SEMINAR ON

PARALLEL COMPUTING ARCHITECTURES

Held at

Telecom Australia Research Laboratories

Melbourne, 24-25 February 1986

Telecom Australia

PREFACE

The idea of a workshop on Data Flow Architectures arose during 1985, due to the support of a number of projects concerned with the development of data flow multiprocessor systems by the Australian Telecommunications and Electronics Research Board (ATERB). After contacting the various groups of researchers it became obvious that the groups knew very little of each other's work. It also became apparent that some of the groups ostensibly working on Data Flow were in fact working on hybrid control flow/data flow architectures and that there were many other groups in Australia working on multiprocessor systems. There was also a group within the Telecom Australia Research Laboratories starting to develop ideas on architectures for high performance packet switches.

The initial preparations for a small workshop to bring together experts in the field of multiprocessor architectures commenced in November 1985. By January 1986, interest had grown considerably both from researchers and staff in Telecom and other Government agencies. It became clear that there was a need for the dissemination of information regarding the work that was being performed by each of the research groups at a general level, as well as a need to provide an opportunity for researchers to meet informally for in-depth discussions of a research nature. It was felt that these needs could best be met by holding both a seminar (for information dissemination) which would be open to all those interested in parallel computing architectures (given the limitations of the venue) and also a workshop which would be limited to about 20 people who would be required to be "practitioners" in the field. Telecom considers the topic of Parallel Computing of major importance and has therefore committed resources to the organisation and sponsorship of both the seminar and the workshop.

The seminar brings together people interested in parallel computing from 14 organisations around Australia for the exchange of information for the mutual benefit of all concerned. It is hoped that the seminar and workshop will provide a forum where Australian research groups will obtain an appreciation of each other's work and the requirements of Telecommunication and other authorities, leading to a deepening of relationships.

In the process of organising the seminar and workshop considerable use has been made of the Australian Computer Science mail Network (ACSNET). It is hoped that this network (and its successors) can be used as a vehicle for communication and co-operation in the future to overcome the rather large geographical separation between the various interested parties. In particular, I look forward to many more conferences and workshops being organised over ACSNET, and in the future the electronic publication of the proceedings.

My thanks are due to numerous people who have co-operated in the organisation of the seminar and workshop. I would like to express my gratitude to the speakers for their enthusiastic support and their willingness to write papers for this proceedings at short notice. Many thanks are due to the Director Research, Mr Harry Wragge, the Assistant Director Switching and Signalling Branch, Mr Peter Gerrand, my Section Head, Robin Court and colleague Paul Kirton for their support and advice, without which the holding of this event would not have been possible. In particular the financial support of Telecom Australia in this regard is especially appreciated.

Finally, I would like to express my deepest thanks to my organising committee: Mrs Evelyn Swenson for providing backup and preparing this proceedings; and Ms Michelle Lambert, Mr Steven Tarpkos, Ms Sandra Van Den Heuvel and Mrs Joy Scambler for their invaluable help with all the time consuming administrative tasks associated with running this seminar and workshop on Parallel Computing Architectures.

Jonathan Billington
Convenor
21/2/86

RMIT DATA-FLOW PROJECT

G.K. Egan, C. Baharis, M. Rawling, A. Young
Department of Communication and Electronic Engineering
Royal Melbourne Institute of Technology

1. HISTORICAL BACKGROUND

The RMIT Data-flow Project had its origins in 1976 as part of doctoral research [Egan (1)] at the Department of Computer Science, University of Manchester, England. The research followed the initial work at the University by Treleaven [Treleaven] and ran in parallel to the more widely publicised work of Gurd and Watson [Gurd]. A pilot four processing-element data-flow machine was commissioned in and ran data-flow graphs in 1978 some time prior to the Manchester Data-flow Machine.

Research has continued on a modest scale at RMIT primarily as a series of simulation studies until recently when an injection of funds and manpower permitted the construction of a significant multi-processor.

2. THE RMIT VARIANT OF DATA-FLOW

The RMIT architecture [Egan (1)] is based on a variant of the data-flow model [Adams]. The characteristics of this data-flow variant which underlies the languages and implemented by the simulator and data-flow multi-processor are outlined in the following points.

1) Tokens are strongly typed and of variable length. They carry not only the data used by the graph during its evaluation but also the node-descriptions which comprise the graph.

2) Node-functions are weakly typed and accept a set of argument token-types; this increases graph generality while reducing the size of the node-function set. Type coercion is performed automatically where 'sensible'.

3) Input-output is accomplished using pre-defined nodes. The names of these nodes are associated with particular input or output devices, which in turn are associated with particular processing-elements. As these nodes already 'exist' within the system, they must be linked into the graph at load or evaluation time; this is done by sending response-destination tokens to the appropriate nodes.

4) Because of the association of input-output devices with particular processing-elements, and the system's probable geographic distribution, graphs are partitioned and the partitions allocated statically to processing-elements.

5) Information describing exceptions occurring during graph evaluation is communicated to the graph using the token-type ?, or don't-know.

6) The system supports shared sub-graphs in sufficient generality to allow multiple recursion. Tokens involved in concurrent invocations of a shared sub-graph are separated by means of a copy number. The copy number is computed and appended only to tokens actually sharing the sub-graph. Less sophisticated processing-elements at the periphery of the system need not support the mechanism.

7) Streams proposed by Weng are supported [Weng].

3. LANGUAGES

Two textual languages and their associated compilers have been developed. The first is a dialect of Lisp called Newspeak developed by Wathanasin [Wathanasin] and the second is DL1 a single assignment language developed by Richardson to support his research on manipulator control [Richardson (2)]. DL1 is currently the major project language.

4. SIMULATION FACILITIES

A simulator/emulator written in Pascal models systems of 128 processing-elements and runs under Unix and CDC NOS. The simulator uses the same communication and function evaluation times as those derived from benchmarks running on the prototype processing-element [Rawling]. In this way more realistic assessment of communication overheads can be obtained than with the simple unit time step estimates used by some groups.

5. DATA-FLOW MULTI-PROCESSOR

A data-flow multi-processor is currently being constructed/ commissioned. The initial configuration is 32 general purpose M68000 microprocessor based processing-elements. The elements emulate the RMIT data-flow architecture while providing sufficient flexibility for the study of alternative data-flow and other architectures.

5.1 Processing Elements

The multi-processor's general purpose processing-elements [Young] are based on the prototype general purpose element developed by Rawling and Zuk [Rawling]. Each processing-element is comprised of one (Figure 5.1.1) or two (Figure 5.1.2) M68000 based units. In the normal configuration, one unit performs the matching/structure accessing task and the second the node-function evaluation task. The alternative single unit configuration allows exploration of non data-flow applications where either the matching or the evaluation task is less significant.

A special purpose processing-element [Chan] for floating-point intensive computation has been developed. Depending on the application a number of these may be included in the processing-element 'mix'. The question however of whether floating-point capacity in the face of communication overheads should be distributed is still under study.

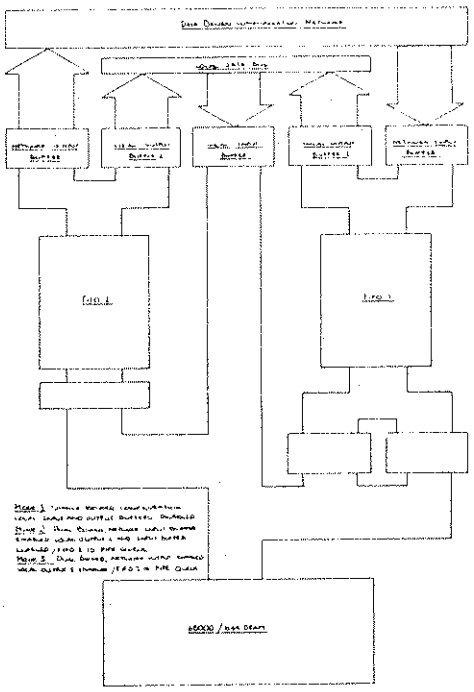

Figure 5.1.1 Single Processor Configuration

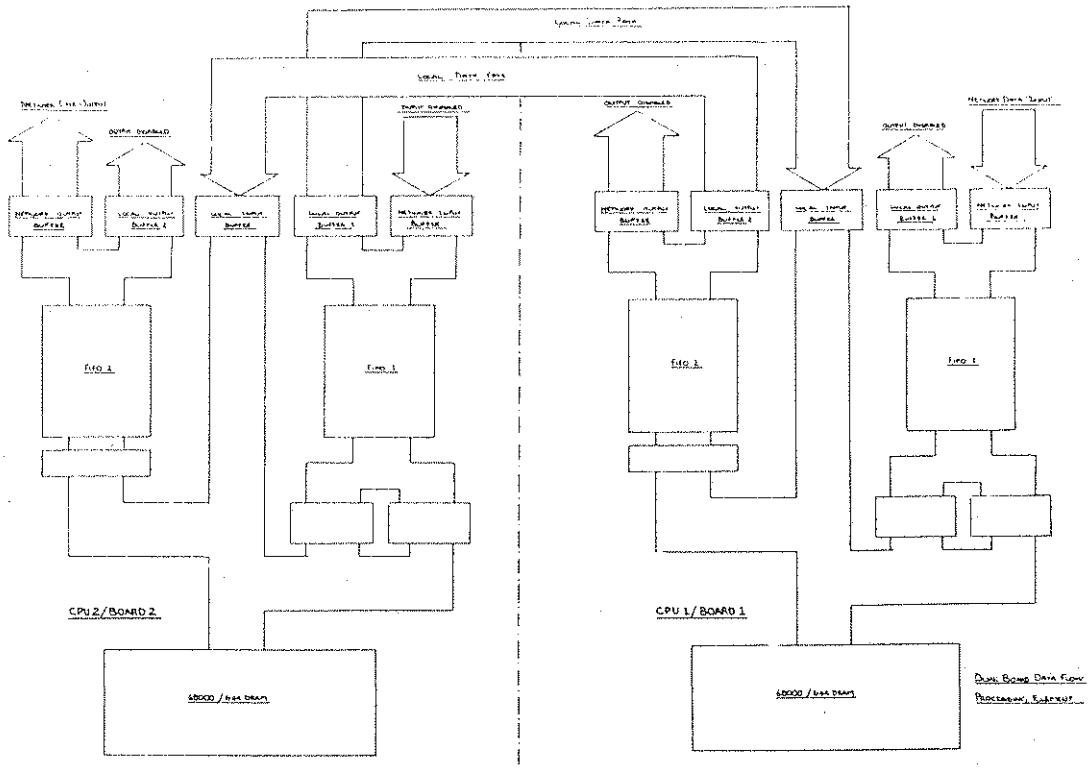

Figure 5.1.2 Two Processor Configuration

5.2 Communication Network

The communication network [Young] for the facility is a multi-stage packet-switched network (Figure 5.3) based on 2x2 self-timed crossbar switches. Communication between switches is asynchronous with a single level of buffering at each switch. The nominal time through each switch is 100nS per packet word. Packet words are 16 bits wide; an additional 17th bit is used to signal end of packet. The most likely packet length in the data-flow applications studied to date is short (5 words).

The current switches in the facility's network are implemented in conventional TTL logic with PAL based state machine control. The final implementation is likely to be custom CMOS with a target time through switch at <100nS. An NMOS prototype is currently being processed. With M68000 based processing-elements initially emulating the architecture the communication network's bandwidth (~640 MBytes/S peak) will be under utilised.

Nodes for a ring network for smaller configurations of 16 processing-elements or less has been designed.

6. APPLICATION STUDIES

Three major application studies have been undertaken and are briefly described below. A number of smaller applications in signal processing and logic net evaluation have also been undertaken.

6.1 Numerical Petri Nets

Studies under a Telecom contract (Egan(3)] indicated that the RMIT architecture, designed to directly evaluate functions expressed as directed graphs, could be readily adapted to execute petri nets. The studies showed however that the more complex firing conditions and memory reference requirements of numerical petri nets would result in substantial overheads in determining enabled transitions.

When used as a numerical petri net or petri net processing-element the matching unit would be responsible for determining when a transition is enabled and the evaluation unit is responsible for operations on net reference data; reference data would reside in the evaluation unit's data space.

In the case of numerical petri nets the determination of enabled transitions may require substantial computation. In this case any computation associated with the firing of enabled transitions may be comparatively small and therefore the evaluation unit may be under utilised.

With the possibility of computational load imbalance within the prototype processing-element for numerical petri nets, petri nets, and under some circumstances for data-flow graphs, considerable debate led to the adoption of the scheme which allows processing-elements to be based on either one or two M68000 based units.

The manpower required to write appropriate translators to convert numerical petri nets to data-flow graphs was outside the level of project funding and prevented extensive studies of what increase in efficiency might be achieved using data-driven architectures. Inspection however indicated that the numerical petri net examples provided exhibited little exploitable concurrency. The reduction in computing time for these nets (T6) using a data-flow machine would be less than a decimal order of magnitude compared with a conventional sequential computer of similar technology. Whether other protocols would exhibit greater exploitable concurrency remains an open question.

Changes to algorithms used within the numerical petri net analyser at Telecom resulted in a substantial reduction in the computing time required for the analysis of numerical petri nets; this made the original simulation efficiency goals less pressing. It was agreed that future studies would shift from a simulation emphasis to how timing analysis and tentatively direct execution of protocols under live communication traffic might be performed using the data-flow multi-processor; the option for these studies remains open.

6.2 Laser Range-finder Based Vision

For this study [Richardson(1)][Egan(2)] the data-flow computing system, laser-rangefinder and object space were simulated on a large conventional computing system [Morris]. The simulated range-finder [Ishii] is a random access device permitting direct determination of the co-ordinates of an illuminated point on an object, or the background.

The algorithm used was to interrogate the centre of the space and then the centre of each quarter of the space and so on in a four way simultaneous recursion. If an object was illuminated, the distance to the background being known, copies of the co-ordinates of the illuminated point were sent simulataneously to a number of sub-graphs, or procedures, each of which was designed to recognise a particular object class; the search for other objects continued simulataneously to a mesh size smaller than assumed size of the smallest object. Given an illuminated point the recognition sub-graphs sent further requests to the rangefinder to either confirm or refute that the object was of the class recognised by them. The outputs of the recognition sub-graphs were the co-ordinates of the objects centroid, its dimensions and the probability of it being a particular object class.

The exploitable concurrency in this application is dependent on the number of object classes and objects in the space. Concurrency is constrained by the shared laser rangefinder. Processing-element activity to recognise three objects from three classes is given in Figure 6.2.1.

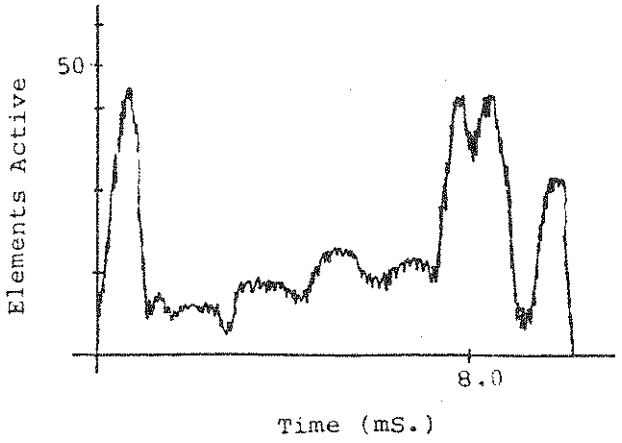

Figure 6.2.1 Object Recognition (3 object classes)

Resource sharing (the rangefinder) and merging of results from a number of concurrently executing sub-graphs are the issues of continuing interest arising from this early application.

6.3 Manipulator Control

For this study [Richardson(2)][Egan(4)] the data-flow computing system, manipulator and task environments were again simulated. The Unimation VAL language [Unimation] was used to describe manipulator tasks.

VAL is a BASIC like language and does not provide language constructions to describe parallel or concurrent actions the evaluation of single high level statements may involve tasks which do exhibit exploitable concurrency e.g. a statement specifying cartesian motion with fixed tool tip orientation to a new location.

VAL task descriptions were interpreted by a program written in Pascal executing on a conventional computing system. Some VAL statements such as incrementing counters were performed directly by the interpreter. For other more complex tasks the interpreter transmitted data or command tokens to the data-flow computing system. The data-flow system then performed the computations, including those associated with servo actions, necessary to implement the VAL statement.

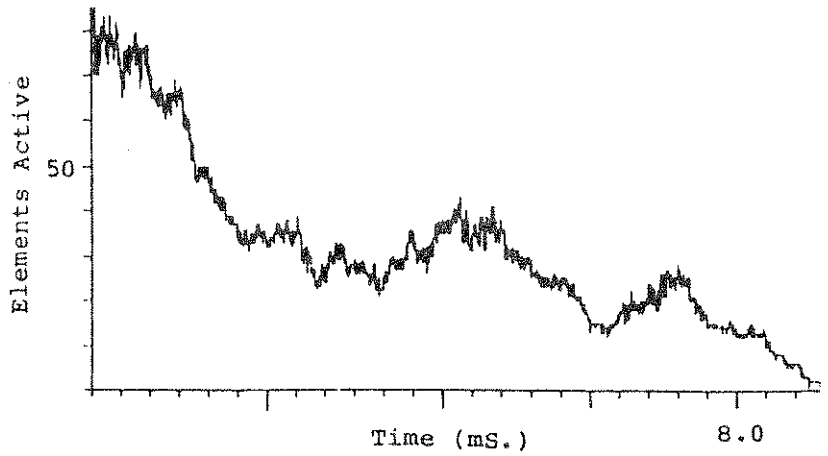

Figure 6.3.1 Cartesian Motion Concurrency

Manipulator control tasks exhibited significant concurrency (Figure 6.3.1) particularly for cartesian motions where the computation of the next joint solution could be commenced before computation of the previous solution was completed. The data-flow computing system is not constrained to execute only one task and may for example support object recognition tasks and manipulator control tasks simultaneously. The study showed that the parallelism in typical manipulator tasks can be exploited easily using the multi-processor.

7. CURRENT PROJECT STATUS

The project currently has two academic members of staff, one professional officer and three research students associated with it. Most of the projects effort is currently being directed into completing the commissioning of the data-flow multi-processor.

REFERENCES

[Adams] D.A. Adams, 'A Model for Parallel Computations', in Hobbs (ed) Parallel Processor Systems, Technologies and Applications, Spartan Books, 1970, pp311-333.

[Chan] Chan, S.K., & Lim, K.K., 'Data-flow Machine Mathematics-element', Design 3 Report, Department of Communication and Electronic Engineering, Royal Melbourne Institute of Technology, Nov. 1984.

[Egan(1)] Egan, G.K., 'Data-flow: Its Application to Decentralised Control', Ph.D. Thesis, Dept. of Computer Science, University of Manchester, 1979.

[Egan(2)] Egan, G.K., and Richardson, C.P. 'Object Recognition Using a Data-Flow Computing System', Microprocessing and Microprogramming 7, North-Holland, 1981.

[Egan(3)] Egan, G.K., 'Simulation of Numerical Petri Nets Using Data-driven Computer Architectures', Reports 1,2,3&4 Telecom Australia, Contract No.63228, 1983-84.

[Egan(4)] Egan, G.K. and Richardson, C.P., 'Manipulator Control Using a Data-driven Multiprocessor Computer System', National Conference and Exhibition on Robotics, Melbourne, Aug. 1984.

[Gurd and Watson] Gurd, J. and Watson, I., 'Data Driven System for Data-flow Computing', Part 1 and Part 2 in Computer Design, Vol.19, No.6 and 7, 1980.

[Ishii] Ishii, M., and Nagata T., 'Feature Extraction of Three-dimensional Objects and Visual Processing in Hand-eye System', Pattern Recognition, Vol. 8, p229.

[Morris] Morris, D., and Ibbett, R.N., 'The MU5 Computer System', Macmillan Computer Science Series, 1979.

[Rawling] Rawling, M.W., and Zuk, E.A., 'Data-Flow Processing Element', Design 3 Manual, Department of Communication and Electronic Engineering, Royal Melbourne Institute of Technology, 1982.

[Richardson(1)] Richardson, C.P., 'Object Recognition using a Dataflow Machine: Algorithms for a Laser Range-finder', M.Sc. Dissertation, Dept. of Computer Science, University of Manchester, 1979.

[Richardson(2)] Richardson, C.P., 'Manipulator Control Using a Data-Flow Computing System', Ph.D. Thesis, Dept. of Computer Science, University of Machester, 1981.

[Treleaven] Treleaven, P.C., 'Exploitation of Parallelism in Computer Systems', Ph.D. Thesis, Dept. of Computer Science, University of Manchester, 1977.

[Unimation] Unimation Inc., 'User's Guide to VAL', Unimation Inc., Danbury, Conn., Feb. 1979.

[Wathanasin] Wathanasin, S., 'Proposed Language for the Multiprocessor', Internal document, Dept. of Computer Science, University of Manchester, 1978.

[Weng] Weng, K.S., 'Stream-oriented Computation in Recursive Data-flow Schemas', Technical memo 68, Laboratory for Computer Science, Massachusetts Institute of Technology, Oct. 1975.

[Young] Young, A., Internal Engineering Documentation, Department of Communication and Electronic Engineering, Royal Melbourne Institute of Technology, June 1984.

Contact

The contact for the RMIT project is:

Dr. G.K. Egan
Principal Lecturer,
Digital Systems and Computer Engineering,
Department of Communication and Electronic Engineering,
Royal Melbourne Institute of Technology,
P.O. Box 2476V, Melbourne, 3001.
Phone: (03)660 2090 ACSNet: rcoge@Unison6