J. Indian Inst. Sci., Mar.–Apr. 1991, **71**, 195–228.

IISc THESES ABSTRACTS

Thesis Abstract (Ph.D.)

A broadcast cube-based multiprocessor architecture for solving partial differential equations
by C. Siva Ram Murthy.
Research supervisor: V. Rajaraman.
Department: Computer Science and Automation.

1. Introduction

A large number of mathematical models in engineering and physical sciences employ partial differential equations (PDEs). Models to perform numerical weather prediction, simulation of nuclear reactors and fusion reactors, analysis of blood flow in the human body, computation of air flow around an aircraft or aerospace vehicle, propagation of noise through the atmosphere, simulation of petroleum reservoirs, etc., are some examples based on PDEs. The sheer number of operations required in numerically integrating PDEs in these applications has motivated the search for faster methods of computing. The conventional uniprocessor computers are often unable to fulfill the performance requirements of these computation-intensive problems.

Currently, many problems, such as those mentioned above, are being solved on array processors or single instruction multiple data stream (SIMD) computers (BSP, MPP, etc.), and vector or pipeline computers [CRAY X-MP, ETA10, NEC SX-2, VP-100, etc.]. The performance of array processors is impressive only for specific problem environments. Vector computers incorporate substantial amount of parallelism and pipelining in their central processing units (CPUs). However, the vector computers attain high speeds only with complex and expensive CPUs.

A very promising approach to tackle the problems stated above is to adapt the computer structure to the inherent parallelism of these problems, *i.e.*, to solve them by multiprocessor systems or multiple instruction multiple data stream (MIMD) computers. Since MIMD computers are inherently more flexible than array processors and vector computers, they are suitable for a much larger class of problems. With the recent advances in VLSI technology it has now become feasible to design and implement multiprocessor systems with a large number of processors or processing elements (PEs).

The principal contributions of the present study are: (i) development of BCS (broadcast cube system), a cost-effective message-based multiprocessor system for solving important computation-intensive problems such as systems of linear algebraic equations and PDEs, and (ii) development of an efficient strategy for assigning the set of (cooperating) tasks in a program to the PEs in the BCS. The performance of the BCS has been evaluated by both analytical and simulation studies.

2. Methodology used and results

A message-based multiprocessor system which we call the broadcast cube system has been proposed for solving important computation-intensive problems such as systems of linear algebraic equations

and PDEs[1]. The BCS has the following important features: (i) Being based on the conceptually simple bus interconnection scheme it is easily understood. The use of homogeneous PEs which can be realized as standard VLSI chips makes the hardware less costly. (ii) The interconnection network is simple and regular. The network can easily be extended to a vast number of PEs by adding buses with new PEs on them and by slightly increasing the number of PEs on existing buses. The interconnection pattern is highly redundant to support fault tolerance in the event of PE failures. (iii) In terms of the switching delays, the delay a message undergoes between a pair of PEs connected to a common bus is zero. The maximum delay between any pair of PEs is one unit and thus a strong localization of communicating tasks is not needed to avoid long message delays even in networks of thousands of PEs. A simulator for performance evaluation of parallel algorithms to be executed on the BCS has been implemented[2]. A comparative study of the BCS with other multiprocessor systems has also been made[3].

One of the key problems in the design of any incompletely connected multiprocessor system is to appropriately assign the set of tasks in a program to the PEs in the system. The task assignment problem, in which the program tasks have precedence and communication constraints among them, is intractable and this entails heuristic approaches. In this study, a simple and efficient strategy (task-assignment algorithm) for assigning program tasks with precedence and communication constraints to the PEs in the BCS has been developed[4–6]. The effectiveness of this strategy has been demonstrated by comparing the results with (i) the lower bound on the execution time of a program, and (ii) a random assignment. The task assignment algorithm has been shown to produce optimal assignments for PDE problems. An important merit of this algorithm is that a topological change in the BCS due to failures or bringing in of additional PEs into the system has no effect on the basic assignment procedure.

Optimal partitioning of the problems, solving systems of linear algebraic equations and PDEs, into tasks and their assignment to the PEs in the BCS have been given. Efficient parallel algorithms for solving these problems on the BCS have been designed[7,8]. The performance of the parallel algorithms has been evaluated by both analytical and simulation methods. The results indicate that the BCS is highly effective in solving systems of linear algebraic equations and PDEs. The performance of these algorithms on the BCS has also been compared with that of their implementations on hypercube machines which are gaining popularity among the message-based multiprocessor systems. The results show that the performance of the BCS is better than that of the hypercubes for linear algebraic equations and compares very well with PDEs, with a modest number of PEs despite the constant PE connectivity of three in the BCS.

Heat transfer and fluid flow simulation, and global weather modelling are two important practical problems that involve solution of nonlinear PDEs. The effectiveness of the BCS in solving these has been demonstrated[9,10].

3. Conclusions

A cost-effective message-based multiprocessor system, which we call the broadcast cube system, has been proposed for solving important computation-intensive problems such as systems of linear algebraic equations and PDEs. A simulator for performance evaluation of parallel algorithms to be executed on the BCS has been implemented. A strategy (task-assignment algorithm) for assigning program tasks with precedence and communication constraints to the PEs in the BCS has been developed and its effectiveness demonstrated. This task-assignment algorithm has been shown to produce optimal assignments for PDE problems. Optimal partitioning of the problems, solving systems of linear algebraic equations and PDEs, into tasks and their assignment to the PEs in the

BCS have been given. Efficient parallel algorithms for solving these problems on the BCS have been designed. The performance of the parallel algorithms has been evaluated by both analytical and simulation methods. The results indicate that the BCS is highly effective in solving systems of linear algebraic equations and PDEs. The performance of these algorithms on the BCS has also been compared with that of their implementations on popular hypercube machines. The results show that the performance of the BCS is better than that of the hypercubes for linear algebraic equations and compare very well with PDEs, with a modest number of PEs despite the constant PE connectivity of three in the BCS. Finally, the effectiveness of the BCS in solving nonlinear PDEs occurring in two important practical problems, (i) heat transfer and fluid flow simulation, and (ii) global weather modelling, has been demonstrated.

References

1. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — A multimicroprocessor architecture for solving partial differential equations, *Microprocessing Microprogramming*, 1987, **20**, 113–118.
2. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — Analytical and simulation studies of a multiprocessor system for high-speed numerical computations, *Math. Computers Simulation*, 1990, **32**, 393–401.
3. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — Multiprocessor architectures for solving PDEs, *J. IETE*, 1988, **34**, 172–184.
4. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — Task assignment in a multiprocessor system, *3rd SIAM Conf. on Parallel Processing for Scientific Computing*, CA, USA, December 1–4, 1987.
5. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — Task assignment in a multiprocessor system, *Microprocessing Microprogramming*, 1989, **26**, 63–71.
6. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — On the assignment of precedence and communication-constrained tasks in a multiprocessor system, *Computer Sci. Infor.*, 1989, **19**, 17–22.
7. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — Iterative solution of linear algebraic equations on a multiprocessor system, *Proc. IEEE Conf. on Computers and Communications Technology toward 2000*, Seoul, Korea, August 26–28, 1987.
8. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — An architecture of a Navier-Stokes computer and its evaluation, *Proc. 6th IMACS Inter. Symp. on Computer Methods for Partial Differential Equations*, PA, USA, June 23–26, 1987.
9. SIVA RAM MURTHY, C. AND RAJARAMAN, V. — A multiprocessor architecture for solving nonlinear partial differential equations, *Math. Computers Simulation*, 1988, **30**, 453–464.
10. SIVA RAM MURTHY, C., MOONA, R. AND RAJARAMAN, V. — A multiprocessor architecture for weather modelling, *Proc. 2nd Inter. Conf. on Supercomputing*, CA, USA, May 3–8, 1987.

Thesis Abstract (Ph.D.)

Design and implementation of multidimension multilink multicomputer hardware and software by Rajat Moona.
Research supervisor: V. Rajaraman.
Department: Computer Science and Automation.

1. Introduction

Multicomputer systems, especially those using low-cost microcomputers, provide a very cost effective solution to the continuing demand for computing power. A multicomputer may be designed to maximize the throughput of multiple jobs or to speedup a single job. Our concern is with the second type of systems where a single job is partitioned into multiple cooperating tasks and run on multiple computers. Of the many issues to be considered in the design of such a system, the most important ones are the design of interconnection network and the method of synchronizing the working of computers in the system.

2. Contribution of the thesis

We propose and implement a multidimensional multilink system (MMS) architecture which uses message-passing paradigm between computing elements (CEs). The merits of this architecture are its simplicity, regularity and rich connectivity among CEs. Many existing message-passing architectures can be emulated very effectively on MMS architecture. A software environment for this architecture is also designed and implemented.

MMS architecture is made using multiple fully connected multicast networks in multiple dimensions. CEs in each dimension are connected through a fully connected network. The number of CEs in this network is called 'drop' parameter of MMS. The network also allows a selective broadcast (multicast) by which a CE can address a few CEs connected through a fully connected multicast network.

Two performance measurement tools for this architecture are also presented. The first one, an analytical model based on mathematical equations, can be used to model very simple and regular problems. It assumes a load-balanced task partitioning of a given problem. This model is used for modelling some problems which are also written and run on the MMS implementation. The second performance-measurement tool, simulation model, can be used for measuring the performance of arbitrarily structured problems. The simulation model is written in SIMULA[1] programming language and can be used for measuring the performance of concurrent programs written in SIMULA. Both these tools give performance parameters like speedup and processor utilization.

The work also discusses various routing algorithms for MMS architecture. This architecture supports a multicast network using which algorithms are given for a point-to-point communication where a CE sends data to another CE which may not be connected, and a broadcast communication where a CE sends data to all the CEs in the system.

Two variations of this architecture, one with four CEs and another with nine, have been implemented. The design of communication unit is independent of the processor used in a CE and a communication unit can be used with any CE if it provides a standard IBM PC[2] bus. This scheme has an advantage by which a processor in CE can be replaced by a better performance processor. Currently the implementation is based on IBM-PC motherboards with Intel's iAPX 88[3] processor at 5MHz clock frequency but the communication unit has also been tried with 80386[4]-based PC/AT

motherboards. This implementation is very cost-effective. Implementation of communication module (CM) can handle a length of interconnection cables up to 20 feet. Using this interface, various computers in a laboratory room can be connected thus providing a multicomputer configuration. This structure then behaves as a small network of computers on which machines can be used either as a set of sequential computers working independently or as a single parallel multicomputer.

An integrated software environment has been designed for MMS architecture and implemented. This environment is fairly general purpose and can be interfaced to any existing language compilers. This makes the implementation user friendly. The existing commercial compilers and the commercial computers can be hooked in a parallel computing environment. Software environment models a program as multiple cooperating tasks where each task is run on a different CE in parallel. It provides subroutine calls to find the configuration of the MMS architecture using which one can write programs that are independent of the MMS architecture. Further, a small subset of MMS can be specified through drop and dimension parameters. For example, on a 3-drop, 2-dimension system, a 2-drop, 2-dimension system can be emulated by specifying drop ≈ 2 and dimension $= 2$. The environment supports software calls for creating remote tasks, terminating tasks, interprocessor communication and certain support routines to help in programming. These support routines return the topological parameters. The interface to the software environment has been developed for various languages. We then discuss the following three benchmarks programs that are coded in 'C' programming language: 1. Numerical integration, 2. Bubble sort, 3. Matrix multiplication.

We discuss the task partitioning for these problems, allocation on various CEs and show how a program can be written using the programming environment that is independent of the MMS configuration. We also present analytical modelling for these programs, and compare the results from simulation and implementation.

3. Results

The MMS system proposed and implemented is a low-cost parallel computing bench and is effective in terms of the synchronization between tasks and the communication networks. The communication between two processors is as fast as the communication between a processor and its memory. It is concluded that the proposed architecture is a viable and useful architecture for message-passing multicomputers. For the benchmarks given, speedup ranges from 0.75 to ≈ 1 times the number of processors.

References

1. BIRTWISTLE, G. M. *et al* — *SIMULA Begin*, Auerbach, Philadelphia, 1973.
2. — *Personal computer X T system*, Technical Reference Manual, IBM, USA, 1981.
3. — *iAPX 86,88 User's manual*, Aug. 1981, Intel.
4. — *80386 Hardware reference manual*, 1986, Intel.

Thesis Abstract (Ph.D.)

Design of buses and arbitration schemes for a multicomputer system by M. S. Shiva Kumar.
Research supervisor: V. Rajaraman.
Department: Computer Science and Automation.

1. Introduction

A cost-effective solution to meet the demand of large computing power is to use a large number of low-cost, low-speed processors in a parallel computing system. Speed improvement of several folds in execution time of a program can be obtained by exploiting parallelism available in the code at the execution level. Multicomputer systems based on multistage networks are costlier and their expandability is limited, as these networks are expensive to design and have limited fan out[1-3]. Hence time-shared bus architecture is quite attractive for cost-effective implementation of a high-performance multiple processor system. We propose an intercomputer, bus-based, message-passing communication system which is a cost-effective solution to interconnect many computing elements (CEs) in a multicomputer system. We have implemented an eight-node token broadcast bus multicomputer (TBBM) system using Intel 8088 microcomputers. A hierarchical-multiple TBBM, as an extension to TBBM, is proposed and its performance evaluated[4].

2. Main contributions

The salient features of our implementation are the following: (a) Interprocessor communication is by broadcast over 32-bit, high-bandwidth, dual-parallel buses in the form of tokens; (b) Separate buses for data and acknowledgement tokens reduce the bus traffic by about half the load; (c) Neither of the buses carry any address lines; (d) The 'processor hiding' concept used in the implementation permits replacement of existing CEs with new generation CEs when they become available with minimum upheaval in the system setup; (e) The broadcast buses have high bus-utilization efficiency – between 87.62 and 99.91% for minimum and maximum token sizes, respectively; (f) The token transmission and reception is done 'on-the-fly' by the bus interface unit independently in parallel with computations; and (g) Bus access is by contention and the arbitration is overlapped with bus transactions.

TBBM architecture is a radically different approach to multicomputer architecture. The existing multicomputers use local area networks or point-to-point serial links in incompletely connected communication structure which have a communication delay of the order of a few milliseconds which throttles the speedup possible with high-performance processors. In contrast to this, we propose a parallel bus interconnection structure and message-transfer protocol, which has a broadcast delay of the order of a few microseconds for the same volume of message and provides high communication bandwidth.

Parallel bus arbitration by encoder-decoder combination has the lowest arbitration delay, but the priorities of CEs are fixed *apriori* and hence inherently it is an unfair scheme. We propose gated parallel, pseudorandom priority and parallel request sweep-service arbitration schemes, all the three of which utilize the conventional encoder-decoder combination but have fairness. The arbitration delay, average waiting time and average queue length are used as performance measures to evaluate these schemes. Simulation results show that the gated parallel and pseudorandom priority arbitration schemes have satisfactory performance while the parallel request sweep-service scheme has the best performance among them. The pseudorandom priority arbitration scheme is programmable to satisfy

a specific request origination pattern from the CEs. This feature is demonstrated for two specific patterns.

The data bus and acknowledgement bus are modeled as M/M/1 and M/D/1 queueing systems, respectively. This illustrates that waiting time of tokens increases rapidly with higher bus utilization. Since the bus mastership change-over time has direct influence on waiting time, it is desirable to keep it low. In our implementation, the switch-over time is optimized to a small value so that a high bus utilization efficiency between 99.91 and 87.62% is achieved for maximum and minimum token sizes, respectively.

Analytical model for the TBBM architecture is developed by assuming that for a given algorithm, both the computation and communication overheads are uniformly distributed among the computing elements. Our analysis demonstrates that the speedup obtained cannot be improved by increasing the number of CEs, N, if N exceeds the ratio of the computation time, t_c, to the communication time, t_c. Thus, the optimum number of CEs is equal to t_c/t_c and the best achievable speedup is also equal to the same value.

Even a high-speed bus is unable to support an arbitrarily large number of CEs without performance degradation due to limited bus bandwidth. Hence, an extension to the TBBM architecture is proposed where clusters of TBBM system modules are connected using hierarchical, multiple global bus structures. Message controller modules are used for routing of tokens from level-*i* to level-*j* bus (*i*#*j*). The bus acquisition is by contention resolution using arbiters for each of the buses. The HM-TBBM is modeled as a network of queues for computing average waiting times and average queue lengths at different message controllers. A level 2 HM-TBBM system has 392 CEs and 9 buses, which provide an overall communication bandwidth of 1.17 GigaBytes/second. The token broadcast delay at each level is of the order of 470 microseconds and hence the overall delay is only about 2.35 milliseconds for 512-word, 32-bit token. The specification of aggregate communication bandwidth of a system do not reflect the actual bandwidth available per channel and the hardware complexity for expansion. Hence, we define two parameters, bandwidth per CE and bandwidth per CE per port, to capture these effects. We observe that in hypercube the bandwidth per CE remains constant at 2, and bandwidth per node per port decreases with increase in the dimensions. In contrast, the HM-TBBM system has both increasing bandwidth per CE and bandwidth per CE per port with increasing number of hierarchical levels. This is due to the fact that the number of ports per CE remains constant in HM-TBBM system but it increases in hypercube with increase in the number of dimensions.

3. Conclusions

We propose a multicomputer architecture and implement it with the available technology. The system provides a small interprocessor communication delay, high bus bandwidth, bus utilization efficiency and fair arbitration scheme. Also the proposed HM-TBBM system, having only two levels, is shown to provide an overall communication bandwidth of 1.17 GBytes/sec and a maximum delay of only 2.35 millisec for a 512-word, 32-bit token, thus providing higher performance compared to second generation multicomputers.

References

1. Athas, W. C. and Seitz, C. L. Multicomputer: Message passing concurrent computers, *Computer*, Aug. 1988, **21**, 9–24.

2. AGRAWAL, D. P., JANAKIRAMAN, V. K. AND PATHALE, G. K. Evaluating the performance of multicomputer configurations, *Computer*, May 1986, **19**(5), 23–37.

3. REED, D. A. AND GRUNWALD, D. C. The performance of multicomputer interconnection networks, *Computer*, June 1987, **20**(6), 63–73.

4. SHIVA KUMAR, M. S., SRINIVASAN, M. P. AND RAJARAMAN, V. A flexible test bed for multimicroprocessor system studies, *Third Int. Conf. Super Computers*, Boston, May 1988, Vol. 3, pp 135–145.

Thesis Abstract (Ph.D.)

Systolic architectures for realistic 3D graphics by Praxedes Canute Mathias.
Research supervisor: L. M. Patnaik.
Department: Computer Science and Automation.

1. Introduction

The interactive computer graphics system as a tool for man-machine interaction has gained wide applicability in many scientific and engineering areas, one of the exciting and potentially important areas being the synthetic generation of realistic images of solid objects. To have a feedback on the solids modelled by an interactive designer, the generation of images and their display have to be carried out in at least 1/30th of a second. This requirement is governed by the characteristics of the eye and the graphics monitor on which the image is displayed[1].

Highly parallel implementations for real-time display are made possible by advances in very large scale integration (VLSI) technology. In this work, we address the fast implementation of three important graphics tasks, namely, removal of hidden surfaces in the display of 3D objects represented by planar surfaces[1], generation of curves and surfaces[2], and evaluation of polynomial expressions[3,4] using systolic arrays towards the design of a high-performance systolic graphics system.

Systolic architectures have all the desirable properties one looks for in parallel architectures[5]. Their only drawback is that they are not general-purpose computing systems. They are high-speed arrays having high effective bandwidth and are ideally suited for compute-bound problems. Another attractive feature is that all these benefits are available at very low cost.

2. Hidden surface removal

A fundamental problem in the realistic display of modelled objects on a 2D monitor screen is occulting hidden portions of objects. This problem is generally referred to as hidden surface removal (HSR)[1]. The objects are modelled as simple polygons. Though the HSR problem is simple in statement, the excessive time required to execute the HSR algorithm for even moderately complex 3D objects places a severe constraint on the real-time display system.

We propose linear arrays for the scanline-based hidden surface removal. Three schemes are presented based on the segment comparison, priority relation, and plane sweep paradigm. In the first, linear systolic array LSA_1, concurrency is exploited by comparing segments in parallel. The data locality is maintained by back-and-forth segment transfers among neighbouring cells. The first linear array is studied in detail by simulation to evaluate its performance. Two conclusions are drawn: (i) it

is sufficient to have a small number of registers in the priority queue of each cell, and (ii) a small number of data transfers between cells in each systolic cycle is sufficient. The second linear array LSA_2 eliminates the priority queue and the back-and-forth data transfers between the cells by stating the problem in the framework of a priority relation. A third linear systolic array LSA_3 based on plane sweep paradigm is also proposed. The execution time of the sequential plane sweep algorithm is improvised upon using a systolic array priority queue data structure with a plane sweep of a single vertical line. The execution time is further improved using multiple sweep lines. Two linear arrays are described for the multiple sweep line-based HSR algorithm. One is a semi-systolic array with a daisy chain of AND gates to determine the terminating condition of the HSR algorithm. The other array is a pure systolic array, with terminating condition determined by a tree of adders. All the arrays have been compared from the point of view of space and time performance. We use the coherence present in the scene, advantages of VLSI technology, and the overwriting feature on the frame buffer, to arrive at the design of high-performance linear arrays for hidden surface removal.

3. Curve and surface generation

While the display of solid objects modelled by polygons is widely used in practice, polygons are not suitable for modelling curved or smoothly varying objects. Bezier and B-spline curves and surfaces have been widely used to model smoothly varying objects[2] such as the body of a car or a ship's hull. However, the generation and display of Bezier and B-spline curves and surfaces are compute-intensive processes. They are studied and systolic implementations for them have been proposed. Two properties of Bezier curves are made use of. From the recursion property, we derive triangular systolic arrays which compute the points on the Bezier curve exactly. The other, namely, the degree elevation property, has a higher speed but only closely approximates the curve. This high-speed feature is attractive in an interactive design environment. Systolic arrays for derivative calculations of Bezier and B-spline curves are also discussed. These derivative evaluations are very useful in computing the tangent, normal, and curvature used in scene models involving reflections, refractions, and for shading. The systolic implementations of the Bezier and B-spline curves are extended to get pyramidal and prism architectures. We also suggest methods for obtaining the points on the surface from the arrays for curve generation by suitable pipelining of the control vertices but at the expense of speed. Different systolic architectures are presented for computing exactly the points on the curves and surfaces, or closely approximating the curves and surfaces by a large number of control points, for quick response. Suggestions are made for an efficient VLSI layout for these systolic arrays. These arrays can provide real-time response in interactive CAD/CAM applications at low cost.

4. Frame buffers

Frame buffers are universally used in raster graphics display systems. We study 'smart' frame buffers[6,7] in which the memory is augmented with simple additional processing elements such as adders, complementors, delay units and comparators. These smart frame buffers are able to compute polynomials $P(x, y)$ at very high speeds and have potential applications in modelling and real-time display of objects. We present two types of organisations for frame buffers. One is a two-dimensional mesh or square grid array with hexagonal interconnections and the other is a set of linear arrays. The square grid array computes the value of the polynomial from the results of the three neighbouring cells. The linear array frame buffer is derived from a sequential algorithm and requires less number of elements in each cell. The space requirement of the frame buffers can be reduced considerably to yield a multiple wavefront-simplified cell array. The complexities of the organisations in terms of their area *period and area *period^2 are compared. The advantage of the organisations described is that the computation in any cell is carried out using the results of the neighbouring cells only and

without the need for any broadcast. Hence, it is very attractive to realise the frame buffer in the form of a VLSI chip where only one cell has to be designed and the rectangular layout of the chip can be carried out by tiling. It is also very convenient to design a printed circuit board for the frame buffer using these chips, as the layout for the interconnections can be easily done by the same tiling process. The VLSI array can be laid out on the chip as a rectangle such that there is a one-to-one correspondence between the coordinates of the pixel on the screen and of the pixel registers on the chip. Hence, a light-emitting device constituting the screen can become a part of the chip thus eliminating the scanning process.

5. Experimental systolic arrays

To show that the architectures studied in this work are not mere theoretical issues, a test bed using processing modules with Intel 8086 microprocessors is set up[8]. The processing modules are built to be sufficiently general for use in a variety of systolic architectures and in a multiprocessor system.

We consider four problems for implementation on systolic arrays. They are: (1) Generation of Bezier curves, (2) Generation of B-spline curves, (3) Inversion of the B-spline curves, and (4) Hidden surface removal.

A triangular array for the Bezier and B-spline curve generation and a linear array for the problem of inversion in B-spline curves are set up using six processor cells. A speedup of nearly six has been observed for these experiments. A linear array LSA_1 for HSR using eight processing cells is also studied on the experimental setup. The results obtained are very similar to those from the simulation studies. This validates the simulation study which is also carried out for more complex scenes.

6. Ray tracing and other graphics problems

The time required for a sequential ray-tracing algorithm[9] is of the order of the product of the number of surfaces in the scene and the total number of rays traced. The ray-surface intersection is the most time-consuming part of the ray-tracing algorithm requiring a large number of floating point calculations. Since for typical scenes the number of rays traced is thousands or tens of thousands and each ray has to be tested with all the surfaces in the scene, the execution time on a uniprocessor becomes prohibitively large.

We conduct preliminary investigations for the implementation of the systolic ray-tracing algorithm. We introduce the systolic ray-tracing tree to compute the ray-surface intersections in parallel. We then present a systolic mesh of trees in which each systolic tree is assigned a small region of the subdivided 3D space. We present linear systolic arrays for some of the widely used algorithms in raster graphics. These include the line drawing, circle generation, polygon scan conversion, clipping and geometric transformations.

7. Conclusions

The generation and display of realistic 3D objects in computer graphics have been emerging as important fields in computer science with their impact felt in many disciplines of science and engineering. The ability to achieve this by means of high-speed VLSI devices using the techniques of parallel processing, and at low cost is reported. We propose systolic arrays for the important compute-intensive problems in the generation and display of realistic 3D pictures.

References

1. SUTHERLAND, I. E., SPROULL, R. F. AND SCHUMACKER, R. A. — A characterisation of ten hidden surface algorithms, *Computing Surv.*, 1974, **6**, 1–55.
2. BOHM, W., FARIN, G. AND KAHMANN, J. — A survey of curve and surface methods in CAGD, *Computer-Aided Geometric Des.*, 1984, **1**, 1–60.
3. MATHIAS, P. C. AND PATNAIK, L. M. — A systolic evaluator for linear, quadratic and cubic expressions, *J. Parallel Distributed Computing*, 1988, **5**, 729–740.
4. MATHIAS, P. C. AND PATNAIK, L. M. — Systolic evaluation of polynomial expressions, *IEEE Trans.*, 1990, **C-39**, 653–665.
5. KUNG, H. T. — Why systolic architectures?, *IEEE Computer*, 1982, **15**, 37–46.
6. FUCHS, H., POULTON, J., PAETH, A. AND BELL, A. — Developing pixel planes. A smart memory-based raster graphics system, *Proc. 1982 Conf. on Advanced Research in VLSI*, MIT, Dedham, MA, Artech House, pp 137–146.
7. GOLDFEATHER, J. AND FUCHS, H. — Quadratic surface rendering on a logic enhanced frame-buffer memory, *IEEE Computer Graphics Applic.*, 1986, **6**, 48–59.
8. MATHIAS, P. C., PATNAIK, L. M. AND SUDHA, R. — Systolic architectures in curve generation, *Computer Graphics*, 1989, **13**, 561–567.
9. ARVO, J. AND KIRK, D. — A survey of ray tracing acceleration techniques. Tutorial notes on introduction to ray tracing, *Siggraph 88*, 1988, pp 1–46.

Thesis Abstract (M.Sc. (Engng))

Integrated analytical models for parallel and distributed computing systems by C. R. Meenakshi Sundaram.
Research supervisor: Y. Narahari.
Department: Computer Science and Automation.

1. Introduction

Analytical modelling plays an important role in the design and development of parallel and distributed computing systems. In this context, product form queueing networks (PFQNs) and generalized stochastic Petri nets (GSPNs)[1] represent two principal modelling tools. Computationally efficient solution algorithms exist for PFQNs, leading to their wide use. However, the inability of PFQNs to capture non-product form features such as synchronization, blocking, and dynamic routing hampers their utility as a general-purpose modelling tool. GSPNs constitute a flexible tool for modelling computer system behaviour involving synchronization, concurrency, and conflict phenomena. The representational power of GSPNs allows one to capture in a natural way non-product form features. However, the problem of state-space explosion inherent in GSPN analysis restricts the use of GSPNs in modelling large-scale practical systems. So, both these techniques have advantages and disadvantages in terms of computational complexity and flexibility of modelling. Recently, a new technique that combines the efficiency of PFQNs and the representational power of GSPNs has been considered in literature[2].

The basic idea behind this new technique is the following. If the non-product form features of a model are localized in a submodel, then this submodel is studied in isolation, keeping the number

of jobs it uses fixed. Then a high-level model is constructed for the entire system in which this submodel is represented as a flow-equivalent server. The performance model in the integrated technique has two parts, namely, a PFQN part and GSPN part. If the high-level model is a PFQN model, then some of its stations will be the flow-equivalent representations (derived using GSPNs) of the submodels that incorporate non-product form features. If the high-level model is a GSPN model, then some of its timed transitions will be the flow-equivalent representations (derived using PFQNs) of the submodels that are product-form solvable. We shall refer to this technique as integrated analytical modelling in the sequel.

In this research work, we show how integrated analytical modelling provides an efficient technique for evaluating parallel and distributed computing systems. The specific problems we consider are the following:

(i) Incorporating time variance of parallelism in performance models of dataflow architectures.
(ii) Analytical evaluation of dynamic routing policies for load balancing in distributed computing systems.

In both of the above applications, the integrated analytical models provide an accurate description of the systems and lead to very efficient performance analysis.

2. Modelling of dataflow architectures

When the parallelism in applications is characterized at a high level of detail, the performance estimates based on them will be superior in nature. Based on a single parameter characterization, namely, the average parallelism, a closed queueing network model for the Manchester dataflow architecture has recently been developed[3]. More recently, it has been found[5] that decisions based on a 4-parameter characterization, namely, minimum, maximum, average, and variance of parallelism are superior to those based on single parameter characterizations.

Based on the above 4-parameter characterization, we have developed performance models for the execution of applications in the Manchester dataflow architecture, using integrated analytical modelling. The integrated model has a PFQN part and a GSPN part. The product form features have been captured by the PFQN part. The non-product form feature, namely, time variance of parallelism has been captured by the GSPN part.

We have evaluated the model for different values of A, the average parallelism. In each of the cases, we allowed V, the variance of parallelism to vary from 0 to 2A. When compared with the model based on single-parameter characterizations, our results reported error in the performance measures as high as 26% for the values of V close to the value of 2A.

Some of the findings of the study are the following:

(i) The average parallelism is a good characterization only as long as the variance of the parallelism is small compared to the average parallelism. However, there could be significant difference in performance measures when the variance is comparable to the average parallelism.
(ii) When compared with the exact GSPN modelling technique, significant state-space reduction was observed in the integrated approach. Also, the performance estimates based on the integrated approach are quite accurate.

3. Modelling of dynamic routing policies

The problem of balancing the load over the entire system so that its overall performance is optimized is an important research problem in distributed computing systems. Routing an arriving job to the

processor with the shortest queue length (SQR, shortest queue routing) in the case of homogeneous systems and to the processor with the shortest expected delay (SEDR, shortest expected delay routing) in the case of heterogeneous systems have been found to be close to optimal heuristic load-balancing strategies.

Several approximations[5] have been developed to compute performance measures of distributed computing systems employing these dynamic routing schemes. These approximations have the following drawbacks:

(i) Different techniques are employed for different performance measures.
(ii) Good approximations are generally obtained only under heavy or light load assumptions.
(iii) They are computationally expensive.

GSPN models can effectively overcome problems (i) and (ii) above but modelling the entire system using GSPNs where only a part of the system employs these strategies leads to state-space explosion.

In our research work, we have developed integrated analytical models for capturing these dynamic routing schemes. A GSPN model for the subsystem employing these strategies is developed and studied in isolation keeping the number of jobs it uses fixed. Then a high-level PFQN model for the entire system is developed in which the subsystem is represented by a flow-equivalent server derived using the GSPN submodel. Such an integrated approach can be employed for modelling any general distributed computing systems with dynamic routing schemes.

For demonstrating the efficacy of our integrated analytical models and for numerical experimentation, we have considered closed central server models with SQR and SEDR routings. Central server models with state-dependent routing have been considered extensively in literature[6]. Even though we have considered closed queueing models, our results are applicable to open queueing models in which the maximum number of jobs that can be present in the submodels employing these dynamic routings is a finite value. Through evaluation of this class of models for different values of job population and service capacities of the nodes of the model, we have shown that:

(i) With increase in the population of the central server model, the cardinality of the state space of the exact GSPN model grows exponentially. But, the cardinality of the state space of the GSPN submodel for SQR and SEDR is significantly less. For example, in a model with four processors, the GSPN submodel has a maximum of six states for SQR, and a maximum of four states for SEDR, for all the populations.
(ii) When the population is high, the results obtained using the integrated analytical model showed an error as low as 0.1%, compared to the exact models.

We also have compared the performance of these dynamic routing schemes with some simple load-balancing policies that can be modelled exactly by queueing networks[6]. SQR and SEDR are found to give at least 10% better performance than the others. These comparisons have been made possible, because the integrated analytical models provided us efficient approximate analytical solutions having an accuracy comparable to that of the exact modelling technique.

References

1. MARSAN, M. A., BALBO, G. AND CONTE, G. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems, *ACM Trans. Computer Systems*, 1984, **2**, 93–122.

2. GHANTA, S. *On the integration of queueing network and generalized stochastic Petri nets for the performance evaluation of computer systems*, Ph.D. Dissertation, Department of Computer Science, University of Minnesota, Minneapolis, December 1984.

3. GHOSAL, D. AND BHUYAN, L. N. Analytical modelling and architectural modifications of a dataflow computer, *Proc. Fourteenth Annual Symp. on Computer Architecture*, June 1987, pp 81–89.

4. SEVCIK, K. Characterizations of parallelism in applications and their use in scheduling, *Performance Evaluation Rev.*, 1989, **17**, 171–180.

5. NELSON, R. AND PHILIPS, T. K. An approximation to the response time for shortest queue routing, *Performance Evaluation Rev.*, 1989, **17**, 181–189.

6. TOWSLEY, D. Queueing models with state-dependent routing, *J. ACM*, 1980, **27**, 323–337.

Thesis Abstract (M.Sc. (Engng))

Performance analysis and scheduling of stochastic fork-join jobs in a multicomputer system

by Rajeev Shorey.
Research supervisors: Anurag Kumar and L. M. Patnaik.
Department: Electrical Communication Engineering.

1. Introduction

We study the performance of a particular type of parallel program, a stochastic fork-join job, on a multiprocessor system consisting of homogeneous interconnected processors. Each such job consists of a series of forks and joins, such as those that might be created by *parbegin* and *parend* constructs in parallel programming languages. Each fork gives rise to a random number of tasks that can be processed independently on any of the processors. The job terminates after a random number of fork-joins. Our objective is to study the response times of such jobs under certain stochastic assumptions.

The main difference between the fork-join model we have described and those studied in the literature[1,2] is that in the previous work the task structure of the arriving jobs is *deterministic* (*i.e.*, the number of tasks in a fork is a constant) and the tasks map exactly on to the processing elements, *i.e.*, one task to each element. For example, in the most commonly studied model, each job brings N tasks, there are N processors, and one arriving task is scheduled on each processor. Such a model is, however, inappropriate for a general-purpose parallel processing system. In Nelson *et al*[3], the number of tasks in a job is a random variable, but the model analysed is that of a centralized parallel processing system where the N servers are all fed by a single queue, whereas what we model and analyse is a distributed parallel processing system with a queue associated with each of the N servers.

What makes our model more complex to analyse than those studied previously is the fact that, in general, the task batches allocated to the N queues are dependent. This, in turn, causes the task-batch service times to be dependent.

2. The fork-join model

We specialize to the situation where there is only one fork-join in each job. Upon arrival at a processor, the tasks of the job are assigned to the processors using an allocation policy. A job leaves the system as soon as all its tasks complete their service. We are interested in the mean job response time.

In Fig. 1, we show probabilistic task allocation. An arriving task is allocated to processor i with probability p_i.

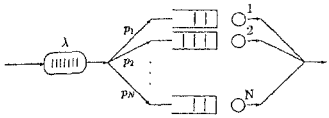

FIG. 1. Multicomputer model: All jobs arrive at a central scheduler.

3. Upper bounds to the mean job response time

Job arrivals are assumed to be Poisson with rate λ. The arriving batch is split probabilistically into subbatches which are then assigned to the processing elements. The arriving batch is thus multinomially partitioned over the N processors, each task going to the ith processor with probability p_i. Thus, a possibly empty, subbatch of tasks is allocated to each queue. Note that, in general, these subbatches are dependent.

Clearly, the job sojourn time is the maximum of the sojourn times of the nonempty subbatches.

Assume that the stability condition is satisfied. Denote by $W^{(i)}$ to be the stationary waiting time of the subbatch at the ith queue, and $D^{(i)}$ the service time of the subbatch at the ith queue, $i = 1, 2, \ldots, N$. Then, denoting the sojourn time of a fork-join job by T, we have:

$$T = \max_{1 \leqslant i \leqslant N} (W^{(i)} + D^{(i)}) I_{\{D^{(i)} > 0\}} \tag{1}$$

where $I_{(\cdot)}$ is the indicator function of $(\cdot)$.

Following Baccelli *et al*[2], we use the properties of associated random variables[4] to compute upper bounds to the mean job response time ($E[T]$).

We first prove that if the subbatches at the N queues are associated, then the subbatch service times are also associated. It then follows that the subbatch sojourn times are associated. The associativity upper bound to the mean job response time then follows.

We then attempt to find batch-size distributions which result in associated subbatches. It is proved using known results that the components of a multinomial partition of a geometric random variable are associated[4]. It follows that for geometric batch size, upper bound to the mean job response time can be computed.

4. Lower bounds to the mean job response time

A trivial lower bound to $E[T]$ is the mean delay in each component queue. It is quite clear that this bound is only a function of λ (or ρ) and not of N, the number of processors. We next search for lower bounds which are a function of N.

Consider now a queueing system with a single queue that is served in an FCFS manner by the N servers. Consider a single batch arriving at this system and finding it empty. We prove that the sojourn time of the batch in the centralized system is lower in the sense of stochastic order than the sojourn time of the batch in the original system.

Notice that the above lower bound amounts to comparing the two queueing systems in the limit as $p \to 0$, and thus ignores the queueing delays. We next seek a lower bound which varies with both ρ and N.

We use the technique in Baccelli *et al.*[2] to show that if the task service time distribution in the original distributed system (with exponential task service time) is replaced by a deterministic distribution, having the same mean, then the sojourn time in the resulting system is a lower bound in the sense of convex increasing order.

Now, consider yet, another system in which tasks with deterministic service times are all put into a single queue, that is served in a first-come-first-served fashion by the N servers. This represents a system with centralized queueing of tasks and distributed service[2]. It is easily seen that the task sojourn times in the deterministic service time system with centralized queueing are smaller sample pathwise than in the deterministic service time system with distributed queueing[5,6].

The centralized system with deterministic service time ($M^x/D/N$ queue) is exactly analysable for geometric batch sizes. Exact analysis of the lower bound system for Poisson batches is difficult; however, the subbatches, allocated to queue, are independent and the lower bounds suffice[7].

5. Multiple fork-join jobs

We study a simple approximation for the fork-join queueing model where the arriving jobs now have a multiple fork-join structure. We model the multiple fork-join queue as a fork-join N processor system with feedback. After the completion of a join, with probability p the job is fed back to the main arrival stream and with probability $(1-p)$, the job leaves the system. Thus, the number of fork-joins in a job is geometrically distributed with parameter p, *i.e.*, $P(K=k)=p^{k-1}(1-p)$, $k \geq 1$, $0 \leq p \leq 1$, where K is the random variable corresponding to the number of fork-joins in the job.

The analysis of the multiple fork-join queueing system is considerably harder. The reason is that although the external arrival process is Poisson, owing to feedback the total arrival process at the queueing system is not Poisson[8].

Since the multiple fork-join queue with feedback is not easily amenable to analysis, we have studied an approximation for this system motivated by an approximation for the M/G/1 queue with feedback.

6. Comparison of two JSQ allocation policies

We study two versions of the join-the-shortest-queue (JSQ) JSQ by batch and JSQ by task. In JSQ by batch, an arriving batch (with a random number of tasks) is assigned to the shortest of the N queues in the multicomputer system. In JSQ by task, the tasks of an arriving batch are ordered in an arbitrary way and are then successively assigned to the shortest queue as if they were a stream of task arrivals.

Through simulation and analysis of these two JSQ allocation policies, we have seen that for low and moderate utilizations of the queue, JSQ by task yields a strictly lower value of mean job response time as compared to JSQ by batch. However, at high utilizations, the two policies approach each other in mean delay performance. Thus, if communication delays were included in the model, JSQ by batch would yield lower mean response time than JSQ by task for high loads. This suggests an adaptive allocation policy: do JSQ by task at low and moderate loads and do JSQ by batch at high loads.

In the limit when $\rho \to 0$, it is proved using the arguments that JSQ by task yields a strictly lower value of mean job response time as compared to JSQ by batch[9]. The behaviour at heavy loads can be explained using diffusion approximation results.

7. Conclusion

Motivated by models for parallel programs on multicomputer systems, we have studied the fork-join queueing model. We have extended the scope of this class of models by allowing a random number of tasks in each fork, task allocation policies, and multiple forks in a job. For the probabilistic task allocation policy we have obtained upper and lower bounds to the mean job sojourn time and have compared them with simulation results. We have also studied the JSQ allocation policy and have compared two versions of it.

References

1. Nelson, R. and Tantawi, A. N. — Approximate analysis of fork/join synchronization in parallel queues, *IEEE Trans.*, 1988, **C-37**, 739–743.
2. Baccelli, F., Massey, W. A. and Towsley, D. — Acyclic fork-join queueing network, *J. ACM*, 1989, **36**, 615–642.
3. Nelson, R., Towsley, D. and Tantawi, A. N. — Performance analysis of parallel processing systems, *IEEE Trans.*, 1988, **SE-14**, 532–539.
4. Barlow, R. E. and Proschan, F. — *Statistical theory of reliability and life testing: Probability models*, 1975, Holt, Rinehart and Winston.
5. Wolff, R. W. — An upper bound for multi-channel queues, *J. Appl. Probability*, 1977, **14**, 884–888.
6. Kuri, J. and Anurag Kumar — On the optimal allocation of customers that must depart in sequence, submitted to *Operations Res. Lett.*
7. Baccelli, F. and Makowski, A. — Simple computable bounds for the fork-join queue, In *Proc. Conf. Information Science Systems*, Johns Hopkins Univ, Baltimore, Md., March 1985, pp 436–441.
8. Wolff, R. W. — *Stochastic modeling and the theory of queues*, 1989, Prentice-Hall.
9. Weber, R. R. — On the optimal assignment of customers to parallel servers, *J. Appl. Probability*, 1978, **15**, 406–413.

Thesis Abstract (M.Sc. (Engng))

A compiler and symbolic debugger for Occam by M. Chelliah.
Research supervisor: Y. N. Srikant.
Department: Computer Science and Automation.

1. Introduction

Parallel computing is being advocated as the only solution to many problems in key areas like weather forecasting, remote sensing, space technology, etc. To harness the enormous computing power of concurrent systems, software support in the form of languages allowing the user to express parallelism is being proposed. We have chosen one of them, Occam, which is sufficient for most of the parallel programming needs and is relatively easy to implement as a vehicle for parallel algorithm testing and operating system implementation.

Occam is a descendant of CSP[1] proposed by C.A.R. Hoare with a few convenient modifications such as allowing channels to be used for communication and procedures. Even though a large number of CSP-based languages have been implemented, to the best of authors' knowledge, only a handful of Occam implementations are seen in literature[2,3]. Among them, INMOS implementation[2] receives considerable hardware support, as the transputer provides instructions even for process creation and interprocess communication.

We have developed a compiler and symbolic debugger for Occam. Our compiler generates code for the 68020 microprocessor and this code runs on a UNIX-based minicomputer. The 68020 does not provide any instructions for process creation and communication. Hence, we have developed our own runtime system to achieve these tasks and various others required by the implementation.

A programming system cannot be considered complete without a debugger. Debugging plays a major role in the life cycle of program development. A source-level (symbolic) debugger is presumed to be more user-friendly than an assembly level one. But, unfortunately, developing a source-level debugger from scratch is non-trivial. Hence, in developing any parallel debugger, it is a prevalent custom to use an already existing source-level debugger as the core to provide conventional debugger features like breakpoint setting, variable inspection, etc.[4,5]. We have adopted the well-established Unix system V.2 source-level debugger 'Sdb' as the foundation for developing a parallel debugger for Occam.

2. Compiler implementation

The compiler has been implemented using well-known software tools available with us. The front end consists of a lexical analyser developed using LEX and a parser with semantic analysis and intermediate tree generation developed using YACC. The back end, *i.e.*, the code generator was developed using a code generator generator (CGG) available in our department. The input specifications to the CGG consist of various tree patterns which occur in the tree intermediate code and the corresponding assembly code to be generated and its cost. The output from the CGG is a code generator which accepts the tree intermediate code generated by the front end and generates 68020 assembly code for the MAGNUM minicomputer. Runtime support has been developed entirely in C and made available as a library. This is linked to the assembly module to generate an executable version of the input Occam program.

3. Parallel debugger features

The following features have been provided in our parallel debugger and we believe that these features are sufficient in practice.

1. Read and write the variables of individual processes.
2. Dynamic scheduling allowing user interaction.
3. Deadlock and livelock detection and helping the user to analyse the situation.
4. Choosing a guard of an ALT construct to which control should flow at the beginning of each ALT transaction.
5. Inspecting channels.
6. Providing diagnostic messages depicting context switching and interprocess communication.

4. Conclusions

We have designed and implemented a compiler and a debugger for Occam. This is a valuable tool for testing parallel programs. The compiler generates high-quality code because of the use of good code-generation techniques.

References

1. HOARE, C. A. R.	Communicating sequential processes, *CACM*, 1978, **21**, 666–677.
2.	*INMOS Occam programming manual*, 1987, Prentice-Hall.
3. FISCHER, A. J.	A multiprocessor implementation of Occam, *Software Practice Experience*, 1986, **16**, 875–892.
4. GAIT, J.	A debugger for concurrent programs, *Software Practice Experience*, 1985, **15**, 539–554.
5. GRIFFIN, J. H. *et al*	A debugger for parallel processes, *Software Practice Experience*, 1988, **18**, 1179–1190.

Thesis Abstract (M.Sc. (Engng))

HIRECS: Hypercube implementation of relaxation-based circuit simulation by Srilata Raman.
Research supervisor: L. M. Patnaik.
Department: Computer Science and Automation.

1. Introduction

The increasing complexity of VLSI chips has made computer aids indispensable to the design of the chips. A variety of computer-aided design (CAD) tools that shorten the design cycle time and ensure the reliability of the chips are available. Of the various CAD tools available to a designer, verification tools occupy the most significant position in the design process. Circuit-level simulation is a verification tool that is of special interest to the design engineer. Circuit-level simulation provides precise waveform information such as frequency response, time-domain waveforms, and sensitivity information of the circuit. However, circuit simulation is a highly compute-intensive task as it involves solving thousands of ordinary differential equations (ODEs) describing the VLSI circuit under consideration. The time required for performing circuit simulation grows almost exponentially as the circuit size increases. There is thus an urgent need to speed up this important task. This work is an effort towards this using a parallel computer architecture, the hypercube.

The work focusses on the design and development of HIRECS (hypercube implementation of relaxation-based circuit simulation). HIRECS is based on the relaxation approach of solving the ODEs describing the circuit. The relaxation-based approach provides the circuit variables as accurate waveforms as the standard circuit simulation based on the direct method[1], with up to two orders of magnitude speed improvement. The relaxation method decomposes the problem of solving a large system of equations describing the circuit into the problem of solving several smaller subsystems. The natural decomposition makes the relaxation algorithms amenable to parallel implementation. HIRECS employs the waveform relaxation (WR) algorithm[2], that is relaxation is applied at the differential equation level. The special feature of WR algorithm is that the latency of the circuit can be exploited better, effecting a saving in the total computation time. The target parallel architecture is the hypercube, which uses message-passing scheme of communication among processors.

2. Why hypercube?

1) The solution of circuit equations requires communication of data corresponding to the circuit variables after every iteration. This may result in communication bottlenecks unless multiple paths are available among the processors. Hypercube offers such multiple paths[3].

2) For circuits containing feedback loops too much communication may be required between distant processors. The connectivity of an architecture like mesh may not be sufficient for solving such circuits. The hypercube topology has sufficiently high connectivity that can solve such tightly coupled circuits without significant performance degradation.
3) Unlike tree architecture where the root processor becomes a bottleneck in communication, owing to the symmetry of the hypercube, no processor poses such a problem.
4) Hypercube has the advantage of being a general-purpose machine that is commercially available.

Considering all the above factors, we have used the hypercube topology for the parallel implementation of the circuit simulation algorithm.

For solving the circuit equations, HIRECS partitions the circuit to be analysed into a number of tightly coupled subcircuits. The subcircuits are evaluated using the more efficient direct method while relaxation is applied among the subcircuits. A novel circuit partitioning scheme forms an essential constituent of HIRECS.

3. Circuit partitioning

The task of partitioning a circuit into suitable subcircuits is a combinatorial optimization problem. HIRECS partitions the circuit using a scheme based on simulated annealing, a heuristic-based approach. The simulated annealing-based approach ensures that the final solution obtained is close to the global optimum whereas the standard iterative methods have a tendency to get stuck at a local optimum[4]. The circuit partitioner developed has the following important features:

1) It generates almost uniform sized partitions that result in uniform distribution of load among the processors.
2) It places tightly coupled nodes in one subcircuit such that the resultant subcircuits show fast convergence.
3) It attempts to minimize the distance between the processors containing subcircuits that require interprocessor communication thus minimizing communication overheads.

The simulated annealing-based circuit partitioning scheme is modified to run on the hypercube architecture. The performance of the parallel partitioning algorithm is shown in Fig. 1. It shows the speedup obtained for different number of processors.

4. Salient features of HIRECS

1) The concept of 'windowing' has been incorporated in HIRECS to effect a saving in the memory requirement. In this approach, the entire simulation time interval is divided into smaller time intervals called 'windows'. Relaxation is carried to convergence over each window before proceeding to the next window.
2) It has a novel synchronization scheme called partial synchronization incorporated in the parallel implementation of the relaxation algorithm. In this scheme, synchronization of the processors of the hypercube is carried out at the end of evaluation of variables at all time points over a window after convergence. Partial synchronization combines the advantages of totally synchronous and asynchronous relaxation.
3) It is tailored to the simulation of MOS circuits, though it can be extended to other technologies as well.

HIRECS runs on a DEC-1090 system and is developed using the programming language SIMULA. In the development of the hypercube simulator, HIRECS adopts the basic design methodology of modelling concurrent processes on a uniprocessor system. Though hypercube implementation of

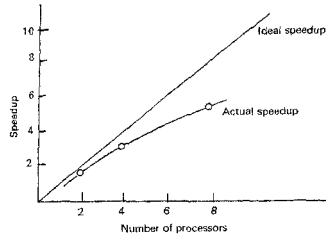

FIG. 1. Speedup of SA algorithm as function of number of processors.

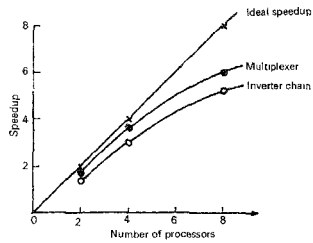

FIG. 2. Speedup *versus* number of processors.

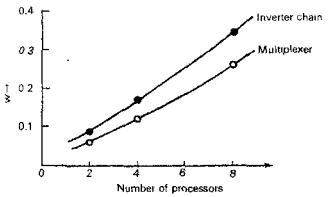

FIG. 3. Wasted time *versus* number of processors.

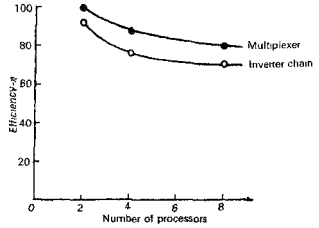

FIG. 4. Efficiency *versus* number of processors.

circuit simulation has been tried out in an earlier work CONCISE[5], the implementation of HIRECS differs significantly from it. One important difference is that CONCISE uses point relaxation whereas HIRECS is based on block relaxation. The latter has better convergence properties. Moreover, CONCISE uses a concurrent program that does not take into account architectural features of hypercube like message passing and communication overheads. HIRECS, on the other hand, simulates all the architectural features of the hypercube.

5. Conclusions

Performance studies of HIRECS based on parameters such as speedup, efficiency, and utilization of processors have been carried out as a function of the number of processors (Figs 2–4). The performance evaluation of HIRECS in the simulation of some benchmark circuits like inverter chains and multiplexers indicates that significant speedup, almost linear, can be obtained by using a hypercube. It has not been possible to try circuits with large number of nodes due to non-availability of data for such circuits. However, a safe prediction can be made to the effect that for circuits with large number of nodes, and with adequate number of processors considerable saving in the computation time can be achieved.

References

1. NAGEL, L. W. *SPICE2: A computer program to simulate semiconductor circuits*, Technical Report ERL-M 520, Electronics Research Laboratory, Univ. of California, Berkeley, 1975.

2. LEELARASMEE, E. *The waveform relaxation method for time-domain analysis of large scale integrated circuits*, Memo UCB/ERL M82/40, Electronics Research Laboratory, Univ. of California, Berkeley, 1982.

3. SAAD, Y. AND SCHULTZ, M. H. *Topological properties of hypercubes*, Research Report, YALEU/DCS/RR-389, Yale Univ., June 1985.

4. KIRKPATRICK, S., GELATT, C. D. Jr. AND VECCHI, M. P. Optimization by simulated annealing, *Science*, 1983, **220**, 671–680.

5. SVEN, M. *CONCISE: A concurrent circuit simulation program*, Research Report: Coden: Lutex/(Tete-1003)/1–116, Deptt of Applied Electronics, Lund Institute of Technology, Sweden, 1986.

Thesis Abstract (M.Sc. (Engng))

Reliability analysis and fault diagnosis of a spacecraft control system by V. Nagesh.
Research supervisor: N. Viswanadham and K. R. Ramgopal (ISAC).
Department: Computer Science and Automation.

1. Introduction

This work is primarily concerned with the reliability analysis and fault diagnosis of a real-life spacecraft attitude and orbit control system (AOCS). AOCS is an important subsystem of a spacecraft and its reliable performance and fault-tolerance capabilities are crucial to the success of space mission.

AOCS consists of several subsystems/elements and any malfunction/defect in them is likely to adversely affect the performance of a spacecraft. We consider the consequence of failure of all the components of the AOCS on the spacecraft performance. Further, since reliability is one of the vital subsystem design parameters in all space-borne systems, we also compute the reliability measures for AOCS.

An important subsystem such as AOCS should be fault-tolerant so that failures in the individual elements will not have disastrous effect on the spacecraft performance. A required level of performance could always be guaranteed by appropriate redundancy management and use of online fault diagnosis methods. We outline a systematic methodology for the design of fault diagnostic methods for a real-life spacecraft attitude-control system.

2. Failure analysis techniques

We introduce two existing methods of failure analysis techniques, namely, failure modes and effects analysis (FMEA) and fault-tree analysis (FTA)[1]. FMEA is an inductive method of analysis, where the analyst induces a failure into the system at the lowest level, *i.e.*, at the component level and traces the complete path through which the fault propagates. For this reason, FMEA can be classified as a bottom-to-top technique. FTA is a deductive method of analysis and it can be identified as a top-down technique. In FTA, the analyst starts from a fault at the system level and systematically comes down to lower level component failures. After discussing FMEA and FTA, we present a

methodology for fault-tree construction. We detail a scheme for coding a fault tree into LISP and present the LISP functions for obtaining the minimal cutsets of a fault tree. Also, we discuss the well-known MOCUS algorithm to derive the minimal cutsets of any fault tree. Finally, we present the binary decision-tree algorithm that is used in this work to compute the reliability of the control system.

3. Reliability analysis of spacecraft control system

Initially, we present the FMEA of the real-life control system together with significant findings of the analysis. Next, we present the exhaustive fault tree for the spacecraft control system. We discuss the methodology adopted in estimating the probability of occurrence of all the basic events of the fault tree. Also, we obtain the list of minimal cutsets using MOCUS and subsequently the probabilities of their occurrence. Finally, we show that the total reliability of the spacecraft control system is 0.9346512, for a mission time of three years. We also list the criticality/importance factors of all the basic and intermediate events in the fault tree.

4. Fault diagnosis of spacecraft control system

We present algorithms for fault diagnosis using fault trees and expert systems. First, we consider the spacecraft attitude and orbit control system and later, a more complex hypothetical example. Initially, we review the existing fault-tree-based fault diagnosis techniques[3,4] and later present an algorithm FTDIGNOSE developed by us for fault diagnosis. Next, we give a brief introduction to expert systems and details of the expert system approach for fault diagnosis. We present the production rules representing knowledge base of the expert system and also the details of the problem-solving strategy adopted by us. For problem solving, we have adopted bidirectional chaining which is a combination of data- and goal-driven strategies. Such a strategy would be valuable in resolving first the critical faults[5].

5. Conclusions

We have presented the results of failure analysis of the spacecraft control system using FMEA and fault trees. We have shown that the spacecraft control system has a reliability of 0.9346512 at the end of three years. Further, we have presented two methods for fault diagnosis, one using fault trees and the other expert systems. These results are immensely useful in the context of Indian space program since real-life data and configurations have been used wherever possible. Implementation of the fault diagnosis and reconfiguration schemes in spacecraft is currently under consideration by several countries. Our results presented here are useful in the design of practical computer control systems for spacecraft.

Our results could be extended in several ways. One can combine the dynamic fault detection schemes available in control theory with those of static fault trees to get more reliable fault detection schemes. Expert system implementations have been largely academic exercises. Building an actual expert system using the heuristic knowledge of various designers/ground control personnel cascaded with algorithmic fault-diagnostic schemes would certainly prove valuable. Also development of efficient and fast search procedures in the context of AOCS would be worthwhile.

References

1. HENLEY, E. J. AND KUMAMOTO, H. *Reliability engineering and risk assessment*, 1981, Prentice-Hall.

2. LAMBERT, H. E. AND YADIGOROGLU, G. Fault trees for diagnosis of system fault conditions, *Nucl. Sci. Engng*, 1977, **62**, 20–34.

3. LEES, F. P. *Computer support for diagnostic tasks in the process industries, human detection and diagnosis of system failure*, 1981, Plenum Press.

4. ALI, M. AND SCHARRHORST, D. A. Sensor-based fault diagnosis in a flight expert system, *2nd Conf. on AI Applications*, IEEE, 1985, pp 49–54.

Thesis Abstract (M.Sc. (Engng))

Measures of depth of collision between convex objects and their efficient computation by K. Sridharan.
Research supervisor: S. Sathiya Keerthi.
Department: Computer Science and Automation.

1. Introduction

In applications such as robot motion planning and VLSI layout, there is a need to describe the spatial position of one object, say A with respect to another, B. The focus to this day has been on the study of proximity relationships when the objects do not intersect and computation of the minimum distance, $d(A, B)$ between them. Especially, when A and B are convex, efficient algorithms have been presented in two domains, *viz.*, computational geometry[1] and mathematical programming[2]. When A and B intersect, $d(A, B) = 0$. The problem of computing depth of collision between two intersecting objects has not received much attention. This research attempts to give various measures to characterise the penetration between intersecting objects and efficient algorithms for their computation. The work concentrates on developing algorithms when A and B are convex polyhedra.

The study of this problem is rewarding from two different angles. First, from a theoretical view point, quest for elegant and efficient algorithms leads to interesting characterisations of the geometric objects involved. Second, computation of depth of collision is a substep in important problems such as collision detection and planning of collision-free motions for robots using penalty functions.

2. Contributions of the work

We have first considered a simplistic way of defining depth of collision as the diameter of the largest sphere inside $A \cap B$. We have provided reasons to show that this measure is inadequate. An improved definition of depth of collision is then presented based on the ideas of Buckley and Leifer[3] and Cameron and Culley[4]. We call this $D_2(A, B)$ and it corresponds to the minimum relative translation of the objects needed to just separate them.

Buckley and Leifer and Cameron and Culley have not been successful in obtaining efficient algorithms for $D_2(A, B)$. We have established several interesting properties of $D_2(A, B)$ and utilized them to come up with an efficient algorithm for $D_2(A, B)$ in two dimensions. Our algorithm is based on a result of Lozano-Perez[5] and runs in $O(m + n)$ time where m and n are the number of vertices in A and B, respectively.

We discuss the extension of the 2-D algorithm to higher dimensions and point to the combinatoric complexity of the extended algorithm.

We then propose a general scheme to compute $D_2(A, B)$ in two and higher dimensions. This is based on the well-known Fourier elimination method for solving a system of linear inequalities. The approach is quite simple but has an exponential time complexity.

Both the methods discussed for the computation of $D_2(A, B)$ do not yield efficient algorithms in three and higher dimensions. This has led us to ask the following question: Are there alternative measures for the depth of collision which have good properties that can be computed efficiently? We have been successful in answering this question.

The motivation behind our definition of the new measures is to use norms other than the Euclidean norm in the definition of depth collision. This results in a sequence of new measures: $D_p(A, B)$, $1 \leqslant p \leqslant \infty$. We have analysed the properties of the new measures and compared them with $D_2(A, B)$.

Efficient algorithms have been presented for computing two of the new measures namely, $D_1(A, B)$ and $D_\infty(A, B)$. If facial representations of A and B are given, $0(a + b)$ time algorithms have been developed, where a and b are the number of faces in A and B, respectively. The algorithms use a linear programming formulation for the problems and use Megiddo's result[6] to establish the time bound. The performance of the algorithms has been tested on the DEC-1090 system and the results are quite good.

Special algorithms have been developed to compute $D_1(A, B)$ and $D_\infty(A, B)$ when A and B are polygons. The algorithms run in $O(\log m \times \log n)$ time.

3. Conclusions

Several extensions to our work are possible. One of the tantalizing open problems is the efficient computation of $D_1(A, B)$ in three and higher dimensions. Another direction is to extend our algorithms to non-convex polyhedra. Further, the computed depth measures can also be appropriately utilized in a scheme for collision-detection and collision-avoidance.

References

1. PREPARATA, F. P. AND SHAMOS, M. I. *Computational geometry*, 1985, Springer Verlag.
2. GILBERT, E. G., JOHNSON, D. W. AND KEERTHI, S. S. A fast procedure for computing the distance between complex objects in three-dimensional space, *IEEE Trans.*, 1988, **RA-4**, 193–203.
3. BUCKLEY, C. E. AND LEIFER, L. J. A proximity metric for continuum path planning, *Proc. Ninth Int. Conf. on AI*, 1985, pp 1096–1102.
4. CAMERON, S. A. AND CULLEY, R. K. Determining the minimum translational distance between two convex polyhedra, *IEEE Int. Conf. on Robotics and Automation*, 1986, pp 591–596.
5. LOZANO-PEREZ, T. Spatial planning: A configuration space approach, *IEEE Trans.*, 1983, **C-32**, 108–120.
6. MEGIDDO, N. Linear programming in linear time when the dimension is fixed, *J. ACM*, 1984, **31**, 114–127.

Thesis Abstract (M.Sc. (Engng))

Algorithms for testing planarity of chordal graphs and hamiltonicity of planar chordal graphs by C. R. Subramanian.
Research supervisor: C. E. Veni Madhavan.
Department: Computer Science and Automation.

1. Introduction

Our work comprises the development of efficient sequential and parallel algorithms for certain graph-theoretic problems. In contrast with many graph-theoretic problems which can be solved by polynomial-time sequential algorithms, there are a large number of graph problems which are known to be NP-complete. Some approaches to coping with NP-complete problems are based on the study of approximation algorithms, probabilistic algorithms, and algorithms for restricted classes of problems. Also with the availability of powerful and cheaper processing elements, it is of interest to design efficient algorithms, which guarantee reasonable speedups, for parallel computers.

2. Contributions of the thesis

In this work, we develop the following results.

1. A characterization of planar chordal graphs.
2. As a consequence, a linear time sequential algorithm and an $O(\log^2 n)$ time n processor CREW parallel algorithm for testing planarity of chordal graphs.
3. A result on the structure of hamiltonian cycles in planar chordal graphs and an $O(n^2)$ time sequential algorithm for testing hamiltonicity of planar chordal graphs.
4. An $O(\log^2 n)$ time, $n^2/\log^2 n$ CREW parallel algorithm for finding the lowest common ancestors (LCA) of pairs of vertices in a rooted directed tree.
5. An improved algorithm for listing complete subgraphs of order k in a graph.

We establish conditions for the existence, in a chordal graph, of subgraphs homeomorphic to complete graphs K_n ($n \geqslant 4$), complete bipartite graphs $K_{m,n}$ ($m,n \geqslant 3$) and wheels W_n($n \geqslant 3$). The proofs of the above results are based on the following result which we prove by induction: If there are m vertex disjoint paths from a vertex d to all the vertices of a complete subgraph $\{a_1, \ldots, a_m\}$ of a chordal graph, then there is a vertex e belonging to one of these paths, which is adjacent to all of $\{a_1, \ldots, a_m\}$. Using the above results we develop the following characterization.

For a chordal graph $G = (V, E)$ with a perfect elimination ordering (PEO) $v_{p(1)}, \ldots, v_{p(n)}$, the following are equivalent:

(i) G is planar,
(ii) G contains no subgraph isomorphic to K_5 or $K_{3,3}$,
(iii) For each i, $i = 1, 2, \ldots, n$,
  (a) $|N(v_{p(i)}) \cap V(G_{n-i})| \leqslant 3$
  (b) If $|N(v_{p(i)}) \cap V(G_{n-i})| = 3$, then there is atmost one vertex in G_{n-i} which is adjacent to all the vertices of $N(v_{p(i)}) \cap V(G_{n-i})$. Here G_i is the subgraph induced by the set $\{v_{p(n-i+1)}, \ldots, v_{p(n)}\}$.

Using this characterization, we develop a new linear-time planarity testing algorithm. This algorithm is conceptually simpler than the standard planarity testing algorithms for arbitrary graphs. The sequential algorithm can be parallelized to yield an $O(\log^2 n)$ time, n processor CREW parallel algorithm. For characterizations of chordal graphs based on PEOs we refer to Golumbic[2]. In a

recent work, Klein and Reif[3] propose an $O(\log^2 n)$ time n processor CRCW parallel algorithm for this problem using a different approach.

We also show, using the results on the existence of homeomorphic subgraphs in a chordal graph, that the fixed subgraph homeomorphism problem can be solved in polynomial time for chordal graphs with the pattern graph being a K_n, $K_{m,n}$ or a wheel W_n. The running time of this algorithm is a polynomial whose degree is equal to the number of vertices in the pattern graph. This is significant from a practical point of view when compared to the approach based on the results of Robertson and Seymour[4] where the degree of the polynomial is an exponential tower on the size of the pattern graph.

We next develop some results on the hamiltonicity of chordal graphs. Using these results, we develop an $O(n^2)$ algorithm for the hamiltonian cycle problem on planar chordal graphs.

We show by exploiting the PEO of a planar chordal graph that we can build inductively a hamiltonian cycle for the graph if one exists, or report nonhamiltonicity otherwise. We show that in chordal graphs, the subgraph H_i induced by $\{v_{p(i)}, \ldots, v_{p(n)}\}$, is hamiltonian if and only if the subgraph H_{i+1} has a hamiltonian cycle in which some neighbouring $v_{p(i)}$ appear consecutively. We then introduce the notions of admissible pairs for every vertex, and admissible hamiltonian cycle. Our inductive procedure is based on these two ideas.

We present new approaches to the two problems: (i) finding LCA of vertex pairs in a rooted directed tree, and (ii) listing certain types of subgraphs of a graph.

The LCA problem has many applications in other problems, such as finding dominators in a directed rooted acyclic graph and updating minimum spanning trees. We present a new CREW parallel algorithm for the above problem which takes $O(\log^2 n)$ time using $n/\log^2 n$ processors. Our algorithm is based on a simple recursive strategy. We analyse the time and processor complexities, establish a bound on the number of processors, and show that the algorithm is optimal, *i.e.*, the product of time and processor complexity is $O(n^2)$. There are several other algorithms for this problem, a recent one being that of Tsin[5].

The problem of listing certain kinds of subgraphs of a graph arises in many applications. Chiba and Nishizeki[1] give algorithms for listing triangles, quadrangles and complete subgraphs of order k. Their algorithm takes $O(k.a(G)^{k-2}m)$ time where $a(G)$ is the arboricity of the graph G. It is based on the strategy of visiting the vertices in a non-increasing order of their degrees.

We present another strategy based on the smallest last ordering of vertices. Associated with each graph G is a parameter $\delta(G)$, defined by $\delta(G) = \max\{\delta(H) : H \subseteq G\}$, where $\delta(H)$ is the minimum degree of H. We establish that $\delta(G)$ and $a(G)$ are related by $a(G) \leqslant \delta(G) \leqslant 2 \cdot a(G) - 1$. Using this bound and the strategy based on the smallest last ordering of the vertices, we obtain an algorithm for listing all complete subgraphs of order k in $O(k(\delta(G)/2)^{k-2}m)$ time. Our algorithm is thus as efficient as the algorithm of Chiba and Nishizeki[1], in general. However, when $\delta(G) \simeq a(G)$, our algorithm is faster by a factor of 2^{k-2}.

References

1. CHIBA, N. AND NISHIZEKI, T. — Arboricity and subgraph listing algorithms, *SIAM J. Computing*, 1985, **14**, 210–223.

2. GOLUMBIC, M. C. — *Algorithmic graph theory and perfect graphs*, 1980, Academic Press.

3. KLEIN, P. N. AND REIF, J. H. — Efficient parallel planarity testing, *IEEE Conf. on Foundations of Computer Science*, 1986, pp 465–477.

4. ROBERTSON, N. AND SEYMOUR, P. D. Disjoint paths – a survey, *SIAM J. Algebraic Discrete Meth.*, 1985, **6**, 300–305.

5. TSIN, Y. H. Finding lowest common ancestor in parallel, *IEEE Trans.*, 1986, **C-35**, 764–769.

Thesis Abstract (M.Sc. (Engng))

Intelligent backtracking in logic programming by V. Rajasekar.
Research supervisor: M. Narasimha Murty.
Department: Computer Science and Automation.

1. Introduction

Logic programming is widely used because of its declarative nature. However, the naive execution models for logic programs are very inefficient due to the non-determinism involved. Several models that perform intelligent backtracking to improve efficiency are proposed in literature. The method proposed here has a new approach. The information required to determine the backtrack literal is collected during the process of unification. This helps in determining better backtrack literals. To do this a new type of unification, which is a slight variant of the normal left-to-right unification, is defined.

Most execution models for logic programs adopt depth-first search strategy as it requires less overhead than the breadth-first search strategy. Prolog is a logic programming language that uses the depth-first search strategy. However, the depth-first search strategy has a major disadvantage—it may not be able to determine those solutions that are to the right of an infinite branch of the search tree. Hence, the completeness proofs are with respect to that of Prolog.

2. Motivation

Although various intelligent backtracking mechanisms eliminate a lot of redundancies, still some more are left. The following example will elucidate this point.

Example 1: Consider the following program:

$P_0(X, Y, Z, W, K)$:- $P_1(Z)$, $P_2(Y, X)$, $P_3(W, Z)$, $P_4(K)$, $P_5(X, Y, Z, W)$.
$P_1(z1)$:-.
$P_2(y2, x1)$:-.
$P_2(y1, x1)$:-.
$P_3(w1, z1)$:-.
$P_3(w2, z1)$:-.
⋮
$P_3(wn, z1)$:-.
$P_4(k1)$:-.
$P_4(k2)$:-.
⋮
$P_4(km)$:-.
$P_5(x1, y1, z1, w1)$:-.
$P_5(x2, y2, z2, w2)$:-.

When P_5 fails, the substitution would be $\{Z/z1, X/x1, Y/y2, W/w1, K/k1\}$. Since the B-List[3] of the

failed literal P_5 contains (P_1, P_2, P_3) the backtrack literal determined by Vipin Kumar's method would be P_3. In the variable-based intelligent backtracking method[4,5], the most recently instantiated variable is W, instantiated at P_3 and hence the backtrack literal determined is also P_3. However, the overheads required for the variable-based backtracking are lesser compared to that of Vipin Kumar's method. Note that both these methods do not reevaluate P_4 and hence eliminate some redundant executions.

The system backtracks to P_3, reevaluates it and proceeds further. P_5 fails again and the process repeats itself until P_3 also fails and the backtracking is done to P_2. It is possible to capture this information before hand and backtrack directly to P_2. This would eliminate a lot of redundant executions.

3. Definitions

Definitions of failure list (F-list), step number of a variable, ordering of variables, failure due to constraints, and instantiated order unification, can be found in Rajasekar and Narasimha Murty[5] and Rajasekar[6].

4. Data-based intelligent backtracking: Method

Consider example 1. When P_5 fails, the backtrack literal to be selected should be P_2. When the variable Z is bound to $z1$ by P_1 and when Y is bound to $y2$ by P_2, it can be seen that P_5 can never succeed, immaterial of the values assumed by the variables W and K. This is possible because the set of candidate clauses of P_5 does not have any clause of the form $P_5(\_, y2, z1, \_)$. In the presence of such a clause, the backtracking mechanism should select P_3 only. Otherwise, there is a possibility of missing some solution. The idea is to capture such information and backtrack in an intelligent fashion. To perform this, we introduce the concept of *status flags*.

Each variable associated with a subgoal contains a flag. This is used to indicate whether the unification was successful or not. Initially, all the variables of a literal are set to false. Now, the subgoal is unified with a set of candidate clauses. The instantiated-order unification is adopted and a flag is set to true if any of the corresponding term unification succeeds. Otherwise, the status remains false. For example, let the subgoal be $P(X, Y, Z, W)$. Let the set of candidate clauses be $P(x1, y1, z1, w1)$ and $P(x2, y2, z2, w2)$. Let the substitution be $X/x1$, $Y/y2$, $Z/z1$, $W/w1$ and the ordering be $Z \leqslant X \leqslant Y \leqslant W$. Initially, all status flags are set to false. Hence, the status vector for the above subgoal would be (F, F, F, F). On unifying with the first candidate clause, the resulting status would be (T, F, T, F). Now the second clause is considered for unification with the status (T, F, T, F). Since the first term $z1$, $z2$ does not unify, there is no change to the status flags. Now the possible failed variables would be those variables whose status flags are set to true and the first instantiated variable whose status is false. All the possibly failed variables are added to a list called the failure list (F-list). The most recently instantiated variable is chosen and the step where this variable got instantiated is chosen as the backtrack step and the subgoal evaluated at this step is chosen for reevaluation. In example 1, when P_5 fails, the status would be (T, F, T, F). Hence, the variables added to the F-list are Z, X, Y. The most recently instantiated variable is X, Y. Hence the backtrack step is the step where the variables X and Y got instantiated. Hence P_2 is reevaluated. Note that P_3 is not reevaluated, thereby saving a lot of redundant executions.

5. Conclusions

The model proposed eliminates a lot of redundant executions. Also, another type of redundancy elimination, called forward redundancy elimination, has been discussed in detail in the thesis[6].

Completeness proofs have been presented for these. It has also been shown by examples that the proposed method can eliminate a lot of redundant executions in an AND-Parallel execution model.

References

1. PEREIRA, L. M. AND PORTO, A. — Selective backtracking for logic programs, *5th Conf. on Automated Deduction, Lecture Notes in Computer Science* #87, Les Arcs, France, 1980.

2. CHANG, J. H. AND DESPAIN, A M. — Semi-intelligent backtracking of PROLOG based on a static data dependency analysis, *Proc. IEEE Symp. on Logic Programming*, Aug. 1985

3. VIPIN KUMAR AND LIN, Y.-J. — A framework for intelligent backtracking in logic programs, *Proc. Sixth Conf. on Foundations of Software Technology and Theoretical Computer Science*, New Delhi, India, Springer-Verlag, *Lecture Notes* #241, Dec 1986.

4. RAJASEKAR, V. AND NARASIMHA MURTY, M. — Variable-based intelligent backtracking, *Proc. Applications of Artificial Intelligence– VII Conference*, Florida, USA, March 28–30, 1989.

5 RAJASEKAR, V. AND NARASIMHA MURTY, M. — *Efficient execution models for logic programs*, Technical Report IISc-7CSA-89-15, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India, Mar. 1989.

6. RAJASEKAR, V — *Intelligent backtracking in logic programming*, M.Sc. (Engng) Thesis, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India, Dec. 1989.

Thesis Abstract (M.Sc. (Engng))

Algorithmic knowledge for a knowledge-based clustering environment by Atul Negi.
Research supervisor: M. Narasimha Murty.
Department: Computer Science and Automation.

1. Introduction

Clustering is a process of partitioning a given set of objects into meaningful groups. It has numerous applications in different fields like image processing, numerical taxonomy, etc. Often, the user's dilemma[1] is in choosing a relevant clustering technique, from the numerous existing, for the analysis of the data at hand. The choice is difficult because of the presence of potentially large number of uncertain factors that affect the process of clustering. Amongst the many factors of clustering algorithms, whose knowledge helps in producing a meaningful classification, we include foremost: i) Time and space complexity, ii) Versatility, iii) Contextual and conceptual knowledge. The investigations in this work stress unifying abstractions of clustering knowledge such as problem-solving paradigm (divide-and-conquer[2]), data structures common to algorithms (k-d tree)[3] and cluster property concepts (admissibility criteria). Due to the computation-intensive nature of clustering, more attention was directed towards algorithmic efficiency, for better utilization of computational resources.

2. Results and discussion

The prime motivation behind this work, apart from the user's dilemma, was to find a greater scope of application of the numerous hierarchical and non-hierarchical algorithms for clustering[4], by making

use of the three above-mentioned factors. The aim was also to find a concise and functional representative formalism for the domain.

A divide-and-conquer method for clustering, which follows the 'problem reduction strategy' of problem solving, was proposed. *k*-group admissibility of the method was formally established. Experimentation results validated both the formal claims of computational efficiency and versatility of classification. The *k-d* tree data structure for nearest-neighbor (NN) computation was shown to be a common factor in the efficient implementation of a number of clustering algorithms. A clustering algorithm using 'biased NN search' was proposed which employs the *k-d* tree structure. Various concepts necessary for clustering and the supporting activity of finding structure in the data were examined. An object-oriented knowledge representation scheme was proposed here for the representation of various hierarchical, non-hierarchical and NN-computation-based clustering algorithms.

To keep the analyst in active control of clustering process while allowing the benefit of the above-mentioned schemes for clustering and organization of clustering knowledge, it was thought to integrate these under a knowledge-based environment for clustering.

References

1. DUBES, R. AND JAIN, A. K. Clustering techniques: The user's dilemma, *Pattern Recognition*, 1976, **8**, 247–260.
2. HOROWITZ, E. AND SAHNI, S. *Fundamentals of computer algorithms*, 1984, Computer Science Press.
3. FRIEDMAN, J. H., BENTLEY, J. L. AND FINKEL, R. A. An algorithm for finding best matches in logarithmic expected time, *ACM Trans. Math. Software*, 1977, 3, 209–226.
4. ANDERBERG, M. R. *Cluster analysis for applications*, 1973, Academic Press.

Thesis Abstract (M.Sc. (Engng))

Hybrid knowledge-based system for process plant fault diagnosis by Saugata Pramanik.
Research supervisors: P. Venkataram and V. Rajaraman.
Department: Electrical Communication Engineering.

1. Introduction

Knowledge-based system (KBS) appears to be the most appropriate approach for effectively aiding the operator in arriving at a quick and accurate diagnosis in plant operations. Most of the current KBSs[1] in process plants are built on expert knowledge compiled in the form of production rules. These systems lack flexibility due to their process-specific nature and are unreliable when faced with unanticipated faults. However, attempts[2,3] have been made to integrate knowledge based on experience and 'deep' process knowledge to make the diagnostic system flexible and capable to tackle novel circumstances.

In this work, we propose a hybrid[4] knowledge framework which includes a process-independent diagnostic mechanism (DM) based on causal and qualitative reasoning, and a hybrid knowledge base which integrates information based on experience and 'deep' knowledge of the process plant. In this effort, we itemized process-specific and process-common knowledge to create a separate diagnostic strategy to deal with each of them. This framework is flexible and allows a unified design methodology for fault diagnosis of process plants.

2. Hybrid knowledge organization

The entire process knowledge may be thought of as collection of knowledge 'chunks' corresponding to functional subsystems or functional blocks (FBs), for instance, boiler system, condenser system, water-regeneration system and electrical auxiliaries system that typically occur in power plants. The knowledge is hybrid as it involves: (i) Experiential and 'deep' knowledge of the process, (ii) Process-common and process-specific knowledge.

Experiential knowledge captures empirical associations and expertise for process diagnosis. 'Deep' knowledge consists of structural knowledge that captures the physical layout (components' description and connectivity), and behavioral knowledge that captures the steady-state model (causal interactions among parameters) of the process. The process-specific knowledge includes experiential knowledge about common faults, behavioral and structural knowledge of the process. The process-common knowledge comprises fault models (FMs) for various types of components (*viz.*, reactors, tanks, pumps streams, heat-exchangers, valves, etc.) commonly occurring in any process plant.

The process behavioral knowledge is qualitatively represented in the form of signed directed graph (SDG) which is then converted into a set of rules (SDG-rules[5]) added with control premises for the purpose of diagnostic reasoning. Frames are used to represent structural knowledge, while rules are used to capture experiential knowledge about common faults. (Figure 1(a–e) illustrates the hybrid knowledge representation.) An interface program, *viz.*, knowledge acquisition interface (KAI) was

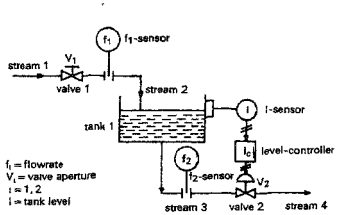

FIG. 1a. A buffer tank system (BFT).

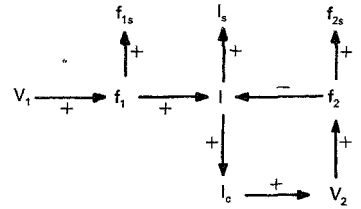

FIG. 1b. Signed directed graph (SDG) of the BFT. *Note*: Sensors are represented by separate nodes indicated by subscript 's'.

```
var(f1,X1):- dat(f1s,X1), X1<>0, tl(f1);
             test (f1), loop(v1), var(v1,X1).
var (f1,0):- el.(f1)

var(l,X2):- dat(ls,X2), X2<>0, tl(l);
            test(l), loop (f1), var(f1,X2);
            test(l), loop(f2), var(f2,Y2), X2 = Y2.
var(l,0):- el(l).

var (lc,X3):- loop(l), var(l,X3).

var(v2,X4):- loop(lc), var(lc,X4).

var(f2,X%):- dat(f2s,X5), X5<>0,tl(f2);
             test(f2), loop(v2), var(v2,X5).
var (f2,0):- el(f2).
```

FIG. 1c. SDG-rules corresponding to the SDG of Fig. 1b.

Frame type	Value
Attribute	*Value*
Name	: tank1
Inlet	: stream2
Outlet	: stream3
Level	: l
Temperature	: t1
Concentration	: c1
pH	: ph1

```
tank(tank1,
     inlet([stream2]),
     outlet([stream3]),
     level(l),
     temperature(t1),
     concentration(c1),
     ph(ph1).
```

FIG. 1d. Frame representation of tank 1 and its Prolog assertion.

```
IF tank1 ⟨level⟩ is "low" AND valve 1⟨state⟩ is "open"
THEN valve2 ⟨state⟩ is "stuck-closed".

rule(valve2, "stuck-closed",
        parameter_condition([dat(level, −1)]),
        component_condition([scomp(valve1, "open")])).
```

FIG. 1(e). A heuristic rule and its Prolog assertion.

created to acquire and convert (i) behavioral knowledge into a set of rules, and (ii) structural knowledge and experience-based heuristics into a set of facts.

3. Diagnostic mechanism (DM)

DM is based on a steady-state model of the process and is composed of three consecutive phases for locating a fault (Fig. 2). The first phase is malfunction block identification (MBI) which locates

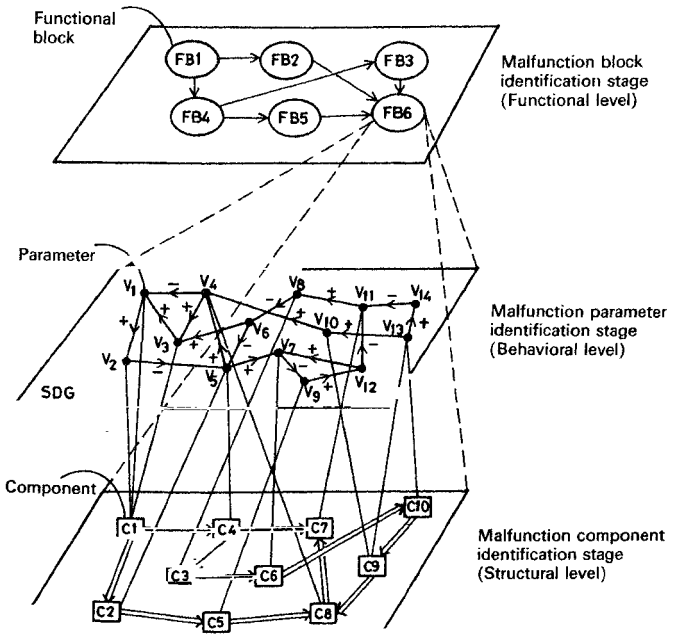

FIG. 2. Overview of the diagnostic mechanism (DM).

malfunctioning subsystem or malfunction block (MB) that is responsible for causing the process malfunction. It is based on alarm data whenever violation of process parameters occurs. Once the suspected MB is identified, the second phase, *viz.*, malfunction parameter identification (MPI) is invoked to locate the parameters which indicate the prime cause(s) of the fault in it. This is achieved by the MPI algorithm which correlates symptoms (abnormalities) in the process with the help of SDG-rules that are evaluated using the process data. As a result, all side effects are eliminated from further investigation and a reduced set of symptoms (malfunction parameters) is obtained. Finally, malfunction component identification (MCI)[6] phase uses the malfunction parameters to locate the faulty component. Initially, 'shallow' reasoning based on experiential knowledge is invoked. If it fails, DM calls forth FMs in conjunction with structural knowledge to find the malfunctioning component.

4. Implementation of methodology

We implemented the prototype system on an IBM PC-XT using Turbo Prolog programming environment which facilitated fast development and required less diagnostic time. The system has been tested with several simulated subsystems of process plants, the details of which are available in the original dissertation.

5. Conclusions

Hybrid knowledge increases diagnostic efficiency and enhances reasoning from first principles in unanticipated situations. DM is process-independent and, therefore, is capable of adapting to various types of plant configurations. Since, hybrid knowledge base and DM are separate, modification of either of them can be done independently. DM also provides explanation facility for justifying the line of diagnostic reasoning to the human operator. Finally, the methodology addressed in this work provides a groundwork to troubleshoot analog systems such as circuit boards, communication networks, and electrical power systems.

References

1. ROWAN, D. A. — *Chemical plant fault diagnosis using expert system technology: A case study*, IFAC, Kyoto, Japan, Sept. 28–Oct. 1, 1986.
2. FINK, P. K. AND LUSTH, J. C. — Expert system and diagnostic expertise in the mechanical and electrical domains, *IEEE Trans.*, 1987, **SMC-17**, 340–349.
3. RICH, S. H. AND VENKATASUBRAMANIAN, V. — An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis, *Computers Chem. Engng*, 1988, **12**, 903–921.
4. PRAMANIK, S. AND VENKATARAM, P. — A hybrid knowledge structure for process plant fault diagnosis, *Proc. 4th. Natn Convention of Computer Engineers*, Institute of Engineers (India), Calcutta, Dec. 1988.
5. PRAMANIK, S. AND VENKATARAM, P. — Diagnosis of steady-state process behavior using rules derived from signed directed graph, *J. Indian Inst. Sci.*, 1990, **70**, 269–281.
6. PRAMANIK, S. AND VENKATARAM, P. — A knowledge-based approach to the problem of locating the malfunction component in a process plant, *Computers Ind. Engng* (communicated).