stephen Edwards
- Ron Galicia
- Mudit Goel
- Michael Goodwin
- Bilung Lee
- Jie Liu
- Michael C. Williamson
- Yuhong Xiong

**Undergraduate Students**
- Sunil Bhave
- Luis Gutierrez

**Key Outside Collaborators**
- Shuvra Bhattacharyya (Hitachi)
- Joseph T. Buck (Synopsys)
- Brian L. Evans (UT Austin)
- Soo Shin (Seoul N. Univ.)
- Tom Lane (SSS)
- Thomas M. Parks (Lincoln Labs)
- José Luis Pino (Hewlett Packard)

**Sponsors**
- DARPA
- MICRO
- The Alta Group of Cadence
- Hewlett Packard
- Hitachi
- Hughes
- LG Electronics
- NEC
- Philips
- Rockwell
- SRC

---

**Types of Computational Systems**

**Transformational**
- transform a body of input data into a body of output data

**Interactive**
- interact with the environment at their own speed

**Reactive**
- react continuously at the speed of the environment

*This project focuses on design of reactive systems*
Interactive, High-Level Simulation and Specification

Properties of Such Specifications

- Modular
  - Large designs are composed of smaller designs
  - Modules encapsulate specialized expertise
- Hierarchical
  - Composite designs themselves become modules
  - Modules may be very complicated
- Concurrent
  - Modules logically operate simultaneously
  - Implementations may be sequential or parallel or distributed
- Abstract
  - The interaction of modules occurs within a “model of computation”
  - Many interesting and useful MoCs have emerged
- Domain Specific
  - Expertise encapsulated in MoCs and libraries of modules.

Heterogeneous Implementation Architectures

Two Approaches to the Design of Such Systems

- The grand-unified approach
  - Find a common representation language for all components
  - Develop techniques to synthesize diverse implementations from this
- The heterogeneous approach
  - Find domain-specific models of computation (MoC)
  - Hierarchically mix and match MoCs to define a system
  - Retargetable synthesis techniques from MoCs to diverse implementations

The Ptolemy project is pursuing the latter approach

- Domain specific MoCs match the applications better
- Choice of MoC can profoundly affect system architecture
- Choice of MoC can limit implementation options
- Synthesis from specialized MoCs is easier than from GULs.

Heterogeneity is a major source of complexity in such systems.
Some Problem-Level Models of Computation

- Gears
- Differential equations
- Difference equations
- Discrete-events
- Petri nets
- Dataflow
- Process networks
- Actors
- Threads
- Synchronous/reactive languages
- Communicating sequential processes
- Hierarchical communicating finite state machines

Example — Process Networks

Strengths:
- Good match for signal processing
- Loose synchronization (distributable)
- Determinate
- Maps easily to threads
- Dataflow special cases map well to hardware and embedded software

Weakness:
- Control-intensive systems are hard to specify

Sequential Example — Finite State Machines

Strengths:
- Natural description of sequential control
- Behavior is decidable
- Can be made determinate (often is not, however)
- Good match to hardware or software implementation

Weaknesses:
- Awkward to specify numeric computation
- Size of the state space can get large

Note: Dataflow is a special case.
Essential Differences — Models of Time

- Synchronous/reactive
- Continuous time
- Discrete time
- Multirate discrete time
- Partially-ordered discrete events
- Totally-ordered discrete events
- Synchronous/reactive

Key Issues in these Models of Computation

- Maintaining determinacy.
- Supporting nondeterminacy.
- Bounding the queueing on channels.
- Scheduling processes.
- Synthesis: mapping to hardware/software implementations.
- Providing scalable visual syntaxes.
- Resolving circular dependencies.
- Modeling causality.
- Achieving fast simulations.
- Supporting modularity.
- Composing multiple models of computation.

Choosing Models of Computation

**Validation methods**

- By construction
  - property is inherent.
- By verification
  - property is provable syntactically.
- By simulation
  - check behavior for all inputs.
- By testing
  - observation of a prototype.
- By intuition
  - property is true, I think.
- By assertion
  - property is true. That’s an order.

It is generally better to be higher in this list

Usefulness of Modeling Frameworks

The following objectives are at odds with one another:

- Expressiveness
- Generality

vs.

- Verifiability
- Compilability/Synthesizability

The Conclusion?
Heterogeneous modeling.
**A Mixed Design Flow**

- System-level modeling
- Cosimulation
- Symbolic
- Discrete event
- Imperative
- FSMs
- Dataflow
- Logic

**Synthesis**

- Compiler
- Software synthesis
- ASIC synthesis
- Logic synthesis

**Execution**

- Execution model
- Logic model
- ASIC model

**Contributions (contd.)**

- A synchronous-reactive modeling technique that is modular and can be combined with dataflow, finite-state machines, and discrete-event modeling.
- A hierarchical finite-state machine model of computation that can be combined with dataflow, discrete-event, and synchronous reactive modeling.
- A mathematical semantic framework for comparing models of computation, and analysis within this framework of the discrete-event semantics of VHDL and the formal semantics of dataflow.
- Public distribution of three major versions of the Ptolemy software and two versions of the Tycho user-interface framework. This software serves as our laboratory and as a major vehicle for technology transfer.

**Major Contributions under RASSP**

- Static scheduling of synchronous dataflow (SDF) graphs for optimum memory utilization, for partitioning into mixed hardware/software implementations, and for synthesis of VHDL.
- Mixed modeling and design of hardware, embedded software, and the test environment.
- Integrated symbolic processing with numeric and demonstrated heterogeneous design tools that leverage commercial tools such as Matlab, Mathematica, and VHDL simulators.
- Generalizations of dataflow to multidimensional streams and to process networks.
- Robust dynamic dataflow scheduling for bounded memory.
- Visual programming and use of higher-order functions.
- Optimized synchronization for multiprocessors.

**Mixing Control and Signal Processing — *Charts**

Choice of domain here determines concurrent semantics

Hierarchy is free
**Example: DE, Dataflow, and FSMs**

**Technology Transfer**

Our policy of free and open software distribution has proven to be a very effective facilitator for technology transfer.

- 1995 — The Alta Group at Cadence announces software using Ptolemy dataflow and mixed dataflow/discrete-event technology (SPW 3.5).
- 1995 — DQDT uses and extends Ptolemy VHDL generation for ASIC designs.
- 1995 — BDTI uses the Ptolemy kernel to integrate commercial tools (SPW and Bones from Alta).
- 1996 — Lockheed/Martin develops architectural tradeoff analysis tool based on Ptolemy.
- 1997 — Hewlett-Packard (EEsof) announces “HP Ptolemy,” an integration of Ptolemy dataflow technology with analog RF and microwave design and modeling tools.
- 1997 — BNED, Technologies Lyre, White Eagle Systems, ...

**Ptolemy Software as a Tool and as a Laboratory**

Ptolemy software is
- Extensible
- Publicly available
- An open architecture
- Object-oriented

Allows for experiments with:
- Models of computation
- Heterogeneous design
- Domain-specific tools
- Design methodology
- Software synthesis
- Hardware synthesis
- Cosimulation
- Cosynthesis
- Visual syntaxes (Tycho)

**Further Information**

- Software distributions
- Small demonstration versions
- Project overview
- The Almagest (software manual)
- Current projects summary
- Project publications
- Keyword searching
- Project participants
- Sponsors
- Copy of the FAQ
- Newsgroup info
- Mailing lists info

http://ptolemy.eecs.berkeley.edu