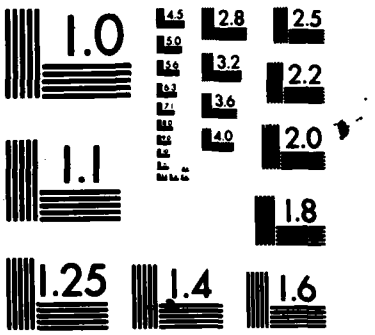
AD-A171 246 EFFICIENT EXECUTION OF FUNCTIONAL LANGUAGE PROGRAMS: 1/1
ALGORITHM DESIGN AND..(U) NORTH CAROLINA UNIV AT CHAPEL
HILL DEPT OF COMPUTER SCIENCE D F STANAT ET AL. JUL 86
UNCLASSIFIED ARO-20785.6-EL DAAG29-83-K-0090 F/G 9/2 NL



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(2)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ARO 20785.6-EL	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): "Efficient Execution of Functional Language Programs: Algorithm Design and Program Optimization."	5. TYPE OF REPORT & PERIOD COVERED: Final Report, 6 Jun 83 - 19 Mar 86 / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Donald F. Stanat, Gyula A. Mago'	8. CONTRACT OR GRANT NUMBER(s): DAAG29-83-K-0090
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science, University of North Carolina at Chapel Hill, New West Hall, Chapel Hill, North Carolina 27514	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office, Post Office Box 12211, Research Triangle Park, NC 27709	12. REPORT DATE: July, 1986 / 13. NUMBER OF PAGES: 8 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

AD-A171 246

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DTIC ELECTE AUG 27 1986 S D B

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Algorithm Design
Program Optimization
Functional Language Architecture
Machine Design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The most significant results of the work in efficient execution of functional languages and program optimization are in the areas of improving the design of the FFP machine and facilitating the use of a variety of problem-solving paradigms. Investigations also led to the preliminary design of a special-purpose architecture for tracking objects in 2 or 3-space.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Efficient Execution of Functional Language Programs: Algorithm Design and Program Optimization

Final Report

Donald F. Stanat
Gyula A. Magó

July, 1986

U. S. Army Research Office

DAAG29-83-K-0090

Department of Computer Science
University of North Carolina at Chapel Hill

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

I. Goal of Project

The goal of this project was to investigate algorithm design and program optimization for the functional language architecture. Because the architecture has not yet been realized in hardware, these studies were necessarily either analytical or based on simulations.

The problem was attacked from a variety of directions. Most fundamental was that of machine design; since the beginning of the grant period, we have come to understand the implications of the design better, and as a consequence, describe it better and propose better implementations. Additionally, we have refined and extended the design, providing answers to a number of questions that had not been previously addressed. Additionally, we worked on the question of making the machine an appropriate one for solving a broader class of problems than had previously been considered. These investigations were all done in conjunction with our considerations of particular applications that were chosen on the basis of their widespread use and their exhibiting characteristics of fundamental importance.

II. Research Results

The most significant results of our work in efficient execution of functional languages and program optimization are in the areas of improving the design of the FFP machine and facilitating the use of a variety of problem-solving paradigms. Our investigations also lead to the preliminary design of a special-purpose architecture for tracking objects in 2 or 3-space. Bracketed integers in the following refer to the publications list that accompanies this report.

A. Efficient Execution

The efforts to improve the design of the FFP machine were directed toward the crucial issues of increasing parallelism and speeding up communication. Our accomplishments fall in the following areas:

1. Virtual memory. The original FFP machine design included only a main memory. We investigated of a number of possible schemes for providing a virtual memory. These schemes range from a complex one in which the FFP machine can be made to look like a data flow machine, to a very simple scheme in which the machine is considered to operate only on a local part of an unbounded memory; i.e., the machine evaluates expressions that are located in a 'sliding window' on the (unbounded) virtual memory. The last scheme has been chosen for implementation in the first prototype.

   These investigations were reported in a conference paper by Frank, Siddall and Stanat [1].

2. Program representation. The original FFP machine design stored at most one atomic or syntactic symbol in each cell of the fine-grained multiprocessor. We investigated the improvement of processor utilization through the use of other representations. It was found that storing more than one atomic symbol in a single processor is unwise because the increased processing required in some cases results in too great a disparity among the processing required by the different cells of the machine. However, it was also found that storing one or more syntactic symbols in a cell, whether or not an atom is present, could be handled in a relatively straightforward way without any substantial effect on the speed of the cell. Thus it appears most attractive to store at most one atom in each cell, together with adjacent associated syntactic symbols.

   Preliminary results of this work were reported in a paper by Middleton [5] as well as a paper by Magó and Middleton [3]; the complete report will be contained in Middleton's doctoral dissertation, to be completed in the next few months.

3. Routing algorithms. Because the FFP machine is a small-grain multiprocessor, communication plays an important role in program execution. An investigation of routing algorithms within the machine has been conducted. Although these results were obtained specifically for the FFP machine, they may have an impact on other tree multiprocessing architectures as well.

This investigation is essentially complete and in the process of being written up. Results will be reported in the doctoral dissertation of Anne Presnell, which we expect to be completed in the next six months.

B. Developing New Problem-Solving Paradigms

Part of our effort was directed toward providing a broader basis for solving problems on the FFP machine by providing alternative problem-solving paradigms.

1. Language Extensions

   i. The FFP machine is a language-based architecture, designed to execute directly the FFP languages proposed by Backus. Because these languages as defined by Backus admit only finite data structures, they are not suitable for an important class of programs such as operating systems. We have laid a theoretical basis for extending the domain of application for these languages by extending FFP to include infinite data structures.

   A preliminary report of these results was contained in a conference presentation by Teresa Thomas and Donald F. Stanat [10]; additional results will be described in Thomas's doctoral dissertation, expected to be completed in the next three months.

   ii. Other language extensions developed specifically for the FFP machine were also considered; the goal was to investigate the what could be accomplished if the programmer had machine-specific instructions that would affect how programs would be executed.

   A variety of mechanisms were investigated and reported in a paper by David Middleton and Bruce Smith [6].

2. An important question concerning the FFP machine is the following: how specific is the architecture to these languages? (We are only interested in this question for non-von Neumann languages, which are properly viewed as architecture-based languages.) In particular, will it be more effective to translate programs in other languages to FFP, or can the FFP machine be modified so that it can directly execute other high-level languages? Presumably the second approach would be more attractive if it is feasible, since program translation may obscure some problem characteristics such as opportunities for parallelism. Hence our initial efforts have not addressed translation issues, but instead have concentrated on what kinds of modifications of the FFP machine would allow the direct execution of other languages.

   We have conducted preliminary investigations into the execution of logic languages and Scheme, a modern version of LISP. Results of the investigation are encouraging but incomplete. Early results for logic languages were described in a conference presentation by Bruce Smith [7]; additional results will be described in his doctoral dissertation, which is expected to be completed within the next year. Results for Scheme will be described in the doctoral dissertation of Kent Dybvig, which should be complete within the next six months.

C. Applications

Evaluation of any machine must ultimately be based on its performance in particular applications. Investigation of how the FFP machine could be used to solve applications sometimes provides insight into possible machine improvements as well as alternative approaches to the application. The application of simultaneously tracking a collection of objects in two or three space was considered as a possible use of a large multiprocessor such as the FFP machine. Our investigation of this problem led to a description of a special purpose architecture for that problem. At this stage it is not clear whether the problem would be handled significantly better by a special purpose architecture (as compared to the FFP machine); additional work may answer this question.

The tracking machine architecture will be described in a technical report by Frank and Stanat [2]; this should be completed in the next few months and will be submitted for publication.

Publications, Conference Presentations and Technical Reports

[1] Frank Geoffrey A., William E. Siddall and Donald F. Stanat: "Virtual Memory Schemes for an FFP Machine" *Proceedings of the International Workshop on High-Level Computer Architecture '84* (Los Angeles, California, May 23-25, 1984).

[2] Frank, Geoffrey A., and Donald F. Stanat: "A Tracking Machine" Technical Report Number TR86-012, Department of Computer Science, University of North Carolina at Chapel Hill.

[3] Magó, G.A.: and D. Middleton: The FFP Machine—A Progress Report. *International Workshop on High-Level Computer Architecture 84* (Los Angeles, California, May 23-25, 1984).

[4] Magó, Gyula A., and Donald F. Stanat: "The FFP Machine" A chapter in *Advanced Microprocessors and High-Level Language Computer Architecture*, edited by V. Milutinovic, to be published by Computer Science Press.

[5] Middleton, David: "Alternate program representation for the FFP Machine" *Proceedings of the Eleventh EUROMICRO Symposium on Microprocessing and Microprogramming*, Brussels, September 3-6, 1985. pp 85-93.

[6] Middleton, David and Bruce T. Smith: "FFP Machine Support for Language Extensions" *Proceedings of the Nineteenth Hawaii International Conference on System Sciences*, Honolulu Jan 1986. Architecture volume, pp 59-65.

[7] Smith, Bruce T. "Logic Programming on An FFP Machine" *Proceedings of the International Symposium on Logic Programming* (Febr. 6-9, 1984, Atlantic City, New Jersey).

[8] Smith, Bruce T., and Kent Dybvig: "A Semantic Editor" *ACM Sigplan '85 Symposium on Programming Languages and Programming Enviroments*, Seattle, Washington, June 25-28, 1985.

[9] Thomas, T.A., A semantic domain with infinite objects. Technical Report 83-009, Dept. of Computer Science, University of North Carolina, 1983.

[10] Thomas, Teresa A., and Donald F. Stanat: "An FP Domain with Infinite Objects" *Conference on Mathematical Foundations of Programming Semantics at Kansas State University*, Manhattan, Kansas, April 11-12, 1985. To be published in Lecture Notes in Computer Science by Springer-Verlag.

Participating Scientific Personnel and Advanced Degrees Earned (or nearly completed).

Principle Investigators (UNC-Chapel Hill)

Gyula A. Magó
Donald F. Stanat

Research Associates (UNC-Chapel Hill)

Leigh Pittman

Research Triangle Institute (Subcontract)

Dr. Geoffrey A. Frank

Graduate Students (UNC-Chapel Hill)

Lakshmi Dasari
Kent Dybvig (Ph. D. Expected 1986)
William Gibson
David Middleton (Ph. D. Expected 1986)
Amos Omondi
William Partain
Anne Presnell (Ph. D. Expected 1986)
Bruce T. Smith (Ph. D. Expected 1986)
Teresa A Thomas (Ph. D. Expected 1986)
William A. Warren (Master of Science Degree)

END

DTIC

9-86