The Ptolemy Project

Shuvra Bhattacharyya
Joseph T. Buck
Wan-Teh Chang
Brian L. Evans
Steve X. Gu
Sangjin Hong
Christopher Hylands
Asawaree Kalavade
Alan Kamas
Allen Lao
Bilung Lee
Edward A. Lee
David G. Messerschmitt
Praveen K. Murthy
Thomas M. Parks
José Luis Pino
Farhana Shiekh
S. Sriram
Juergen Teich
Warren W. Tsai
Patrick J. Warner
Michael C. Williamson

Miniconference Morning Program

9:00  Overview of the Ptolemy project  
Edward A. Lee

9:20  The Ptolemy kernel  
Edward A. Lee

9:40  Design Methodology Management  
Asawaree Kalavade

10:10 Symbolic Computation in System Simulation and Design  
Brian Evans

10:30  Break

10:50 VHDL Code Generation for Simulation and Synthesis  
Mike Williamson

11:10 Optimization Issues in Embedded Software Synthesis  
Shuvra Bhattacharyya

11:40 Combined Code and Data Memory Minimization  
Praveen Murthy

12:00  Lunch

Miniconference Afternoon Program

1:00  Parallel Implementation  
S. Sriram

1:20  Real-Time Prototyping  
José Pino

1:40  Mixing Dataflow with Control  
Wan-Teh Chang

2:10  Break

2:30  An introduction to a Mathematical Model of Dataflow  
Tom Parks

2:50  The Process Networks Domain  
Tom Parks

3:00  Application to the Infopad Project  
Sam Sheng

3:20  NetPlan: A Network Planning Tool on Ptolemy  
Zhigang Qin

3:40  Application to Distributed Telecommunications Services  
William Li

4:00  Application to Video Networking  
Allen Lao

4:20  Preview of Ptolemy versions 0.5.2 and 0.6  
Alan Kamas

5:00  Adjourn

System-Level Design of Signal Processing Systems

Ptolemy Research

- Design complexity management.
- Visual, algorithm-level system design.
- Formal methods for dataflow systems.
- Programming language semantics.
- Software and hardware synthesis.
- Parallel architectures, partitioning, and scheduling.

This highly multidisciplinary project addresses system-level design and implementation of signal processing systems.
Implementation of Signal Processing Systems

**Hardware/Software Synthesis**
- Design of heterogeneous embedded systems.
- Real-time systems.
- Synthesis of software from dataflow graphs.
- System-level hardware design.
- Cosimulation of hardware and software.
- Codesign of hardware and software.

The design philosophy in Ptolemy is heterogeneous, allowing for effective use of specialized design tools within a general system-level design environment.

Heterogeneous Modeling and Design

**Key Principles**
- Extensible, object-oriented kernel.
- No model of computation is implemented in the kernel.
- Models of computation are implemented in modular domains.
- A domain interacts with another without knowing its semantics.

Multiple models of computation may be used in the same system. Here, dataflow is used for signal processing, while a timed discrete-event system models a communication network.

Heterogeneity in System-Level Design

**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
Domains in Ptolemy

- SDF
- BDF
- DDF
- PN
- process networks
- dynamic dataflow
- Boolean dataflow
- synchronous dataflow
- MDSDF
- multidimensional SDF
- Thor
- circuit simulation
- DE
- discrete-event
- communicating processes
- design methodology management
- PTOLEMY KERNEL

Major Activities

- **Formal methods**
  - Dataflow (process networks, synchronous, Boolean, multidimensional, ...)
  - Control (hierarchical FSMs, Esterel, synchronous languages, ...)
  - Partitioning and scheduling of dataflow graphs (optimize IPC, memory, ...)
  - Programming languages (higher-order functions, polymorphism, ...)

- **Algorithm-level design methodology**
  - Mixing models of computation (discrete-event, FSMs, dataflow, imperative, ...)
  - Animation and visualization (Tcl/Tk, Matlab, xv, ...)
  - Mixing domain-specific tools (filter design, Matlab, Mathematica, ...)
  - Visual programming (dataflow, FSMs, regularity, recursive, functional, ...)

- **System-level design methodology**
  - Synthesis of embedded software (high-level, assembly, ...)
  - Design complexity management (data, tool, flow, methodology, ...)
  - Hardware/software codesign (DesignMaker, GCLP partitioning, ...)
  - Architecture design and performance modeling (OT principle, VHDL, ...)

Major Contributions in 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**.

Where to From Here?

- **Real-time scalable computing.**
- **Scalable embedded systems design.**
- **Design migration from abstract to concrete.**
- **Formal methods based on partial orders.**
- **Hybrid systems: combining FSM with dataflow.**
- **Modeling and analysis of random systems.**
- **Design of nondeterminate systems.**
- **Complexity management.**
- **Design visualization and documentation.**
- **Partial evaluation and incremental compilation.**
- **Models for back-end signal interpretation.**
- **Heterogeneous scheduling.**
Activities during the Next Year

- **Software infrastructure**
  - Generalize Wormhole mechanism with dynamic switching.
  - Redesign the code generation mechanism for better retargeting.
  - Improve support for scripted runs.
  - Improve design visualization.
  - Generalize type checking and automatic type conversion.
  - Generalize parameter handling.
  - Enhance interactive graphics.
  - On-line design documentation.
  - Simulation data management.

- **Heterogeneous design**
  - Design of a hierarchical finite-state machine controllers
  - Embed controllers designed using Esterel.
  - System-level performance modeling for heterogeneous hardware.
  - Complete the design methodology management domain.

Activities During the Next Year (cont.)

- **Models of computation**
  - Explore interacting semantics of synchronous languages and dataflow.
  - Generalize multidimensional dataflow and explore synthesis issues.
  - Apply Boolean dataflow technology to VHDL-based synthesis.
  - Develop a process network domain that supports nondeterminacy.
  - Add enhancements from the Navy’s PGM to the dynamic dataflow domain.
  - Implement and understand cyclo-static dataflow.

- **Applications**
  - InfoPad
  - Radar
  - Image and video processing
  - Communications

- **Algorithm-level design**
  - Interface to Mathematica for symbolic computation.
  - Integrate system rewriting in Mathematica with DMM domain.

---

1994 Funding for the Ptolemy Project

- **Industry**
  - Bellcore
  - BNR
  - Dolby
  - Hitachi
  - Mentor
  - Mitsubishi
  - NEC
  - Pacific Bell
  - Philips
  - Rockwell

- **Institutional Sponsors:**
  - ARPA & the US Air Force under RASSP.
  - the Semiconductor Research Corporation.
  - the National Science Foundation.
  - the State of California MICRO program.

- Cash basis, 1994 only

World Wide Web Server

- Complete distribution of version 0.5.1, including all source code.
- Distribution of Ptiny Ptolemy, a small demonstration version.
- An evolving quick tour of Ptolemy with animations of simple demos.
- User’s manual in hypertext form.
- Publications from the Ptolemy group.
- Keyword searching for publications.
- Directory of project participants and sponsors.
- Copy of the FAQ and info about mailing lists and newsgroups.

http://ptolemy.eecs.berkeley.edu