National Conference and
Exhibition on Robotics, 1984,
Melbourne, 20-24 August

Manipulator Control Using a Data-driven Multi-processor Computer System

G.K. EGAN
Senior Lecturer, Department of Communication & Electronic Engineering, Royal Melbourne Institute of Technology
and
C.P. RICHARDSON
Department of Computer Science, Victoria University of Manchester

1 INTRODUCTION

Manipulator systems of the near future can not rely on conventional Von Neumann computer architectures for unlimited computational power. In spite of this substantial increases in computing power will be necessary if manipulator systems are to support adequately higher levels of decision making, sensors and continuous path control [1].

A number of schemes employing several conventional computational elements or a conventional processor augmented with special purpose transform processors to control manipulator systems [2] have been used. The computational model underlying conventional computing architectures is itself inadequate and not very useful [3]. Certainly the data communication mechanism in systems using several conventional computational elements is difficult to integrate with the sequential computing model. The systems engineer is thus faced with basing theoretically sound distributed control solutions on unsound computing systems.

The study described in this paper applies a computing system based on the Data-flow model of computation to the task of manipulator control.

2 DATA-FLOW

The Data-flow model of computation was originally developed by Karp and Miller at IBM's Thomas J. Watson Research Centre [4] and subsequently expanded by Adams to include conditional evaluation [5].

The Data-flow model uses a finite directed-graph to describe a computation. The edges or arcs of the graph are queues of data directed from one node to another. The nodes represent functions which map input data onto output data or results.

Data flows down the arcs as packets or tokens, each node requiring a specific number of tokens to trigger the node function's evaluation. The evaluation or firing consumes tokens from the input arcs and places result tokens on the output arcs. The number of nodes eligible to fire at any instant depends only on the availability of data. Node functions include the normal arithmetic, mathematical, logical operators and functions which conditionally control the path of data through the graph.

The models of Karp and Miller, and Adams have the property of determinacy: it does not matter in what order eligible nodes fire or how long the node-functions take to be evaluated; the result of the computation is always the same. This makes the Data-flow model well suited to a computing system of loosely coupled processing elements.

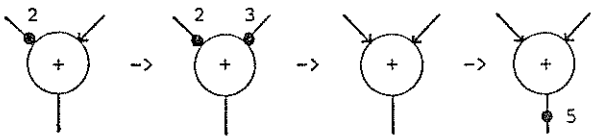


Figure 1 Firing Example

2.1 A Control System Example

Data-flow computations are expressed in a graphical form which closely resembles the traditional block diagrams of control engineering. As with a block diagram, the overall data-flow graph may be decomposed into more manageable sub-graphs; this process is continued until some desired level of functional complexity is reached. Eventually the decomposition process will lead to sub-graphs which are the node functions recognised by the computing system.

As an example, which may occur some way down the overall decomposition process, we take the case of a PID compensator [6] imbedded in a single-loop controller.

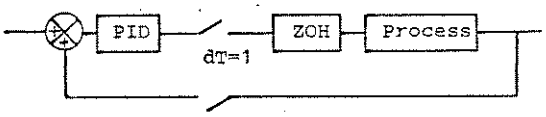
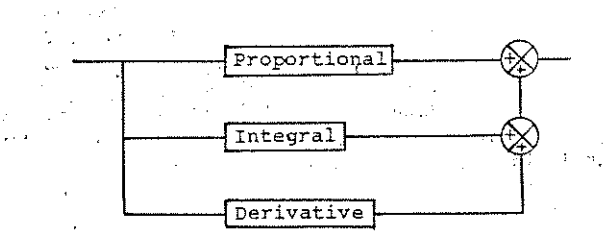
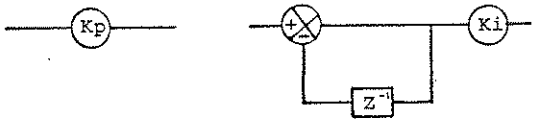


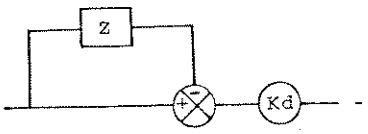
Figure 2 Digital-PID Based Controller

The PID block can be decomposed into sub-blocks for the proportional, integral and derivative components of the compensator.





Proportional Integral



Derivative

Figure 3 Block Diagram Decomposition

Similarly the PID data-flow graph can be decomposed into sub-graphs.

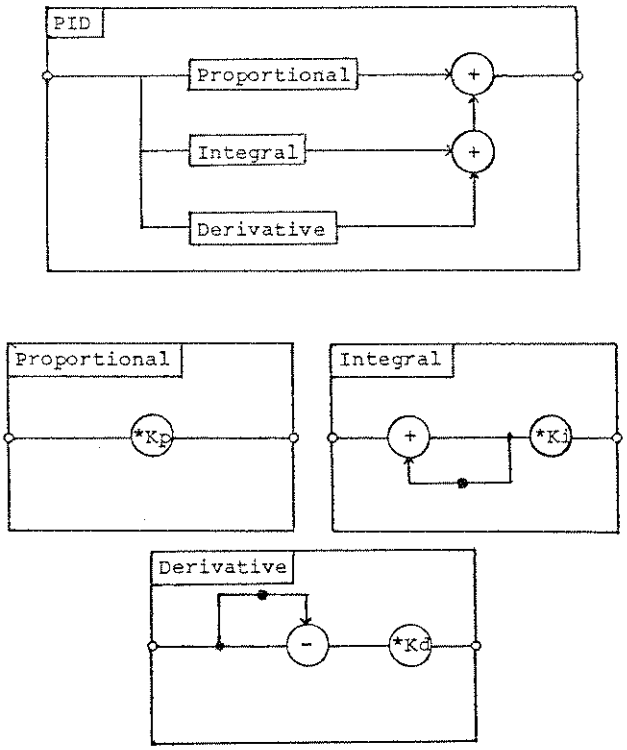


Figure 4 Data-flow Graph Decomposition

Placing an initial token on an arc has the effect of delaying subsequent tokens on that arc by dT. Initial or priming tokens placed on arcs give the state of the graph at t=0.

2.2 Data-flow Computing Systems

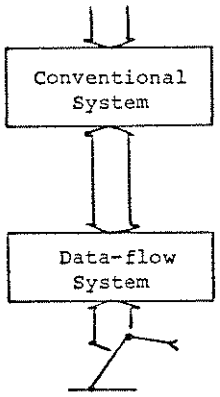
Several computing systems directed at general purpose computations and based on the Data-flow model have been proposed [7,8,9,10]. The computing system used in the study described in this paper was developed specifically for distributed control and advanced automata applications [11,12,13, 14,15,16]. A prototype machine has been constructed [17] with some structures being implemented as VLSI circuits [18,19]. The computing system consists of a number of connected computational elements. The communication of data between elements is described fully by the model. Computation is driven solely by the availability of data and therefore maps readily onto the stimulus-response environments of real-time control.

3 MANIPULATOR CONTROL

For this study the data-flow computing system, manipulator and task environments was simulated on a large conventional computing system [20]. The Unimation VAL language [21] was used to describe manipulator tasks; our current studies use the Pascal Language. The VAL language is a primitive language similar to BASIC. Data types are restricted to real, integer and two types of location variables which are used to specify manipulator positions. All variables are global, there are no procedure parameters and recursive structures are not allowed. The control structures are very rudimentary consisting of conditional and unconditional control transfers (jumps) to integer statement labels. VAL programs describe a sequence of actions with some sequences being conditional on external stimuli or the result of a computation.

While VAL 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.



Manipulator

Figure 5 Simulated System

The graph within the data-flow system contained all the necessary sub-graphs for manipulating joint solutions and transformations. There were three types of location variable known to the graph; each type of variable represented a different level of abstraction. The variable types were:

1) Joint Solutions which specified the setpoint of each servo,

2) Transformations which specified the tool-tip co-ordinates and orientation in a three by three matrix and

3) Descriptions which were records containing the tool-tip co-ordinates and orientation angles O,A and T.

The sub-graph structure used for translating a description into the servo setpoints is shown in figure 6. It is important to note that, given sufficient computational elements, all sub-graphs may be active simultaneously when computing solutions for say cartesian motions.

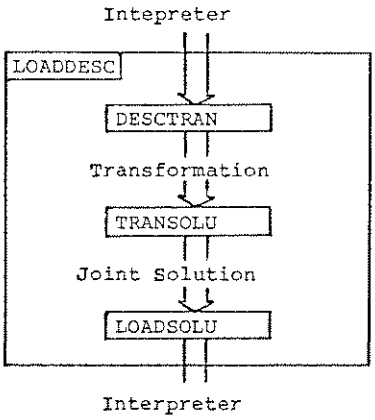


Figure 6 Description-Solution Sub-graph

Consider the simple task of de-palletising, one of several such tasks studied. Nine work-pieces were arranged in an orthogonal pattern on a rectangular pallet. The task was to remove the work-pieces from the pallet and deliver them to a machining point.

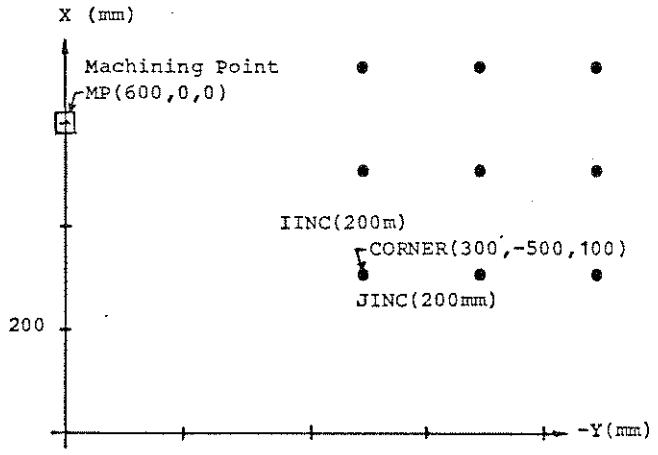


Figure 7 De-palletising Task Layout

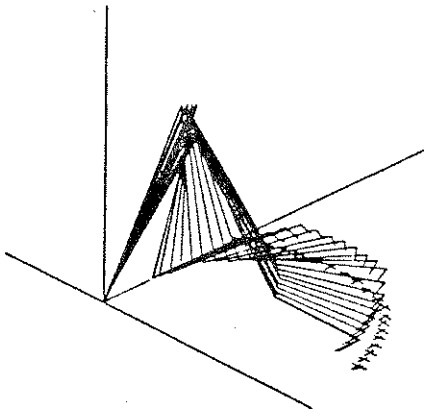
The two main routines of the VAL program describing the de-palletising task are shown below. The first routine is called NEXTPIECE and computes the transformation of the next work-piece to be moved and assigns it to the location variable NEXT. The second routine is called CLEARPALLET and describes the order in which the work-pieces are to be moved.

```
Subroutine 'NEXTPIECE'
     SETI I=I+1
     IF I.GT.3 THEN 100
     SET NEXT=IINC:NEXT        (*transform to next
                               (*item
     GOTO 300

100  SETI I=1
     SETI J=J+1
     IF J .GT. 3 THEN 200
     SET NEXT=JINC:ISTART      (*transform current
                               (*item to next row
     SET ISTART=NEXT

300  RETURN
200  RETURN 1                  (*finished return
                               (*skip one statement

Subroutine 'CLEARPALLET'
     READY                     (*reset arm
     SETI I=1
     SETI J=1
     SET ISTART=CORNER
     SET NEXT=ISTART           (*first piece

100  APPRO NEXT,50             (*joint interpolated
                               (*to 50mm from piece
     MOVES NEXT                (*cartesian to piece
     CLOSEI 0                  (*grasp
     DRAW 0,0,100              (*cartesian vertical
                               (*100mm
     DEPART 100                (*along gripper axis
     MOVES  MP                 (*cartesian to machine
                               (*point
     OPENI 100                 (*release work-piece

     GOSUB NEXTPIECE           (*location of next
     GOTO 100

     RETURN
```

Figure 8 shows motion plots of the manipulator and tool tip descriptions for two VAL statements.

	X	Y	Z	O	A	T
Start	600	0	0	90	0	0
Finish	300	-850	100	0	0	0

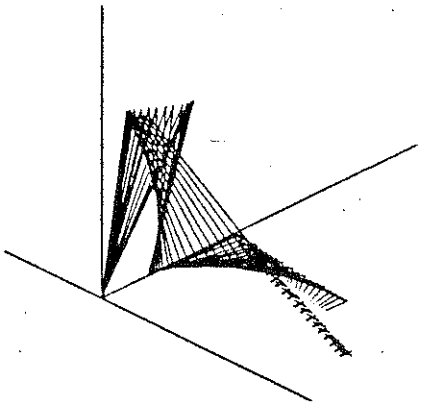
Duration 0.638S



APPRO NEXT,50
Joint Interpolated Motion

	X	Y	Z	O	A	T
Start	300	-600	200	0	0	0
Finish	600	0	0	90	0	0

Duration 1.302S



MOVES MP
Cartesian Motion

Figure 8 Manipulator Motion Plots

4 RESULTS

Interpretation of VAL statements requiring computation of joint solutions from tool tip descriptions used the LOADDESC subgraph. A plot of computational element activity for the LOADDESC sub-graph is shown in figure 9. An average of 6 elements were active with peaks of 30. If further descriptions for a continuous motion are sent to the LOADDESC sub-graph before the computation for the previous descriptions are completed the element activity is much higher as shown in figure 10. The data-flow system exploits the task parallelism easily.

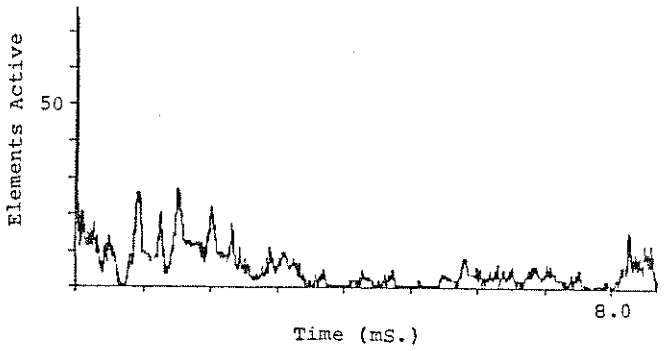


Figure 9 LOADDESC Parallelism

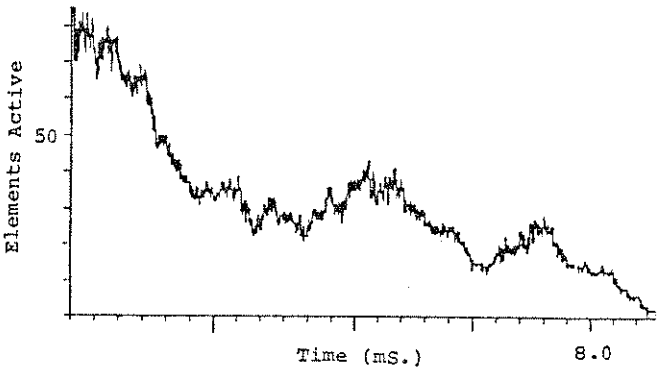


Figure 10 LOADDESC Continuous Path Parallelism

5 CONCLUSIONS

A computer system which uses the data-flow model has been successfully applied in two major studies:

1) object recognition using a laser range-finder [12,14] and

2) manipulator control [15].

The second study, described in this paper, used the Unimation VAL language to describe the manipulator tasks which involved joint interpolated and cartesian continuous path motion. The VAL task descriptions were interpreted by a conventional processor which sent task data and commands to a data-flow computing system. The data-flow system computed the transformations and generated the low level servo commands for the modelled Unimation series 6000 manipulator [22].

Manipulator control tasks exhibited substantial parallelism 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 data-flow computing system.

6 ACKNOWLEDGEMENTS

The authors wish to thank Professor T. Kilburn for the use of the computing facilities at the Department of Computer Science at the Victoria University of Manchester. C.P. Richardson wishes to thank the Government of Barbados for its financial support.

7 REFERENCES

[1] Coiffet, P., et al., 'Real Time Problems in Computer Control of Robots', Proceedings of the 7th. International Symposium on Industrial Robots, pp145-152, Oct. 1977.

[2] d'Auria, A. and Salmon M., 'SIGMA -An Integrated General Purpose System for Automatic Manipulation', Proceedings of the 5th International Symposium on Industrial Robots, pp185-202, Sept. 1975.

[3] Backus, J., '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.

[4] Karp, R.M., and Miller, R.E., 'Properties of a Model for Parallel Computations: Determinacy, Termination and Queueing', SIAM J. Applied Mathematics, Vol.11 No.6, pp1390-1411, Nov. 1966.

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

[6] Ogata, K., Modern Control Theory, Prentice-Hall Inc., Englewood Cliffs, N.J., 1970.

[7] Davis, A.L., 'Architecture of DDM1: A Recursively Structured Data Driven Machine', Technical Report, University of Utah, 1977.

[8] Dennis, J.B. and Misunas, D.P., 'A Preliminary Architecture for a Data Flow Processor', Proceedings of the Second Annual IEEE Symposium on Computer Architecture, p126-, 1975.

[9] Syre, J.C., 'Pipelining, Parallelism and Asynchronism in the LAU System', Proceedings of the IEEE International Conference on Parallel Processing, pp87-92, Aug. 1977.

[10] 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.

[11] Egan, G.K., 'A Study of Data-flow: Its Application to Decentralised Control, Ph.D. Thesis, Department of Computer Science, Victoria University of Manchester, 1979.

[12] Richardson, C.P., 'Object Recognition Using a Data-flow Machine: Algorithms for a Laser Range-finder', M.Sc. Dissertation, Department of Computer Science, Victoria University of Manchester, 1979.

[13] Egan, G.K., 'A Decentralised Computing System Based on Data-flow', Proceedings of the I.E.E.E. Industrial Control and Instrumentation Conference, Philadelphia,

March 1980.

[14] Egan, G.K. and Richardson, C.P., 'Object Recognition Using a Data-flow Computing System', EuroMicro Microprocessing and Microprogramming 7, North-Holland, 1981.

[15] Richardson, C.P., 'Manipulator Control Using a Data-flow Computing System', Ph.D. Thesis, Department of Computer Science, Victoria University of Manchester, 1981.

[16] Walkington, M., 'A Graphical Data-flow Compiler', Design 2 Report, Department of Communication and Electronic Engineering, Royal Melbourne Institute of Technology, 1983.

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

[18] Biggs, I., 'An NMOS VLSI Queue for the Data-flow Multiprocessor', Design 3 Report, Department of Communication and Electronic Engineering, Royal Melbourne Institute of Technology, 1983.

[19] Siow, A., 'An NMOS VLSI Communication Switch for the Data-flow Multiprocessor, Design 3 Report, Department of Communication and Electronic Engineering, Royal Melbourne Institute of Technology, 1983.

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

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

[22] Dunne, M.J., 'An Assembly Experiment Using Programmable Robot Arms', Proceedings of the 7th International Symposium on Industrial Robots, Tokyo, Oct. 1977.