An Overview of the Ptolemy Project

Edward A. Lee
Professor and
Principal Investigator

UC Berkeley
Dept. of EECS

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
Raja Nagarajan
John Reekie
Dick Stevens (on leave from NRL)

Students
Sunil Bhave
Cliff Cordeiro
John Davis
Stephen Edwards
Ron Galicia
Mudit Goel
Michael Goodwin
Luis Gutierrez
Bilung Lee
Michael C. Williamson
Yuhong Xiong

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

Cooperating Industrial Sponsors
The Alta Group of Cadence
Dolby Laboratories
Hitachi
Lucky/Goldstar
Lockheed Martin ATL
Mitsubishi
Motorola
NEC
Philips
Rockwell
Semiconductor Research Corporation

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**

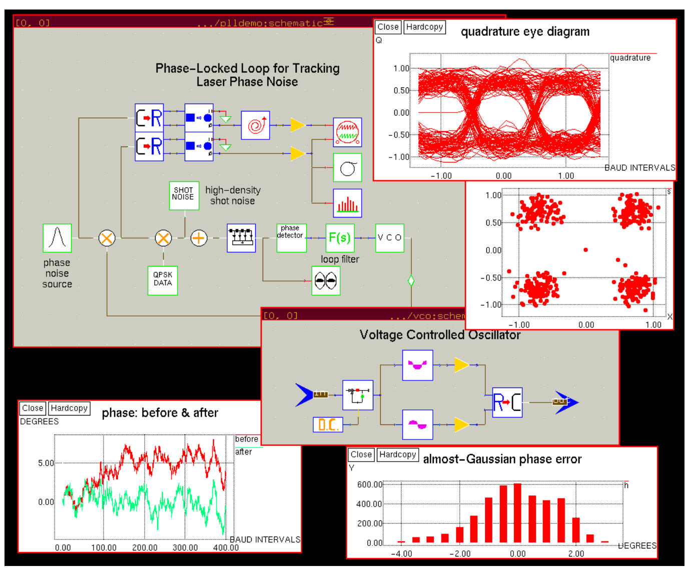
This project focuses on the design of reactive systems

- **real-time**
- **embedded**
- **concurrent**
- **distributed**
- **adaptive**

Background

Ptolemy Research

- Visual, system-level heterogeneous design.
- Animated interactive and real-time simulation.
- Formal methods for dataflow and discrete-event systems
- Programming language semantics.
- Software and hardware synthesis.
- Parallel architectures, partitioning, and scheduling.

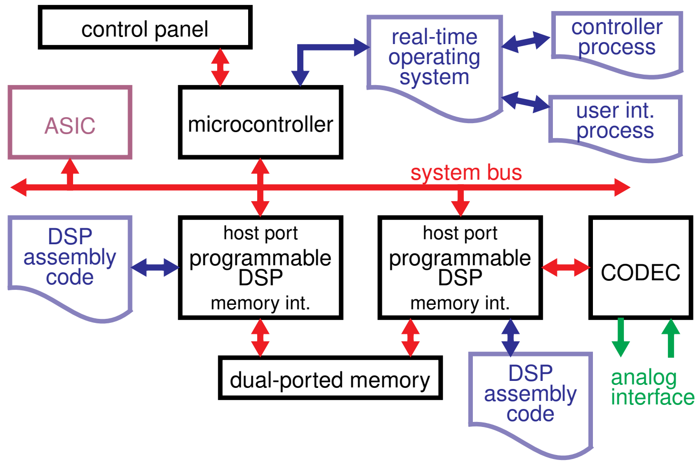


This highly multidisciplinary project has been addressing system-level design and implementation of reactive real-time systems, with emphasis on embedded systems since 1990. It provides the infrastructure on which we will build.

System-Level Heterogeneous Design

Hardware/ Software Codesign

- Synthesis of embedded software simulated on an instruction set model of the processor.
- Cosimulation with VHDL simulators.
- Synthesis of synthesizable and simulatable VHDL from dataflow.
- Partitioning and scheduling.

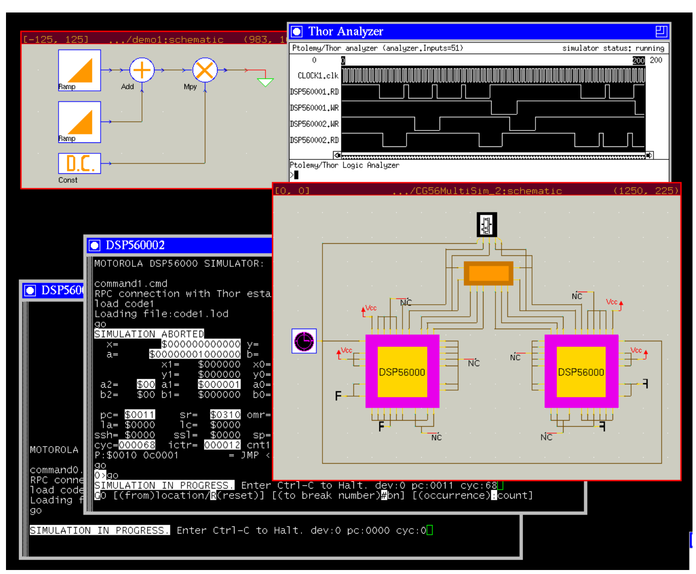


The class of systems addressed in the Ptolemy project under RASSP was embedded systems with one or more programmable processor.

An Early Co-Simulation Environment

Demonstrated

- Instruction set model of the processor.
- RTL model of chip input/output.
- High-level software development.
- Co-simulation with VHDL simulators.
- Co-simulation with functional ASIC models.
- Co-synthesis using Hyper and architectural synthesis.



The image above shows an early (1992) Ptolemy simulation of a multiprocessor system where the software is synthesized (including partitioning) from a block diagram and hardware is simulated at the RTL level.

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.

Meret Oppenheim, *Object*, 1936

It is generally better to be higher in this list

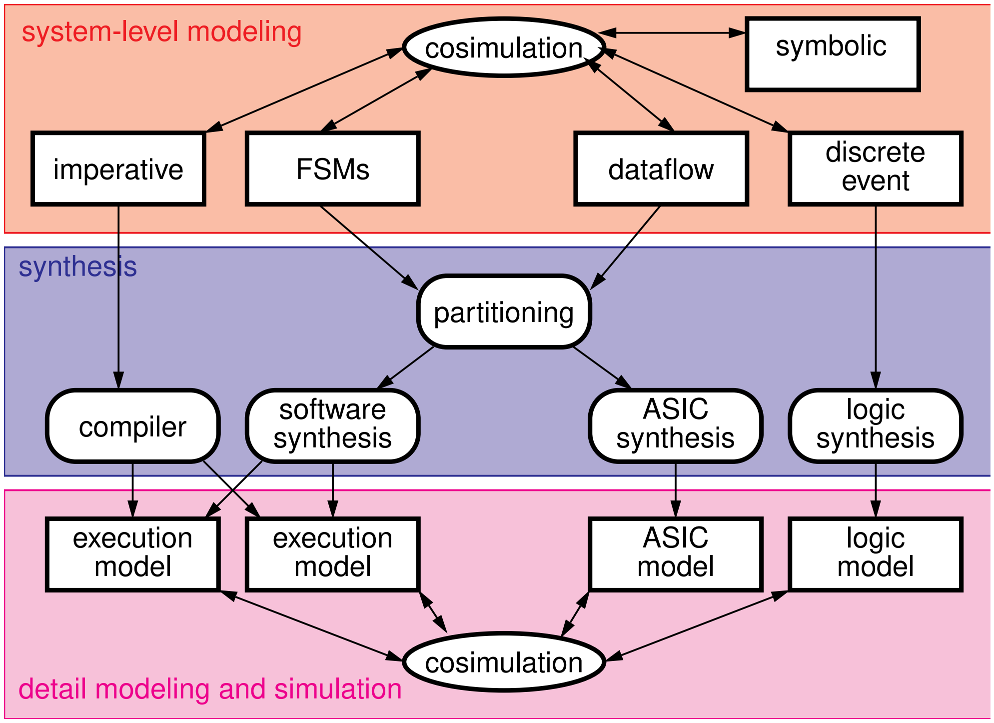
Usefulness of a Model of Computation

- Expressiveness
- Generality
- Simplicity
- Compilability/Synthesizability
- Verifiability

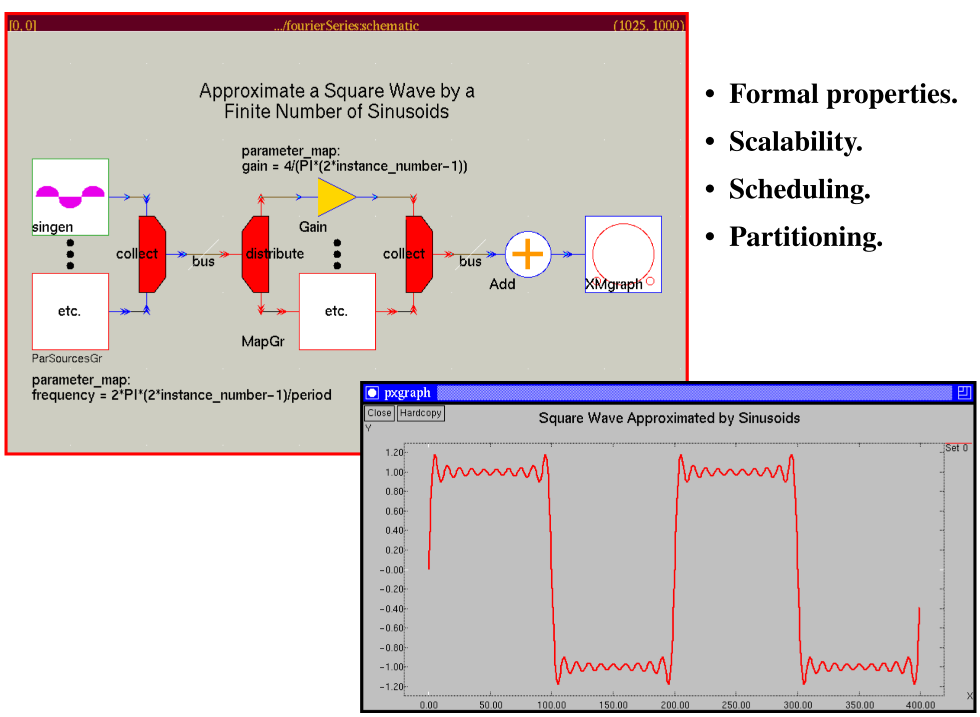
The Conclusion

One way to get all of these is to mix diverse, simple models of computation, while keeping compilation, synthesis, and verification separate for each MoC. To do that, we need to understand these MoCs relative to one another, and understand their interaction when combined in a single system design.

A Mixed Design Flow



Visual Design

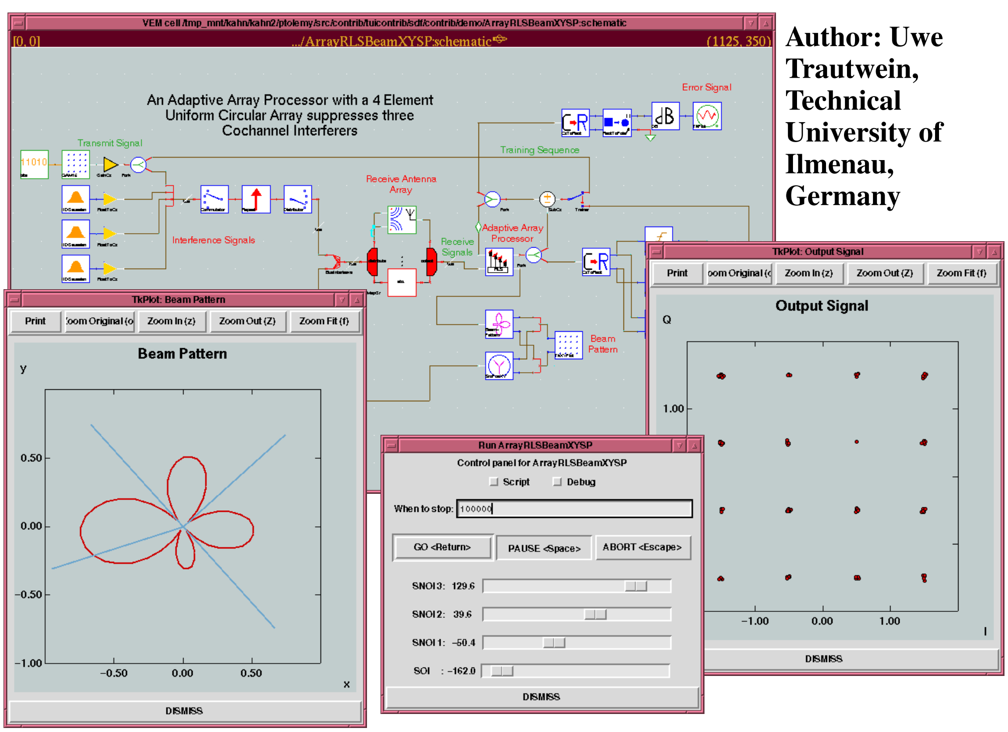


- **Formal properties.**
- **Scalability.**
- **Scheduling.**
- **Partitioning.**

Our Contributions to Dataflow Modeling

- Compile-time scheduling of *synchronous dataflow* graphs with optimized partitioning and memory utilization.
- Specification of the *Boolean dataflow (BDF) model*, which is Turing complete.
- Proof that the existence of a finite complete cycle and a bounded memory implementation for BDF is *undecidable*.
- *Heuristics* for constructing finite complete cycles and bounded memory schedules most of the time.
- *Multidimensional* generalization to dataflow models.
- *Process network* model generalization to dataflow.
- *Visual programming* formulation and use of *higher-order functions*.

Interactive, High-Level Simulation and Specification



Author: Uwe Trautwein, Technical University of Ilmenau, Germany

Ptolemy as a Tool and as a Laboratory

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

Allows for experiments with:
- Models of computation
- Domain-specific tools
- Design methodology
- Software synthesis
- Hardware synthesis
- Cosimulation

Rationale for heterogeneity: specialized models are
- More useful to the system-level designer
- More amenable to hardware and software synthesis
- More amenable to formal methods

Extensive Documentation

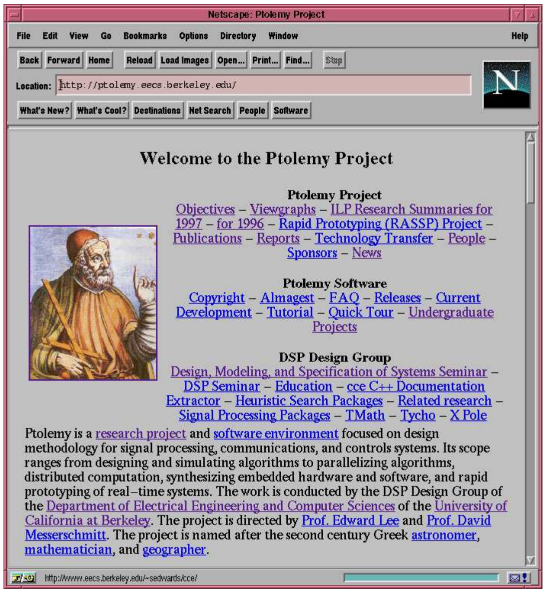
The Almagest

Volume 1
User's Manual
pigi
ptcl
domains
vem
pxgraph
installation

Volume 2
Programmer's Manual
software organization
writing stars
infrastructure
data types
Tcl/Tk
domains
code generation

Volume 3
Kernel Manual
detailed documentation
of all C++ classes
defined in the kernel.
Essential for defining
Targets and Domains.

Further Information



- Software distribution
- Small demonstration version
- Project overview
- *The Almagest* (the 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