G. Egan

IREE

IREECON INTERNATIONAL

MELBOURNE 1981

18th INTERNATIONAL
ELECTRONICS
CONVENTION & EXHIBITION

AUGUST 24th — 28th, 1981
EXPO CENTRE
ROYAL MELBOURNE SHOWGROUNDS

THE INSTITUTION OF RADIO AND ELECTRONICS ENGINEERS AUSTRALIA
The Science Centre, 35-43 Clarence Street, Sydney, N.S.W. 2000
Telephone: (02) 294051

A Data-Flow Computing System For Decentralised Control and Advanced Automata Applications

G.K. Egan, SMIREE
R.M.I.T.
C.P. Richardson
University of Manchester

The FLO multi-processor computing system is based on the Data-flow model of computation and directly executes, or evaluates, computations described by means of finite directed graphs.

Two of a number of studies involving the system are object recognition using laser range-finder and manipulator (robot) control.

SUMMARY

The FLO multi-processor computing system [4] belongs to a class of computing systems based on the Data-flow model of computation [1]. Data-flow computing systems directly execute or evaluate computations described by means of finite directed graphs.

Graphs may be partitioned and the partitions allocated to the system's processing-elements; graph connectivity provides the information needed to transmit data between and within elements. Processing-elements in the system need not be identical and may be based on conventional micro-processors; communication links between elements may be asynchronous bit-serial.

Because control is asynchronous and fully decentralised the system lends itself readily to geographic distribution, giving it marked advantages over conventional control-flow systems which rely on centrally available control information, are not readily distributed, and have poor underlying mathematical models [2].

Extensive simulation studies have been conducted with system configurations of up to 128 processing-elements and a small pilot system has been constructed. A number of studies involving the system are either in progress or have been completed. The FLO system and two related application studies, those of object recognition [6] [8] manipulator or robot control [9], are described briefly in this paper.

THE COMPUTING SYSTEM

FLO is a fully 'decentralised' data-flow computing system based on a variant of Adam's model [1]. Decentralised means that the system does not rely on any centrally available information for its operation. FLO unlike other major data-flow computing systems [4] is intended primarily for the decentralised control of distributed systems [5].

The system uses finite directed-graphs to describe computations. The edges or arcs of the graphs are queues of data directed from one node to another. The nodes represent functions which map input data onto output data. Data flows down the arcs as packets or tokens, each node requiring a specific number of tokens to trigger the node functions evaluation. The evaluation or firing consumes tokens from the input arcs and places results on the output arcs. The number of nodes eligible for firing at any instant depends only on the availability of data.

The FLO system may be characterised as follows:

1. Tokens are strongly typed and of variable length.
2. Node-functions are weakly typed and accept a set of argument token types. 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.
4. Because of the association of input-output with particular processing-elements, and the system's probable geographic distribution, graphs are partitioned and the partitions allocated statically to processing-elements.
5. Storage nodes are provided to allow the graph to retain 'semi-permanent' information.
6. Information describing exceptions occurring during graph evaluation is communicated to the graph using the token-type ?, or don't know.
7. The system supports shared sub-graphs in sufficient generality to allow multiple recursion.
8. The hardware underlying the system consists of an arbitrarily large number of processing-elements, communicating over channels which may be asynchronous and unsophisticated. The processing-elements themselves may be based on conventional microprocessors.

Major system studies so far have been simulation based. The Simulator can evaluate graphs with the order of 16,000 nodes while modelling systems of 128 processing-elements; graphs are evaluated at approximately 500 nodes/second.

To investigate the feasibility of basing systems on conventional computing elements a pilot system of four processing-elements using 8-bit microprocessors has been constructed. 'Typical' graph evaluation involves interpretation overheads of the order of 20%. With the assistance of hardware arithmetic units computational throughput increases but the interpretation overheads rise to around 80%. These overheads are satisfactory where throughput is not the major consideration.

OBJECT RECOGNITION

Ishii and Nagata used a laser based range-finder coupled with a multiple hypothesis algorithm to recognise objects in an object space [7]; each hypothesis corresponded to an object in the set of recognisable objects.

Their approach was well suited to the FLO system as the laser range-finder could be used to directly interrogate the object-space and the hypotheses could be tested in parallel [6].

The range-finder is located at the origin of a cartesian space. Objects rest on the positive x-y plane the laser may be vectored in the positive x-y (θ) and y-z (ϕ) planes. The range-finder returns the distance from the origin to the spot illuminated by the laser.

The five stages of the recognition processes were:

1. Finding an object: This process was implemented as a doubly recursive sub-graph taking as input parameters an interval in θ. A vertical line bisecting the interval is scanned. Should a hit be obtained (indicated by a range discontinuity) the value of θ is returned. Otherwise the two intervals generated by bisection are processed simultaneously in the same manner.

2. Delimiting the object: Bounding values of θ left, θ right, ϕ top and ϕ bottom are obtained through iteractive routines which scan the object vertically or horizontally at a particular interval seeking the edge. When the edge is passed, the interval is halved and the direction of scanning reversed.

3. Resolution of Clusters: The range profile is inspected for discontinuities. If a discontinuity is found, the bounding parameters are adjusted so that the objects furthermost away in the cluster are ignored and the parameters for the nearest object are sent on to the hypothesis testing sub-graphs.

4. Hypothesis testing: The actual range profile is compared with a predicted range profile; statistical analysis gives the probability that the object is the one expected.

5. Results: The probabilities issued by the hypothesis sub-graphs are examined for the most probable object.

The study highlighted the need for additional research in optimal graph partitioning strategies.

MANIPULATOR CONTROL

A study of manipulator or robot control using the FLO system [9] was a logical extension to the object recognition study as they both aimed to make manipulator stations more flexible, reducing reliance on expensive work piece jigging.

The need for flexibility has exposed weaknesses in single processor control of manipulators [3] which while adequate for simple fixed sequence control are quickly overloaded when called upon to perform continuous manipulator path control in real-time. This has led to an interest in multi-processor control of manipulator stations.

The control structure adopted in the study is hierarchical and may be described as follows:

1. A conventional processor interprets control programs written in the Unimation VAL language [10]; commands in this language correspond to high level goals such as MOVE <location, orientation>. The processor directs the VAL command parameters into the FLO system as tokens.

2. The top level of the data-flow control graph is collection of sub-graphs one for each actionable VAL command. Sub-graphs corresponding to VAL commands such as MOVE, APPROACH GRASP etc. generate intermediate manipulator servo commands. These intermediate commands specify, for example target velocity profiles for individual servos.

3. The bottom levels of the control graph correspond to subgraphs controlling individual servos and monitoring manipulator status. Servo subgraphs are responsible for maintaining specified velocity profiles or driving the servo at maximum speed from set-point to set-point.

4. Servo and other bottom level manipulator status monitoring sub-graphs are responsible for direction motion complete and/or fault tokens to the higher level sub-graphs.

It should be noted that apparently simple VAL commands calling for a specific velocity profile point to point and fixed manipulator end effector orientation involve computation which is far from trivial.

CURRENT RESEARCH

In addition to the continuing study on manipulator control, the following areas of research are being actively pursued:

1. Application-specific graphical languages.

2. 'Fast' processing-elements for applications which require high computational throughput.

3. Optical communication techniques for systems of closely coupled processing-elements.

4. A detailed study of system aspects associated with time and non-determinacy.

5. The application of FLO to 'knowledge-based' or 'expert-systems'.

ACKNOWLEDGEMENTS

The authors would like to thank the University of Manchester's Department of Computer Science for the provision of computing and engineering facilities during the course of the research leading to this paper.

REFERENCES

[1] D.A.Adams, "A Model for Parallel Computations" in Hobbs [ed] Parallel Processor Systems, Technologies and Applications, Spartan Books 1970.

[2] J.Backus, "Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs", CACM Vol. 21, No. 8, pp613-641, Aug. 1978.

[3] P.Coiffet et al., "Real Time Problems in Computer Control of Robots", Proceedings of the 7th Industrial Robots, pp145-152, Oct. 1977.

[4] G.K.Egan, "A Study of Data-flow: Its Application to Decentralised Control", Ph.D thesis, Dept. of Computer Science, University of Manchester, 1979.

[5] G.K.Egan, "A Decentralised Computing System based on Data-flow', Proceedings of the IECI' 80 Conference, Mar. 1980.

[6] G.K.Egan and C.P.Richardson, "Object Recognition Using a Data-flow Computing System", accepted for publishing EUROMICRO Journal.

[7] M.Ishii and T. Nagata, "Feature Extraction of Three Dimensional Objects and Visual

Processing in a Hand-Eye System Using a Laser Tracker", Pattern Recognition, Vol. 8, pp229-237, 1970.

[8] C.P.Richardson, "Object Recognition using a Data-flow Machine: Algorithms for a Laser Range-finder", M.Sc. Dissertation, Dept. of Computer Science, University of Manchester, 1979

[9] C.P. Richardson, "Manipulator Control Using a Data-flow Machine", Ph. D. Thesis, Dept. Computer Science, University of Manchester, to be submitted.

[10] B.Shimano, "VAL: A versatile Programming and Control System", Proceedings of the COMPSAC'79 Conference, pp.878-883, Nov. 1979.

A General-Purpose Dynamic System Simulator

H.B. Gatland
University of Auckland
C.W. Fotherby
Cossor Electronics Ltd., Essex, UK

A 16-bit microprocessor system with analogue input and output channels has been designed to operate as a dynamic simulator. Once the system parameters have been entered the computer can be regarded as a real-time representation which provides analogue outputs appropriate to the analogue inputs.

INTRODUCTION

The analogue computer was for many years used as the primary means of simulating dynamic systems. Although such systems have the ability to provide direct real-time integration, there are limitations largely due to the reliance on the physical properties of components and the measurement of voltages. Digital methods are now widely used for dynamic system simulation purposes.

When simulation is not required in real-time a special language can be used with a general-purpose digital computer. A language such as CSMP (Continuous System Modelling Program) can be employed (1). This uses the conventions relating to an analogue computer approach, but does not permit inter-connection with real components. Because of ready access to general purpose computers this is an economical method of simulation. The analogue system, using separate integration, is a parallel-oriented system which permits high speed integration of a number of variables simultaneously. This allows real-time simulation of many physical systems (2). To improve the precision of the analogue integrator, digital versions have been built. As integration is not a direct arithmetic operation, and has to be accomplished by a continuous summation process, the digital integrator is more complicated than the operational amplifier type. However the precision of a digital system can be improved by increasing the binary word length, and the integration gain can be controlled without any physical modification. Digital integrators when used for analogue simulation still require interconnections to be made. This can be accomplished with patch-cords as with the traditional analogue system, or by digital-gating. The former is awkward, even for small simulations, if frequent changes are needed. The latter is complex if a high degree of flexibility is required, but can be preprogrammed and set-up automatically.

STATE EQUATION MODEL

General purpose digital computers can be programmed to solve the state-equations of a particular plant or dynamic system (3). The state-equations are usually expressed in matrix form and have a standard structure.

If a new system is to be simulated, the appropriate new matrix coefficients are entered, and the response of the system to applied inputs, or initial state, can be calculated step-by-step. For calculation of the new state, involved matrix operations are necessary, and if the solution is to be in real-time enabling a real process to be modelled, a powerful digital processor is required. This has restricted this approach to minicomputer systems, with the programs developed in a high-level language (4). However, with the availability of powerful microprocessors it is currently possible to build special-purpose digital simulators which have provision for analogue inputs and outputs, and externally behave in a similar manner to the traditional analogue computer. An advantage is that no interconnections are required as the necessary relationships are established by the matrix representing the system.

HARDWARE ARRANGEMENT

As shown in Figure 1 the system follows the form of a digital computer. However, as the simulator operates either in real-time, or some multiple of real-time, the central processing unit (CPU) is controlled by an interrupt generator which can be used to speed-up or slow-down the simulation. As with any computer system the memory is divided into program and data sections. The former is 6K words of EPROM which contains all the programs necessary to perform the simulations, as well as providing a monitor facility. These programs are available immediately the system is switched on. The system constants are entered through the VDU key-board and are stored in the data RAM memory.

Two devices are shown in Figure 1 using the serial interface. These are the VDU which is used for both entering programs and operating the monitor and displaying data for the interactive programs. The second serial device is the cassette store which is used to retain operator-developed programs. These are initially entered using the monitor facility, and can be subsequently stored on tape for later use. A parallel interface is provided with 2 sixteen-bit ports available on the front panel.

For the system to operate as a simulator, 16 channels for analogue inputs and 16 channels for analogue outputs are available at the front panel, and are under the command of the analogue I/O controller. The channels are automatically assigned when the system is being set-up and the operator inserts the number of inputs, and number of outputs required.

One 12-bit A/D converter, with sample-and-hold and a 16 channel multiplexer, is used for the analogue inputs. The analogue outputs use a single 12-bit D/A converter which continually monitors an auxilliary data store holding the current outputs. The D/A outputs are held on 16 sample-and-hold amplifiers which provide the output sources.

USING THE SIMULATOR

With POWER ON or RESET the monitor prompt is displayed by the VDU. The simulation program is called by typing S, and the data for the simulation is entered as each request is answered. System parameters are entered as the A, B, C and D matrices.