[2] A. Banerjea, E.W. Knightly, "Using Calendar Queues for Discrete Event Scheduling in Ptolemy", *Final Report for EE290T*, University of California, Berkeley, CA 94720, December, 1993.

[3] S. Bhattacharrya and E. A. Lee, "Looped Schedules for Dataflow Descriptions of Multirate Signal Processing Algorithms," **to appear** in *Formal Methods in System Design*, (updated from UCB/ERL Technical Report, May 21, 1993).

[4] S. Bhattacharyya and E. A. Lee, "Memory Management for Synchronous Dataflow Programs," to appear in *IEEE Tr. on Signal Processing*, May 1994. Updated from: Technical Report UCB/ERL M92/128, EECS Dept., UC Berkeley, November 18, 1992.

[5] S. Bhattacharyya, J. T. Buck, S. Ha, and E. A. Lee, "A Scheduling Framework for Minimizing Memory Requirements of Multirate DSP Systems Represented as Dataflow Graphs," in *VLSI Signal Processing VI*, IEEE Special Publications, New York, 1993.

[6] S. Bhattacharrya, J. Buck, S.-H. Ha, E. A. Lee, "Generating Compact Code from Dataflow Specifications of Multirate DSP Algorithms," UCB/ERL Technical Report M93/36, May 21, 1993.

[7] J. Bier, E. Goei, W. Ho, P. Lapsley, M. O'Reilly, G. Sih and E.A. Lee, "Gabriel: A Design Environment for DSP," *IEEE Micro Magazine*, October 1990, Vol. 10, No. 5, pp. 28-45.

[8] J. Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, "Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems", *International Journal of Computer Simulation*, special issue on "Simulation Software Development," January, 1994.

[9] J. T. Buck, *Scheduling Dynamic Dataflow Graphs with Bounded Memory Using the Token Flow Model*, Ph. D. Dissertation, Dept. of EECS, University of California, Berkeley, CA 94720, 1993.

[10] J. T. Buck and E. A. Lee, "Scheduling Dynamic Dataflow Graphs with Bounded Memory Using the Token Flow Model," *Proc. of ICASSP* '93, Minneapolis, MN, April, 1993.

[11] J. T. Buck, "The Ptolemy Kernel, A Programmer's Companion for Ptolemy 0.4," Memorandum UCB/ERL M93/8, January 19, 1993.

[12] M. Chen, "Developing a MDSDF domain in Ptolemy", *Final Report for EE290T*, University of California, Berkeley, CA 94720, December, 1993.

[13] A. Kalavade, and E.A. Lee, "A Hardware/Software Codesign Methodology for DSP Applications," *IEEE Design and Test*, September 1993.

[14] E. A. Lee, "Computing and Signal Processing: An Experimental Multidisciplinary Course", *Proc. of ICASSP-94*, Adelaide, Australia, April, 1994.

[15] E. A. Lee, "Representing and Exploiting Data Parallelism Using Multidimensional Dataflow Diagrams," *Proc. of ICASSP* '93, Minneapolis, MN, April, 1993.

[16] S. Sriram and E. A. Lee, "Design and Implementation of an Ordered Memory Access Architecture," *Proc. of ICASSP* '93, Minneapolis, MN, April, 1993.

[17] E. A. Lee, "A Design Lab for Statistical Signal Processing," *Proceedings of ICASSP '92*, San Francisco, March, 1992.

[18] E. A. Lee, "Consistency in Dataflow Graphs", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 2, No. 2, April 1991.

[19] E. A. Lee, W.-H. Ho, E. Goei, J. Bier, and S. Bhattacharyya, "Gabriel: A Design Environment for DSP", *IEEE Trans. on ASSP*, November, 1989.

[20] E. A. Lee and D. G. Messerschmitt, "Synchronous Data Flow," *IEEE Proceedings*, September, 1987.

[21] E. A. Lee and D. G. Messerschmitt, "Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing," *IEEE Transactions on Computers*, January, 1987.

[22] P. Murthy, S. Bhattacharyya, E.A. Lee, "Minimizing Memory Requirements For Chain-Structured Synchronous Dataflow Programs", *Proc. of ICASSP-94*, Adelaide, Australia, April, 1994.

[23] P. Murthy, *Multiprocessor Code Synthesis in Ptolemy*, **MS Report**, Dept of EECS, UC Berkeley, July 1993.

[24] J. Ousterhout, "Tcl: An Embeddable Command Language," *USENIX Conference Proceedings*, Winter, 1990.

[25] J. Ousterhout, "An X11 Toolkit Based on the Tcl Language," *USENIX Conference Proceedings*, Winter, 1991.

[26] J. L. Pino, "Software Synthesis for Single-Processor DSP Systems Using Ptolemy", **MS Report** UCB/ERL M93/35, Dept of EECS, UC Berkeley, May 1993.

[27] S-I Shih, "An Application for MDSDF",*Final Report for EE290T*, University of California, Berkeley, CA 94720, December, 1993.

[28] K. White, *XPole: An Interactive Graphcial Signal Analysis and Filter Design Tool*, **MS Report**, Dept of EECS, UC Berkeley, May 1993.

[29] E. Wu, "Scheduling SDF Graphs with I/O Timing Constraints", *Final Report for EE290T*, University of California, Berkeley, CA 94720, December, 1993..

exception. We have, in the process, encountered both strengths and weaknesses in this approach. For example, we frequently wish to parameterize a system specification, but find that the value of the parameter affects the structure of our graphical representation. We believe that such limitations are due primarily to the semantics of the graphical models we are using, and not to their graphical syntax. In view of this, we are more systematically study the potential of graphical representation of signal processing systems. One aspect of this work concerns the use of higher-order functions in graphical languages. This work has not progressed sufficiently to demonstrate concrete results.

2.1.4 Dynamic dataflow

Our most heavily used model of computation for signal processing is synchronous dataflow (SDF) [19][20]. In this model, an application is described as a graph where nodes represent computations ("actors") and arcs represent the flow of data ("streams"). The actors produce and consume a fixed and known amount of data on each arc each time they fire. The synchronous dataflow model has the compelling advantage that the firing pattern of the actors can be completely determined at compile time.

Algorithms with predictable control flow have been successfully addressed using the synchronous dataflow (SDF) model of computation. Recently, however, our effort has broadened to include applications where control flow is not predictable. The objective is to preserve the benefits (especially efficiency) of predictable control flow whenever possible, but to support dynamic decision making, dynamic real-time response, and asynchrony. This will broaden the application domain to include telecommunications systems, real-time control, and hardware and software co-design. To do this, we are pursuing two lines of inquiry that avoid discarding the SDF model of computation in favor of one that is more general. The first is to mix models of computations, gaining generality through heterogeneity. The Ptolemy system is focussed on supporting this. The second is to extend the analytical techniques of SDF to dynamic dataflow graphs. A *token flow model* [17][10][9] has been devised that replaces many numeric operations that worked under the SDF model with symbolic operations. The dependence of control-flow on Booleans is represented symbolically.

2.2. Overview of recent publications and software

The most visible output from this group is the Ptolemy software system, described in detail in [8][11] and in the manual. Version 0.5 of Ptolemy was released in February of 1994. It is distributed through our Industrial Liaison Program office and electronically by anonymous FTP. The manual (called *The Almagest*) has grown to more than 700 pages and occupies four volumes. The software has formed the testbed for a number of research projects, including three masters projects [22][25][27] and one Ph.D. dissertation [9] completed during this reporting period.

J. Buck's Ph.D. dissertation [9] makes fundamental contributions to the theory and practice of scheduling dataflow graphs. Buck proves that determining whether dataflow graphs can be executed with bounded memory is equivalent to solving the general halting problem, and thus cannot be reliably completed in finite time. P. Murthy's masters report [22] describes parallel code generation for a four-processor programmable DSP system from Star Semiconductor. J. Pino's masters report [25] describes a family of optimizations performed in synthesizing assembly code for programmable DSPs. K. White's masters report [27] describes a software system for interactively designing and analyzing discrete- and continuous-time filters and the mappings between them.

One undergraduate independent study project was also based on Ptolemy; Wei-Jen Huang, who worked on the Tcl/Tk interface.

In the Fall of 1993, we organized a graduate seminar in which we examined the fundamentals of the technology underlying Ptolemy. The design of this seminar is reported in [13]. Some notable projects resulting from it are reported in [2][1][28][12][26].

We have made considerable progress on formal and heuristic techniques for efficient code generation from synchronous dataflow graphs [3][4][5][6][21]. Most of these involve coupling of scheduling techniques with optimized code generation. We recently devised a multidimensional extension of synchronous dataflow [14] that promises to significantly improve our ability to exploit data parallelism.

Our work on hardware/software codesign [13] has been getting quite a bit of attention from both industry and academia. However, we feel that we have barely scratched the surface of this problem.

In addition, Ptolemy is being used in both our graduate and undergraduate signal processing classes. In the Spring semester of 1994, approximately 100 students (half graduate and half undergraduate) will each perform between 6 and 8 algorithmic design projects using Ptolemy on a network of DEC workstations. The graduate experiments are described in [16].

3. CURRENT PROJECTS

3.1. Wormhole interfaces for heterogeneous targets

When Ptolemy executes synthesized code on an attached processor, it currently has only rudimentary mechanisms for controlling that execution. We are generalizing the wormhole interface in Ptolemy so that a real-time program running on attached hardware will appear to the user as part of the process running on the workstation. This should make the real-time hardware transparently accessible, greatly enhancing our ability to rapidly prototype systems.

3.2. Real-time control

The dataflow capabilities in Ptolemy have advanced much further than other models of computation useful for prototyping. We are developing multithreaded dataflow and hierarchical finite state machine models for mixing real-time control with signal processing.

3.3. Optimized code generation

Synthesizing efficient assembly code for programmable DSPs has proved to be a rich area for innovation. We are currently focusing on problems associated with multirate systems that have radically different sample rates in different parts of the system.

4. REFERENCES

[1] A. Abnous, "Modeling the ALOHA Radio Network in the Discrete Event and Communicating Processes Domains of Ptolemy", *Final Report for EE290T*, University of California, Berkeley, CA 94720, December, 1993.

first problem to be addressed here is one of algorithm representation.

2.1.1 *Animation in simulation*

One aspect of this is to represent the dynamics of a signal processing system using interactive, animated graphics. We have linked the interpreted language Tcl [23] and its associated X window toolkit [24] with the Ptolemy system. This provides a powerful, extensible environment within which users can construct customized, animated, interactive simulations. In figure 1, we show an example of how this is used.

2.1.2 *Matrix manipulations*

A second aspect of the effort to raise the level of algorithm representation in Ptolemy is the inclusion of a comprehensive matrix manipulation mechanism, cleanly coupled with the block diagram representation. Figure 2 shows one application of this matrix class. Here, the so-called MUSIC algorithm is being implemented for identifying sinusoids in noise. This is a very high-level representation of the algorithm.

2.1.3 *Graphical programming*

Although it was not originally part of the intent of the Ptolemy project to explore graphical programming, we have found ourselves representing applications at least partially graphically almost without

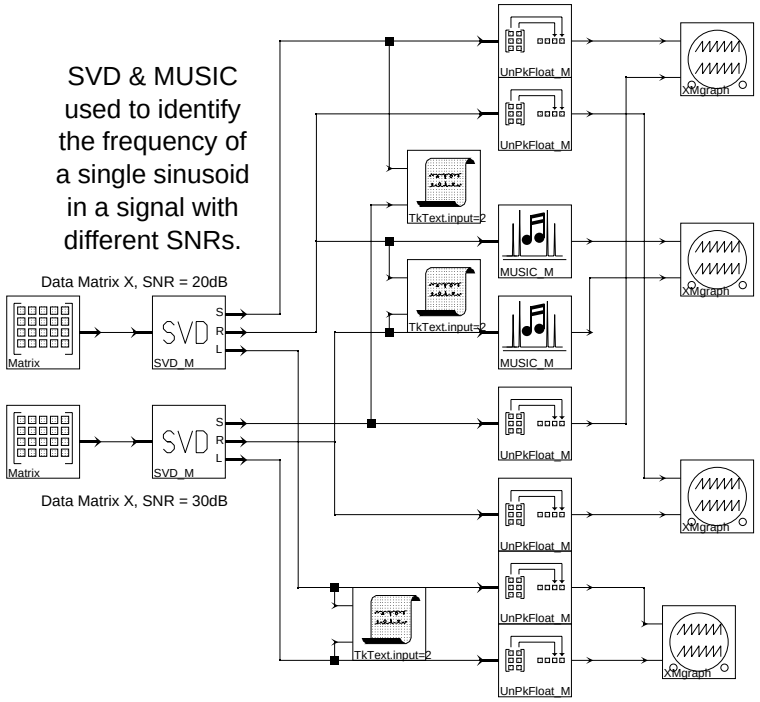

Figure 2. An application of the Matrix class in Ptolemy, used for high-level representation of algorithms.

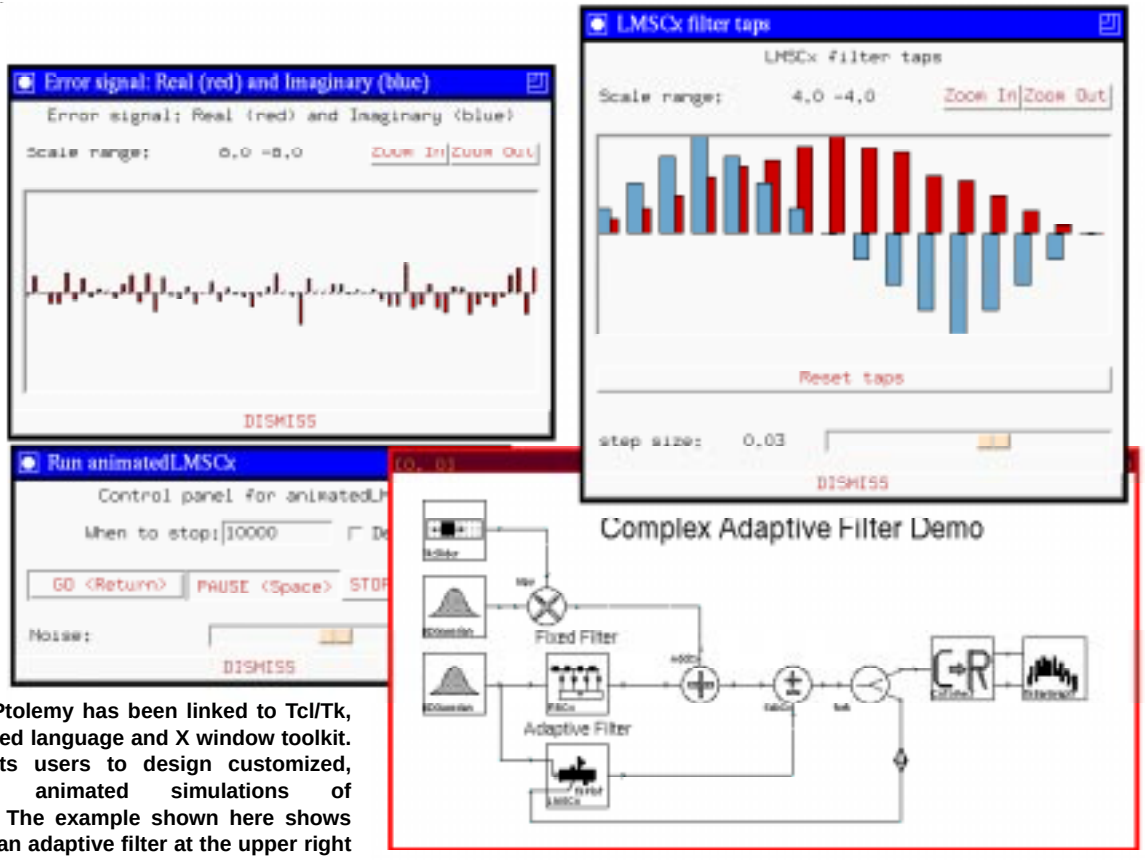

Figure 1: Ptolemy has been linked to Tcl/Tk, an interpreted language and X window toolkit. This permits users to design customized, interactive, animated simulations of algorithms. The example shown here shows the taps of an adaptive filter at the upper right as the filter adapts. The user controls the noise and adaptation step size.

DESIGN METHODOLOGY FOR DSP

Edward A. Lee

Department of Electrical Engineering and Computer Science
University of California, Berkeley CA 94720

Final Report 1992-93, Micro Project #92-084
Industrial Sponsors: Bell Northern Research, Dolby Laboratories,
Motorola, Philips Research, Rockwell International, and Star Semiconductor.

ABSTRACT

This project explores design methodology for simulation and real-time parallel computation for applications using digital signal processing. The goal is to facilitate rapid prototyping of complex algorithms by developing tools that are both efficient in their use of hardware and easy for an algorithm designer to learn and use. The first five years of the project have been extremely productive, resulting in several new techniques and two working software systems, *Gabriel* and *Ptolemy.* Ptolemy is current being distributed by anonymous FTP and through our Industrial Liaison Program. Ptolemy is also currently being used as an integral part of our graduate course in statistical signal processing, EE225a, our undergraduate signal processing course, EE123, and a research seminar EE290T investigating languages and design methodology for signal processing systems.

1. MOTIVATION

Ptolemy, a system-level design framework, is the successor to Gabriel [7][18], a software environment for real-time signal processing. Ptolemy serves as the nucleus of a variety of distinct projects in simulation, design, and implementation of systems. This MICRO project continues the work begun with Gabriel on design of real-time signal processing systems, but now uses Ptolemy as the software framework. Unlike Gabriel, the Ptolemy framework is flexible enough to accommodate a number of distinct projects, and each contributes infrastructure that the other projects can benefit from. The Ptolemy effort began in January of 1990, under the joint direction of Professors Lee and Messerschmitt. It has grown to directly involve approximately 20 research students and three FTE staff positions. In addition, approximately 130 undergraduate and graduate students are currently using Ptolemy each year as an integral part of two courses, and several other research groups at Berkeley are using Ptolemy to conduct simulation and design integral to their research. The latest version of Ptolemy (designated 0.5[1]) is publicly available and freely redistributable as of February, 1994.

The MICRO project focuses on two issues compatible with the overall objectives of Ptolemy. The first of these concerns representation of signal processing algorithms at a high level of abstraction. In the past, we have concentrated on the use of large grain dataflow graphs for specifying algorithms. The elementary functional blocks are defined in the target language, C++, C, or assembly code for the target processor. We are broadening the options here, with the objective of enabling quick intuitive evaluation of algorithms. The second issue concerns compilation of the specification to generate an efficient real-time implementation. In particular, we are addressing several interesting problems associated with code generation for programmable DSPs, including video signal processors with VLIW architectures.

The ambitious objectives of the overall Ptolemy project include practically all aspects of designing signal processing and communications systems, ranging from the design of algorithms and communication strategies, through simulation, hardware and software design, parallel computing, and real-time prototyping. To manage this, it is essential that the software be highly modular and extensible, so that projects of manageable scope can proceed independently, and nonetheless can combine their results. The MICRO project complements nicely some of the other work using Ptolemy, such as simulation and design of communication networks, and parallel programming of commercial parallel machines, such as the CM-5 from Thinking Machines.

2. RESULTS OF MICRO SUPPORT

Tools for designing real-time systems must be somewhat specialized to a class of applications. This specialization enables tool designers to build in optimizations and customizations suited ideally to the class. For example, very good parallelizing code generators can be made for feedforward algorithms with deterministic control flow. A different code generator should be used for highly dynamic real-time systems, or for asynchronous systems.

Such specialization, however, runs counter to the increasing heterogeneity of system implementations. Custom hardware is mixed with software, and even within each of these two classes, design styles may differ radically for different parts of the system. The principle being pursued in this project is that a suite of specialized tools can be combined to form a general framework. Each specialized tool works with a model of computation that it can understand. And each tool must interoperate with other tools. We have focussed on signal processing applications, and are attempting to define precisely the suite of tools required to get a complete design environment. The MICRO project addresses a subset of the possibilities: those involving real-time implementation on programmable DSPs.

2.1. Algorithm representation

We are exploring methods for quickly and easily evaluating signal processing algorithms, coupled with methods for synthesizing real-time prototypes using programmable processors including the Motorola DSP families and the Philips video signal processor. The

1. Since Ptolemy is research software, distributed free of charge and without support, all versions distributed by the University are designated 0.x.