JOINT ROYAL MELBOURNE INSTITUTE OF TECHNOLOGY AND
COMMONWEALTH SCIENTIFIC AND INDUSTRIAL RESEARCH ORGANISATION
PARALLEL SYSTEMS ARCHITECTURE PROJECT

The RMIT/CSIRO Parallel Systems Architecture Project: Taking Australian Industry into the 1990's and Beyond

TR 112 076 R

D. Abramson †
G.K. Egan ‡

† Division of Information Technology
C.S.I.R.O.
‡ c/o Department of Communication and Electrical Engineering
Royal Melbourne Institute of Technology
P.O. Box 2476V
Melbourne 3001
Australia.

Version 1.0 September 1988

ABSTRACT:

This report considers the potential of parallel computer systems in Australia. It begins by reviewing current technology and illustrating the power of parallel computers. It argues that industry which is developing any compute intensive application should consider the use of parallel computer systems to speed execution of the application. Appropriate software engineering principles should enable the development of products which make effective use of current uniprocessors and multiprocessors, and allow a software lifetime which extends well into the next generation multiprocessors. The report describes the activities of the RMIT/CSIRO Parallel Systems Architecture Project, and gives a profile of activities which will assist in maximising the exploitation of parallel computer systems by Australian Industry.

1. INTRODUCTION

Over the past 30 years the speeds of computer systems have increased dramatically. Even small personal computers can now execute many millions of instructions per second, and supercomputers can execute up to billions of instructions per second. Unfortunately, this growth in speed cannot continue unless a dramatic change is made in the physics of the semi conductors used in the computer systems.

A widely accepted model for increasing the speed of computer systems is to combine many processors into a multiprocessor. This technique relies of the problem being solved being decomposed into smaller subtasks and being mapped onto the various processors. When a multiprocessor is operating at full efficiency, the speed of the total system can be multiplied by the number of processors in the machine.

Given this model, even if the speed of uniprocessors continues to increase, a multiprocessor system by incorporating the same circuit technologies, will always have the potential to solve a problem many times faster. It is this attribute which makes multiprocessor program implementation essential for any computation intensive software being developed; a multiprocessor implementation has the potential to outperform a uniprocessor program. **More importantly because the growth in cost of faster circuit technologies is greater than the increase in performance, it is possible to build multiprocessors which are equal in performance to uniprocessors whilst using cheaper, and possibly more reliable, circuit technologies.** Appropriate use of current software engineering tools allows efficient serial programs to be developed which double as parallel programs given appropriate parallel hardware. Thus, the cost of implementing a parallel program should only be visible in the cost of designing and building the program, which is a small incremental cost over designing a serial program.

Many different multiprocessor architectures have been proposed. There are a number of different categories, but the most usual are Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD).

In SIMD machines, each processor executes the same instruction, but on a different piece of data. This model is very good for performing uniform processing on a large data set. The type of algorithms which suit SIMD machines are "data parallel", i.e. the parallelism is found in the basic data. The most notable example of a SIMD machine is the Connection Machine [1], but the original SIMD concept dates back to Illiac 4 [2]. Examples of problems which contain data parallelism are stress analysis computations, geological analysis, weather modelling and tomography processing. Some less obvious examples include free text processing and database searching [3].

MIMD machines cover a number of different architectures. In this class of multiprocessor each processor executes a different instruction on a different data set. They are suitable for more asynchronous types of algorithms where processes may perform a task and only communicate where necessary. The two main sub categories of machines are *shared memory* and *message passing*. In shared memory systems processes execute asynchronously, and communicate via regions of memory which more than one process can access. Shared memory machines are typically tightly coupled, because each processor requires rapid access to the shared memory. The most common communication mechanism is a shared bus, however, some systems use other interconnection networks [4]. The types of algorithms which suit shared memory machines are those which require access to large shared data structures, and when it is difficult to partition these structures. Examples of commercially available shared memory machines are the Encore Multimax [5], the Sequent Symmetry [6] and the Alliant Fx8 [7].

Message passing machines, on the other hand, are typically loosely coupled. Processes usually execute in isolation of each other, and communicate by passing coded messages. Messages can be sent over a communication structure which has a lower bandwidth than the memory interface of the shared memory machines, and thus the processors may be physically distributed if necessary. The types of algorithms which best suit message passing machines are those in which the information can be partitioned and in which there is little inter-process communication. They also suit *flow through* or *pipelined* algorithms, such as those found in digital signal processing and in transaction processing. In this way, data may enter a stage of a computation, be processed and then sent to the next stage of the computation. Examples of commercially available message passing machines are those based on the Inmos Transputer [8], such as the Meiko computing surface [9].

These classes of machine will be available commercially for some time although parallel computing in general should be regarded as in transition. Changes will occur in the internal structure of systems to accommodate changes in technology as well as the desire for larger systems; shakeouts are to be expected. At the present time, programming such systems is difficult and requires special training and experience. Clear standards have yet to be adopted by manufacturers; as in the past there is currently little incentive for manufacturers to enhance portability. New parallel programming languages are being developed to ease the programming task although it is not yet clear which of a number of competing approaches will prevail.

2. OPORTUNITIES FOR THE EXPLOITATION OF PARALLEL SYSTEMS

Because of their peculiar cost performance advantages parallel computer systems are capturing an increasing segment of the mid and high performance computing market. In recognition of this overseas software houses and manufacturers interested in imbedded computer applications are moving to:

1) exploit the performance growth path for compute intensive software products and,

2) the provision of cost effective computer based products, particularly for communication and other time critical application areas.

A software package requires many man-years of development. For example, a typical engineering package may require 10-15 man years of development before it is released. Any company which is developing a new package in 1988, will see many changes in the speeds and architectures of computer systems during the lifetime of their product. To date, good software engineering practices, coupled with good high level languages and portable operating systems, have been sufficient to allow a software product to survive hardware changes. However, efficient use of parallel machines requires careful design of the algorithms used in the implementation. **It is very unlikely that a software product designed in 1988, without careful design of the internal algorithm, will be able to make effective use of current parallel systems, let alone survive the next generation of machines.** Some example of computation intensive programs are CAD/CAM programs, complex database search operations, operations research and optimisation programs, artificial intelligence and heuristic search code.

Microprocessor devices are now cheap enough that it is feasible to include more than one processor in a product. Many products, such as communication controllers, graphics processors, real time control systems, can effectively use more than one processor. In order to design multiple process programs for execution in such systems, the implementors must have experience with complex parallel programming concepts much of which is resident in research groups. **It is imperative that this experience be transferred to industry.**

There is very little experience in using parallel computers in Australian industry. It is imperative that the level of such experience is increased, plus an awareness of how parallel computers can be effective. There are also very few parallel systems available in Australia. The number of such systems, and well as the range of systems must be increased. It is also important that industry is not tied too heavily to the current generation or brands of parallel machines. Portability in the parallel implementation of algorithms is of utmost importance just as it is in the implementation of sequential systems. **For these reasons, the transfer of such technology should not be left to the computer manufacturers, who have a vested interest in locking users into their products.**

Both of the above uses will allow Australia to make the best use of current and future technology. The first category will allow software products to have a longer lifetime and usefulness. The second category will allow very competitive products to be developed and exported.

3. Parallel Systems Architecture Project 1986 - 1989

The RMIT/CSIRO Parallel Systems Architecture Project was established in 1986 as a joint research project between CSIRO's Division of Information Technology, and the Royal Melbourne Institute of Technology. The project charter was to investigate parallel algorithms, methodologies, languages and architectures. It was formed from an already existing project within RMIT which was investigating the design of multiprocessors based on a data-driven or *dataflow* model of computation [10,11].

Dataflow machines are MIMD machines which solve many of the problems associated with the current generation of MIMD multiprocessors in that they are "incremental computers". Incremental computers are those which, by simply adding more processors, permit linear growth in program speedup in way that is entirely transparent to the programmer. Machines incorporating dataflow principles are likely to form a significant fraction of the next generation parallel machines. The interested reader is referred to [12] for a full discussion of dataflow architectures.

The project has an ambitious programme plan for the three first years of funding. The principal elements of the programme are to develop:

1) an emulation, simulation and parallel emulation facility for the dataflow computational model under study;

2) compilers for various parallel programming languages, as well as conventional imperative languages;

3) a prototype processing element for a high speed dataflow multiprocessor;

4) benchmark parallel applications on the dataflow machine.

A multiprocessor has been constructed for executing the parallel emulator. This multiprocessor has fifteen 68020/68881 based processors connected via a packet-switched multistage interconnection network, and is hosted by a Sun 3/260 workstation. As well as running the dataflow emulator, this system can execute codes which use message passing for inter-process communication.

There are two parallel functional programming languages being implemented for the machine, SISAL as used in the Lawrence Livermore National Laboratories, and ID as used at the Massachusetts Institute of Technology. These languages simplify the development of parallel programs by providing a deterministic execution environment coupled with parallel programming constructs. An attribute of SISAL is that it can be compiled for more than one type of parallel computer. At present, SISAL code can be run on many uniprocessors, as well as parallel processors including those produced by Sequent, Encore, Alliant and Cray. At present, if various options and run time parameters are set appropriately, then the performance of SISAL code compares favorably with current sequential languages on uniprocessors. With dataflow architectures little or no tuning is necessary.

As well as functional languages an implementation of Guarded Horn Clauses, a dialect of Prolog, is well advanced. Work is also progressing on a conventional imperative language implementation. The latter effort will permit porting of existing codes to the dataflow system, however it is unlikely that the full power of the machine can be exploited using imperative languages.

A new processing-element is being designed to implement the dataflow instruction set. This element will form the heart of a high speed dataflow multiprocessor. The details of the multiprocessor are outside the scope of this paper, but a more complete discussion can be found in [13]. A 16 processing element should have a sustained scalar execution rate of 80 MFLOPS, and a sustained vector rate of 160 MFLOPS.

The dataflow architecture has been evaluated on a number of applications. These include laser rangefinder based object recognition, manipulator control, digital logic simulation [14] and a timetable scheduling [15]. Codes for many applications, written in SISAL, are available from the Lawrence Livermore National Laboratories [16].

The Project is important for a number of reasons. First through its own independant research it has established important contacts with major research groups overseas with the prospect of regular exchange of staff. As these groups provide the driving force for next generation parallel architectures and language products, the Project is more likely to be aware of new directions and, most importantly language and architectural deadends, than through reliance on manufacturer originated product information alone. Second, it has allowed a group of researchers to gain significant experience with current and next generation languages and, the design and construction of parallel machines. Finally the results of the Project's own research, both software and hardware, offers the prospect of commercialisation in the same time frame as next generation systems overseas. It is essential that support for these elements continue if Australia is to capitalise on the commercial prospects offered by current and future parallel machines.

4. Parallel Systems Architecture Project 1989 -

In 1989 the Project will continue its research into the next generation computing machines, but will also broaden its scope by offering assistance to Australian industry in its efforts to capitalise on current generation parallel machines.

The existing work will continue to investigate issues in parallel programming on dataflow machines. It will build up a substantial dataflow multiprocessor as a test bed, and continue development of parallel programming languages. Apart from the intrinsic value of this work, the dataflow machine will be a valuable computing resource which can be used to evaluate and execute parallel algorithms.

In broadening its scope the project transfer parallel computing technology to industry. This activity will take three forms.

1) Running parallel programming workshops and educational activities.

2) Providing a parallel programming tool set for use on current multiprocessors.

3) Providing a centre staffed with experienced personnel and equipped with a range of parallel machines.

4.1 Educational Activities

One of the major problems facing Australian industry is the lack of sufficient staff with the requisite knowledge and expertise in parallel programming to capitalise on the opportunites offered. This problem can be address in two complementary ways. First, as many Universities and colleges buy parallel computers, they will will add sections on parallel programming to their computer science and engineering courses. This change is already occurring. However, the graduates from such courses will not, in general, be a position to contribute in the near term, because there is presently little wide spread use of parallel computers within Australian industry.

The second method is by extension courses for the computer industry. These courses need to be aimed at two different groups; middle to upper management, and technical personnel. Technical personnel require specific architectural and language material while management requires strategic information on directions and possible pitfalls.

The Project is in an ideal position to contribute to both industry and RMIT undergraduate programmes in engineering and computer science. Because it is joint programme with CSIRO based at RMIT, it is an ideal vehicle to act as an interface to the computing industry; RMIT has a sound reputation for continuing education programmes.

4.2 Tool Sets and Language Systems

As already inidicated, there is no clear standard programming languages or methods for current generation parallel systems. This aspect of parallel machines is illustrated in [17] by many different programming environments and languages. This is a highly undesirable situation for a software industry which is attempting to maximise its spread of target machines. The Project can offer two solutions to this problem.

First, through the provision of parallel programming tool kits. These kits isolate application codes from underlying architectures, manufacturer specific language and operating system implementations, while still permitting the exploitation of application parallelism. The feasibility of such kits has been demonstrated by the Argonne Laboratories in the United States [18].

Second, significant research and development has been conducted on the next generation parallel programming languages. The language SISAL, which is still under alpha release, offers many advantages in the design and development of parallel algorithms. One of its major benefits is that is can generate code for a number of parallel machines. Continued research on SISAL at Lawrence Livermore Laboratories should yield a programming system which generates efficient machine independent code. It is reasonable to expect that the SISAL development environment would be used at least in the design and testing of various algorithms even if it is not used for generating production code in the immediate future.

4.3 Access to Representative Parallel Systems

The Project has access to a number of parallel systems although this base needs to be extended. These machines range in architecture from message passing to shared memory to dataflow processors.

The following processors are available for the Parallel Systems Architecture Project.

1) A Dataflow multiprocessor

2) A message passing multiprocessor

3) A large shared memory multiprocessor

4) A Transputer development system

5) A shared memory attached multiprocessor (available 1989)

6) A shared memory/message passing DSP engine (available 1989)

Access to a range of machines allows the exploration of different programming techniques, languages and algorithms.

5. ASSOCIATED GROUPS

There are a number of associated research projects at RMIT investigating different aspects of parallel processing. Each of these groups is concentrating on a different aspect of parallel processing. Within the Department of Communication and Electrical Engineering one group is building a hybrid shared memory - message passing machine for high speed digital signal processing; the primary application for this machine will be in the production of computer generated imagery for flight simulation [19,20]. Another group is working with Transputer arrays, with specific application to digital signal processing [21,22]. Substantial, and demonstrated, design and construction capability exists within the Project for imbedded processor applications. Within the Department of Computer Science, substantial experience exists in the programming of shared memory multiprocessors. An Encore Multimax is being used for parallel program simulation as well as parallel database research.

6. CONCLUSION

The RMIT/CSIRO Parallel Systems Architecture Project, established in 1986 as a joint research project between CSIRO's Division of Information Technology and the Royal Melbourne Institute of Technology has, through its own independant research, built up a considerable pool of skilled staff and established important contacts with major research groups overseas. Through its continuing contact with these groups, groups which provide the driving force for next generation parallel architectures and language products, the Project is aware of new language and architecture directions. Its own research offers the prospect of commercialisation in the same time frame as next generation parallel systems overseas.

Over the next 5 to 10 years most computer manufacturers will add multiprocessors to their equipment lines. Unless the software industry pays careful attention to the structure and internal design of future software systems, it wil not be possible to realise the power of such computers. Drawing on its links with overseas government and industry supported groups, the skill base it has built up under its own research programmes its location at RMIT the Project wishes to offer its expertise and knowledge to assist Industry's efforts to capitalise on the opportunities open to Australia.

References

[1] Hillis, D. "The Connection Machine", MIT PRess, Cambridge, Massachusetts, 1985.

[2] Bouknight, W., Denenberg, S., McIntyre, D., Randall, J. Sameh, A. and Slotnick, D. "The Illiac IV System", Chapter 20, Computer Structures: Primciples and Examples, Siewiorek, D., Bell, G., and Newell, A. McGraw Hill, 1984.

[3] Stanfill, A.C., Kahle B. "Parallel free-text search on the Connection Machine system", Communications of the ACM, Vol 29, No 12, December 1986, pp 1229-1239.

[4] BBN Advanced Computers, "Inside the Butterfly Plus", Cambridge, MA, October 1987.

[5] Schanin, D. "The design and development of a very high speed system bus - the Encore Multimax Nanobu", Proceedings of the 1986 Fall Joint Computer Conference, Dallas, Texas, November 1986, Edited by Harold S. Stone, pp 410-418.

[6] Crawford, F., Crawford, J. "Parallel programming facilities on the Sequent series of machines", Australian Unix system User Group Newsletter, Vol 9, No 2, April 1988, pp 17-28.

[7] Perron, A. R. and Mundie, A.C. "The architecture of the Alliant FX/8 computer", Digest of Papers of the Thirtyfirst IEEE Computer Society International Conference (CompCon 86 Spring): Intellectual Leverage, Edited by Alan G. Bell, San Francisco, California, February 1986, pp 390-393.

[8] Inmos, Transputer Reference Manual, Prentice Hall, 1988.

[9] Barton, A.E. " Data concurrency on the meiko computing surface.", Proceedings of the International Conference on Parallel Processing for Computer Vision and Display, Leeds, United Kingdom, January 1988.

[10] Abramson, D. and Egan, G., " An Overview of the RMIT/CSIRO Parallel Systems

Architecture Project", Australian Computer Journal, Vol 20, No 3, August 1988.

[11] Abramson, D and Egan, G, "The RMIT Dataflow Computer: A Hybrid Architecture", to appear in the British Computer Journal.

[12] IEEE. "Dataflow and Reduction Architectures", IEEE Computer Society, Order number 759, 1987

[13] Abramson, D. and Egan, G., "Design considerations for a High Speed Dataflow Multiprocessor", TR 112-073, RMIT Department of Communication Engineering, 1988.

[14] Abramson, D. A. "Using a dataflow computer for functional logic simulation", Third International Conference on Supercomputing, Boston, Massachusetts, May 15-20, 1988.

[15] Abramson, D. A. "Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms", submitted for publication.

[16] Feo, J., "The Livermore Loops in SISAL", Lawrence Livermore National Laboratories Technical report UCID-21159, 1987.

[17] Babb, R. "Programming Parallel Processors", Ed. Robert G. Babb II, Addison Wesley, 1988.

[18] Dongarra, J.L. and Sorensen, D. "Algorithm Design for High-Performance Computers", Proceedings of 1986 IBM Europe Institute: Seminar on Parallel Computing", Oberlech, Austria, August 11-15, 1986, Elsevier Science Publishing Company.

[19] Beckett, P. "A Multi-Processing Architecture for CGI", Joint Microelectronic Research Centre- ARL Contract Report, August, 1985, Royal Melbourne Institute of Technology.

[20] Beckett, P. "An Investigation into Parallel Processing Architectures for CGI", Proc IREECON 85, October 1985.

[21] Gregory, M. and Hulskamp, J.P., "Fast Waveform Vector Quantisation using Dynamic Codebook Systolic Encoders and its OCCAM implementation", Australian Transputer and OCCAM User Group Proceedings (23-24 June 1988),ISBN 0 86444 143 6

[22] Gregory, M. and Hulskamp, J.P., "Dynamic Codebook Systolic Encoders Performing Fast Waveform Vector Quantisation", Submitted to the I.E.E.E. 1989 International Conference on Acoustics, Speech and Signal Processing (ICASSP-89)