[17] Sundararajan Sriram, *Minimizing Communication and Synchronization Overhead in Multiprocessors for Digital Signal Processing*, Tech. Report UCB/ERL 95/90, **Ph.D. Dissertation**, Dept. of EECS, University of California, Berkeley, CA 94720, November 7, 1995. (http://ptolemy.eecs.berkeley.edu/papers/sriramThesis)

[18] J. L. Pino, S. S. Bhattacharyya and E. A. Lee, "A Hierarchical Multiprocessor Scheduling System for DSP Applications,"*Proc. IEEE Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 29 - November 1, 1995. (http://ptolemy.eecs.berkeley.edu/papers/hierStaticSched-asilomar-95)

[19] T. M. Parks, J. L. Pino, and E. A. Lee, "A Comparison of Synchronous and Cyclo-Static Dataflow," *Proc. IEEE Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 29 - November 1, 1995. (http://ptolemy.eecs.berkeley.edu/papers/csdfVSsdf)

[20] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, "Optimal Parenthesization of Lexical Orderings for DSP Block Diagrams," in *Proc. IEEE Workshop on VLSI Signal Processing*, Osaka, Japan, October 16-18, 1995. (http://ptolemy.eecs.berkeley.edu/papers/PganRpmcDppo/).

[21] S. S. Bhattacharyya, S. Sriram, and E. A. Lee, "Resynchronization for Embedded Multiprocessors," ERL Technical Report UCB/ERL M95/70, University of California, Berkeley, CA 94720, September, 1995. (http://ptolemy.eecs.berkeley.edu/papers/synchOpt/).

[22] Asawaree Kalavade, *System Level Codesign of Mixed Hardware-Software Systems*, Tech. Report UCB/ERL 95/88, **Ph.D. Dissertation**, Dept. of EECS, University of California, Berkeley, CA 94720, September, 1995. (http://ptolemy.eecs.berkeley.edu/papers/kalavadeThesis)

[23] B. L. Evans, D. R. Firth, K. D. White, and E. A. Lee, "Automatic Generation of Programs That Jointly Optimize Characteristics of Analog Filter Designs," *Proc. of European Conference on Circuit Theory and Design*, August 27-31, 1995, Istanbul, Turkey, pp. 1047-1050. (http://ptolemy.eecs.berkeley.edu/papers/filter_design_ecctd95.ps.Z)

[24] B. L. Evans, S. X. Gu, A. Kalavade, and E. A. Lee, "Symbolic Computation in System Simulation and Design," Invited Paper, *Proc. of SPIE Int. Sym. on Advanced Signal Processing Algorithms, Architectures, and Implementations*, July 9-16, 1995, San Diego, CA, pp. 396-407. (http://ptolemy.eecs.berkeley.edu/papers/spie95symbcomp.ps.Z)

[25] S. S. Bhattacharyya, S. Sriram, and E. A. Lee, "Minimizing Synchronization Overhead in Statically Scheduled Multiprocessor Systems," *Proc. of IEEE Int. Conference on Application Specific Array Processors*, July 24-26, 1995. (http://ptolemy.eecs.berkeley.edu/papers/synchOpt/).

One masters thesis has been completed related to Tycho [5].

2.7. Plans

The proposed project in the next year will be continuing its focus on one big issue: system-level design that mixes real-time signal processing with control. By control we mean sequential and concurrent decision making that responds to external events, such as user input or sensor data, or internal events, such as intermediate computational results. Applications include knowledge-based adaptive signal processing, startup and take-down of signal processing systems, interaction between signal processing subsystems and host computers, and user-driven multimedia systems.

This work includes both analytical techniques, based on process algebras, programming language semantics, and scheduling theory, plus practical implementations in the Ptolemy software environment. The Ptolemy software, developed under this project, serves as an extensible and well-equipped laboratory for this research. We are currently focusing on the combination of dataflow semantics, hierarchical finite state machines, and synchronous/reactive languages. In all three cases, languages with both visual and textual syntaxes are being explored, and the semantics of the interactions between these languages is being studied.

3. PUBLICATIONS — July 1995 - December 1996

[1] P. K. Murthy, *Scheduling Techniques for Synchronous and Multidimensional Synchronous Dataflow*, Technical Report UCB/ERL, **Ph.D. Dissertation**. EECS Department, University of California, Berkeley, CA 94720, December 1996. (http://ptolemy.eecs.berkeley.edu/papers/murthyThesis/)

[2] S. S. Bhattacharyya, P. K. Murthy and E. A. Lee, "*Software Synthesis from Dataflow Graphs*., Kluwer Academic Publishers, Norwell, Mass, 1996.

[3] S. S. Bhattacharyya, S. Sriram, and E. A. Lee, "Resynchronization of Multiprocessor Schedules: Part 1 -- Fundamental Concepts and Unbounded-latency Analysis,," Memorandum UCB/ERL M96/55, Electronics Research Laboratory, U. C. Berkeley, October, 1996.

[4] S. S. Bhattacharyya, S. Sriram, and E. A. Lee, "Resynchronization of Multiprocessor Schedules: Part 2 -- Latency-constrained Resynchronization,," Memorandum UCB/ERL M96/56, Electronics Research Laboratory, U. C. Berkeley, October, 1996.

[5] F. Sheikh, "Visualizing Architecture and Algorithm Interaction in Embedded Systems,," MS Thesis, EECS Department, University of California. Berkeley, CA 94720, September 17, 1996. (http://ptolemy.eecs.berkeley.edu/papers/visualizing)

[6] E. A. Lee, and A.Sangiovanni-Vincentelli, "Comparing Models of Computation," Proc. of ICCAD, San Jose, CA, Nov. 10-14, 1996.

[7] A. Kalavade and E. A. Lee, "Complexity Management in System-Level Design," *Journal of VLSI Signal Processing Systems*, Vol. 14, No. 2, November 1996. (http://ptolemy.eecs.berkeley.edu/papers/96/jvlsi-dmm)

[8] R. Mani, H. S. Nawab, J. M. Winograd, and B. L. Evans, "Integrated numeric and symbolic signal processing in a heterogeneous design environment," to appear *Proc. SPIE Int. Sym. on Advanced Signal Processing Algorithms, Architectures, and Implementations*, Denver, CO, August 12-15, 1996. (http://ptolemy.eecs.berkeley.edu/papers/96/ipusPtolemy/)

[9] S. S. Bhattacharyya, S. Sriram, and E. A. Lee, "Latency-Constrained Resynchronization for Multiprocessor DSP Implementation," *Proc. ASAP Conference*, Chicago, August 19-21, 1996. (http://ptolemy.eecs.berkeley.edu/papers/96/resync)

[10] E. A. Lee and A. Sangiovanni-Vincentelli, "The Tagged Signal Model --A Preliminary Version of a Denotational Framework for Comparing Models of Computation," ERL Memorandum UCB/ERL M96/33, University of California, Berkeley, CA 94720, June 4, 1996. (http://ptolemy.eecs.berkeley.edu/papers/96/denotational/).

[11] S. S. Bhattacharyya, S. Sriram, and E. A. Lee, "Self-Timed Resynchronization: A Post-Optimization for Static Multiprocessor Schedules," *Proc. IPPS*, April 1996. (http://ptolemy.eecs.berkeley.edu/papers/96/parallel)

[12] K. Chiang, B. L. Evans, W. T. Huang, F. Kovac, E. A. Lee, H. J. Reekie, D. G. Messerschmitt, and S. Sastry, "Real-Time DSP for Sophomores," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Atlanta, GA, May 7-10, 1996, vol. 2, pp. 1097-1100. (http://ptolemy.eecs.berkeley.edu/papers/96/realTimeCourse)

[13] P. K. Murthy and E. A. Lee, "An Extension of Multidimensional Synchronous Dataflow to Handle Arbitrary Sampling Lattices," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Atlanta, GA, May 7-10, 1996, vol. 6, pp. 3306-3309. (http://ptolemy.eecs.berkeley.edu/papers/genMdsdf)

[14] J. L. Pino, M. C. Williamson, and E. A. Lee, "Interface Synthesis in Heterogeneous System-Level DSP Design Tools," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Atlanta, GA, May 7-10, 1996, vol. 2, pp. 1268-1271. (http://ptolemy.eecs.berkeley.edu/papers/interfaceSynthesis)

[15] G. Arslan, B. L. Evans, F. A. Sakarya, and J. L. Pino, "Performance Evaluation and Real-Time Implementation of Subspace, Adaptive, and DFT Algorithms for Multi-Tone Detection," *Proc. Int. Conf. on Telecommunications*, Istanbul, Turkey, April 15-17, 1996. (http://ptolemy.eecs.berkeley.edu/papers/96/dtmf_ict)

[16] T. M. Parks, *Bounded Scheduling of Process Networks*, Technical Report UCB/ERL-95-105. **PhD Dissertation**. EECS Department, University of California. Berkeley, CA 94720, December 1995. (http://ptolemy.eecs.berkeley.edu/papers/parksThesis)

Graphical programs can be either interpreted or compiled. It is common in signal processing environments to provide both options. The output of compilation can be a standard procedural language, such as C, assembly code for programmable DSP processors, or even specifications of silicon implementations. We have put considerable effort into optimized compilation.

2. RESULTS OF MICRO SUPPORT

2.1. Formal methods

The focus of our work on formal methods is to understand models of computation that can be applied to system-level design of embedded signal processing systems. The major focus, therefore, is on concurrent models and models that coexist well with huge computational loads and real-time constraints.

In this reporting period, this work produced one Ph.D. thesis [16], one conference paper [19], one invited embedded tutorial [6], and one technical memorandum [10].

2.2. Control and signal processing

In previous years this project concentrated on the computational aspects of signal processing systems, and thus focused on models of computation such as dataflow that are particularly well suited. More recently, our attention has broadened to include control and sequential decision-making aspects of system design. We are pursuing four approaches for combining control-oriented computation with data-oriented computation: hierarchical concurrent finite-state machines, the synchronous/reactive model of computation, dynamically evaluated higher-order functions, and nondeterminate interaction between concurrent processes.

Although only one conference paper related to this work was published in this reporting period [14], a number of presentations have been made and several papers are the publication pipeline.

2.3. System-level design

In the area of system-level design, we have improved the synthesis of VHDL from dataflow graphs and developed a VHDL synthesis mechanism that translates dataflow block diagrams into one of two different styles of VHDL code, one style optimized for synthesis and one for simulation. We have also developed a model of computation for multidimensional multirate signal processing, have made some contributions to symbolic signal processing, and have developed a method for reducing the synchronization overhead in concurrent systems.

In this reporting period, this work produced two Ph.D. theses [1][22], one journal paper [7], and six conference papers [8] [13] [15] [18] [23] [24].

2.4. Scheduling and code generation

We have made major improvements in heterogeneous scheduling and code generation. We have rethought heterogeneous code generation in the context of a new hierarchical scheduling framework. The objective is to more effectively mix synthesized software and VHDL models with simulations built in other Ptolemy domains. We have successfully demonstrated cosimulation of VHDL models, synthesized DSP assembly code, and synthesized embedded C code.

We have completed a new loop scheduler in Ptolemy that works on acyclic SDF (synchronous dataflow) graphs and constructs single appearance schedules that are optimized for buffer memory consumption. In collaboration with Shuvra Bhattacharyya (now of Hitachi America), we have developed a new, general, latency constrained resynchronization (LCR) algorithm for 2 processor systems that is capable of handling delays. This algorithm is provably optimal, and the proof is relatively simple.

In this reporting period, this work produced one book [2], one Ph.D. thesis [17], four conference papers [9] [11] [20] [25], and three technical memoranda [3] [4] [21].

2.5. Ptolemy software

The Ptolemy software serves as both a laboratory for experimentation and a mechanism for disseminating results. During this reporting period, we have made a large number of enhancements to the software, and have completed one major release, numbered 0.6, completed on April 12, 1996.

The Ptolemy 0.6 release consists of approximately 3000 files containing 400,000 lines and 9 Mb of source code (compressed). See our World Wide Web server http://ptolemy.eecs.berkeley.edu or finger ptolemy@ptolemy.eecs.berkeley.edu for complete information.

The documentation for Ptolemy0.6 is available in PostScript, HTML and PDF, together with updated summary sheets, answers to frequency asked questions, a quick tour, and a tutorial. We have set up a Usenet read news group called comp.soft-sys.ptolemy. Postings to our mailing list ptolemy-hackers@ptolemy.eecs.berkeley.edu are cross-posted to the comp.soft-sys.ptolemy and visa-versa. Postings to the news group and the e-mail group are searchable from our World Wide Web site.

2.6. Tycho software

Tycho is an object-oriented syntax manager with an underlying heterogeneous technical rationale. It provides a number of editors and graphical widgets in an extensible, reusable framework. The editors for textual syntaxes are modeled after emacs in the sense the emacs key bindings are used whenever possible. However, they make more extensive use of menus, windows, and dialogs than emacs. Also, the intent is that visual editors and visualization tools will be fully integrated, something that would be difficult to accomplish with emacs in its current form. Editors for visual syntaxes will be more diverse. The system documentation is integrated, using a hypertext system compatible with the worldwide web. Tycho was originally conceived for use with the Ptolemy system, a heterogeneous design environment from U.C. Berkeley, but it has grown into a system that is useful on its own. Tycho has been used extensively in the development of the Tycho software itself.

Tycho is written in Itcl, also called [incr Tcl], developed by Michael McLennan of AT&T. Itcl is an object-oriented extension of Tcl, a “tool command language” written by John Ousterhout of U.C. Berkeley. The window toolkit Tk and its object-oriented extension Itk are also used extensively.

The Tycho0.1 release is the first public release of Tycho as a stand-alone system. It runs under the vanilla Itcl 2.0 and 2.1 with no changes to the executable. It also runs under the Ptolemy system, a design environment distributed by the same group that developed Tycho. Tycho0.1 can be obtained from http://ptolemy.eecs.berkeley.edu/tycho/Tycho.html.

DESIGN METHODOLOGY FOR DSP

Edward A. Lee, Principal Investigator

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

Final Report 1995-96, Micro Project #95-088
Industrial Sponsors: Alta Group of Cadence Design Systems, Dolby Laboratories,
Rockwell International, Mentor Graphics, Philips

ABSTRACT

The focus of this project is on design methodology for complex real-time systems where a variety of design methodologies and implementation technologies must be combined. Design methodologies are encapsulated in one or more models of computation, while implementation technologies are implemented as synthesis tools. Applications that use more than one model of computation and/or more than one synthesis tool are said to be heterogeneous. Hardware/software codesign is one example of heterogeneous design. Project results have been disseminated via the *Ptolemy* software system, in addition to papers. The overall Ptolemy project is fairly large, with additional support from DARPA, SRC, and a number of other companies, and is strongly collaborative. The MICRO project has focused on real-time signal processing, although the larger project is broader.

1. THE CONTEXT

The objectives of the Ptolemy Project include many aspects of designing signal processing and communications systems, ranging from designing and simulating algorithms to synthesizing hardware and software, parallelizing algorithms, and prototyping real-time systems. Research ideas developed in the project are implemented and tested in the Ptolemy software environment. The Ptolemy software environment, which serves as our laboratory, is a system-level design framework that allows mixing models of computation and implementation languages.

In designing digital signal processing and communications systems, often the best available design tools are domain specific. The tools must be able to interact. Ptolemy allows the interaction of diverse models of computation by using the object-oriented principles of polymorphism and information hiding. For example, using Ptolemy, a high-level dataflow model of a signal processing system can be connected to a hardware simulator that in turn may be connected to a discrete-event model of a communication network.

A part of the Ptolemy project concerns programming methodologies commonly called “graphical dataflow programming” that are used in industry for signal processing and experimentally for other applications. By “graphical” we mean simply that the program is explicitly specified by a directed graph where the nodes represent computations and the arcs represent streams of data. The graphs are typically hierarchical, in that a node in a graph may represent another directed graph. In Ptolemy the nodes in the graph are subprograms specified in C++.

It is common in the signal processing community to use a visual syntax to specify such graphs, in which case the model is often called “visual dataflow programming.” But it is by no means essential to use a visual syntax.

Hierarchy in graphical program structure can be viewed as an alternative to the more usual abstraction of subprograms via procedures, functions, or objects. It is better suited than any of these to a visual syntax, and also better suited to signal processing.

Some other examples of graphical dataflow programming environments intended for signal processing (including image processing) are Khoros, from the University of New Mexico (now distributed by Khoral Research, Inc.), the signal processing worksystem (SPW), from the Alta Group at Cadence (formerly Comdisco Systems), COSSAP, from Synopsys (formerly Cadis), and the DSP Station, from Mentor Graphics. MATLAB from The MathWorks, which is popular for signal processing and other applications, has a visual interface called SIMULINK. These software environments all claim variants of dataflow semantics.

Most graphical signal processing environments do not define a language in a strict sense. In fact, some designers of such environments advocate minimal semantics, arguing that the graphical organization by itself is sufficient to be useful. The semantics of a program in such environments is determined by the contents of the graph nodes, either subgraphs or subprograms. Subprograms are usually specified in a conventional programming language such as C. Most such environments, however, including Khoros, SPW, and COSSAP, take a middle ground, permitting the nodes in a graph or subgraph to contain arbitrary subprograms, but defining precise semantics for the interaction between nodes. We call the language used to define the subprograms in nodes the *host language*. We call the language defining the interaction between nodes the *coordination language*.

Many possibilities have been explored for precise semantics of coordination languages. Many of these limit expressiveness in exchange for considerable advantages such as compile-time predictability. In Ptolemy, a *domain* defines the semantics of a coordination language, but domains are modular objects that can be mixed and matched at will. Thus we gain flexibility without the sloppiness of unspecified semantics in the coordination language.