August 30, 1993

Electrical Engineering and Computer Science

DSP Design Group — Summary of Research Activities

This document contains abstracts from the 1994 Research Summary, summarizing individual research projects. The work divides into three categories: (1) design methodology for signal processing and communications, (2) parallel computation, and (3) signal processing and communications applications. The abstracts are grouped in this order.

University of California at Berkeley

Design Methodology for Signal Processing and Communications:

Compiling Dataflow Programs for Multirate DSP into Compact Code

STUDENTS: **Shuvra S. Bhattacharyya and Joseph T. Buck**

POSTDOC: **Soonhoi Ha**

ADVISOR: **Edward A. Lee**

SPONSORS: **DARPA (J-FBI-90-073), NSF (MIP-9201605), AT&T Bell Laboratories, and SRC(93-DC-008)**

We are investigating techniques to compile dataflow programs for multirate signal processing. We have focused on the synchronous dataflow (SDF) model, in which the number of tokens produced and consumed by each functional block is fixed and known at compile time. In an SDF description, only a partial ordering of the computations, based on data precedences, is specified. This exposes parallelism and does not commit to a specific control-flow structure. We are examining techniques to determine the optimal loop hierarchy for an SDF program based on considerations of program compactness (static instruction count), data space compactness, and throughput.

We have developed formal techniques to construct the most compact loop organization for a large class of SDF graphs [1, 2]. To guide the scheduling process, a compiler may use some form of "clustering", in which multiple functional blocks are hierarchically grouped together, according to different criteria. We have developed a compiler scheduling framework for synthesizing minimum code-size schedules, and for incorporating arbitrary clustering strategies in a way that provably preserves minimum code-size [3]. Two applications of clustering that we have implemented are clustering to reduce buffering cost and clustering to increase the amount of data transfers that occur through registers rather than memory. We have implemented our scheduling framework in Ptolemy [4].

Our code-size minimization techniques do not apply to subgraphs of the input SDF graph that belong to a certain class of SDF graphs called "tightly interdependent graphs". Our preliminary observations suggest that tightly interdependent graphs occur rarely in practice; however we are exploring the minimization of code size for this class of graphs to provide a more complete solution. Other areas of work that we are pursuing are the extension of our scheduling techniques to parallel processing, and to more general dataflow models, such as the token flow model [5].

[1] S. Bhattacharyya and E. A. Lee, "Looped Schedules for Dataflow Descriptions of Multirate DSP Algorithms", Memorandum No. UCB/ERL M93/37, Electronics Research Laboratory, U. C. Berkeley, May, 1993.

[2] S. Bhattacharyya, J. Buck, S. Ha, and E. A. Lee, "Generating Compact Code from Dataflow Specifications of Multirate DSP Algorithms", Memorandum No. UCB/ERL M93/36, Electronics Research Laboratory, U. C. Berkeley, May 1993.

[3] S. Bhattacharyya, J. Buck, S. Ha, and E. A. Lee, "A Scheduling Framework for Minimizing Memory Requirements of Multirate DSP Systems Represented as Dataflow Graphs", to appear in *Proc. IEEE Workshop on VLSI Signal Processing*, Koningshof, Veldhoven, The Netherlands, October, 1993.

[4] J. Buck, S. Ha, E. A. Lee and D. G. Messerschmitt, "Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems", invited

paper to appear in the *International Journal of Computer Simulation*, special issue on “Simulation Software Development”, 1992.

[5] J. Buck and E. A. Lee, “Scheduling Dynamic Dataflow Graphs with Bounded Memory Using the Token Flow Model”, *Proc. ICASSP*, Minneapolis MN, April 1993.

Minimizing Buffering Requirements for Multirate Synchronous Dataflow Graphs

STUDENTS: **Shuvra S. Bhattacharyya and Praveen K. Murthy**

ADVISOR: **Edward A. Lee**

SPONSORS: **Star Semiconductor Corporation, MICRO, NSF (MIP-9201605), and DARPA (J-FBI-90-073).**

We are investigating the problem of minimizing the memory required for buffering along arcs of a synchronous dataflow (SDF) graph given the constraint that a minimum code-size, static uniprocessor schedule is desired. Initially, we are focusing on acyclic SDF graphs. For an acyclic SDF graph, a minimum code-size schedule can easily be constructed, but the number of distinct minimum code-size schedules increases combinatorially with the number of nodes, and these schedules can have vastly differing data space requirements.

For the class of chain-structured SDF graphs, our problem is similar to the problem of most efficiently multiplying a chain of matrices [2], for which a dynamic programming solution exists that has cubic time-complexity. In [1], we show that this technique can be adapted to our memory-minimization problem for chain-structured graphs to yield an algorithm that finds an optimal solution with time complexity that is cubic in the number of nodes. We have also developed a quadratic-time heuristic which usually performs quite well [1]. Both the dynamic programming solution and the heuristic extend easily to the more general class of well-ordered SDF graphs.

Our dynamic programming solution does not extend directly to handle general acyclic SDF graphs. Currently we are trying to establish whether or not our memory-minimization problem is NP-complete for general acyclic SDF graphs, and we are devising and evaluating various heuristics for the problem.

[1] P. Murthy, S. Bhattacharyya, and E. A. Lee, “Minimizing Memory Requirements for Chain-Structured Synchronous Dataflow Programs”, submitted to *ICASSP ’94*.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, “Introduction to Algorithms”, McGraw Hill, 1990.

The Token Flow Model: Compile-Time Scheduling of Dynamic Dataflow Graphs

STUDENT: **Joseph T. Buck**

ADVISORS: **Edward A. Lee and David G. Messerschmitt**

SPONSORS: **SRC(93-DC-008) and Rockwell**

Dataflow graphs have proven to be a very effective representation for problems in digital signal processing, both because the representation is natural for DSP researchers and because the representation exposes the parallelism of the algorithm and does not impose excessive constraints on the order of execution. When the actors in the graph are restricted to be synchronous (meaning that the number of tokens produced and consumed by each actor is fixed and known at “compile time”), powerful techniques exist for demonstrating the consistency of the graph, determining memory requirements, and scheduling its execution on one or multiple processing elements. These techniques are used effectively in Ptolemy and its predecessor system, Gabriel.

However, the synchronous dataflow (SDF) model is overly restrictive for digital signal processing, since some data-directed decision-making is usually required. Accordingly, this research project introduces a new, more general model called the token flow model, which was first conceived by Lee [1]. This model (also known as “Boolean-controlled dataflow” or BDF) admits graphs containing dynamic actors — that is, actors in which the number of tokens produced and consumed on each arc is a function of the data values of certain Boolean tokens. This larger class of graphs is Turing-equivalent, while SDF graphs are not. Accordingly, it permits a wider class of algorithms to be mapped onto multiple processors or onto VLSI hardware.

We have discovered necessary and sufficient conditions for BDF graphs to have bounded-length schedules, and sufficient conditions for such a graph to be scheduled in bounded memory [2]. We have extended techniques used for analysis of Petri nets to determine which arcs of a BDF graph require unbounded storage (and therefore dynamic memory allocation). Bounded cycle length is important in hard-real-time problems to assure that deadlines are always met; bounded memory permits static memory allocation, which is particularly important when mapping DSP algorithms that have decision-making onto VLSI hardware.

A clustering technique has also been developed that, given a dataflow graph, decomposes it into traditional control structures such as if-then-else, do-while, and iteration. This technique has been implemented in the Ptolemy system. Graphs that are completely clusterable are executed using a static schedule; remaining graphs are clustered as much as possible and then the clusters are dynamically executed. Implementation of single processor code generation using the BDF model is now being designed.

Implementation of the BDF model in Ptolemy for a single processor is being completed; this will permit data-dependent actors to be used in all code generation domains. Multi-processor scheduling techniques such as those of Sih [3] will be extended to handle BDF graphs with bounded-length schedules using a minimax optimality criterion (we aim to make the worst case run as rapidly as possible).

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

[2] 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.

[3] Gilbert C. Sih, “Multiprocessor Scheduling to Account for Interprocessor Communication,” Ph.D. Thesis, ERL, UC Berkeley, CA 94720, April 22, 1991.

Specification of Control Flow in Ptolemy

STUDENT: **Wan-Teh Chang**

ADVISORS: **Edward A. Lee and David G. Messerschmitt**

SPONSOR: **SRC (93-DC-008), Sony**

Real-time signal processing systems often have multiple operating modes, and have to respond under timing constraints to user commands or dynamic events from the environment. The subsystems responsible for ensuring correct system behavior with the environment have very complex control flow, and so a Ptolemy domain specifically designed for the specification of control flow will be crucial for the design of control-intensive subsystems. It is also important that this domain for control flow can interact smoothly with the dataflow domains, which are used to describe the datapaths, so that the design of the whole system can be integrated easily. Some possible applications include the simulation and rapid prototyping of software systems such as user interfaces and telecommunications switching software, and the synthesis of controllers.

We are investigating three approaches. The first approach is a domain with finite-state machine (FSM) semantics. The way it is used is as a hierarchy of alternating FSM and dataflow domains. What is special about this approach is that dataflow subsystems are embedded inside the states in the FSM domain. The second approach is a control flow domain in which the scheduler can be customized by the user on a galaxy by galaxy basis. The third approach is to implement FSM stars for the existing dataflow domains. The FSM stars will take logic equations or some logic synthesis interchange format like kiss or BLIF-MV as a parameter.

Multidimensional Dataflow in Ptolemy

STUDENT: **Michael J. Chen**

ADVISOR: **Edward A. Lee**

SPONSOR: **NSF (MIP-9201605), Comdisco Systems, and Hitachi**

The purpose of this project is to investigate the concept of multidimensional dataflow (MDF). Instead of considering data as a stream of single values, MDF handles data constructs of more than one dimension [1, 2]. An example is a system that works with data in the form of matrices. New algorithms for scheduling MDF graphs are being investigated, as is an efficient implementation of a data passing mechanism. Questions such as “what does a multidimensional delay mean” and “how to implement efficient buffering for code generation” are also being studied. The initial goal is to develop a two dimensional synchronous dataflow domain in Ptolemy [3] and to explore the possible uses of such a domain. A C++ matrix class has been written and will be incorporated into the new domain as the primary data structure for data passing.

[1] 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.

[2] E. A. Lee, “Data Parallelism in Graphical Signal Flow Representations of Algorithms”, Memorandum No. UCB/ERL M92/110, Electronics Research Laboratory, U. C. Berkeley, August 1992.

[3] 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.

System-level Design using Ptolemy

STUDENT: **Asawaree Kalavade**

ADVISOR: **Edward A. Lee**

SPONSOR: **SRC (92-DC-008) and Bell Northern Research**

This project aims to develop a systematic design methodology for systems containing a mixture of hardware and software components [1]. In the traditional design approach, a design is partitioned into hardware and software early in the design process, and the hardware and the software components are then developed independently. The lack of a unified system representation and a simulation model makes it difficult (if not impossible), in this approach, to consider alternative hardware/software partitions and to study their trade-offs. Hence, an integration of the hardware and software designs is necessary. Our goal is to develop a design manager/advisor, within Ptolemy, that would guide the designer through the hardware/software codesign process.

Towards this end, hardware synthesis capability has been added to Ptolemy by means of a Silage code generation domain. The generated Silage code is fed to the Hyper [2] synthesis tools, which synthesize the custom hardware. A preliminary version of a Design Assistant has also been developed. The Design Assistant starts with a synchronous dataflow (SDF) description of the algorithm, queries the designer for a hardware/software mapping, and proceeds to automatically synthesize the software, hardware, and the interface between the hardware and the software components. We are currently focusing on a DSP-core based ASIC as the target system architecture. The methodology will then be extended to the design of heterogeneous multiprocessor systems [3].

[1] Asawaree Kalavade, "A Hardware/Software Codesign Methodology for DSP Applications," to appear in *IEEE Design and Test*, September 1993.

[2] J. M. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast Prototyping of Datapath Intensive Architectures," *IEEE Design and Test of Computers*, June 1991, pp 40-51.

[3] Y. S. Wu and L. J. Wu, "An Architectural Framework for Signal Flow," *Proceedings of the International Conference on Digital Signal Processing*, Florence, September 1984.

Representing and Exploiting Data Parallelism in Graphical Signal Flow Representations of Algorithms

PROFESSOR: **Edward A. Lee**

SPONSOR: **Hitachi**

Signal flow graphs with dataflow semantics have been used in signal processing system simulation, algorithm development, and real-time system design. Dataflow semantics implicitly expose function parallelism by imposing only a partial ordering constraint on the execution of functions. They are also capable of representing data parallelism. We are investigating how the synchronous dataflow model can be used to graphically define algorithms while exposing their data parallelism to a compiler. A "recursive iterator" notation has been introduced to achieve scalable graphical representations for certain algorithms. The SDF model has also been extended to multidimensional streams to represent and exploit data parallelism in certain signal processing applications. The method appears to be by no means general, but to have broad enough applicability to be useful. The resulting semantics are related to reduced dependence graphs used in systolic array design [1] and to multidimensional streams in the declarative language Lucid [2]. It is more distantly related the stream-oriented functional language Silage [3], and streams in the dataflow language Id and the "synchronous" languages Lustre [4] and Signal [5].

[1] S. Y. Kung, *VLSI Array Processors*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[2] E. A. Ashcroft and W. W. Wadge, "Lucid, a Nonprocedural Language with Iteration", *Comm. of the ACM*, Vol. 20, No. 7, pp. 519-526, July 1977.

[3] P. N. Hilfinger, "Silage Reference Manual, Draft Release 2.0", Computer Science Division, EECS Dept., University of California, Berkeley, CA 94720, July 8, 1989.

[4] P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice, "Lustre: A Declarative Language for Programming Synchronous Systems", *Conference Record of the 14th Annual ACM Symposium on Principles of Programming Languages*, Munich, Germany, January, 1987.

[5] A. Benveniste, B. Le Goff, and P. Le Guernic, "Hybrid Dynamical Systems Theory and the Language SIGNAL", Research Report No. 838, Institut National de Recherche en Informatique at en Automatique (INRIA), Domain de Voluceau, Rocquencourt, B. P. 105, 78153 Le Chesnay Cedex, France, April 1988.

Real-Time Scheduling of Dataflow Programs

STUDENT: **Thomas M. Parks**

ADVISOR: **Edward A. Lee**

SPONSORS: **SRC (92-DC-008) and Bell Northern Research**

The purpose of this project is to apply real-time scheduling techniques to programs represented by dataflow graphs. The infrastructure for multi-threaded simulation and code generation now exists. This will provide the means to verify algorithms that partition dataflow graphs into threads and that schedule those threads for real-time execution.

The current infrastructure relies on operating system support for prioritized, preemptive scheduling of the real-time threads. Graph partitioning algorithms will be developed that minimize the overhead of preemptive scheduling by reducing the number of threads in the system.

Non-preemptive scheduling algorithms will be explored to avoid this overhead wherever possible.

Optimized Code Generation for DSPs

STUDENT: **José Pino**

ADVISOR: **Edward A. Lee**

SPONSORS: **ONR Fellowship, AT&T Bell Labs Fellowship, NEC, and Bell Northern Research**

This past year, code generation modules in Ptolemy have flourished. Last December, the first version of Ptolemy to fully incorporate its code generation capabilities was released. These capabilities include assembly code generation for Motorola 56000 and 96000 digital signal processors and C code generation. For a summary of the currently released code generation capabilities in Ptolemy see [1].

During the next year, we will enhance the code generation capabilities by optimizing the code generated; defining heterogeneous multiprocessor architectures, where the code generated for each processor can be expressed in a different language; and exploring interface issues between systems simulated and code generated by Ptolemy.

To optimize the code generation in Ptolemy, we will incorporate architecture-specific knowledge, such as hardware looping and the number of registers. This information will be used to eliminate excess transfers to and from memory, thus improving the performance of code constructed from fine-grain dataflow graphs. Architecture-specific knowledge will also be used to balance the use of loops and in-line code to efficiently use program memory while not sacrificing speed. This is particularly challenging in multirate systems.

Another major goal in this coming year is to enhance Ptolemy to automatically generate code for heterogeneous multiprocessor systems. The framework will enable the designer to migrate from simulation to code generation while developing an application. At present time, we envision two example test beds to test the framework. The first target consists of a host workstation and a single Motorola 56001 processor. The second target consists of four digital signal processors (Motorola

96002) and a single general-purpose processor (Motorola 68030). A major focus will be to preserve the object-oriented style of Ptolemy.

[1] J.L. Pino, S. Ha, E.A. Lee, and J.T. Buck, “Software Synthesis for DSP Using Ptolemy,” Journal of VLSI Signal Processing, special issue on “Synthesis for DSP,” to appear in Fall 1993.

Advanced Filter Design

STUDENT: **Kennard White**

ADVISOR: **Edward A. Lee**

SPONSORS: **MICRO, Dolby Labs, and Comdisco Systems**

XPole is a graphical filter design program that is intended to give users an intuitive understanding of the relationship between time-, frequency-, and transform-domain representations of filters. It allows users to construct filters by manipulating singularities (poles and zeros) in the Z or S plane, while the program interactively computes the impulse and frequency responses. Standard filters (e.g., Butterworth, elliptical) of arbitrary order are supported, as well as several mappings from S to Z transform domain.

Recent work has concentrated on improving XPole’s user interface and developing an on-line help system called *tkinfo*. Also many of the numeric algorithms and filter design methods have been made more robust and consistent. Future plans include support of root-locus and Nyquist plots, analysis of implementation and quantization effects, design of statistically derived filters, and support of frequency-based filter design algorithms. X11 with Tk is used for the graphical interface, and Tcl for the command and script language.

DSP Hardware Implementation Design through VHDL in the Ptolemy Environment

STUDENT: **Michael C. Williamson**

ADVISOR: **Edward A. Lee**

SPONSOR: **SRC (92-DC-008) and Rockwell**

We have undertaken to improve the coupling of Ptolemy to state-of-the-art low-level design tools. The THOR domain, while able to simulate digital hardware at a low level, does not easily allow for intermediate behavioral descriptions of components whose detailed functionality has yet to be specified. The VHDL standard for hardware description and simulation can serve as a bridge between abstract algorithmic descriptions and detailed low-level hardware designs. Towards this end, we have constructed a VHDL domain in Ptolemy.

We have produced a working VHDL target for code generation, modeled after ongoing work in the Silage code generation domain. This allows for structurally-oriented VHDL code generation, based on systems constructed within Ptolemy which combine existing VHDL entities. This effort provides a basic infrastructure for generating top-level VHDL design specifications. Following that effort, the target has been expanded to allow fully hierarchical designs to be output into complete VHDL specifications. These designs are composed of predefined VHDL blocks connected according to the block specifications, and then elaborated to a full design by the code generator.

We have also constructed a communications interface between the VHDL simulation package and Ptolemy so that VHDL designs may be freely simulated within non-VHDL Ptolemy systems, allowing for heterogeneous design and simulation. This makes it possible for abstract designs to be gradually substituted block-by-block with well-defined VHDL specifications until the design converges to its final VHDL form.

Now that a two-way link between Ptolemy and the VHDL simulator has been achieved, the user will be able to run heterogeneous simulations within Ptolemy which incorporate, entirely or in part, subsystems written in VHDL and simulated on the external simulator. These subsystems may be generated by Ptolemy from block descriptions, or they may incorporate user-written and vendor-supplied VHDL descriptions of existing or hypothetical digital devices and subsystems. Block specifications of algorithms in the Synchronous Dataflow domain and other domains can be gradually expanded and substituted block-by-block with functionally equivalent blocks created in the VHDL domain in order to validate hardware implementation of the original algorithm and to experiment with various hardware design alternatives. These VHDL descriptions, if expanded into synthesizable VHDL, can then be passed to synthesis tools which take VHDL as input, and output gate-level circuit implementations.

Upcoming work will include improvements to the Ptolemy/VHDL co-simulation environment we have constructed, including improvements to the two-way communication between Ptolemy and the VHDL simulator. We also plan to add options to differentiate between pure code generation, fast simulation, and debugging during simulation. Also, efforts are being made to obtain simulation toolsets from other vendors in order to expand our simulation capabilities and to make the VHDL domain in Ptolemy interoperable with more simulation toolsets. A further benefit from this effort is the potential for obtaining larger libraries of pre-defined design units which will improve the speed with which start-up users can move from concept to full specification. In addition, some of these toolsets include synthesis capabilities, which would allow for migrating from algorithm-level designs to synthesis-level designs in less time. While other toolsets offer intermediate-level design specification as a starting point on the way to the synthesis level, Ptolemy's particular advantages as an open-ended heterogeneous simulator can allow the designer more freedom in the early stages of design development.

Parallel Computation:

Design of Heterogeneous Parallel Systems

STUDENT: **Sangjin Hong**

ADVISOR: **Edward A. Lee**

SPONSOR: **ONR and NRL (N00014-92-K-2018)**

The methodology for design and evaluation of parallel architectures with heterogeneous components is being investigated. Specifically, the "communicating processes" (CP) domain in Ptolemy is being used to model hardware systems built on the "ALPS" concept (Alternative Low-level Primitive Structures) [1]. An ALPS architecture uses a communication ring to distribute tasks to computational system resources. A system resource consists of a primitive (functional module) coupled to a standard system interface called the Interface Control Unit (ICU). Such a standard interface makes it possible to have a wide range of hardware resources within a single application - the ICU handles the discrepancies in the requirements of different resources.The ICU implements the control protocols necessary for data communication, primitive activation, and deactivation. This distribution of control, made possible by a self-synchronizing network architecture, avoids the bottlenecks associated with centralized control and synchronization, and provides a measure of fault tolerance.

A signal flow graph (SFG) describing the application is loaded into each ICU in the form of a routing table, which acts as a look-up table for task activation. Data associated with an arc between two nodes of the SFG is characterized by its source and destination node pair — called the virtual circuit number (VCN). This VCN, along with an identifier as to the 'type' of resource needed by this data, constitutes a message control token (MCT). A source node, upon completing execution of its task, broadcasts the MCT(s) to all the system resources on the ring network. A 'free' resource of the requested type grabs this token and sends an acknowledgment to the source, which then transmits the data on the data circus. After transmitting the data, the source becomes free for subsequent computations. The sink processes the received data and initiates the next transfer. The protocol supports multiple sinks (multicast, or forking of data), variable-length data, and multirate processing. If an appropriate sink is not found, the MCT is re-transmitted some time later. The design is highly modular due to the standard interface, and can accommodate more or less arbitrary combinations of processing elements.

The Communicating Processes (CP) domain in Ptolemy has a multi-tasking kernel managing concurrent processes. A process, or thread, is created for each computational system resource and ICU in a particular ALPS design. The CP model of computation makes it easy to write and interconnect a variety of computational resources and simulate the execution of an application to evaluate the particular architectural configuration.

[1] Y. S. Wu and Les J. Wu, "'Signal Flow' Architecture", COMPCON 84, Wash. DC, Sept. 84.

SPONSOR:

Real Time Video Signal Processing in Ptolemy

STUDENT: **Sun-Inn Shih**

ADVISOR: **Edward A. Lee**

SPONSOR: **DARPA (J-FBI-90-073) and Philips**

A programmable video signal processor(VSP), developed at Philips Research Laboratories, can perform a broad range of video signal processing tasks in real time. The processor is a VLIW machine with three ALUs and two memory elements that can operate in parallel at the rate of 27 MHz. They are connected in a highly pipelined fashion. Furthermore, processors can be interconnected in a large network. Tools are provided for software development and hardware simulation of the VSP network.

The short-term objective of this project is to integrate the VSP tools into Ptolemy. The long-term objective is to study and develop implementation methodologies for real-time video signal processing. In particular, we are interested in code generation and scheduling issues for VLIW programmable architectures at extremely high sample rates. Ptolemy's support for heterogeneity will provide an environment for studying the interaction of video algorithms and other signal processing tasks (for instance, signal transmission, network control and intelligent back-end processing of images).

We have defined a VSP Simulation Target with synchronous dataflow (SDF) semantics. A block library has been developed that is compatible with the block library in the existing Philips tools, enabling us to do functional verification by local simulation of the algorithm.

In addition to the VSP Simulation Target, we have also implemented a VSP Target. This Target is capable of generating code for VSP software development tools, which are responsible for the scheduling and the VLIW code generation. This Target has a newly written scheduler that first invokes the SDF scheduler to compute the repetition rate of each of the blocks, and then simply walks through the topology of the system and generates a file in the Philips' intermediate ASCII netlist format. The Philips' tool is invoked for further modification of the application. Currently, we are at the last stage of completing the library and the Targets.

Restricted Application Parallel Architecture Design

STUDENT: **S. Sriram**

ADVISOR: **E. A. Lee**

SPONSORS: **SRC (92-DC-008)**

In this project, we are investigating special purpose parallel architectures for real time signal processing. A systematic, low-cost method for communication between programmable components has been devised. This method is compatible with the assembly and microcode synthesis techniques and the hardware/software codesign approach in Ptolemy.

A significant subclass of DSP applications can be expressed as synchronous dataflow (SDF) graphs, where the number of tokens produced and consumed by each functional node is fixed and known at compile time. A compiler can automatically partition SDF graphs into tasks running on multiple processors and compute near optimal schedules for them. One consequence of this approach is that the ordering of transactions between the processors is known, or can be estimated with reasonable accuracy, at compile time. By determining such an order at compile time and enforcing it at run time, contention for shared resources is eliminated, and there is no need for explicit synchronization between processors.

Based on these ideas, we have designed and built a prototype multiprocessor design called the "ordered memory access" (OMA) architecture, comprised of four Motorola DSP96002 processors on a single printed circuit board. The processors communicate via shared memory. At runtime, processor accesses to shared memory are ordered by a transaction controller. The controller, implemented on a Xilinx FPGA, grants access to shared memory according to the schedule predetermined and downloaded into it at compile time. By ordering processor transactions in this manner, we have been able to achieve low overhead and low latency interprocessor communication with very little added hardware cost.

We have defined a Heterogeneous Target in Ptolemy for the OMA system that consists of the OMA board and also a DSP56000-based host board connected to a Sun workstation. Executable code for such a target is automatically generated from an application specified in a block diagrammatic fashion in Ptolemy, with the I/O handled by the 56000 board and the user commands coming from user interfaces running on the workstation.

We have also developed a formal method for quantifying the impact on the execution time of the timing constraints imposed by ordering shared memory transactions. This has been used to show that, for a given static schedule, there always exists an order on accesses to shared resources such that there is no penalty (in terms of throughput) in enforcing this order as opposed to arbitrating the resources dynamically. We expect these results to lead to optimization techniques for synthesis of parallel systems.

We are currently exploring ways of extending these ideas to applications that involve some dynamics in their flow of control, and hence are not wholly SDF. The prototype architecture allows a processor to force the transaction controller to switch between schedules on the basis of runtime conditions. This will enable us to execute applications that fall into the dynamic dataflow (DDF) or Boolean dataflow (BDF) domains in Ptolemy.

[1] S. Sriram and E. A. Lee, "Design and Implementation of an Ordered Memory Access Architecture", *Proceedings of ICASSP'93*, 1993.

[2] J. C. Bier, S. Sriram, and E. A. Lee, "A Class of Multiprocessor Architectures for DSP", *VLSI DSP IV*, IEEE Press, 1990.

[3 E. A. Lee and J. C. Bier, "Architectures for Statically Scheduled Dataflow", *Journal of Parallel and Distributed Computing*, December 1990.

Synthesis of Parallel Code for the CM-5

STUDENT: **Kennard White**

POSTDOC: **Soonhoi Ha**

ADVISOR: **Edward A. Lee**

SPONSORS: **NSF (MIP-9201605), MICRO, NSF Infrastructure Grant number CDA-8722788, and ONR via NRL (N00014-92-K-2018).**

We have developed a Target in Ptolemy that generates parallel code for the CM-5 from Thinking Machines Inc. The CM-5's processing nodes (Sparc processors with dedicated network interfaces) match well with Ptolemy's large grain parallelization approach. Ptolemy will map an application fitting the synchronous dataflow (SDF) model of computation onto the CM-5's processing nodes and schedule the computation and interprocessor communication (IPC). A variety of different algorithms may be used to perform the mapping and scheduling. Ptolemy generates C code for the computation and IPC; this code is then compiled and executed. The IPC is performed using Active Messages (CMAM) from Prof. Culler's group [1].

The CM-5 target provides a good test bed for evaluating mapping and scheduling algorithms; in particular, we can evaluate how closely the models used by the scheduling algorithms match reality. Future plans includes general research of parallel scheduling algorithms, research into fine grain parallelization, and CM-5 specific research on alternative IPC protocols and finding weaker IPC synchronization methods.

[1] T. von Eicken, D. E. Culler, and S. C. Goldstein, and K. E. Schauser, "Active messages: a mechanism for integrated communications and computation," *Proc. of the 19th Int. Symp. on Computer Architecture*, Gold Coast, Australia, May 1992, also available as TR UCB/CSD 92/675, CS Division, University of California, Berkeley, CA 94720.

Signal Processing and Communications Applications:

Dynamic Assignment of Channel Resources in Multi-User Radio Networks

STUDENT: **Rolando Diesta**

ADVISOR: **Jean-Paul Linnartz**

SPONSORS: **PATH (65H998 MOU 82) and SRC (92-DC-008)**

In radio data networks, efficient use of the available spectrum ensures high user capacity and low access delays for data messages. We are investigating combined random access and spatial frequency reuse to develop “spatial random access” methods for wireless data networks with different transmitter location distributions and propagation characteristics. Dynamic channel resource allocation schemes for multimedia applications are under study to improve spectrum efficiency compared to fixed-channel networks. Collision resolution algorithms for “spatial random access” are being studied with applications in macroscopic networks (e.g., highway/mobile communications) as well as networked (indoor wireless) systems.

The Ptolemy discrete-event (DE) domain was used to create a high-level simulation of the ALOHA multiple access protocol. This protocol was selected because of its well-studied characteristics which led to easy verification of the model’s correctness. This work allowed construction of control tools in the DE domain which can be to model other protocols. New Ptolemy domains currently under development are being evaluated for more efficient specification of communication protocols.

On Intelligent Vehicle Highway Systems (IVHS), the “downlink” (i.e., roadside base station to vehicle communications) was analyzed to see the effect of dynamic cluster sizes on spectrum efficiency in a Rician fading environment with co-channel interference. A model using interacting queues is being developed to extend the analysis.

Dynamic Assignment of Channel Resources in Multi-User Radio Networks

STUDENT: **Michael Goodwin**

ADVISORS: **Edward A. Lee and David Wessel**

SPONSOR: **GAANN (Graduate Assistance In Areas of National Need) fellowship, The California Competitive Technology Program (DOC-C92-048-WESSEL-3/94) and Dolby Laboratories**

Musical sounds can be modeled as the sum of numerous partials, namely sinusoids with time-varying frequency and amplitude. In an FFT-based analysis-synthesis system for musical applications currently under development, the analysis provides a description of the time-evolution of the partials comprising the analyzed sound. This description is converted into a spectral form by accumulating sparse representations of each partial into a register of FFT bins. The representations of the partials, however, are limited in that they are formed according to low-rate discrete-time samples of the amplitude, frequency, and phase parameters; the sample points in time correspond to the times at which an analysis window was applied to the original signal. In order to avoid this discretization, which can produce distortion at the frame boundaries in the overlap-add synthesis procedure, it is necessary to devise a spectral representation of the partials that incorporates the rate of change of the analysis parameters. With regards to the frequency parameter, this problem has been touched upon in the literature on chirp radar, but since changes in musical sounds occur at a much slower relative rate than in radar chirp signals, some of the key results cannot be readily applied. This project involves the investigation of methods for deriving frequency-domain representations of the slow "chirps" that are present in musical sounds.