DESIGN METHODOLOGY FOR DSP

Edward A. Lee

Department of Electrical Engineering and Computer Science

University of California, Berkeley CA 94720

Final Report 1990-91, Micro Project #90-089

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. Two versions of the Ptolemy software system have been widely distributed.

MOTIVATION

Digital signal processing, traditionally the domain of Government labs, large telecommunications companies, and multinational oil companies, is about to break 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 in on scheduling and compilation of parallel computations.

RESULTS OF MICRO SUPPORT

Algorithms with predictable control flow have been successfully addressed using the synchronous dataflow (SDF) model of computation [10] [11]. 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, but this effort is barely beginning. The second is to extend the analytical techniques of SDF to dynamic dataflow graphs. A *token flow model* [15] [16] 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.

The Gabriel program, developed with MICRO support, relies on the SDF model of computation. It should properly be viewed as the second generation of such software environments at Berkeley. The first generation (Blosim [19]) is a simulation environment that uses dynamically scheduled dataflow programming principles. Gabriel, by contrast, is a real-time prototyping environment for multiprocessor DSP systems. Efficiency demands predictable control flow. The third generation (Ptolemy) is the next logical step that we hope will solve many of the problems with the first two systems by 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.

Overview of Ptolemy

Ptolemy, a system-level design framework, is the successor to Gabriel. Its ambitious objectives 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, and real-time prototyping. To accommodate this breadth, Ptolemy must support a plethora of widely differing design styles. Hence, the core of Ptolemy is an object-oriented toolkit and library that makes few assumptions about the system to be modeled; rather, standard interfaces are provided for generic objects and more specialized, application-specific objects are derived from these.

Each design style is supported by one or more *domains*, where a domain is an extensible library of functional blocks and a model of computation. For instance, the SDF domain models synchronous multirate signal processing applications. Asynchronous signal processing applications can be built using the dynamic dataflow (DDF) domain in Ptolemy. An example is shown in figure 1. Here, a phase-locked loop controls the downsampling of an amplitude-shift-keyed (ASK) signal.

The DDF domain uses run-time scheduling, whereas the SDF domain uses compile-time scheduling. In Ptolemy, these two scheduling techniques can be combined so that the cost of run-time scheduling is incurred only where it is absolutely required. This figure also shows the hierarchy supported by the graphical interface.

A radically different application is shown in figure 2. It uses the discrete-event (DE) domain in Ptolemy, in which signals consist of sequences of data objects with associated time stamps. The execution is chronological rather than data-driven. The figure shows a system with two queues and two servers, where the first queue discards incoming events when it fills up, whereas the second queue blocks arrivals instead. The two signal plots monitor the sizes of the queues.

The DE domain in Ptolemy is used not just for queueing systems, but also to model communication networks and protocols. It is equally useful for high-level modeling of hardware systems, where

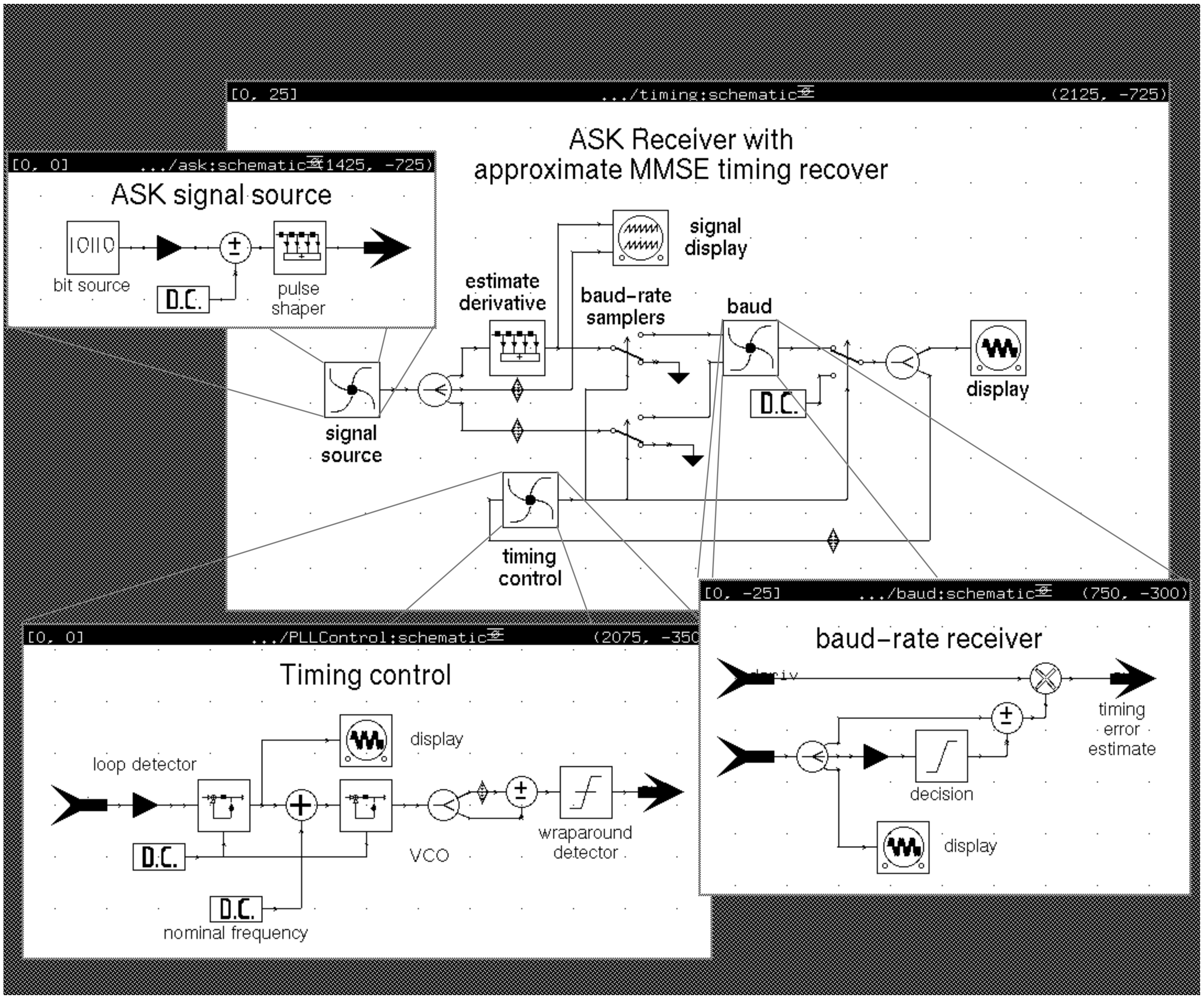

FIGURE 1. An asynchronous signal processing application using the dynamic dataflow domain in Ptolemy.

contention for shared resources, or asynchronous behavior is of interest.

Overview of recent publications and software

The most visible concrete outputs from this group are the Gabriel and Ptolemy software systems, described in detail in [5], [12] and [2]. Version 0.2 of Ptolemy was released in March 1991, and version 0.3 in December. These systems have been distributed through our Industrial Liaison Program office and electronically to hundreds of sites. Perhaps more importantly, they have formed the testbed for a number of research projects, including two masters projects and one Ph.D. dissertation completed just in the last year. Gilbert Sih's Ph.D. dissertation describes innovative compile-time parallel scheduling heuristics that have been implemented in Ptolemy [20]. Shuvra Bhattacharyya produced a Masters thesis on detecting and exploiting iteration in SDF graphs [1]. Phil Lapsley describes in his Masters thesis a block-diagram-level symbolic debugger for real-time DSP programs [9]. Two more master's theses are near completion. Several undergraduate independent study projects were also based on Ptolemy and Gabriel.

We described details of the implementation of Ptolemy at two technical conferences [5] [6]. Applications to multirate signals processing [3] and to speech processing [4] were also published. The use of Ptolemy in the classroom at Berkeley will be described at ICASSP in March [17]. One book chapter [13], one conference paper [7], and one journal article [8] on multiprocessor scheduling have been published, and another has been accepted [21]. Our still theoretical work with the Boolean Dataflow domain in Ptolemy has been described in a journal article [15] and a conference [16]. Our joint work with Comdisco Systems on optimized assembly code synthesis for the Motorola DSP56000 family will also be described at ICASSP [18]. Finally, an early paper on multiprocessor hardware design for compile-time scheduling has been reprinted in a book [14].

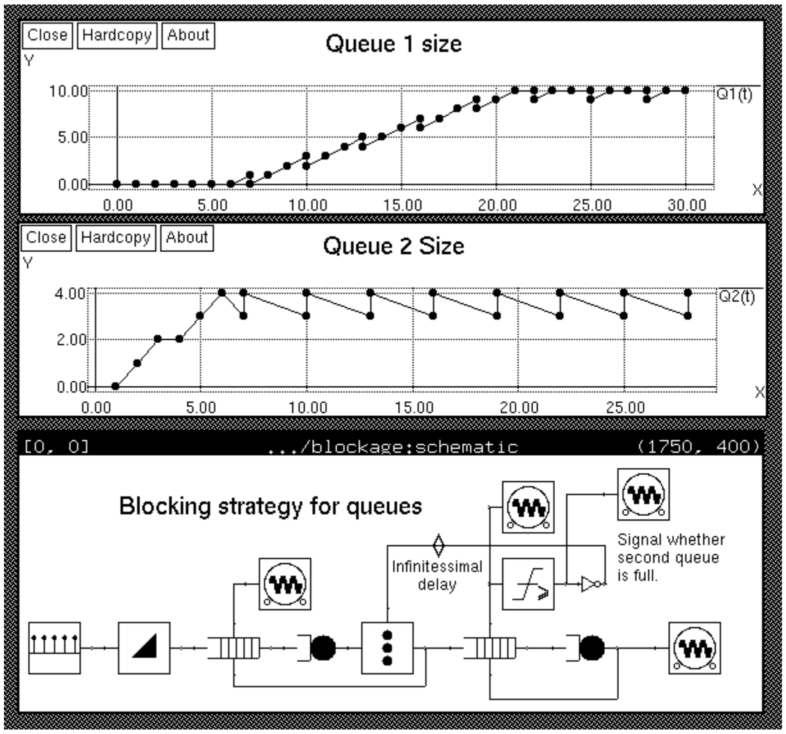

FIGURE 2. A queueing system implemented in the discrete-event domain in Ptolemy.

Status of Ptolemy

In addition to the discrete-event (DE), dynamic dataflow (DDF), and synchronous dataflow (SDF) domains that were included in version 0.2, Ptolemy now includes the Thor and CG domains. The Thor domain is built on the Thor hardware simulator from the VLSI Group at Stanford University. It supports the simulation of circuits from gate level to behavioral level. With the addition of the Thor domain, Ptolemy users can simulate hardware, and can mix hardware simulations with other simulations.

Also the Code Generation (CG) domain has been added. The CG domain provides the base classes for future code generation capabilities. The CG domain itself does not actually generate code for any processor; it does, however, permit experimentation with parallel schedulers and with the code generation infrastructure. A sophisticated parallel scheduler developed by Gil Sih, one of our recent graduates, and a tool for displaying Gantt charts are included with the CG domain.

Ptolemy uses a new facility called a "target" to control the execution of universes; this facility will eventually be used to specify, for example, the behavior of a system for which code is being generated. At this time, it can be used to configure an abstract target for the parallel scheduler, and to select schedulers for the SDF domain. 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. 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.

There are many new stars in version 0.3. In the SDF domain, there are a series of image processing stars that use the packet facility to communicate two-dimensional image objects: Dct, DctInv, DiffImage, DisplayImage, DisplayRgb, Dpcm, DpcmInv, MedianImage, MotionCmp, MotionCmpInv, ReadImage, ReadRgb, Rgb2Yuv, RunLen, RunLenInv, and Yuv2Rgb. Other new SDF stars include Average, BitsToInt, ComplexAverage, Compress, CxDiff, CxGain, CxLMS, CxLMSPlot, CxRaisedCos, CxTable, CxWaveForm, FloatPad, FloatTable, Hilbert, IntTable, IntToBits, LMSPlot, ReadFile, Reverse, Trainer, and Waterfall. There are a number of new demos under pigi that show off these new stars.

New DE stars include And, Arbitrate, FIFOQueue, HandShake, Invert, Match, Or, Packetize, PassGate, PriorityQueue, Stack, TestLevel, and UnPacketize.

The DE domain has been extensively re-worked. Problems in the 0.2 release with the order of event processing have been corrected. Handling of simultaneous events is substantially improved. A new technique is used for DE stars to declare their data dependencies. Also, it is now possible to load dynamically linked stars into pigi if you edit them.

A number of technical improvements have been made to the code. We now uses the g++ “#pragma interface” facility, which results in substantially smaller object files. Also, Ptolemy now compiles under AT&T cfront version 2.1, as ported to Suns.

Ptolemy in the classroom

Last Spring, Ptolemy was used by 24 students in our graduate statistical signal processing class, EE225a. A network of DEC workstations was set up for this purpose. The students were given biweekly assignments to design and implement signal processing systems using Ptolemy. To illustrate the level of sophistication of these experiments, I will describe one of the assignments.

Students first generate an AR random process and construct an optimal one-step forward linear predictor for this process. They then implement a similar adaptive linear predictor using the LMS algorithm, and compare these by measuring the coding gain that can be achieved in each case. One possible solution is shown in figure 3. From left to right, top to bottom, we see a white Gaussian noise generator, an IIR filter to generate the AR process, a fork to broadcast the signal, two unit delays (small diamonds), an adaptive LMS filter, a fixed FIR filter, two subtracters, another fork, three power estimation subsystems (galaxies), three “dB” blocks to convert the power measurement to dB scale, and a display block.

The experiment is repeated with three random processes: the first is an AR process with an extremely peaked power spectrum, the second is an AR process with a much flatter power spectrum, and the third is a segment of voiced speech. The first and last result in much higher prediction gains than the middle signal. The students are asked to explain this phenomenon.

CURRENT PROJECTS

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

Multimedia Network Interfaces for Workstations

Unix workstations are not well suited to hard-real-time computing. We are therefore augmenting our workstations with an Ariel card containing a Motorola DSP56000 and Xilinx FPGA, both of which will be programmed using Ptolemy. To test real-time applications, we are interfacing this card to a 56kbps wireless radio modem and an ISDN terminal adaptor. The DSP will manage encoding of real-time signals such as speech, audio, and video, and will multiplex these signals with IP packets used for data communications. The resulting infrastructure should be ideal for developing multimedia applications.

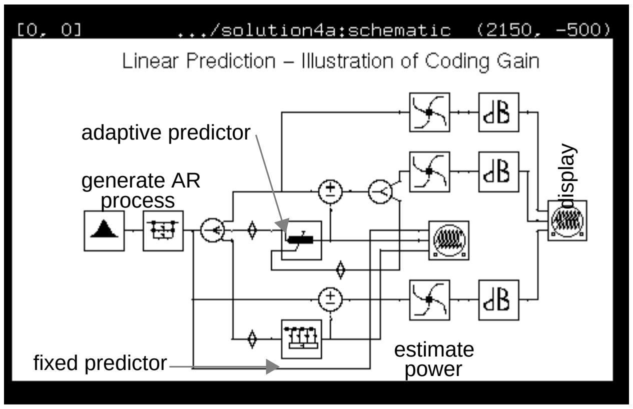
[0, 0] .../solution4a:schematic (2150, -500)
Linear Prediction – Illustration of Coding Gain
adaptive predictor
generate AR process
display
fixed predictor
estimate power
dB
dB
dB

FIGURE 3. : Adaptive and fixed linear prediction on an AR process.

Mixed-Domain Scheduling in Ptolemy

One of the key objective of the Ptolemy design environment is to support coexistence and interaction of diverse computational models, called *domains*. For instance, a dataflow domain can exist within a discrete-event domain, so that for example a signal processing subsystem can be cleanly incorporated within a communication network simulation. A Ptolemaic domain is implemented as a C++ object called a *WormHole*. We have designed Wormholes and their interconnection mechanism for mixed domain applications. A Wormhole is derived from the class Star, and behaves externally like any other star. Internally, however, it encapsulates an entire foreign domain invisible from the outside universe. The internal computational model can be totally different from the external model, in that the specification language, semantics, and scheduling paradigm can be totally different.

Since the kernel makes no assumptions about the internals of a Wormhole but its external abstraction, Ptolemy can support an effectively unlimited number of different scheduling paradigms. Currently, Ptolemy supports four domains: two timed domains (DE, THOR) and two untimed domains (SDF, DDF). We have been performing several simulations that mix those domains and evaluating the design of Wormhole interface. Mixed domains are also being developed for real-time implementation of systems with run-time decision making.

Optimized Code Generation

The goal of this project is to create a new domain in Ptolemy that synthesizes efficient assembly code for programmable DSPs. In this domain the user will be able to target various architectures including stand-alone DSPs, workstations, and DSP/workstation combinations. The DSPs to be targeted initially are Motorola DSP56001 and 96002. Direct assembly code generation for these processors is currently possible using Gabriel, the predecessor to Ptolemy. However, the code generated by Gabriel has too much overhead for many applications, particularly those constructed from fine-grain dataflow graphs, and must be painstakingly written in assembly code. Our approach will be to retarget a public-domain C compiler to produce block definitions that do not commit to the allocation of registers. These block definitions will then be processed by a back-end code generator [18].

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.

Heterogeneous Signal Processing Systems

In this project, we are studying a new type of multi-processor DSP architecture made by Star Semiconductor Corporation, and called the Sproc. This architecture is unique in that a single central memory is shared without contention among four processors on the same die. Programming is via block diagrams with semantics closely related to dataflow graphs. Blocks are "temporally" partitioned among processors, meaning for example that block A will run first on processor 1, then on processor 2, then 3, then 4, then back to 1. A "turnstile" (semaphore) is used ensure that no two processors simultaneously execute the same block. Double buffering is used to ensure that data is not overwritten before it is read. This achieves global synchronization and parallelism via a mechanism quite different from that in Gabriel.

The objective of this project is first to study the Sproc architecture and software methodology and test it against some applications. Then we will experiment with variations on the scheduling, partitioning, and code generation approaches, using Ptolemy as the testbed. The long term objective is to enhance the coding methodology to cleanly support applications that mix real-time control with synchronous signal processing.

REFERENCES

[1] S. Bhattacharyya, "Scheduling Synchronous Dataflow Graphs for Efficient Iteration", Master's Thesis, EECS Dept., Univ. of Calif., Berkeley, May, 1991.

[2] 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.

[3] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Multirate Signal Processing in Ptolemy", *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing*, Toronto, Canada, April, 1991.

[4] J. Buck, S. Ha, E.A. Lee, D.G. Messerschmitt, "Ptolemy: A mixed Paradigm Simulation/ Prototyping Platform", *Proc. of Speech Tech 1991*, New York, NY, April 23-25, 1991.

[5] J. Buck, S. Ha, E. A. Lee, and D.G. Messerschmitt, "Ptolemy: A Platform for Heterogeneous Simulation and Prototyping, *Proc. 1991 European Simulation Conference*, Copenhagen, Denmark, June 17-19, 1991.

[6] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: A Mixed-Paradigm Simulation/Prototyping Platform in C++", C++ Conference, Santa Clara, CA, Nov. 1991.

[7] S. Ha, E. A. Lee, "Quasi-Static Scheduling for Multiprocessor DSP", *Proc. of ISCAS*, Singapore, June 1991.

[8] 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.

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

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

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

[12] 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.

[13] 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.

[14] E. A. Lee and J. C. Bier, "Architectures for Statically Scheduled Dataflow", reprinted in *Parallel Algorithms and Architectures for DSP Applications,* ed. M. A. Bayoumi, Kluwer Academic Pub., 1991.

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

[16] E. A. Lee, "Consistency in Dataflow Graphs", *Proc. of the 1991 Conference on Application Specific Array Processors*, Barcelona, Spain, Sept. 1991.

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

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

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

[20] G. C. Sih, "Multiprocessor Scheduling to Account for Interprocessor Communication", Ph.D. Thesis, ERL, UC Berkeley, CA 94720, April 22, 1991.

[21] G. Sih and E. A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," to appear in *IEEE Trans. on Parallel and Distributed Systems*, 1992.