MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Laboratory for Computer Science
Cambridge, Massachusetts 02139

Computation Structures Group Memo 198

Data Flow Computer Architecture

Jack B. Dennis

(Research Proposal Submitted to the Department of Energy
for the period July 1, 1980 - June 30, 1981)

July 1980

Table of Contents

A. Introduction

The MIT Laboratory for Computer Science (LCS) hereby requests continued support for the second year of our three-year research program on data flow computer architecture being carried out by the LCS Computation Structures Group under the direction of Professor Jack B. Dennis.

The following sections review progress during the first year, and present the goals and plans for our work in the second year.

Our work is being carried out in collaboration with the Lawrence Livermore Laboratory, whose participation is noted below as appropriate, and is covered in reports of the Livermore Laboratory.

B. Review of Progress in the First Year

The proposal for our research program described a set of six projects which form the main thrust of work required to move toward practical and effective data flow computer systems, and supporting studies in areas important to successful realization of our goals.

Projects

1: Data Flow Source Language

This work designed the programming language VAL which is specified in a programmer's reference manual [2]. The language is the first documented and implemented applicative or functional language intended for practical use in numerical computation. The language is designed to allow type checking by the compiler. The unique features of VAL include: an iteration construct that mirrors an attractive programming paradigm known as *tail recursion*; a method of handling exception conditions consistent with the value-oriented character of the language, and a **forall** construct that allows expression of parallel computation of array elements and parallel application of n-argument associative operators.

James McGraw of the Livermore Laboratory, who has provided valuable criticism and guidance in the design of VAL, has written an expository paper [10] on the language using as an example a program for adaptive quadrature.

We have constructed a translator/interpreter implementation of VAL that runs under the CLU programming system on the DEC 20 computer at the MIT Laboratory for Computer Science. The implementation is complete, and McGraw's adaptive quadrative program has been run successfully. In addition, the Simple benchmark program circulated by LLL has been rewritten from Fortran into VAL and run with consistent results for a small test problem.

2: *Construction of an Engineering Model*

To establish feasability of a large-scale data flow computer, it is necessary to model the behavior of a proposed machine so that cost and performance can be evaluated, and issues of program translation, verification and fault tolerance can be investigated. Because data flow computers differ radically from conventional stored program computers, the computation rate of a data a flow machine can be compared with the speed of a conventional machine only in terms of performance for some specific application. Although conventional simulation techniques are suitable for evaluating performance of limited interconnection of hardware units, and for interpreting program execution under simplified assumptions of machine behavior, current simulation capabilites are not up to detailed modelling of a meaningful application running on a hypothetical data flow processor.

Because it is not yet clear to us exactly what structure and instruction code a high performance data flow computer should have, we have chosen to establish a general facility for emulating and evaluating any data flow processor structured using packet communication, rather than build a hardware system specialized to a particular form of machine. The concept of our data flow evaluation facility is described in a paper for presentation at the International Symposium on Computer Architecture [7]. In establishing this facility we are not only interested in evaluating projected cost/performance of proposed systems, but we will use it to address issues of hardware design and construction methodology, in programming language design and compilaton, machine level program structure and instruction set design, and achieving reliability through fault detection and masking.

The facility provides two kinds of building blocks, processing units and router modules, which may be interconnected to form a packet communication model of a system to be studied. The processing units are programmable units based on the American Microdevices 2900 series bit-slice devices which can emulate a wide variety of functions. The facility will include a host computer for loading programs into processing units and monitoring operation of the system. At present, a prototype router module is being checked out, and the board for the first processing unit is ready to be wire-wrapped.

3: *Program Translation*

We are studying the problem of translating programs written in VAL into machine level programs for high performance data flow computers. We imagine that VAL programs will be translated first into an intermediate form closely related to data flow program graphs, and then machine level code will be generated from the intermediate form. Code transformation and optimization can be done at each level of representation: VAL, intermediate form, and machine code. Translating from an applicative language to data-flow program graphs is a solved problem [4, 15], but choosing the best form of program graph to effectively utilize a specific data flow computer is a challenging open problem. During the past year, one thesis project has been completed [12], which studies optimizing transformations of safe data flow graphs. This study relates to the problem of optimum code generation for a high performance data flow computer.

Study of machine level program structure was taken up by Prof. Joe Stoy of Oxford during his visit to MIT last year. His results [14] will be further developed in a future master's thesis project.

4: Data Structures

Our main achievement in the area of data structures is the use of Ackerman's ideas [1] on a generalized data structure processor to develop an overall proposal for a complete general-purpose data flow computer. In this work, Weng [16] shows how a structure controller/packet memory system can support implementation of procedure structures, activation records and data structures, as required in a data flow computer that offers a very general language capability to its users.

5: Architecture Description and Simulation

Work on development of our architecture description language (ADL) progresses as resources are available. Leung has completed a paper [8] on our present concept of the ADL, and he has used the language to specify the router module for our data flow prototype facility [7].

We have found the programming language CLU to be especially useful for writing simulation programs. Simulation of a simple data flow processor and of packet routing networks [13], has been carried out with surprisingly little programming effort.

6: Specification of a Form 2 Data Flow Processor

The objective here is to produce a specification useable for constructing a data flow computer system suitable for Livermore applications. Work towards this end depends on successsful completion of other projects.

Supporting Studies

Fault Tolerance: The packet communication structure of data flow computer systems is a new context in which to study use of redundancy and coding to achieve desired levels of fault tolerance. Leung is completing a doctoral thesis [9] which shows how several levels of fault detection and masking can be implemented in the context of packet communication architecture. He uses a "random-wave-train" fault model which implies that his implementation will work even if faulty units cause inconsistent signals or "runt pulses."

Applications: We have continued our study of the Livermore benchmark code, Simple. The

Fortran code has been translated into VAL with the help of earlier work of Arvind and Bryant [3], and the VAL version of Simple has been run on our VAL implementation for a simple test case yielding results consistent with those of the Fortran code.

The Fast Fourier Transform is the subject of a bachelor's thesis study of the relation between expression of an algorithm as a program graph suitable for efficient execution, and possible forms of expression in VAL [6].

Collaboration With Livermore

James McGraw has continued to be active in promoting the VAL language [11] and in discussing its limitations. John Renneletti, as a user of the language, and as an implementer at LLL, has also provoided useful reactions to the design of VAL. Their advice will be valuable to us as we extend the language and review earlier design decisions.

The physicists at Livermore, Tim Rudy and Chris Hendrickson in particular, have helped us with our analysis of the Simple code.

The possibility of a joint effort involving the Texas Instruments Company, the Livermore Laboratory and MIT, is currently under discussion.

Workshop on Self-Timed Systems

In July 1979, MIT, with support from DOE, sponsored a Workshop on Self-Timed Systems, which served to bring together experts in this field for a unique opportunity to review and assess the state-of-the-art and to chart directions for future development. With the prospect of economical custom fabrication of LSI devices, there appears to be a new opportunity for profitable applications of self-timed principles. In particular, the workshop left one with the strong feeling that self-timed design methodologies will have an important, if not essential, role in the successful production of VLSI devices.

A report on the workshop that summarizes the presentations and discussions is forthcoming [5].

C. Expected Progress in the Second Year

The primary task for the second year is to continue work on those projects essential to our ultimate objective of specifying a practical high performance data flow computer system.

1: Language Development

VAL is an applicative or functional programming language designed to support practical numerical computation on computer systems capable of exploiting parallelism. As such, the VAL language is a pioneering effort in programming language development. We will continue to gather constructive criticism of VAL and review design decisions as a basis for a revised language definition.

The present version of VAL is limited in expressive power since a VAL **function** starts computation only when all argument values are available and produces result values only when all computation has been completed. With this limitation, VAL does not provide complete input/output facilities and does not address the problem of shared data bases. On the basis of work by Weng [16] and others, both at MIT and elsewhere, we are optimistic that an extension of VAL based on streams and a nondeterminate **merge** operator will provide attractive answers that fit naturally within the functional programming framework. Defining such an extension of VAL and building a corresponding extension of our translator/interpreter implementation of VAL will be the goal of this project for the coming year.

2: Prototype Evaluation Facility

Our goal for the second year is to run translated VAL programs on a data flow prototype system emulated using our evaluation facility. In addition, we expect to study applications such as the Simple code using such a prototype system to evaluate code generation and optimization strategies, and to project potential computation speed on a full scale data flow system.

This project involves the following activities:

1. Building and testing processing units and router modules.

2. Completing support software for writing and verifying microcode for the processing units.

3. Setting up a small host computer system to load and monitor an emulated prototype machine.

4. Designing and implementing in microcode an initial instruction set for a prototype data flow computer.

In addition, achieving the goal will depend on work on translation of VAL programs and on the implementation of data structures.

3: Program Translation

This project requires fundamental work on transforming and optimizing data flow programs. We will explore this problem in the abstract -- using concepts derived from our theoretical work on data flow program graphs -- and in concrete terms using application codes such as Simple to test and evaluate our ideas. Studies in this area will be the subject of several graduate thesis projects during the coming year.

4: Data Structures

This project will focus on the form and realization of data structure facilities appropriate to high performance numerical computations such as required in codes for solving PDE's. This will involve deciding what restricted forms of data structure values and operations should be supported, evaluating how memory management should be divided between the translator and the hardware, and designing and evaluating efficient implementations. This work will be based on our earlier studies of data structure semantics and implementation, the design of the VAL language, and our studies of applications.

5: Architecture Description and Simulation

The further development of an architecture description language and simulation facilities based on it will be carried forward as resources permit.

6: Specification of a Full-Scale Data Flow Processor

This portion of our proposed effort will clearly depend on successful results from the other projects. Our work on VAL will produce a full specification of a programming language suitable for high level programming of a full-scale data flow machine, and our work on translation and optimization will provide valuable knowledge for development of a practical programming system for data flow programs. Our work with the prototype evaluation facility and architecture description language will provide the basis for developing precise specifications of hardware units suitable for building a full-scale machine using custom LSI or VLSI technology.

Supporting Studies

We will continue our work on approaches to achieving fault tolerance in packet architecture systems, and on methodologies for the design, verification and realization of asynchronous systems. In addition, we are pursuing fundamental research in formal semantics of languages and systems and architectural principles. These supporting studies are being carried out with support from the National Science Foundation.

Workshop on Applicative Languages and Parallel Computation

Interest in functional programming and applicative programming languages has risen to the point that it is appropriate to bring together people working in this field to exchange ideas, theories and experiences, and to formulate directions of future development. We are planning a workshop conference on this subject to be held on June 6-9, 1980, at the MIT Endicott House. Already about eighteen computer scientists involved in significant work on functional languages and computation outside our MIT Data Flow Project have expressed interest in participating. To be sure of having the best possible group of research scientists from Europe and the U. S. involved in the workshop, we are requesting permission to allocate a portion of our grant for support of this conference. This support will be used to pay useage fees for the Endicott House, and to pay travel expenses for those who could not otherwise participate in the workshop.

References

1. Ackerman, W. A. *A Structure Memory for Data Flow Computers.* Technical Report MIT/LCS/TR-186, M.I.T., Laboratory for Computer Science, Cambridge, Ma., September 1977.

2. Ackerman, W. A., and J. B. Dennis. *VAL: A Value Oriented Algorithmic Language Preliminary Reference Manual.* Technical Report MIT/LCS/TR-218, M.I.T., Laboratory for Computer Science, Cambridge, Ma., June 1979.

3. Arvind, and R. E. Bryant. Design considerations for a partial differential equation machine. *Proceedings of Scientific Computer Information Exchange Meeting,* Livermore, Ca., September 1979, 94-102.

4. Brock, J. D. *Operational Semantics of a Data Flow Language.* Technical Memorandum MIT/LCS/TM-120, M.I.T., Laboratory for Computer Science, Cambridge, Ma., December 1978.

5. Bryant, R. E. *Report on the Workshop on Self-Timed Systems.* Technical Memorandum, M.I.T., Laboratory for Computer Science, Cambridge, Ma., forthcoming.

6. Chien, A. *Structuring the Fast Fourier Transform for Data Flow Computation.* B.S. thesis, M.I.T., Department of Electrical Engineering and Computer Science, Cambridge, Ma.,

forthcoming.

7. Dennis, J. B., G. A. Boughton and C. K. C. Leung. Building blocks for data flow prototypes. To appear in *Proceedings of the 1980 International Symposium on Computer Architecture*, La Baule, France. Also Computation Structures Group Memo 191, Laboratory for Computer Science, M.I.T., Cambridge, Ma., February 1980.

8. Leung, C. K. C. ADL: An architecture description language for packet communication systems. *Proceedings of the 4th International Symposium on Computer Hardware Description Languages*, Palo Alto, Ca., October 1979, 6-13.

9. Leung, C. K. C. *Fault Tolerance in Packet Communication Computer Architecture*. Ph.D thesis, M.I.T., Department of Electrical Engineering and Computer Science, Cambridge, Ma., forthcoming.

10. McGraw, J. R. *Data Flow Computing: The VAL Language*. Lawrence Livermore Laboratory, Livermore, Ca., submitted for publication, November 1979.

11. McGraw, J. R. Data flow computing: Software development. *Proceedings of the 1st International Conference on Distributed Computing Systems*, Huntsville, Ala., October 1979, 242-251.

12. Montz, L. B. *Safety and Optimization Transformations for Data Flow Programs*. S.M. thesis, M.I.T., Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, Ma., February 1980.

13. Ressler, Paul. *Simulation of a Highly Parallel Processor*. S.B. thesis, M.I.T., Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, Ma., January 1979.

14. Stoy, J. E. Private communication.

15. Weng, K.-S. *Stream-Oriented Computation in Recursive Data Flow Schemas*. Technical Memorandum MIT/LCS/TM-68, M.I.T., Laboratory for Computer Science, Cambridge, Ma., October 1975.

16. Weng, K.-S. *An Abstract Implementation for a Generalized Data Flow Language*. Technical Report MIT/LCS/TR-228, M.I.T., Laboratory for Computer Science, Cambridge, Ma., May 1979.

Publications

1. Brock, J. D., and L. B. Montz. Translation and optimization of data flow programs. *Proceedings of the 1979 International Conference on Parallel Processing*, August 1979.

2. Dennis, J. B. The varieties of data flow computers. *Proceedings of the First International Conference on Distributed Computing Systems*, October 1979.

3. Dennis, J. B., and K.-S. Weng. An abstract implementation for concurrent computation with streams. *Proceedings of the 1979 International Conference on Parallel Processing*, August 1979.

4. Dennis, J. B., G. A. Boughton, and C. K. C. Leung. Building blocks for data flow prototypes. To appear in *Proceedings of the 1980 International Symposium on Computer Architecture.*

5. Leung, C. K. C. ADL: An architecture description language for packet communication systems. *Proceedings of the 4th International Symposium on Computer Hardware Description Languages*, October 1979.

6. Leung, C. K. C. *Fault Tolerance in Packet Communication Computer Architecture.* Ph.D thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Ma., forthcoming.

7. Montz, L. B. *Safety and Optimization Transformations for Data Flow Programs.* S.M. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Ma., February 1980.

Personnel

Faculty

J. B. Dennis, Leader
Arvind

Graduate Students

W. B. Ackerman
G. A. Boughton
J. D. Brock
R. E. Bryant
V. K. Kathail
C. K. C. Leung
L. B. Montz
K. K. Pingali
N. Singh
K. Todd

Research Staff

E. Shaw

Support Personnel

A. Rubin