MIT/LCS/TR-216

ANALYSIS OF THE SIMPLE CODE FOR DATAFLOW COMPUTATION

John M. Myers

This blank page was inserted to preserve pagination.

Analysis of the SIMPLE code for dataflow computation

by

John M. Myers, Consultant

May, 1979

This research was supported by Lawrence Livermore Laboratory under contract 8545403.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LABORATORY FOR COMPUTER SCIENCE

Cambridge Massachusetts 02139

This empty page was substituted for a blank page in the original document.

Analysis of the SIMPLE code for dataflow computation

John M. Myers

ABSTRACT

We analyze a problem in hydrodynamics from the standpoint of computation on a dataflow computer that is not yet fully specified, with the objectives of helping to further specify the computer and helping to develop VAL as its source language. Lawrence Livermore Laboratory supplied the algorithm for hydrodynamics, including heat flow, as a 1749-line FORTRAN code called SIMPLE.

The algorithm viewed as 'abstract' (i.e. independent of physical arrangements in space and time for its realization) is shown to imply spatial and temporal structure that must appear in any and all implementations. Both for hardware design and program compilation it is useful to map this structure to grosser levels of description, with the grosser levels reflecting modularity of computational resources conjoined with modularity of the algorithm. Following Holt (1979) we use role diagrams to display spatio-temporal structure at different descriptive levels, so as to guide translation into VAL as well as the analysis of the time to compute.

Inter-resource communication essential to the problem is displayed, and various issues of machine design are defined. Using VAL with one set of extensions, we express the algorithm so that in principle it can be compiled for execution by a dataflow computer. Input-output functions beyond those implied by the SIMPLE code are discussed. A second set of extensions to VAL is advocated to express the conjunction of problem and resource modularity, so as to guide compilation. The dependence of time to compute on the number of processing units is shown for various aspects of the problem.

KEYWORDS: DATAFLOW, ALGORITHM ANALYSIS, PARALLEL COMPUTATION,

COMPUTATIONAL HYDRODYNAMICS, ROLE DIAGRAM.

This empty page was substituted for a blank page in the original document.

Analysis of the SIMPLE code for dataflow computation

CONTENTS

		Page
1.	Introduction: hydrodynamics meets a dataflow computer.	1
2.	The hydrodynamic fields.	5
	Figure 1: Nodes and zones.	6
3.	Communications and the speed and configuration of a dataflow computer.	9
	3.1. General issues.	9
	3.2. Connectivity in the face of resource sharing.	11
	Figure 2: Connectivity of simplified hydrodynamics in one space dimension with one processor assigned to each nodal and zonal calculation.	12
	Figure 3: Constraints on concurrency imposed by sharing of processors.	13
	Figure 4: Grosser view highlighting connectivity between processors.	14
	Figure 5: Alternate view using the notation of buffered communication.	15
	3.3. Fitting the computation to the minds of the analysts: input and output.	18
4.	Modeling the time to compute.	20
	4.1. Choosing an appropriate form of model.	20
	4.2. The need for speed.	22
	4.3. The computational cycle.	23
	Figure 6: Concurrency and connectivity in different phases of the cycle.	25, 26
	4.4. Dependence of time to compute on number of zones and number of processors.	27
	4.4.1. Case definitions.	27
	Case 1: connectivity restricted to nearest neighbor plus "tree".	27

CONTENTS (continued)

Page

Case 2: "general-purpose" communication. 28

4.4.2. Results. 29

Table 1: Form of dependence of time to compute a cycle on number of zones and number of processors. 31

4.5. Input, output, and control over the extraction of features. 32

5. Translation of SIMPLE from FORTRAN into VAL. 34

5.1. The balancing of objectives. 34

5.2. Samples of VAL code. 37

5.2.1. Overall form of the VAL translation of the SIMPLE code. 37

5.2.2. JES_VAL . 38

5.2.3. SIMPLE_VAL. 44

5.2.4. Discussion of functions internal to SIMPLE_VAL. 52

INITIALIZE, EDIT, BOUNDARY_PROJECT, VELOCITY, POSITION, HWORK, ZONE_GEOM. 52

ENERGY_HYDRO. 52

HYDRO_TOTAL. 56

ENERGY_HEAT. 56

HEAT_TOTAL, TIME_STEP, PHYS_REPORT, CYCLE_REPORT. 59

MODIFY. 60

6. Conclusions and possible next steps. 61

6.1. Speed, input-output, and expression of the abstract algorithm. 61

6.2. Implications of the spatio-temporal structure of the algorithm. 62

6.3. The balance between programming ease and efficient use of hardware. 63

CONTENTS (continued)

		Page
6.4.	Extending VAL to support resource allocation.	66
References		68
Appendix A:	Interpreting role diagrams.	69
Appendix B:	Notes on fitting the SIMPLE code into role diagrams and VAL modules.	81
Appendix C:	The SIMPLE code in FORTRAN.	89

This empty page was substituted for a blank page in the original document.

Analysis of the SIMPLE Code for Dataflow Computation

1. Introduction: Hydrodynamics Meets a Dataflow Computer

The equations of physics are prescriptions for calculating; from some presumed starting conditions, they generate a "future". The calculation of this "future" involves many events, each of which "consumes" items -- values of variables -- and "produces" other items. Because an item cannot be consumed before it is produced, these events are subject to constraints of sequencing. These constraints impose a pattern on the calculation.

Although the equations of physics constrain the calculation, they do not fully determine it. The pattern is partly determined also by the method of solution employed and by the structure of the computer. Thus the same (partial differential) equations can result in different patterns of calculation, according to the method of solution and the arrangement of computational resources. For this reason the pattern of computation for a given type of problem, say hydrodynamics, evolves as methods and computational resources evolve. Pattern, method, and resources are coupled in their evolution, with each selected in part to support and to draw on the others.

Over most of history the computer (human or machine) had only a sequential processing capacity, so that computation was necessarily performed one step after another. Thus methods which emphasize concurrency were not called for, and as a result are today relatively unexplored and undeveloped. Not only computers, but also numerical methods have evolved in a context that is weighted toward the sequential, and away from the concurrent.

Via such means as dataflow architecture (see Dennis, 1978), an increase in speed can be brought about by an organization of computational resources that allows concurrency of many events. This report is concerned with fitting -- or refitting -- a pattern that evolved in a sequential context

onto a dataflow computer. The report is based on a case study of an example program written in FORTRAN for a sequential machine for the solution of a problem of hydrodynamics, including heat flow. This program was prepared by Lawrence Livermore Laboratory, and is named SIMPLE. The initially presented questions were:

1.1) What is involved in translating the SIMPLE program from FORTRAN (suitable for a sequential computer) into a dataflow language (the VAL language in particular); and

1.2) Compared to a sequential computer, what speed advantage can be expected from a dataflow computer in the execution of the SIMPLE program?

To realize the potential advantage of a dataflow computer, its program must be free of unnecessary sequencing constraints. Sequencing constraints come from many sources, and their necessity depends on ones point of view. Primarily we report on the narrow view that sees sequencing constraints as imposed by the data dependencies of the FORTRAN program. In this view the "translation" per item 1.1 entails the removal of sequencing