MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Laboratory for Computer Science

Computation Structures Group Memo 160

Data Flow Computer Architecture

(Research Proposal Submitted to the Department of Energy)

May 1978

A. Introduction

The MIT Laboratory for Computer Science (LCS) proposes a three-year program in collaboration with the Lawrence Livermore Laboratory (LLL) to exploit principles of data flow computer architecture first developed in the MIT Computation Structures Group, to achieve high performance (50 to 100 times current speeds) in application to important problems of interest to the Livermore Laboratory. This program will include construction of an engineering model data flow processor to evaluate the practical application of data flow principles, and the development of specifications for a large scale data flow computer able to meet LLL requirements. The substance of this research program is outlined for a three year period beginning January 1, 1979. The attached budget covers the first year of the effort.

In this joint program of MIT and LLL, MIT's role will be the specification, design, and construction of practical data flow computers, and the design and implementation of languages and support software; LLL's role will be in the evaluation, criticism, and documentation of specifications and designs, and in facilitating communication among workers in data flow computation and related fields.

The sections of this proposal present the background leading to the proposed program of research, organized as a series of goal-oriented projects together with supporting studies, and a discussion of activities to be carried out jointly with LLL.

Two of our recent publications [DEN77-1, DEN77-2] explain in some detail the current status of our work on data flow computer and algorithm design, and are attached to this proposal.

B. Background

In November of 1976, a presentation by MIT at LLL led to serious interest in the work being done on data flow computer architecture in the Computations Structures Group at the MIT Laboratory for Computer Science. In July of the next year, a workshop conference organized by MIT and sponsored by LLL brought together researchers on data flow and related fields from institutes in the United States and Europe. At this workshop, members of each group reported on the status and direction of their work. A summary of the reports has been compiled by David Misunas and published as [MIS77].

A one year contract between MIT and LLL has been negotiated to fund further research on the part of both institutions in the application of data flow principles for applications requiring performance beyond the capacity of present machines. The goals of this collaboration are fourfold:

1. To analyze existing speed limited computations to evaluate the prospects for a substantial increase in speed if run on a data flow computer,

2. to support this evaluation by designing a programming language suitable for expressing programs to provide for efficient execution on a data flow computer,

3. to develop schemes for representing data structures and for implementing data structure operations in memory systems capable of highly concurrent operation, and

4. to prepare proposals to the Department of Energy for the continuation of this research program.

The present proposal constitutes our work on number 4 and represents MIT's proposed participation in our future joint effort with LLL to develop practical data flow computer systems. The participation of LLL in the joint program is covered as a portion of work proposed separately to the Department of Energy.

C. Research status

The concept of data driven instruction execution, which is fundamental to the highly concurrent operation of our proposed data flow computers, was suggested by several authors as early as the year 1963 [SHA71] [SEE63]. At the same time, a branch of the theoretical study of program schemas investigated formal properties of concurrency in *parallel program schemata*. The work on schemata evolved directly into present ideas about data flow program representation [KOS73, DEN75-2, ARV75]. However, the early hardware proposals were primitive and incomplete, and could not be used to base a practical implementation project. Once we obtained fuller understanding about the type of programming language a data driven machine would require, [WEN75, BRO78-1, ARV76, KOS75] a number of interesting proposals for corresponding system organizations have appeared [ARV77-1, DAV78, SYR77, RUM75-2]. Dennis and Misunas [DEN73, DEN74, DEN75-1] have published a series of papers and reports covering the evolution of a scheme for data flow computers that we propose to pursue.

The general form of data flow computer foreseen in our work is shown in Figure 1. The machine consists of four major sections connected by channels through which information is sent in the form of discrete packets. The sections are:

Memory section -- consists of Instruction Cells which hold instructions and their operands.

Processing Section -- consists of Processing Units that perform the basic operations on data values.

Arbitration Network -- delivers *Operation Packets* from the Memory Section to the Processing Section.

Distribution Network -- Delivers *Result Packets* from the Processing Section to the Memory Section.

Briefly, instructions held in the Memory Section are enabled for execution by the arrival of their operands in Result Packets from the Distribution Network. Enabled instructions, together with their operands, are sent as Operation Packets to the processing section through the Arbitration Network. The results of instruction execution are sent through the Distribution Network to the Memory Section where they become operands of other instructions. With this overview, we are ready to consider the operation of each section in more detail.

The Memory Section of the processor is a collection of Instruction Cells, which hold a representation of a data flow program. Each Instruction Cell has a unique identifying address, the *Cell identifier*. An occupied Cell holds one *instruction* of a data flow machine level program.

An instruction consists of an *Operation code* and several *destinations* which specify what is done with the results of instruction execution. Each Cell also contains three Receivers which await the arrival of values for use as operands by the instruction. Once an Instruction Cell has received the necessary operand values and acknowledge signals, indicating that previous results of instruction execution have been consumed, the Cell becomes enabled and sends an operation packet, consisting of the instruction and the operand values, to the appropriate Processing Unit through the Arbitration Network.

The Arbitration Network provides a path from each Instruction Cell to each processing unit, and sorts the Operation Packets among its output ports according to the operation codes of the instructions they contain. For each operation packet received, a Processing Unit performs the operation called for by the instruction using the operand values in the packet, and produces one or more *Result Packets* which are sent to Instruction Cells through the Distribution Network. Each result packet consists of a result value and a destination derived from the instruction by the Processing Unit. The Distribution Network delivers Result Packets to Receivers of Instruction Cells as specified by their destination fields; that is, the Result Packets are sorted according to their destinations.

Arrival of a result packet at an Instruction Cell either provides one of the Receivers of the Cell with an operand value or delivers an acknowledge signal; if all Result Packets required by the instruction in the Cell have been received, the Instruction Cell becomes enabled and dispatches its contents to the Arbitration Network as a new operation packet.

Note that the functions performed by the processing unit of a conventional machine are distributed among several sections of the data flow processor. The operations specified by instructions are carried out in the Processing Section, but control of instruction sequencing is a function of the Instruction Cells of the Memory Section, and the decoding of operation codes is partially done within the Arbitration Network. Also unusual is that address fields (destinations) of instructions only specify where results are to go, and do not concern the determination of operand values. Instead of instructions having to ask for their operands, the operand values are sent to the instructions.

We emphasize that all communication between parts of the data flow processor is by packet transmission over the channels shown explicitly in Figure 1; there are no connections other then

those shown in the figure. Furthermore, the transmission of packets over each channel is done using an asynchronous protocol so the four sections of the processor may operate independently, without need for a clock or other central source of timing signals. Systems organized to operate in this manner are said to have *packet communication architecture.*

In particular, note that the Instruction Cells are assumed to be physically independent, so at any time many of them may be enabled. The Arbitration Network can be designed so that many operation packets may flow into it concurrently and be funneled into dense streams of packets directed to processing units. Similarly, the Distribution Network is designed to distribute dense streams of result packets efficiently to the Instruction Cells through highly concurrent operation. In this way, highly parallel operation of the entire processor is achieved, and the appetites of pipelined processing units can be satisfied.

Our work is language based in that each of our proposed architectures is tied to a definite expressive level of data flow programming language: each machine will faithfully execute any program expressed in the corresponding language subject to the usual limitations of memory size and computation speed. Moreover, a programming language designed for data driven computation can have characteristics that are consistent with current thinking about good program structure and language design: these characteristics include freedom from side effects, which yields clarity of meaning, ease of verification and simpler specification of program modules.

Our architectural studies have focused on four forms of data flow computer corresponding to three data flow languages of different levels of expressive power

Form 1. This is the machine shown in Figure 1. It corresponds to a basic language level supporting scalar variables, and having conditional and iteration control structures. Since all data and instructions reside in the Instruction Cells, this form of data flow machine is suitable for fast computations involving relatively small programs and data.

This is the form of data flow machine that has been studied most intensively at MIT. A comprehensive account of its structure, operation, and application to the fast Fourier transform can be found in [DEN77-1], which accompanies this proposal. This machine seems well suited to a wide variety of signal processing applications. Since this form is best understood, it is the basis of many supporting studies in our research group: studies of Instruction Cell design [AMI77], routing network structure [BOU78], byte serial pipelined processing units [FER78], and the problems of generating machine level code [MON78]. The issues requiring resolution before building a prototype form 1 machine are: Design of the machine level instruction code; language translation, specifically algorithms for generating good machine code; and the speed/cost trade-offs to be made

in the logical design of hardware units.

Form 2. A Form 2 data flow machine is obtained by adding to the Form 1 machine a Data Structure Processor consisting of a Structure Controller and a Packet Memory System, as shown in Figure 2. Correspondingly, the language supported is extended to include a general class of data structures and operations for their construction and access. Since a program is still held in the Instruction Cells, the program size limitation of the Form 1 machine still applies. However, the Structure Processor may be designed to handle very large data bases.

A general discussion of Form 2 data flow machines prepared for the Symposium on High Speed Computer and Algorithm Organization [DEN77-2] outlined its application to global weather simulation, a problem requiring high performance and a large data base. Basic research studies on packet memory systems [DEN75-3] and on schemes for representing data structures and implementing operations on them [MIS78], [RUM75-1] have laid the foundations for a more detailed and comprehensive design study by [ACK77] for Structure Processors capable of handling large numbers of concurrent data structure operations. Further evaluative study and consideration of alternative implementation schemes for Structure Processors is needed. Also, we have not yet seriously studied how our implementation schemes would be realized in the most attractive hardware technologies.

Form 3. A Form 3 data flow processor supports the same language level as a Form 2 machine, but allows the execution of large programs. This is accomplished by including an Instruction Memory (probably a form of packet memory system) in the machine as shown in Figure 3, and arranging that only the most active instructions are held in instruction Cells during execution of a program. Thus the Instruction Cells act as a "cache" for the instruction memory.

The basic machinery needed to make the Instruction Cells act as a cache has been presented in [DEN75-1] as an extension to a primitive data flow machine. This work must be revised, extended and evaluated in the context of our more recent concepts of data flow architecture.

At present it appears that the needs of LLL applications demanding high performance computation would best be met by a Form 2 or a Form 3 data flow processor, depending on program size and the degree of speedup required.

Form 4. A Form 4 data flow computer is envisioned as supporting, at the hardware level, all fundamental aspects of data driven computations, including procedures, recursion and data streams. These machines would be sufficiently general to support all services of a general purpose computer system to a community of users.

Solving the conceptual and design problems of a Form 4 data flow computer is a very ambitious task. The major problem holding up progress toward a complete specification of a Form 4 computer is the design of a procedure execution mechanism that will operate effectively in the most general context. Contributions toward a solution to these problems have been made by Misunas [MIS78], and Miranker [MIR77], and the work at Irvine [ARV77-2] is a good source of ideas. Currently, Weng is studying this problem as part of his doctoral research at MIT.

The major focus of our present effort with Livermore is the definition of suitable high level languages for expressing programs for execution on data flow computers. Conventional programming languages such as Fortran, Algol and PL/I are unsatisfactory for this purpose because their semantics depend on the presence of an addressable random access memory, whereas a data flow machine, to achieve highly concurrent operation, does not include such a memory. A consequence of this dependence on an addressable memory is that use of side effects of program modules is essential to writing efficient programs. It is well known that use of side effects in programming leads to programs that are relatively difficult to understand, and hard to establish as correct using formal techniques. Provision of support for a general class of data structures in a language obviates much dependence on side effects in scientific computation (such as the principal uses of "common" in Fortran). We have found that introduction of streams of data as a form of communication between program modules is an attractive alternative to the use of side effects in input-output programming.

In his master's thesis, K.-S. Weng [WEN75] has studied formal properties of a data flow source language including operations on data streams. The source language studied by Weng permits recursive module invocations involving stream arguments, leading to complex execution time data structures. Our current work concerns applying this knowledge to the design of a source language better suited to use with a Form 2 data flow processor which does not include hardware support for recursion.

For this work, we felt that it would be best to take advantage of existing language designs to the extent feasible. The natural starting point for us is the language CLU designed by Professor Liskov and her research group at MIT [LIS77]. CLU is attractive as a starting point for two reasons: first, it is in the line of programming language development including Simula and Pascal, the design of which builds on knowledge of structured programming. Secondly, the semantics of CLU support a very general level of modular programming that demands use of heap storage management. We wish to retain this characteristic in our data flow source language.

Our current approach is to modify CLU so that all sources of side effects are removed, and to augment this base language with provisions for operating on data streams and using streams for

intermodule communication.

During the 1977 fall semester, Professor Dennis taught an MIT graduate course on "Data Flow Computer Architecture" in which the students were asked to study applications problems to evaluate the ease with which algorithms could be expressed in a data flow language and the computation rate that could be achieved by a hypothetical data flow computer. For this exercise, the students were presented with an early version of our proposed language through lectures, sample programs, and documentation of syntax and the basic data types. Ten papers were submitted, of which about half were very good. The ability of the students to clearly express their applications in this language was very encouraging. This experience also suggested areas of the language requiring further development, especially the provisions for expressing computations on streams.

Supporting research

In addition to work on the general architecture of data flow computers and programming language design, we are conducting basic research studies in many areas essential to the success of the proposed research programs.

- *Semantic Theory.* A semantic theory for a programming language provides a sound basis for ensuring that the language has desired properties, and is the basis for formal specification and verification techniques. For data flow programs, a semantic theory is required which deals with concurrency and nondeterminacy. The design of the right theory is an important open area of research. Ackerman, Brock, and Weng [BRO78-2, WEN75] are involved in this research.

- *Architecture Description Language.* To specify a complex machine organization with precision, a formal language is required. We have been developing such a language [LEU78] based on data flow concepts and oriented to the description of structures (such as our proposed data flow computers) having packet communication architecture.

- *Specification and Verification of Hardware.* David Ellis recently completed a Ph.D dissertation [ELL77] on specification and proof methods for systems having packet communication architecture.

- *Simulation of architectures.* A simulation facility for data flow architectures described in a limited architecture description language has been programmed and runs on a DEC PDP 11/70 system at LCS [LEU75]

- *Implementation Studies.* Studies of two critical aspects of our proposed Form I data flow processor have been completed. K. Amikura [AMI77] has developed a detailed design for an Instruction Cell from which it will be possible to determine the complexity, cost and speed of a practical logic design. G. Boughton [BOU78] has studied the structure and throughput of routing networks to determine structures that maintain good characteristics as the networks grow in size.

- *Fault Tolerance.* There has been little careful study of approaches to achieving fault tolerance in the context of general purpose computing systems, and asynchronous operation. Therefore, we have begun an investigation of this area which is the subject of current doctoral research.

D. Proposed work

The research program we propose to carry out is best described as a set of projects, each concerned with a major subgoal essential to the successful development of a practical large scale data flow computer. Successful completion of these projects will yield a detailed specification for a Form 2 data flow computer with confident estimates of cost and performance. In the following we discuss each project in terms of its goal, current status, and the steps required to achieve the goal. Each project will be documented by technical reports.

In addition to the projects above mentioned, we will also pursue related basic research, and we will work with the Livermore Laboratory on evaluation and application of our work.

Project 1: Data Flow Source Language

The design, documentation and evaluation of a user programming language for data flow computation is a matter of highest priority. Such a language specification is essential to providing for evaluation and use of data flow computers in applications. It will serve as the medium for testing completeness and correctness of our machine architectures, for developing program translation algorithms, and for evaluating prospective performance of data flow computers.

Since the different forms of data flow computers support levels of language expressive power, significant differences will exist between source languages designed for the four forms of data flow machine. We will concentrate on the design of a user language for Form 2 and Form 3 data flow machines, as this level of capability is required for Livermore applications and is achievable in a practical machine within the framework of the proposed research program.

Two aspects of the language design are important to obtaining high performance on a Form 2 or Form 3 machine: support for data structures and streams. In both cases, language design decisions interact strongly with details of the proposed architecture, and the language design will necessarily proceed hand-in-hand with the planning of corresponding elements of the hardware.

As mentioned in the background section, the outline of a proposed language has been developed and preliminary documentation of it is being prepared. We anticipate that the language design will evolve through several stages of evaluation and revision, before reaching a completely satisfactory design. The staff of the Livermore Laboratory will work closely with MIT in the evaluation, improvement and documentation of the language.

An implementation of the language on presently available computers will be essential to gain real programming experience with its new characteristics. We propose to program a translator and

interpreter in the language CLU which has been designed and implemented at the MIT/LCS to run on a PDP-10 system. Since the design of our proposed data flow user language is closely related to the design of CLU, much of the CLU software system would be used to simplify construction of an implementation of the data flow language.

Project 2: Construction of an Engineering Model

Our proposed architectures for data flow processors have the very important characteristic that they can be scaled up or down over a large range with no effect on the user language except for performance and the size of programs that may be run. Thus it is feasible to design and construct a modest "engineering model" data flow processor whose programming characteristics are identical with those of a much larger machine. Such an engineering model should be built for many reasons:

1. No digital system using asynchronous control on the scale contemplated has ever been built; it is necessary to develop and evaluate design, construction, and testing methodologies for realizing systems having packet communication architecture.

2. The engineering model will provide a sound basis for estimating the cost and performance of a large scale machine.

3. It will give us experience in running real programs on a data flow processor: this will provide practical demonstration of the correctness of the architecture and program translation algorithms as well as providing experience in testing and verifying application programs.

We propose to build an engineering model of a Form 1 data flow processor, so designed that it could be augmented with a structure processor to become a Form 2 processor. Its construction would use commercially available, state-of-the-art semiconductor devices. Construction of this processor requires some preliminary steps: design of an instruction code that is a reasonable target for translation algorithms, experimental logic design to evaluate cost-performance trade-offs for each section of the machine and simulation to choose an overall structure that yields a good balance of hardware utilization.

Project 3: Program Translation

Programs will be translated from a user language into a data flow machine language in two steps:

UL ---> DFG ---> ML

UL: User language
DFG: Data Flow graphs
ML: Machine language

Data Flow Graphs are a notation we use for the formal study of data flow concepts. If the User Language has appropriate characteristics (principally, absence of side effects), the first step in translation is a well known and straightforward process [WEN75]. The second step, generation of code in the Machine Language of a data flow processor, obviously depends on the instruction code of the processor. Here, entirely new methods of code generation must be developed due to the radical departure from conventional computer architecture. For our proposed Form 1 or Form 2 processors, the machine language program must be constructed so no instruction receives new operand values until the preceding value has been consumed through execution of instructions. Such machine level programs are said to be *safe*. Algorithms for generating safe data flow programs must be developed. This research is now in progress [MON78]. We propose to develop two implementations of our data flow source language: the first will be a translator and interpreter written in CLU; the second will provide language support for the engineering model to be built as Project 2.

Project 4: Data Structures

A consistent formulation of the semantics of data structures and corresponding hardware structures that can achieve efficient, highly concurrent operation is essential to successful realization of a Form 2 data flow processor. As reviewed under "Research Status", the design of support for data structures is an area of study emphasized in our current research. So far, several alternative implementation schemes for adding a data structure capability to a Form 1 data flow processor have been studied, the most detailed being that of Ackerman [ACK77]. Much remains to be done in comparing different schemes and understanding the relation between the architectural features and properties of the user language.

So far, we have not investigated the use of current and new technologies for constructing memory systems to support data structures in data flow computation. Project 4 will include a study of this question. The immediate goal is to develop designs for a Structure Controller and a Packet Memory System for use in a Form 2 data flow processor.

Project 5: Architecture Description and Simulation

We expect to describe the units of data flow processors using the Architecture Description Language (ADL) developed in the Computation Structures Group. The goal of this project is to

complete the design and documentation of our ADL and to continue development of our simulation facility for testing correctness and evaluating performance of our processor designs.

The present documentation of our ADL is [LEU78]; this version of our language is a result of our experience in writing descriptions of architecture units; it has been influenced considerably by data flow ideas. Our current simulation facility supports a very restricted version of our ADL and is implemented on a small machine --- a DEC PDP-11/70. To provide for simulation of more elaborate systems, the simulator must be moved to a PDP-10 system, improved to accept a more complete ADL, and to provide a more convenient user interface.

Project 6: Specification of a Form 2 Data Flow Processor

The goal of this project is to produce a detailed specification for a large scale Form 2 data flow processor which will realize a significant speed advantage for important Livermore applications. The specification will be sufficiently complete to determine cost and to be confident that the projected performance will be achieved. Clearly, our ability to write these specifications depends on the success of the preceding five projects.

The practical construction of a large scale data flow processor will benefit greatly from use of custom designed LSI devices. Hence, preparation of the specification will involve acquainting ourselves with progress in custom LSI services.

Work on the source language design, program translation, data structures and architecture description and simulation is already in progress (projects 1, 3, 4, and 5). Work on the Engineering model will begin as soon as resources are available; we expect that completing a useful experimental machine will require approximately two years of intensive effort. The specification of a large scale Form 2 data flow processor (project 6) will require about one year of effort, but will depend on the success of the other projects in progress in Supporting Studies (see below), especially in the areas of fault tolerance and hardware testing.

Supporting Studies

Supporting work in several areas is needed to ensure successful completion of the projects presented above.

Fault Tolerance -- Because of the component complexity of the projected Form 2 data flow processor, some approach to the detection and/or masking of hardware faults will be needed. We expect to continue our basic research on fault tolerance in systems having packet communication architecture.

Hardware Production Aids -- For the construction of a large scale data flow processor, access to a good set of automated design and production aids will be required. We plan to become familiar with available programs and study their integration with our architecture description language and simulation facility.

Testing and Correcting Asynchronous Systems -- Means are needed for finding design and implementation errors in large asynchronous systems such as our proposed data flow processors. Testing methods will be devised and evaluated in application to the engineering model project.

Fundamental Research

The Computation Structures Group will continue to pursue a program of fundamental research related to data flow computation. This will include theoretical work in formal semantics of languages and systems, the specification and verification of packet architectures, language design and translation, and architectural principles for Form 4 data flow computers.

Collaboration With Livermore

MIT will work closely with the Lawrence Livermore Laboratory in evaluating architectural proposals and language designs for applications of interest to Livermore. Since most of the relevant programs are written in dialects of Fortran and are not readily translatable into programs suitable for data flow computation, a major focus of our collaboration with Livermore will be on re-expressing these computations in the data flow user language being developed at MIT. Livermore plans to acquire a simulation capability to help evaluate the degree of concurrency available for exploitation by a data flow computer.

A major effort at Livermore is planned for developing software tools to aid the programmer in conversion of programs for execution on data flow processors. We expect to assist Livermore in this project.

The data flow workshop held in July 1977 [MIS77] brought together workers in data flow architectures, languages, and related subjects for four days of presentations, discussions, and exchanges of thoughts. Since this meeting was a successful and unique forum for communication in this field, we propose that MIT and Livermore organize and support future meetings of this kind. The next workshop is planned for July 9-13, again at the MIT Endicott House.

Many universities and research institutes have interests and expertise that will prove valuable in the course of our project. We expect that occasional contracts will be made to tap these sources of expertise as they are needed. The Livermore Livermore will administer these contracts, and MIT

will provide appropriate advice and evaluation. Some areas in which such assistance may prove useful include: simulation programs for data flow computation; program translation and optimization; design and implementation of arithmetic operations; fault tolerance techniques; and production aids for custom LSI devices.

Bibliography

[ACK77] Ackerman, W. B., *A Structure Memory for Data Flow Computers*, Laboratory for Computer Science (TR-186), MIT, Cambridge, Massachusetts, August 1977.

[AMI77] Amikura, K., *A Logic Design for the Cell Block of a Data Flow Processor*, Laboratory for Computer Science (TM-93), MIT, Cambridge, Massachusetts, December 1977.

[ARV75] Arvind, and K. P. Gostelow, "A New Interpreter for Data Flow and Its Implications for Computer Architecture," Department of Information and Computer Science (TR 72), University of California - Irvine, Irvine, California, October 1975.

[ARV76] Arvind, and K. P. Gostelow, "Programming in a Viable Data Flow Language," Department of Information and Computer Science (TR 77), University of California - Irvine, Irvine, California, March 1976.

[ARV77-1] Arvind, and K. P. Gostelow, "A Computer Capable of Exchanging Processors for Time," *Information Processing 77: Proceedings of IFIP Congress 77*, (B. Gilchrist, Ed.), August 1977, 849-853.

[ARV77-2] Arvind, K. P. Gostelow, and W. Plouffe, "Environments and Procedures," Department of Information and Computer Science (Data Flow Note 16), University of California - Irvine, Irvine, California, July 1977.

[BOU78] Boughton, G. A., *Routing Networks and Data Flow Architectures*, S. M. Thesis in preparation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Massachusetts, expected January 1978.

[BRO78-1] Brock, J. D., *Formal Semantics of Data Flow Language*, S. M. Thesis in preparation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Massachusetts, expected January 1978.

[BRO78-2] Brock, J. D., and W. B. Ackerman, "An Anomaly in the Specifications of Nondeterminate Packet Systems," Computation Structures Group (Note 33-1), MIT, Cambridge, Massachusetts, January 1978.

[DAV78] Davis, A. L., "The Architecture of DDM1: A Recursively Structured Data Driven Machine," Department of Computer Science (UUCS-77-113), University of Utah, Salt Lake City, Utah, October 1977. To appear in *Proceedings of the Fifth Annual Symposium on Computer*

Architecture, April 1978.

[DEN73] Dennis, J. B., and D. P. Misunas, "The Design of a Highly Parallel Computer for Signal Processing Applications," Computation Structures Group (Memo 101), Laboratory for Computer Science, MIT, Cambridge, Massachusetts, August 1974.

[DEN74] Dennis, J. B., and D. P. Misunas, "A Computer Architecture for Highly Parallel Signal Processing," *Proceedings of the ACM 1974 National Conference*, November 1974, 402-409. Also, Computation Structures Group (Memo 108), Laboratory for Computer Science, Cambridge, Massachusetts, August 1974.

[DEN75-1] Dennis, J. B., and D. P. Misunas, "A Preliminary Architecture for a Basic Data-Flow Processor," *The Second Annual Symposium on Computer Architecture: Conference Proceedings*, January 1975, 126-132. Also, Computation Structures Group (Memo 102), Laboratory for Computer Science, MIT, Cambridge, Massachusetts, August 1974.

[DEN75-2] Dennis, J. B., "First Version of a Data Flow Procedure Language," Laboratory for Computer Science (TM-61), MIT, Cambridge, Massachusetts, May 1975.

[DEN75-3] Dennis, J. B., "Packet Communication Architecture," *Proceedings of the 1975 Sagamore Computer Conference on Parallel Processing*, August 1975, 224-229. Also, Computation Structures Group (Memo 130), Laboratory for Computer Science, MIT, Cambridge, Massachusetts, August 1975.

[DEN77-1] Dennis, J. B., D. P. Misunas, and C. K. C. Leung, "A Highly Parallel Processor Using a Data Flow Machine Language," Computation Structures Group (Memo 134), Laboratory for Computer Science, MIT, Cambridge, Massachusetts, January 1977.

[DEN77-2] Dennis, J. B., and K.-S. Weng, "Application of Data Flow Computation to the Weather Problem," *High Speed Computer and Algorithm Organization*, (D. J. Kuck, D. H. Lawrie, and A. H. Sameh, Eds.), 1977, 143-157. Also, Computation Structures Group (Memo 147), Laboratory for Computer Science, MIT, Cambridge, Massachusetts, May 1977.

[ELL77] Ellis, D. J, *Formal Specifications for Packet Communication Systems*, Laboratory for Computer Science (TR-189), MIT, Cambridge, Massachusetts, November 1977.

[FER78] Feridun, A. M., "Design of an On-line, Byte-level Pipelined Arithmetic Processor," B. S. Thesis in progress, MIT, 1977.

[KOS73] Kosinski, P. R., "A Data Flow Language for Operating Systems Programming,"

Proceedings of ACM SIGPLAN-SIGOPS Interface Meetings, SIGPLAN Notices 8, 9(September 1973), 89-94.

[KOS75] Kosinski, P. R., "Mathematical Semantics and Data Flow Programming," *Conference Record of the Third ACM Symposium on Principles of Programming Languages*, January 1976, 95-103. Also, Computation Structures Group (Memo 135), Laboratory for Computer Science, MIT, Cambridge, Massachusetts, December 1975.

[LEU75] 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: Conference Proceedings*, January 1976, 58-63. Also, Computation Structures Group (Memo 127-1), Laboratory for Computer Science, MIT, Cambridge, Massachusetts, November 1975.

[LEU78] Leung, C. K. C., "ADL: An Architecture Description Language for Packet Communication Systems," Computation Structures Group, Laboratory for Computer Science, MIT, Cambridge, Massachusetts, In preparation.

[LIS77] Liskov, B. H. *et al.*, "Abstraction Mechanisms in CLU," *Communications of the ACM 20*, 8 (August 1977), 564-576.

[MIR77] Miranker, G. S., "Implementation of Procedures on a Class of Data Flow Procedures," *Proceedings of the 1977 International Conference on Parallel Processing* (J. L. Baer, Ed.), August 1977, 77-86.

[MIS77] Misunas, D. P., "Workshop on Data Flow Computer and Program Organization," *Computer Architecture News 6*, 4(October 1977), 6-22. Also, Laboratory for Computer Science (TM 92), MIT, Cambridge, Massachusetts, November 1977.

[MIS78] Misunas, D. P., *A Computer Architecture for Data-Flow Computation*, Laboratory for Computer Science (TM 100), MIT, Cambridge, Massachusetts, March 1978.

[MON78] Montz, L., *Safety and Optimization Transformations for Data Flow Programs*, S. M. Thesis in preparation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Massachusetts, expected September 1978.

[RUM75-1] Rumbaugh, J. E., *A Parallel, Asynchronous Computer Architecture for Data Flow Programs*, Laboratory for Computer Science (TR-150), MIT, Cambridge, Massachusetts, May 1975.

[RUM75-2] Rumbaugh, J. E., "A Data Flow Multiprocessor," *Proceedings of the 1975 Sagamore*

Computer Conference on Parallel Processing, August 1975, 220-223.

[SEE63] Seeber, R. R., and A. B. Lindquist, "Associative Logic for Highly Parallel Systems," *Proceedings of the AFIPS Conference 24*, 1963, 489-493.

[SHA71] Shapiro, R. M., H. Saint, and D. L. Presberg, "Representation of Algorithms as Cyclic Partial Orderings," Applied Data Research (CA-7112-2711), Wakefield, Massachusetts, 1971.

[SYR77] Syre, J. C., D. Comte, and N. Hifdi, "Pipelining, Parallelism and Asynchronism in the LAU System," *Proceedings of the 1977 International Conference on Parallel Processing* (J. L. Baer, Ed.), August 1977, 87-92.

[WEN75] Weng, K.-S., *Stream-Oriented Computation in Recursive Data Flow Schemas*, Laboratory for Computer Science (TM-68), MIT, Cambridge, Massachusetts, October 1975.

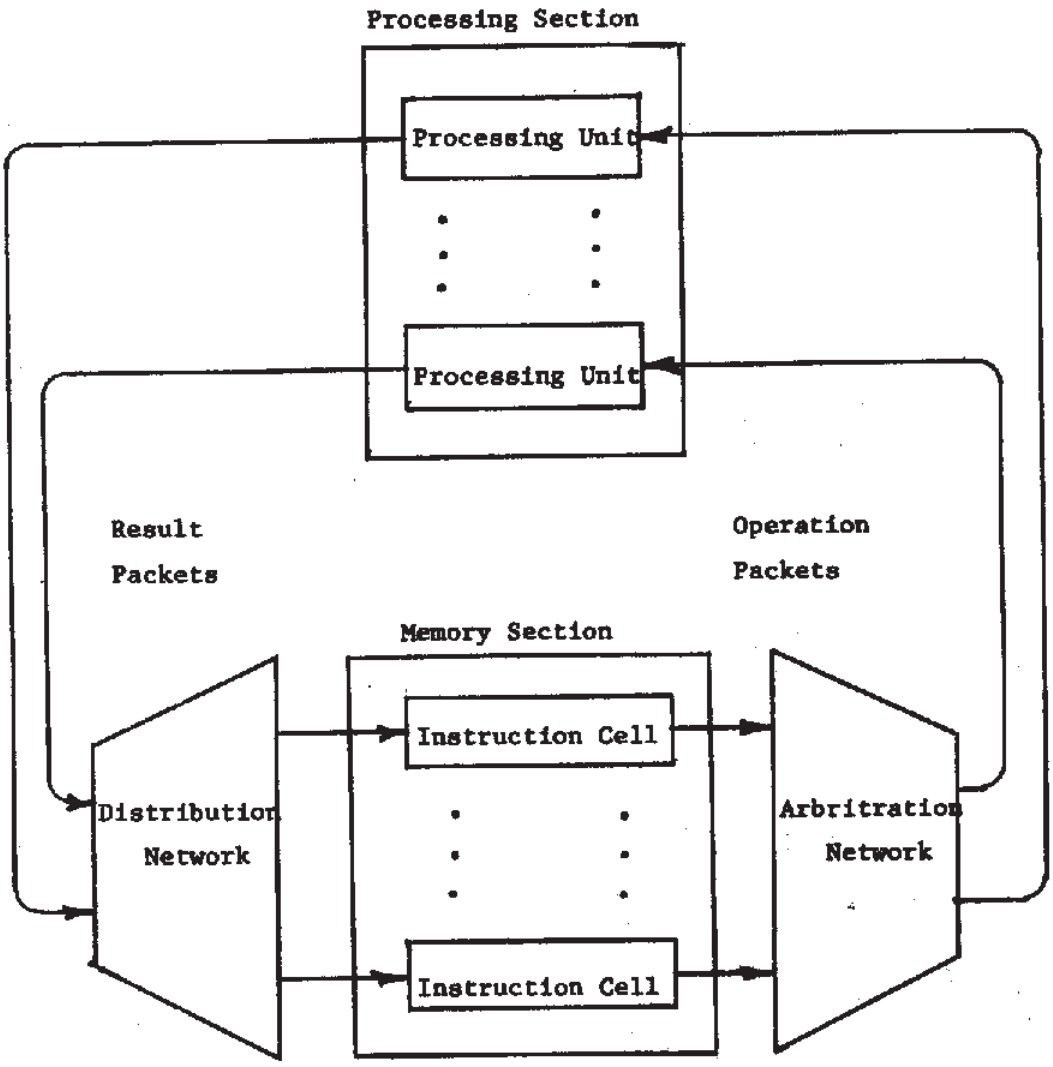

Figure 1. Form 1 Data Flow Processor

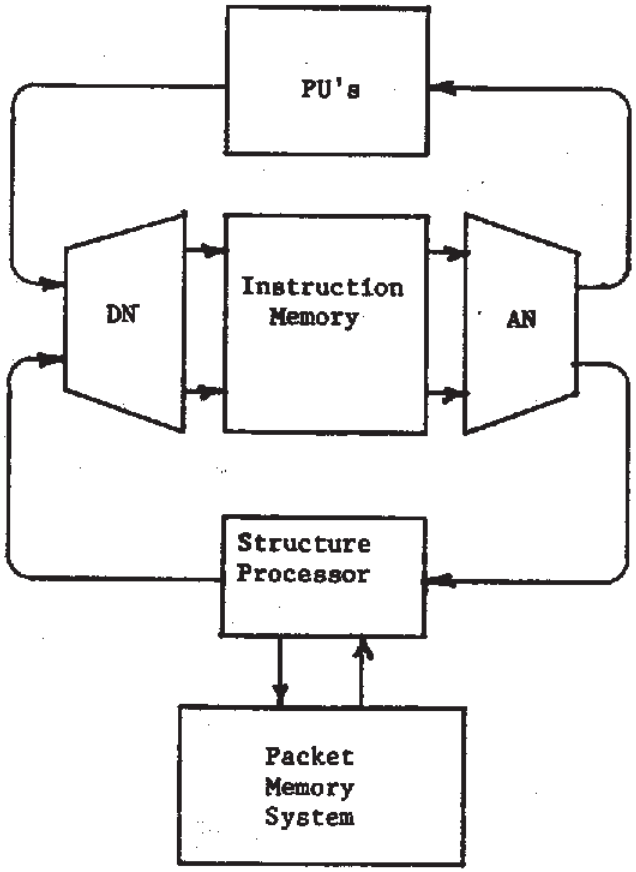

Figure 2. Form 2 Data Flow Processor

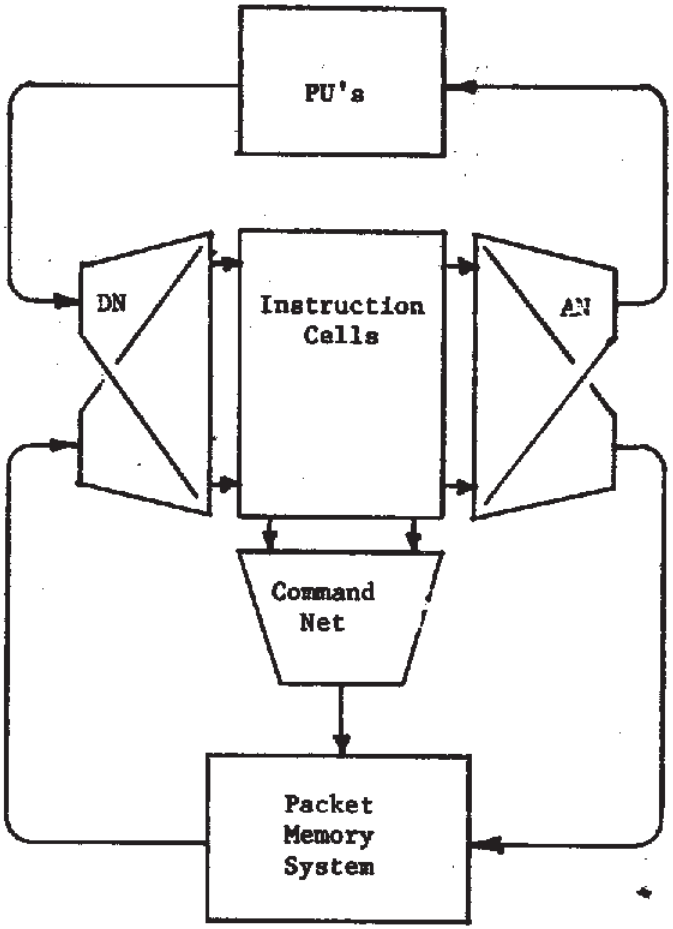

Figure 3. Form 3 Data Flow Processor