DESIGN METHODOLOGY FOR DSP

Edward A. Lee

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

Final Report 1991-92, Micro Project #91-085
Industrial Sponsors: Bell Northern Research, Comdisco Systems, Dolby Laboratories, Motorola, 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. In previous years, we have succeeded with a class of applications with deterministic control structure. This year, we have focussed on a broader class of applications involving run-time decisions and asynchronous real-time operations. The overall research problem divides into investigating human interfaces for specifying real-time systems (the language), developing algorithms for automated implementation (the compilation), and developing suitable target architectures (the architecture). The project has so far been extremely productive. Three versions of the Ptolemy software system have been widely distributed.

1. MOTIVATION

Digital Signal Processing, traditionally the domain of Government labs, large telecommunications companies, and multinational oil companies, has broken out into the much broader computer-user community. To better integrate computers with the telecommunications network, and to invent real-time interfaces better suited to human perception, DSP is mandatory. The beginnings of this evolution are evident; workstations and PCs come with A/D converters, DSPs, telephone and ISDN interfaces, and FAX modems. But to those of us who understand DSP, the evolution is frustratingly slow. What holds it back?

Our opinion, and main motivation for this project, is that general-purpose computing paradigms do not fit DSP very well. The C and Unix world cannot alone drive the DSP engine. Fundamentally different approaches to software and hardware design are required. Intuitively, the DSP community has always known this. The persistence of DSP assembly languages and Fortran at one end of a spectrum, and block diagram languages at the other, speak of a general mismatch with what computer science is offering as the newest and best computing paradigms. Nonetheless, modern software engineering techniques, particularly object-oriented programming, enable construction of high-level, application-specific environments that encapsulate a great deal of expertise.

Dataflow techniques have been applied to DSP in the guise of "block-diagram languages" since its very earliest days. Dataflow representation of algorithms, in fact, is very natural in DSP, appealing *even without the motivation of concurrency.* Of course, automatically exploiting concurrency can only increase the appeal. This project exploits properties of DSP applications to develop design methodologies for the development of hardware and software for real-time DSP. A principal focus of the effort is on scheduling and compilation of parallel computations.

2. RESULTS OF MICRO SUPPORT

Algorithms with predictable control flow have been successfully addressed using the synchronous dataflow (SDF) model of computation [23] [22]. 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* [20][7][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.

Ptolemy is the third generation design environment at Berkeley for signal processing, after Blosim [25] and Gabriel [4][21]. It is unique in permitting *multi-paradigm* computation. This allows us to preserve the benefits of the approach taken in Gabriel, while broadening the application domain. More importantly, it permits in-depth experimentation with a variety of computational models, and with the interaction between computational models in a heterogeneous system.

2.1. Overview of recent publications and software

The most visible concrete output from this group is the Ptolemy software system, described in detail in [6][8] and in the manual. Version 0.4 of Ptolemy was released in December of 1992. It is distributed through our Industrial Liaison Program office and electronically by anonymous FTP. The manual (called *The Almagest*) has grown to more than 500 pages. Our strategy of aggressively automating the documentation has paid off. Perhaps more importantly, the software has formed the testbed for a number of research projects, including four masters projects [14] [15]

[33] [34] and two Ph.D. dissertations [5][11] completed during this reporting period.

Soonhoi Ha's Ph.D. dissertation [11] describes innovative compile-time parallel scheduling heuristics that have been implemented in Ptolemy. Philip Bitar's Ph.D. dissertation [5] explores a communication network design for parallel computers, and makes extensive use of Ptolemy for simulating the system-level design. Asawaree Kalavade's masters thesis [14] develops a Ptolemy-based methodology for hardware/software codesign. Phil Lapsley describes in his masters thesis [15] a block-diagram-level symbolic debugger for real-time DSP programs. Gregory Walter describes a mixed signal processing and packet-switched communication network simulation for real-time transmission of speech over ATM networks [33] (see figure 1). Anthony Wong developed a library of functional blocks for the Motorola DSP96k processor [34].

Several undergraduate independent study projects were also based on Ptolemy, including one by Saurav Chatterjee, who mixed real-time code generated by Ptolemy for a microcomputer running the VxWorks operating system with workstation and Macintosh controller programs. Another undergraduate, Mike Chen, completed a project on Xilinx programming for the Ariel S56x card, and has now joined our group as a graduate student. Mei-Tjing Huang worked on a robust implementation of Elliptic filter design for inclusion in our forthcoming filter design addition to Ptolemy. Chih-Tsung Huang ported the entire Gabriel library of code generation stars for the Motorola DSP56k and DSP96k processors to Ptolemy. Alireza Khazeni has built a prototype fixed-point simulation capability into Ptolemy.

We have made considerable progress on formal and heuristic techniques for efficient code generation from synchronous dataflow graphs [1][2][3][28]. We have worked with one of our sponsors, Comdisco Systems, who has enhanced and commercialized some of this technology [29]. We have also made fundamental advances on code generation from dataflow graphs with run-time dynamics in the control flow [7][9][12]. Our effort on parallel scheduling has yielded three useful parallel schedulers implemented in Ptolemy [11][12][30][31]. These use principles outlined in [19] to minimize the effort expended at run time by maximizing the compile-time analysis. We recently devised a multidimensional extension of synchronous dataflow [16][17] that promises to significantly improve our ability to exploit data parallelism.

We have presented or will present Ptolemy-related work at a number of technical conferences [7][9][10][13][16][18][32]. In this reporting period, six papers have appeared or been accepted in archival journals [1][6][12][28][30][31].

In addition, Ptolemy is being used in both our graduate and undergraduate signal processing classes. In the Spring semester of 1992, approximately 90 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 [18].

2.2. Status of Ptolemy

In addition to the discrete-event (DE), dynamic dataflow (DDF), synchronous dataflow (SDF), and the Thor hardware simulation domain, Ptolemy 0.4 includes a number of "code generation" domains. Each is intended for prototyping of real-time implementations. CGC synthesizes C code, and is currently being targeted to the AT&T DSP3, the Thinking Machines CM5, and the TI TMS320C40. The "Sproc" domain synthesizes parallel implementations of signal processing systems on the Sproc multiprocessor from Star Semiconductor. The CG56 domain synthesizes assembly code for the Motorola DSP56000 family. We use it extensively to build real-time implementations on the Ariel DSP56x s-bus card and a prototype multiprocessor system from Dolby Laboratories, one of our sponsors. The CG96 domain synthesizes assembly code for the Motorola DSP96000 family and is currently being targeted to our own four-processor system [32]. A sophisticated parallel scheduler developed by Gil Sih [30][31], one of our recent graduates, is compatible with all these code generators. A scheduler that supports run-time dynamics has been developed and code generation domains compatible with it are being developed [11].

Ptolemy uses a new facility called a "target" to control the execution of universes; this facility is used to specify the behavior and requirements of a system for which code is being generated. It can be used to configure an abstract target for the parallel sched-

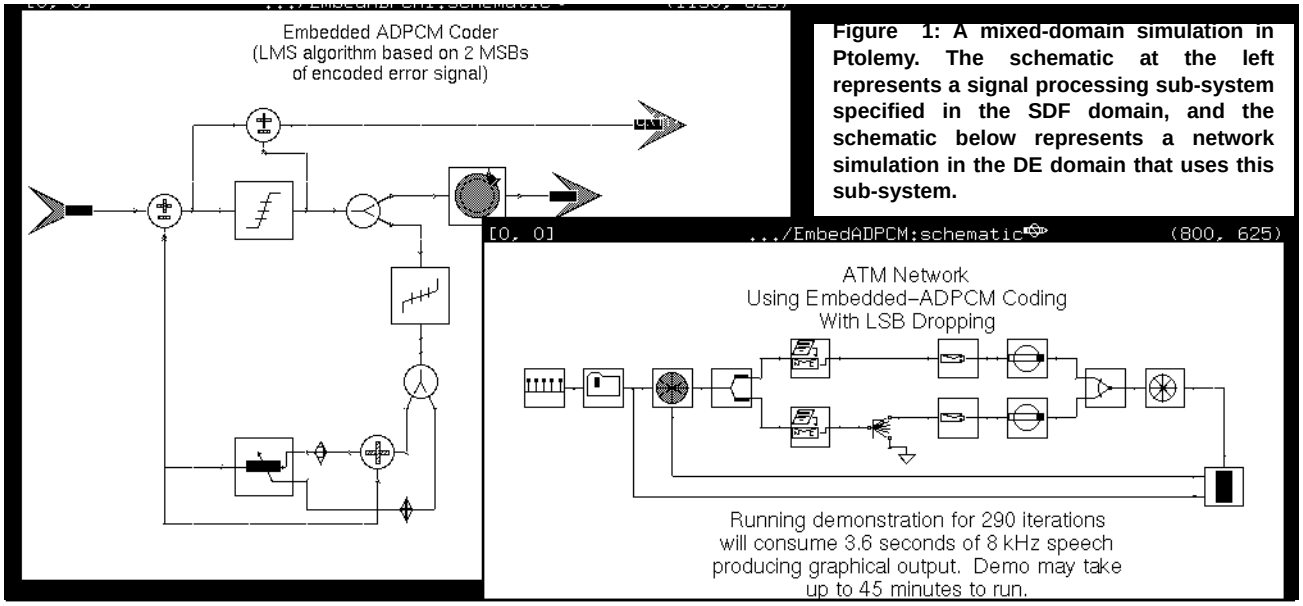

Figure 1: A mixed-domain simulation in Ptolemy. The schematic at the left represents a signal processing sub-system specified in the SDF domain, and the schematic below represents a network simulation in the DE domain that uses this sub-system.

uler, and to select schedulers for the SDF domain. End-users can develop new targets that model their own hardware. We have developed several workstation targets, including one distributed simulation target, and targets for the Ariel DSP56x, the CM5, the AT&T DSP3 (with help from AT&T), and our own OMA [32]. In addition, targets can be simulated hardware specified in the Thor domain [13][14]. Both the interpreter and graphical interface have new commands to manipulate targets.

Ptolemy also now supports "packets", which are particles of arbitrary type. This facility permits stars to exchange arbitrary objects, not just integer, real, and complex data as before. This can be used, for example, to support vectors, arrays, images, video frames, or packets in the simulation of a communication network. This facility was used for the ATM network simulation in [33]. Note, however, that vectors and arrays have always been supported in the SDF domain through the use of SDF principles. Many of the demos operate on vectors and arrays.

A number of technical improvements have been made to the code. As a result, Ptolemy now compiles under AT&T cfront, as ported to Suns, and has been successfully retargeted to HP snake workstations.

3. CURRENT PROJECTS

The following specific projects are funded by the MICRO program in cooperation with the industrial sponsors.

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. Extensible graphical interfaces

The graphical interface of Ptolemy is being enhanced by the incorporation of an interpreted language and associated Motif-compliant X window toolkit called Tcl/Tk [26][27]. This will enable, for example, the designer of a custom star to customize the parameter editing. As a specific demonstration, we have developed a prototype program called Xpole (done by Kennard White) that edits digital filter parameters by graphically manipulating pole-zero diagrams. In addition, the designer of a "target" can customize the control panel that controls the execution of code on that target, as demonstrated in [15].

3.3. 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.

3.4. Run-time dynamics

We are developing scheduling techniques that minimize run-time overhead while allowing dynamic flow of control. One approach that we are pursuing is to hierarchically mix diverse scheduling techniques. Another approach we are pursuing extends the SDF formal methods to dynamic dataflow graphs.

3.5. 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. One approach we are pursuing is based on a scheduler invented at Star Semiconductor, one of our sponsors. It uses a statically constructed compact representation of the schedule to dynamically invoke blocks in a appropriate order. Another technique we are pursuing is based on the formalism of synchronous dataflow. The objective is to construct compact schedules with nested iteration that balance code space and data memory requirements.

3.6. Image and video signal processing in ptolemy

Dataflow representations work well for one-dimensional signal processing applications because streams of tokens are a natural representation for signals. However, multidimensional signals are not so easily represented. The aim of this project is to find graphical representations for multidimensional signal processing algorithms that are amenable to automated parallel implementation. Approaches that are being studied include multidimensional streams, data-parallel representations, and a variety of graphical representations for iterated computations. A major objective is to be able to synthesize real-time implementations on VLIW video signal processors such as the Philips VSP.

3.7. Heterogeneous system design

We are enhancing Ptolemy's capability to mix design styles by incorporating VHDL and Silage code generation domains for hardware design. The VHDL domain will couple to commercial VHDL simulators. The Silage domain will couple to high-level synthesis tools developed at Berkeley in Prof. Rabaey's group.

4. REFERENCES

[1] S. Bhattacharyya and E. A. Lee, "Scheduling Synchronous Dataflow Graphs for Efficient Looping," **to appear** in *J. of VLSI Signal Processing*, 1993.

[2] S. Bhattacharyya, Soonhoi Ha, and E. A. Lee, "Single Appearance Schedules for Synchronous Dataflow Programs," Memorandum UCB/ERL M93/4, January 10, 1993.

[3] S. Bhattacharyya and E. A. Lee, "Memory Management for Synchronous Dataflow Programs," Technical Report UCB/ERL M92/128, EECS Dept., UC Berkeley, November 18, 1992

[4] 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.

[5] P. Bitar, "Combining Windows, A Performance Evaluation of Design Options," **Ph.D. Dissertation**, EECS Dept., University of California, Berkeley, CA 94720, December, 1992.

[6] J. Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, "Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems", **to appear** in *International Journal of Computer Simulation*, special issue on "Simulation Software Development," 1993.

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

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

[9] J. Buck and E. A. Lee, "The Token Flow Model," presented at *Data Flow Workshop*, Hamilton Island, Australia, May, 1992.

[10] J. Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, "Ptolemy: A Mixed-Paradigm Simulation/Prototyping Platform in C++", *Proc. C++ At Work Conference*, Santa Clara, CA, November, 1991.

[11] S. Ha, "Compile-Time Scheduling of Dataflow Program Graphs with Dynamic Constructs," **Ph.D. Dissertation**, EECS Dept., University of California, Berkeley, CA 94720, April 1992.

[12] Soonhoi Ha and E.A. Lee, "Compile-Time Scheduling and Assignment of Dataflow Program Graphs with Data-Dependent Iteration," *IEEE Transactions on Computers*, November, 1991.

[13] A. Kalavade, E.A. Lee, "Hardware/Software Co-Design Using Ptolemy - A Case Study", Proceed*ings of the IFIP International Workshop on Hardware/Software Co-Design*, Grassau, Germany, May 19-21, 1992.

[14] A. Kalavade, "Hardware/Software Codesign Using Ptolemy", **MS Report**, Electronics Research Laboratory, University of California, Berkeley, CA 94720, December, 1991.

[15] P. D. Lapsley, "Host Interface and Debugging of Dataflow DSP Systems", **MS Report**, Electronics Research Laboratory, University of California, Berkeley, CA 94720, December, 1991.

[16] E. A. Lee, "Multidimensional Streams Rooted in Dataflow", *Proc. IFIP Working Conference on Architectures and Compilation Techniques for Fine and Medium-Grain Parallelism*, Orlando, FL, January, 1993.

[17] E. A. Lee, "Data Parallelism in Graphical Signal Flow Representations of Algorithms", Technical Report UCB/ERL M92/110, EECS Dept., UC Berkeley, August 13, 1992.

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

[19] E. A. Lee, "Static Scheduling of Data-Flow Programs for DSP," in *Advanced Topics in Data-Flow Computing*, ed. J.-L. Gaudiot and L. Bic, Prentice-Hall, 1991.

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

[21] 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.

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

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

[24] Brian Link, *The Waveguide Toolkit*, **MS Report**, EECS Dept., University of California, Berkeley, CA 94720, December, 1992.

[25] D. G. Messerschmitt, "A Tool for Structured Functional Simulation", *IEEE Journal on Selected Areas in Communications*, SAC-2(1), January, 1984.

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

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

[28] J. Pino, S. Ha, E. Lee, J. Buck, "Software Synthesis for DSP Using Ptolemy", invited paper in the *Journal on VLSI Signal Processing*, special issue on "Synthesis for DSP", **to appear**.

[29] D. G. Powell, E. A. Lee, W. C. Newman, "Direct Synthesis of Optimized DSP Assembly Code from Signal Flow Block Diagrams," *Proceedings of ICASSP '92,* San Francisco, March, 1992.

[30] G. C. Sih and E.A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures", **to appear**, *IEEE Trans. on Parallel and Distributed Systems*, 1993.

[31] G. C. Sih and E. A. Lee, "Declustering: A New Multiprocessor Scheduling Technique," **to appear** in *IEEE Trans. on Parallel and Distributed Systems*, 1993.

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

[33] Gregory Walter, *ATM, Speech Coding, and Cell Recovery*, **MS Report**, EECS Dept., University of California, Berkeley, CA 94720, December, 1992.

[34] Anthony Wong, "A Library of DSP Blocks and Applications for the Motorola DSP96000 Family", **MS Report**, Plan II, EECS Dept., UC Berkeley, CA 94720, May, 1992.