MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LABORATORY FOR COMPUTER SCIENCE
CAMBRIDGE, MASSACHUSETTS

COMPUTATION STRUCTURES GROUP MEMO 215

HIGH PERFORMANCE DATA FLOW COMPUTERS

A Proposal to the Department of Energy

JACK B. DENNIS

MARCH 1982

Introduction

In June, 1982, the MIT Computation Structures Group will complete a three-year research program on data flow computer architecture sponsored by the Department of Energy. In this period we have:

- Designed and documented the programming language VAL as a high-level programming language for programs to be executed on data flow computers.
- Constructed an "Engineering Model", a hardware system for the evaluation of proposed architectures of data flow computers.
- Designed and implemented procedures for translating the constructs of VAL into data flow machine language.
- Developed promising approaches to handling the data structures of large numerical computations.
- Analyzed several application problems for their suitability for efficient execution on data flow computers.
- Explored design methodologies for packet architectures and developed preliminary specifications and designs for several basic units of data flow computers.

With support from the National Science Foundation, we have contributed to the specific theoretical characterization of data flow languages and computations, and of architectures known as packet communication architectures.

Our work has shown that data flow computers will be able to demonstrate high performance through exploitation of large scale parallelism for computational problems in signal processing and in the solution of the partial differential equations of fluid systems. For example, we have studied the NASA global weather model--a benchmark program written in Fortran. This code has been rewritten in VAL and we have developed machine code structuring principles that permit the VAL code to be mapped onto the processing and memory resources of a static data flow supercomputer. The conclusion of our analysis is

that a data flow computer system with 256 processing elements would be able to carry out this computation at about 180 megaflops, and would complete one time step in each five seconds of computing time for the 144 by 87 by 9 grid. Since the configuration assumed does not require high performance memory to achieve this speed, and since all logic and arithmetic is done by LSI parts, the hypothesized machine should be much smaller and consume less power (by a factor of ten or more) than a conventional supercomputer of the same speed. Furthermore, additional speed can be achieved by increasing the number of units in the machine until the complexity of wiring the interconnection network limits the possible gain. (This is not likely before the machine has several thousand processing elements.) The analysis of the weather model benchmark program and a discussion of the level of performance possible using a data flow supercomputer is presented in [14], which is attached to this proposal. Other applications under study are a Navier Stokes code for aerodynamic simulation provided by the NASA Ames Research Center, and the Simple code provided by the Livermore Laboratory.

We propose to continue this project with the goal of building practical high-performance data flow computers during the next three to five years. The work involved includes the design and engineering of the physical hardware for data flow machines, the design of an efficient machine instruction set, and the development of program transformation and optimization techniques for programs expressed in VAL. This work is already in progress and we are confident of its completion and availability for the construction and use of practical data flow computer systems.

The principal work to be done, however, is the design of the five to ten custom LSI circuit chips from which a broad range of data flow machines may be built. This work includes selecting, acquiring, and/or programming the necessary computer aided design tools, acquiring facilities for testing and evaluating custom fabricated devices, and developing the approach to be adopted for achieving full hardware fault detection. Many MIT computer science graduate students now have training in VLSI design, so a pool of

resources--people and software--is developing at MIT.

We will also need access to fabrication facilities for LSI devices. The MIT VLSI Program is an important resource for us, but it is not clear whether our requirements can be met through this channel alone. We have initiated some contacts with east coast industry as possible partners in custom device fabrication. The Digital Equipment Company, IBM, and Bell Laboratories are possible resources.

For the proposed project, the character of much of our work will change from design studies, problem analyses and basic research, to hardware design and software development. This requires appointment of qualified full-time staff personnel to lead critical areas of the project effort. Fortunately, several very talented people will be available to us in the near future who are enthusiastic and highly qualified to take on these new responsibilities.

In the following section we briefly review the progress and status of our current effort. We discuss work being done elsewhere. Then we outline in greater detail the program of effort required to meet the goals.

Research Status

In our proposal for the research presently supported by the Department of Energy we outlined six projects devoted to essential groundwork for successful development of practical large-scale data flow computers. Here we review our achievements in each of these six areas.

Project 1: Data Flow Source Language. The programming language VAL has been designed expressly for programming applications for data flow computation. It is a *functional* language, by which we mean that each textual unit of a VAL program just defines a process for computing output values from input values--there is no other effect [1]. This property permits easy analysis of VAL programs for parts that can be executed

concurrently. A preliminary reference manual for VAL was published in 1979 [3], and the language has been implemented in the form of a translator and interpreter, which are written in CLU and run on the DEC System 20 machine at the MIT Laboratory for Computer Science. This implementation serves as a standard for the language and has allowed evaluation of the language through the programming of many examples and several benchmark applications [25].

The VAL implementation has been extended with an experimental code generator that constructs formal program graphs from the abstract parse trees produced by the VAL translator, then translates the program graphs into data flow instructions. At present a program written in VAL can either be executed by the interpreter, or, if it is not too large or complicated, translated into a collection of data flow machine instructions and loaded into the processing elements of a data flow architecture under study using the Engineering Model.

Project 2: Construction of an Engineering Model. The concept, purposes, and design of our Engineering Model were described in [12]. This facility is now operational with four microprogrammed processing units and a four-by-four routing network through which result packets may be sent between emulated data flow processing elements. Programs involving several hundred data flow machine instructions have been compiled from VAL source programs into an experimental instruction set [30] and run on the Engineering Model. During the next month, expansion of the engineering model to eight processing units interconnected by an eight-by-eight packet routing network will be completed, and it will be possible to run significantly larger programs for evaluation. Our experience with this facility will provide valuable data for instruction set design, for strategies of program optimization and code generation, and for developing suitable methodologies for hardware specification, design, debugging, verification, and fault tolerance.

Project 3: Program Translation. Our first steps on the problem of translating VAL into

data flow machine code were limited to programs that do not process data structures. Montz [26] showed how to reduce the number of acknowledge arcs in data flow programs while supporting pipelined operation at the maximum rate. Todd [32] developed translation rules for converting the basic VAL constructs for conditionals, iteration, and distribution into pipelined data flow machine code.

The availability of the engineering model has focused interest on the problems of choosing good program structures and generating efficient machine code. Since VAL is a functional programming language in which side effects are excluded, the analysis of data dependencies for purposes of program optimization is straightforward.

Project 4: Data Structures. A major subject of current study is how computations involving structured data should be handled on data flow computers. William Ackerman is pursuing doctoral research on the concept of spatial interleaving of arrays. In this scheme large arrays are divided into many fragments which are processed at different sites within a data flow computer. Programs to be analyzed are represented as abstract program graphs and transformations performed on them to put the program in a form that can run efficiently on a general class of parallel computers.

In a master's thesis, Gao is developing the basis of a general scheme for using pipelined operation of data flow machine code to match parallelism in an application to parallelism in the machine [16]. We have successfully applied this scheme to the global weather model [14] and are evaluating the extent to which this paradigm of machine program structure can be applied to other computations that have a less regular structure.

Project 5: Architecture Description and Simulation. As our ideas about data flow computer architecture have crystallized, we have become interested in the problem of writing satisfactorily precise specifications of hardware units. Some experiments with various notations appear in earlier publications [18, 19]. This work has now come to

fruition: Clement Leung and Willie Lim have completed the Preliminary Reference Manual for PADL [24]--a specification and design language for systems employing the packet communication architecture that characterizes our concept of data flow computers. The PADL language will be used to specify hardware units for projected prototype data flow computers. Already PADL has been used to specify the basic router module, and to develop a preliminary design for the array memory module of our proposed static data flow architecture.

Project 6: Specification of a Form 2 Data Flow Processor. Our studies have shown that the originally proposed Form 2 data flow processor is the best basis for developing practical machines for large-scale numerical computation. We now call this machine organization the *static* data flow architecture. It is significantly different from the Form 2 machine as a result of studies of program structure, hardware design tradeoffs, and analysis of specific applications.

1. Data structures are to be supported as arrays allocated to contiguous sets of locations in conventional random access memory devices. A more general scheme of representation had been proposed but is too complex for machines of the near future [2].

2. We have found that the static architecture is considerably more flexible than we had appreciated. It appears possible to implement most features of appropriate high-level functional languages through use of suitable machine level program structures [29, 31].

3. The appropriate size of the instruction memory of data flow processing elements is, on the basis of application codes we have studied, much larger than we had thought, and this requires rethinking the hardware designs developed earlier.

Further discussion of the proposed static architecture for data flow computers is provided under Proposed Research.

In addition to the six principal projects, supporting studies have been carried out on several topics where new knowledge is needed to provide a solid basis for a successful

implementation project.

Analysis of Applications. In studying the problems of generating good machine code programs for data flow computers, we are using several benchmark programs supplied by research sponsors. The attached paper [14] includes an analysis of a global weather model code from NASA. This code has been translated by hand from FORTRAN into VAL, and the hydrodynamics portion of the VAL program mapped into data flow machine code. Other large application codes being studied include a Navier-Stokes code, also from NASA, and the Simple code provided by the Livermore Laboratory.

Packet System Architecture. Our work on the basis of packet system architecture spans the range from theoretical to pragmatic. Tam-Anh Chu is evaluating alternative designs for the processing element of a static data flow supercomputer. Willie Lim is investigating alternative approaches to developing LSI designs suitable for digital systems having packet architecture. Our group has already devoted considerable effort to developing design methodologies using the principles of self-timed systems [28, 19], and studying application of this approach to the key module types for the static data flow architecture [9, 10]. Another approach being studied is based on the "stoppable clock" -- each LSI chip is internally timed using a clock circuit that can start and stop reliably in response to the presence or absence of signals indicating tasks to be performed. The testability of modules employing asynchronous operation is also of interest. Router modules for the prototype evaluation facility have been built with testing features that allow diagnosis of a large routing network without disassembly [22, 23].

Routing Networks. In his doctoral research G. Andrew Boughton [6] has completed the first major part of his doctoral research on the structure and performance of packet routing networks by sharpening his analysis of the case in which the cost and delay of individual connections between routing modules is constant. The second phase of this research concerns how these cost and delay factors would affect the structure and performance of

VLSI realizations in which a major part of a packet network would be implemented within a single chip.

Semantics of Nondeterminacy. Our work in the area of formal semantics of computer languages and systems includes the doctoral research of Dean Brock, who uses a concept called "scenarios" to capture the ordering information essential to giving a unique description of the behavior of a non-determinate concurrent program [8]. His ideas overcome the flaw discovered in the use of mathematical relations for this purpose.

A General Purpose Data Flow Computer System. Professor Dennis is developing the formal description of a general purpose computer system of unusual characteristics using a novel technique of writing the formal operational semantics for computer systems [13]. The proposed computer system is advocated as an exploration of a combination of assumptions that seems very attractive: many interactive users sharing centrally stored information; concurrency at the level of individual instructions; all information accessed through a universal mechanism using unique identifiers; and *no data changes* -- that is, data are created, used in defining other data, and discarded when no longer of interest. Our plans for this work are given in a paper submitted for publication [15].

Related Work

Of all the computer architectures currently under consideration, only the data flow projects combine development of hardware architecture with development of an appropriate, sound programming language and methodology. Of the data flow projects, it seems to us that only the project we propose here has a realistic chance of realizing supercomputer performance within the next few years.

Among the projects of interest are the Stanford/Livermore S-1 machine, the HEP machine of Denelcor, and CDC's Advanced Flexible Processor. The S-1 is a conventional shared-memory multiprocessor [34] and has the usual problems- it uses expensive fast

circuitry and does not expand gracefully to large numbers of processing elements. No programming methodology for exploiting many processors on one problem has been developed for this architecture.

The Heterogeneous Elements Processor (HEP) offers an intermediate position between conventional multiprocess realizations and data driven instruction execution [7]: the design includes a relatively efficient means for communicating data from one process to another. Consequently, a finer level of concurrency is achievable without loss of performance through high cost of managing process interaction. The HEP architecture retains the problems of shared main memory, and I am not aware of the development of a suitable programming methodology.

The Advanced Flexible Processor (AFP) developed by the Control Data Corporation uses a totally different principle of machine level program organization. It is very suited to the pipelined operation of many processors performing parts of a large computation [11]. However the structure of the processor restricts the variety of computations that may be supported efficiently and the generation of the needed control words could be a severe problem. No suitable programming methodology has been developed for this machine, but one interesting possibility is the design and implementation of a compiler for a functional language such as VAL.

Work on multiprocessor systems is also being carried out at New York University, and at the University of Texas in Austin. The work at NYU is based on certain processor/memory interconnect configurations which are well matched to the data-access patterns of some important algorithms [27]. It has not been shown that the high performance possible in principle can be realized within the framework of a stored program, general purpose machine. The work at Texas attempts the implementation of many ideas--some good, some bad--and is not likely to produce significant results unless the effort is focused on a more coherent overall concept of system operation [17]. In both

projects I have not seen an adequate guiding methodology of program structure.

Other projects in data flow computation are related to the proposed effort. The project at the Texas Instruments Company was the closest in spirit to our present proposal. That effort provided a practical demonstration that a data flow computer can provide a speed-up equal to the number of processing elements. However, the TI project did not address effectively the problem of handling the data bases associated with large numerical computations. Recently, this group has been pursuing a different concept of data flow architecture better suited to process control and military command and control systems.

Professor Arvind (a colleague at MIT) has proposed a tagged token architecture that uses a novel approach to handling structured data and envisions hundreds of processing elements, each complete on a single VLSI chip [5]. A small associative memory is to be used in each processing element to perform the matching of tags.

At Manchester University in England, Gurd and Watson have assembled a prototype data flow computer which uses the tagged token principle [33]. This machine has been demonstrated with up to four processing elements In each processing unit a large associative store is used to perform token matching and to hold data structures.

We are aware of interesting work being done in Japan, especially the work of Amamiya of the Nippon Telephone and Telegraph company [4]. Although several experimental machines have been built, it is not clear that any of these are the basis for a practical, large-scale data flow machine, and there is at present no firm industrial commitment to the development of data flow machines. Nevertheless, the considerable interest in the subject in Japan is likely to lead to strong efforts in the future.

Proposed Effort

We propose to design and fabricate a basic set of LSI parts from which data flow machines may be constructed. The class of data flow machines we propose to construct is

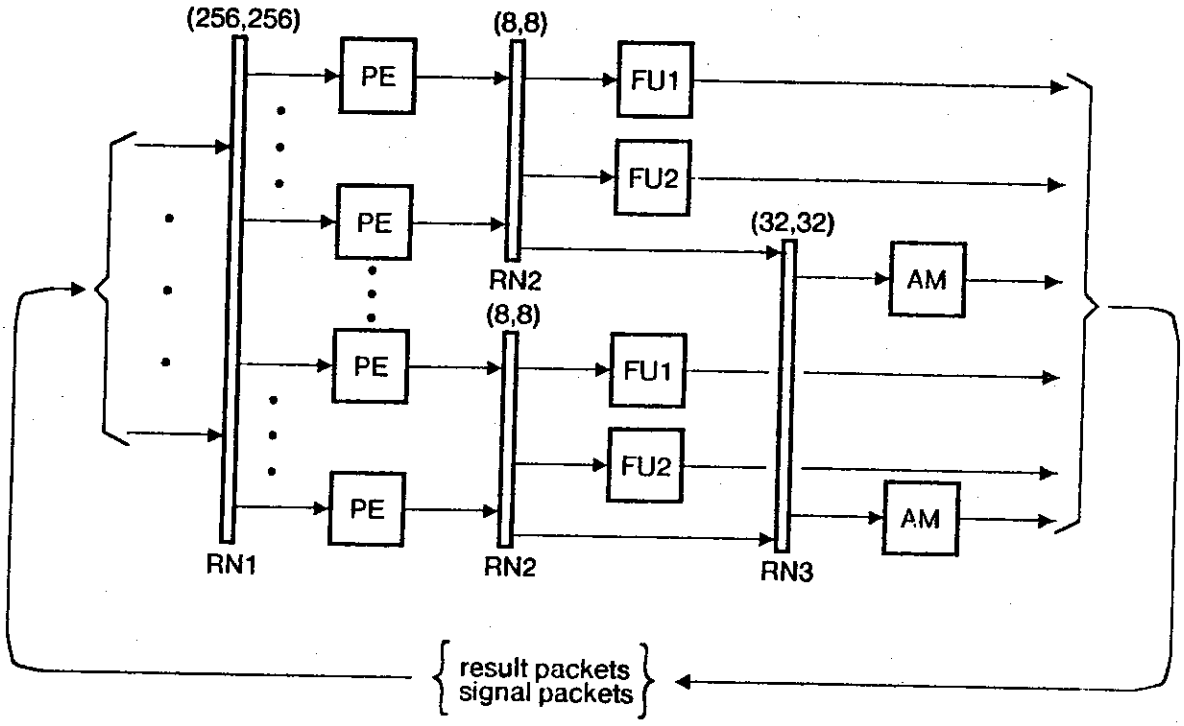

Fig. 1. Structure of the proposed prototype machine.

illustrated in Fig. 1 and explained further in the attached paper. The main units of the machine are:

- PE: Processing Elements. These units hold instructions of the machine level program. They receive result packets from other instructions and determine which instructions are ready for execution.

- FU1,FU2: Functional Units. These units perform the basic scalar operations.

- AM: Array Memories. These memory units hold the array values that form the structured data of the computation.

- RN1: Routing Network (256-in, 256-out). This network delivers result packets to the processing elements that contain their target instructions.

- RN2: Routing Networks (8-in, 8-out; 32 networks). This network transmits ready instructions (with their operands) to an appropriate functional unit, or to an array memory unit via RN3 if an operation on structured data is called for.

- RN3: Routing Network (32-in, 32-out). This network routes instructions that operate on structured data to the array memory module that holds the relevant data.

Each of the routing networks is built of individual two-by-two router units, each being a single LSI part. Network RN1, the largest network, consists of eight stages with 128 two-by-two routers in each stage--a total of 1024 router units.

The thesis to be tested is that high performance can be achieved through massive parallelism at less expense than through ultra-fast circuits. Hence the technology in which the basic units of the proposed machines will be fabricated is NMOS or CMOS. In this case machines can be built from RAM chips and the five basic custom LSI devices listed in Fig. 2.

Machine Unit	*Part Count*
Cell Block (256)	
Cell Block Controller (custom)	256
RAM Chips (8 apiece)	2048
Array Memories (32)	
Array Memory Controller (custom)	32
RAM Chips (32 apiece)	2048
Adders (four per section) (custom)	
Multipliers (three per section) (custom)	
Routing Networks	
RN1 (256 by 256)	1024
RN2 32 networks (8 by 8)	384
RN3 (32 by 32)	80
Subtotals	
Routers (2 by 2) (custom)	1488
Other Custom Chips	512
RAM Chips	4096
Total	6096

Fig. 2. Part counts for a data flow supercomputer.

Due to the pin limitations on practical devices, we assume that packets are transmitted is byte-serial format using 16-bit bytes. The 225 MHz computation rate is based on a byte transmission rate of eight MHz. which is within the performance range of connections to NMOS and CMOS devices.

These numbers are for a data flow computer which, according to our preliminary analysis of the global weather code, could perform that computation at the rate of one time step in each five seconds of computation -- a speed of about 225 MIPS or 180 MFLOPS. For a machine of higher performance, these numbers may be simply doubled for each doubling of computation speed except that the distribution network will require one more stage for each doubling of performance. For example, to achieve 360 MFLOPS capacity, in addition to multiplying all part counts by two, the addition of 256 two-by-two routers to make up a ninth stage of the distribution network is required.

In a static data flow computer instructions of the machine-level program are assigned to processing elements (instruction cell blocks) by the compiler and program loader, and this allocation is fixed for the duration of the computation. The operands of instructions are stored as part of the instructions, so an instruction cannot be reused until the result of its previous execution is consumed. For this reason, pipelining of data through the instructions of the machine-level program is an important means for achieving high performance. A carefully designed optimizing compiler is required.

In bringing a large scale data flow machine into operation for the first time design errors and faults in operation are certain to be discovered. We wish to make sure that a clear distinction can be made between three classes of difficulties: software problems such as erroneous source programs and incorrect compilation; hardware design errors; and failures of hardware elements. Because the envisioned machines are highly asynchronous, it is particularly important that failure of hardware be distinguished from other sources of error. For this reason, we plan to implement full-coverage hardware fault detection in the

proposed machines, adapting the methodology developed by Leung [20, 21]. Simulation will be used to verify the functional correctness of hardware designs.

The schedule of work is shown in Fig. 3. In addition to support for graduate students and staff, the proposed budget requests equipment funding as follows:

Year 1: (1982-83) $50K
Equipment to support design, testing and verification of LSI logic circuits.

Year 2: (1983-84) $100K
Mask generation and circuit fabrication costs for custom device development; test equipment.

Year 3: (1984-85) $300K
LSI device production for prototype machine; packaging, wiring, power supply, cabinetry, and cooling equipment; host computer for programming support systems.

1982-83

- Design instruction set.
- Develop fault detection scheme.
- Preliminary module design (PE and AM).
- Fabricate and evaluate 2-by-2 router chip.
- Acquire CAD tools.
- Design and acquire test facility.
- Develop compilation techniques.

1983-84

- Final designs for modules.
- Logic simulation for design verification.
- Fabrication of experimental chips.
- Mechanical and packaging design for prototype machines.
- Compiler implementation.

1984-85

- Device production.
- Prototype assembly and checkout.
- Develop programming support system.
- Evaluation.

Fig. 3. Project work schedule.

References

1. Ackerman, W. B. Data flow languages. *Computer 15*, 2 (February 1982), 15-25.

2. Ackerman, W. B. A Structure Memory for Data Flow Computers. Technical Report TR-186, Laboratory for Computer Science, MIT, Cambridge, MA 02139, August, 1977.

3. Ackerman, W. B. and Dennis, J. B. VAL -- A Value - Oriented Algorithmic Language: Preliminary Reference Manual. Technical Report TR-218, Laboratory for Computer Science, MIT, Cambridge, MA 02139, June, 1979.

4. Amamiya, Makoto, et al. Data Flow Machine Architecture. Musashino Electrical Communication Laboratory, Nippon Telephone and Telegraph, May, 1980.

5. Arvind, and Kathail, V. A Multiple Processor Dataflow Machine That Supports Generalized Procedures. Eighth Annual Symposium on Computer Architecture, May, 1981, pp. 291-302.

6. Boughton, G. A. Routing Networks for Packet Communication Systems. Ph.D. Thesis Proposal. Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, ,January, 1981.

7. Brinton, J. B. System grows from 10 to 160 MIPS. *Electronics 55*, 4 (1982), 161-163.

8. Brock, J. B., and W. B. Ackerman. Scenarios: A Model of Non-determinate Computation. Lecture Notes in Computer Science, Formalization of Programming Concepts, Springer-Verlag, New York, 1981, pp. 252-259.

9. Chu, T.-A. Determination of Throughput Rates for the Cell Block. Computation Structures Note 43, Laboratory for Computer Science, MIT, Cambridge, MA 02139, November, 1981.

10. Chu, T.-A. Circuit Analysis of Self-Timed Elements for NMOS VLSI Systems. Master Th., Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, June, 1981.

11. Control Data Corporation. Advance Flexible Processor: Design Specification. Information Sciences Division, Control Data Corporation, Minneapolis, Minnesota, 1979.

12. Dennis, J. B., Boughton, G. A., and Leung, C. K. C. Building Blocks for Data Flow Prototypes. Proceedings of the 7th Annual Symposium on Computer Architecture, May, 1980, pp. 1 - 8.

13. Dennis, J. B. An Operational Semantics for a Language with Early Completion Data Structures. In *Formal Description of Programming Concepts*, Springer-Verlag, Berlin, 1981.

14. Dennis, J. D., G. R. Gao, and K. W. Todd. A data flow Supercomputer. Laboratory for Computer Science, MIT, Cambridge, MA 02139, 1982.

15. Dennis, J. B. . Paper in preparation.

16. Gao, G. R. An Implementation Scheme of Array Operations in Static Data Flow Machine. S.M. Thesis Proposal. Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, ,April, 1981.

17. Kapur, R. N., U. V. Premkumar, and G. J. Lipovski. Organization of the TRAC Processor-Memory Subsystem. AFIPS Conference Proceedings, May, 1980, pp. 623-629.

18. Leung, C. K. C., D. P. Misunas, A. Neczwid, and J. B. Dennis. A Computer Simulation Facility for Packet Communication Architecture. Third Annual Symposium on Computer Architecture, Institute of Electrical and Electronics Engineers, Piscataway, N. J., 08854, 1976, pp. 58-63.

19. Leung, C. K. C. ADL: An Architecture Description Language for Packet Communication Systems. Proceedings of the 4th International Symposium on Computer Hardware Description Languages, IEEE, October, 1979, pp. 6-13.

20. Leung, C. K. C., and Dennis, J. B. Design of a Fault Tolerant Packet Communication Architecture. The 10th International Symposium on Fault Tolerant Computing, IEEE, July, 1980, pp. 328-335.

21. Leung, C. K. C. Fault Tolerance in Packet Communication Architectures. Technical Report TR-250, Laboratory for Computer Science, MIT, Cambridge, MA 02139, September, 1980.

22. Lim, W. Diagnostic Hardware of the Prototype 2 x 2 Router. Laboratory for Computer Science, MIT, Cambridge, MA 02139, 1982.

23. Lim, W. A Test Strategy for Networks of 2 x 2 Routers. Laboratory for Computer Science, MIT, Cambridge, MA 02139, 1982. Forthcoming.

24. Lim, W., and C. K. C. Leung. PADL: A Packet Architecture Description Language Preliminary Reference Manual. In preparation.

25. McGraw, J. Data flow computing: the VAL language. *ACM Transactions on Programming Languages and Systems 4*, 1 (January 1982).

26. Montz, L. B. Safety and Optimization Transformations for Data Flow Programs. Technical Report TR-240, Laboratory for Computer Science, MIT, Cambridge, MA 02139, January, 1980.

27. Schwartz, J. T. Ultracomputers. *ACM Transactions on Programming Languages and Systems 2*, 4 (October 1980), 484-521.

28. Singh, N. P. A Design Methodology for Self-Timed Systems. Master Th., Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, January, 1981.

29. Stoy, J. E. . Unpublished memorandum

30. Todd, K. W. An Interpreter for Instruction Cells. Laboratory for Computer Science, MIT, Cambridge, MA 02139, July, 1981.

31. Todd, K. W. . Unpublished memorandum

32. Todd, K. W. High Level VAL Constructs in a Static Dataflow Machine. Master Th., Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, February, 1981.

33. Watson, I. and J. Gurd. A Practical Data Flow Computer. *Computer 15*, 2 (February 1982), 51-57.

34. Widdoes, L. C., Jr. The S-1 Project: Developing High-Performance Digital Computers. COMPCON Spring 1980, February, 1980, pp. 282-291.

Current Support

National Science Foundation

Title: Data Flow Computer Architecture
Principal Investigator: Jack B. Dennis
Period: 1 March, 1980 to 28 February, 1983
Amount: Second year: $81K; Third Year: $90K"

National Aeronautics and Space Agency

Title: High Speed Data Flow Architecture for the Solution of the Navier-Stokes Equations
Principal Investigator: Jack B. Dennis
Period: 1 February, 1981 to 1 March, 1982
Amount: $25K"

Proposals Pending

National Science Foundation

Title: Experimental Computer System Based on Functional Programming Concepts
Principal Investigator: Jack B. Dennis
Period: 1 April, 1982 to 31 March, 1984
Amount: First year: $248K; Second year: $209K

NASA: We may continue work with the Ames Research Center on analysis of Aerodynamic simulation codes and there is a possibility of support for further analysis of weather simulation codes from the Goddard Institute for Atmospheric Research.