J Indian Inst. Sci., Mar.–Apr. 1991, 71, 171–194.

An annotated bibliography of research in parallel computing at the Indian Institute of Science, 1985–1991

S. K. GHOSHAL AND V. RAJARAMAN
Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560012, India.

Received on April 4, 1991.

Abstract

This paper briefly describes research activities in parallel computing conducted at the Indian Institute of Science between 1985 and 1991. The activities include parallel computer architecture, parallel programming languages, parallel algorithms and task scheduling in multiprocessors. There have been ninety publications in this area which are cited in this paper with small annotations giving the gist of each article.

Key words: Parallel computing, annotated bibliography, parallel computer architecture, parallel programming language, parallel algorithms, task scheduling in multiprocessors.

1. Introduction

Parallel computing activities at the Indian Institute of Science have been pursued by many research groups spread over a number of departments. The research has led to a large number of Ph.D. and Master of Science theses and published papers in many refereed journals. The primary purpose of this article is to classify the publications into various fields and briefly state the main theme of the papers. The research activities may be classified into a number of broad areas. In the following sections, we specify these areas and subareas. The published papers are arranged using this classification. The main theme of each paper is stated after the cited reference. We have included articles published in journals and those presented at conferences, and have excluded technical reports as most of them eventually get published. While every attempt has been made to contact individuals working in parallel computing at the Institute and obtain their publications we may have missed some due to ignorance or non-accessibility of the papers. We trust that we will be excused.

2. Parallel computer architecture[1–50]

Many groups at the Institute have been proposing architectures for parallel machines and building prototype hardware. The prototypes have been used as a test bed to develop

operating systems, parallel compilers, parallel programming environments and application programs. Even though commercial parallel machines are widely available in the West, the situation is quite different in India. Commercial machines have not been available and there was a motivation to build machines and eventually transfer the technology to the industry. Besides, test beds are necessary to try out various programming ideas on actual hardware rather than on idealised simulators. The prototype machines constructed fall into two categories: Shared memory parallel computers and message-passing parallel computers. Computers named MULTIMICRO[1] and SHAMP[2] fall in the first category. MMS[3], BBS[5], hypercubes based on Intel microprocessors[6] and transputer-based machines[10] fall in the second category. Some architectures are also based on the implementation of an appropriate specially designed highly parallel hardware algorithm as a board-level back-end processor to a popular and widely used host machine such as the IBM PC/AT[12]. An SIMD machine called CLAP-4, whose processing elements are capable of executing *Boolean, simple near-neighbor* and *recursive near-neighbor* instructions[13] has also been designed and built at IISc for image-processing applications.

Besides the general-purpose computers, research has also been conducted in designing parallel machines for special purposes such as logic programming[14-16], executing programs written in the paradigm of communicating sequential processes[17] and solving ordinary differential equations[19]. These papers[14-19] have been put in the class—special-purpose parallel computers.

Other researchers examine computer architectures (*e.g.*, Biswas *et al*[20]) which use concepts that are expensive or difficult to implement with current state-of-the-art in technology. In such cases simulation is used as a tool. The proposed machine is simulated and its performance evaluated. Excellent simulation strategies[27] are formulated and general-purpose tools crafted[28] to meet these goals. Systolic architectures[29-35] and dataflow[36] architecture machines fall into this category. We have listed papers in these and related areas (*e.g.*, simulating a system of neurons[37] or multiprocessors for high-speed numerical computations[41]) under Architectural proposals—Simulation and analysis[20-41].

Analysing the performance of parallel computers by using idealized models and using these as guidelines to design actual systems is an interesting research area[42-47]. After an actual multiprocessor is constructed in hardware, it is benchmarked on a variety of application problems to evaluate and suggest ways of improving it[50]. We include a separate section listing papers in this area[42-50].

3. Languages for parallel computers[51-60]

Most parallel computers are designed using von-Neumann architecture processing elements which are sequential processors. Programs are written in one of the well-known programming languages such as Fortran or Pascal designed for sequential computers. There are ways to extract parallelism from these programs[51]. Sometimes, new constructs are added to these languages to enable multiple processes to be spawned on different processors, *e.g.*, as in Shah and Patnaik[28]. Novel paradigms of concurrent computation like Linda[58] are implemented

on practical message-passing architectures for extending conventional programming languages like C for parallel programming. The process of compilation itself can be speeded up too by using parallel processing techniques[52-54]. Non-von-Neumann architectures[59] and axiomatization of distributed processes[60] result in new languages too. Issues in designing parallel languages and compilers for them are an important research area[51-60].

4. Algorithms for parallel computers[61-79]

Over the years, the design of algorithms for computers has assumed great importance. With the advent of parallel computers, it is often found that algorithms which are designed for sequential single processor computers are not appropriate for parallel computers. Appropriate abstractions for the problem-solving strategy are required at times[61]. Thus, a large number of researchers are now examining algorithms appropriate for parallel computers. The algorithms should be suitable to be mapped on to multiple processors. For an example, look at an algorithm for parallel hypertext retrieval[62]. There is also a close relationship between an architecture and an algorithm which is appropriate and efficiently maps on to the architecture.

Neural networks may be thought of as a densely interconnected set of tiny processing elements. These are massively parallel systems. These networks mimic (albeit in a gross way) the dense interconnection of neurons in a human brain and have fascinated researchers. Many researchers have been attempting to design algorithms to solve problems using neural nets[80-83]. Such algorithms 'train' the network to recognize patterns and solve complex optimization problems which are intractable on sequential machines. Research in this area at the Institute is at an early stage. Papers in this area[80-83] are also listed in the algorithms section.

5. Task scheduling in parallel computers[84-88]

Many algorithms may be modelled as directed graphs in which nodes represent computations (or tasks) and arcs data exchanged between tasks. An interesting research problem is mapping such a task graph on to a parallel computer with a specified interconnection between processors. No optimal solution exists for this problem and many heuristics are evolved[84,85]. The heuristics differ between architectures.

As may be seen from the above survey a large variety of problems have been examined by researchers at the Institute. There is a growing body of literature and the bibliography is a reasonable representation of the work done in this subject[84-88].

6. Acknowledgements

We would like to thank all the authors whose papers are cited here for generously helping us by lending their papers and for personal discussions.

7. Annotated bibliography

7.1. *Parallel computers built in hardware*

1. SRIDHAR, M. K., SRINATH, R. and SURESH, K., Parallel numerical computations: the case for MIMD architectures, *Proceedings of the CSI Workshop on Issues in Parallel Computing*, January 1989. This paper elaborates the design philosophy of the MULTIMICRO parallel computer. Details of the system architecture are given along with the application programming methodology. The software environment provided by the MMk kernel is described. MULTIMICRO is a multibus II-based shared memory architecture. It has eight 80386/387-based processing elements. Each processing element has 2–4 MB local memory. The size of the shared memory is 4–8 MB. The architecture supports fork-join parallel programming model for Fortran-77 and C. The MULTIMICRO parallel processor uses an indigeneously developed parallel operating systems kernel called MMk and provides an efficient message-passing system for interprocessor communication and synchronization, while providing for a uniform shared memory address space. The MMk kernel has since been transferred for commercial exploitation to M/s Wipro Information Technology Ltd, Bangalore, by the Department of Electronics, Government of India, the sponsors of the project. Some of the applications that have been successfully ported on to the MULTIMICRO include power systems analysis, physics, molecular dynamics and thermal modelling of satellites, etc. Although the machine was designed for specific applications in power systems simulation, a large class of numeric applications have been successfully ported on to the MULTIMICRO.

2. GHOSAL, D. and PATNAIK, L. M., SHAMP: An experimental shared memory multimicroprocessor system for performance evaluation of parallel algorithms, *Microprocessing and Microprogramming*, 1987, **19**, 179–192. The design and implementation of a shared bus shared memory multiprocessor system designed and built for graphics applications is given in this paper. It has five Intel single-board computers based on 8086. An Intel microcomputer development system was used for performing input/output operations. Multibus is used for interconnecting these. An efficient bus-arbitration algorithm was devised for the system and implemented in hardware. The system has both global memory and local dual-ported memory. An interprocessor communication mechanism is implemented through asynchronous hardware interrupts. A multiprocessor operating system is designed as a supervisor for process management, I/O management and message interpretation. The on-board timers on the single board computer were adapted to provide accurate timings of different events so that an appropriate performance monitoring can be done. Execution time, idle time, active time and synchronization time were measured for various graphics applications.

3. MOONA, R. and RAJARAMAN, V., Multidimensional multilink multicomputer: A general-purpose parallel computer, *This Journal, This Issue*. MMS stands for multidimensional-multilink multicomputing system. It is a computer architecture built with IBM PC/XT- or PC/AT386-based motherboards interconnected using message-passing FIFO links. MMS can be realized using PC motherboards stacked in a specially

designed cabinet. It can alternatively be implemented on a given set of full-fledged PCs located within a short geographical distance, with the PCs retaining their functionality as uniprocessors. The price of this conversion is about Rs. 3,000 per processing element and involves the plugging in of the communication interface card developed at IISc. The components used in its hardware, except the first-in first-out (FIFO) memory chip which is imported, are available locally. The intercomputer communication link can transfer data between processors at a rate which is faster than the rate at which a processor can interchange data with its own primary storage. The link also supports the primitives of blocked message-passing communication with hardware aids for automatic synchronization. It supports multicasting in hardware. MMS offers a rich interconnection topology among the processing elements. Various configurations of the MMS are possible and have been implemented. Programs can be developed on the MMS architecture so that they can optimally utilize all the processing elements on any MMS configuration for many classes of application problems. The paradigm of representation and execution of parallel programs in the MMS architecture is an extension of Hoare's CSP where the sender process does not have to wait for the receiver to be ready. MMS was first implemented using MS-DOS as the base operating system. Under MS-DOS the following languages were provided with user-callable libraries for parallel processing: Pascal, Fortran, Cobol, Prolog and C. Recently, Unix has been implemented on the MMS. C is available on the Unix system for user programming. A Fortran will be implemented soon. Our own Pascal compiler-generating native code for the MMS will be ported to this system shortly. A number of routines for numerical integration, curve-fitting and solution of linear programming problems have been implemented on various configurations of the MMS architecture with satisfactory speedups.

4. GHOSHAL, S. K., GUHA, S., ARIFF, S. M. and RAJARAMAN, V., Simple low-cost multiprocessor based on message-passing FIFO links, *Microprocessors and Microsystems*, 1990, **14**, 297–300. The motivation for using IBM PC motherboards as processing elements and byte-wide data paths for interprocessor communication links are elaborated. The design and implementation of the FIFO links used in MMS are given. A few topologies that the MMS can embed are discussed. Application programming methodologies are elaborated along with a description of the Pascal-callable firmware library for parallel programming. The paper concludes by pointing out a few bottlenecks in the system architecture and suggests steps for alleviating them.

5. KUMAR, M. S. S., SRINIVASAN, M. P. and RAJARAMAN, V., A flexible test bed for multi-microprocessor system studies, *Proceedings of the CSI Workshop on Issues in Parallel Computing*, January 1989. Current trends in computer architecture design indicate that distributed memory computer systems will play a vital role in satisfying the ever-increasing computing needs at a low cost. Multicomputer systems are distributed memory computer systems designed using off-the-shelf components and readily available general-purpose single-board computers. The crux of the design is in obtaining a good processor interconnection scheme to reflect the needs of the targetted multicomputer applications. Message-passing architectures are extensively used in exploiting *coarse-grain* parallelism in a wide range of applications. The BBMS consists

of a general-purpose front-end computer system that caters to all the general programming requirements of the user and a back-end consisting of computing elements (CEs) interconnected with each other and the host through the *message broadcast bus* (MBB). This is a multicomputer system with eight homogeneous processors interconnected through a high-speed time-shared bus. Each computer has a 32-bit CPU with 2MB RAM and some peripherals. The bus architecture is such that it reduces the communication overhead and speeds up the performance in a communication-intensive environment. Simulation results confirm that at even in fine-grain environment it performs efficiently. The system is more suitable for solving problems described in the form of task graph. Communication and arbitration overlaps the CPU execution. Hence, the communication overheads are minimum. The system can operate both in MSDOS and UNIX environments. Extensions are provided to C language to enable users to write application programs. An extensive instrumentation is built to make measurements under execution condition without any overhead. This machine can be used to test the performance of algorithms, to detect hot spots in communication medium, to validate simulated results, etc.

6. DAS, S. R., VAIDYA, N. H. and PATNAIK, L. M., Design and implementation of a hypercube multiprocessor, *Microprocessors and Microsystems*, 1990, **14**, 101–105. A low-cost experimental eight-processor hypercube that can be used as a test bed for investigating the performance of various parallel algorithms is described in this paper along with its software communication kernel. The format of the message packet is described along with the experimental configuration of the hypercube. A dual-ported random access memory chip is used to implement an interprocessor communication mechanism that may be viewed as a mailbox. The implementation of the hypercube host is described. Various programming methodologies usable on the architecture are discussed.

7. JAGADISH, N., MOHAN KUMAR, J. and PATNAIK, L. M., An efficient scheme for interprocessor communication using dual-ported RAMs, *IEEE Micro*, 1989, **10**, 10–19. The hardware implementation of an efficient scheme for interprocessor communication using dual-ported RAMs is described in this paper. This scheme is what has been used by Das *et al*[6]. A three-dimensional hypercube along with the network controller is described, and the details of the dual-ported RAM chip are given. The memory map of each processing element is described, along with the block diagram of a node-processor hardware. A circuit diagram of the wait-state generator is provided and the message-transfer protocol is given. How the scheme can be extended to a 64-node configuration is described. The advantages of the scheme over others employed on practical machines like the Intel hypercube are discussed.

8. MOHAN KUMAR, J. and PATNAIK, L. M., Extended hypercube: A hierarchical network of hypercubes, *to appear in IEEE Transactions on Parallel and Distributed Systems*. The paper presents an introduction to the topology of the extended hypercube and analyses its architectural potential in terms of message routing and execution efficiency of a class of highly parallel algorithms. Topological properties and performance in terms of computation/communication ratio are also discussed.

9. MOHAN KUMAR, J. and PATNAIK, L. M., Fault-tolerant message routing and error detection schemes for the extended hypercube, *to appear in Proceedings ICPP-91*. This paper discusses fault-tolerance aspects of the extended hypercube architecture. The use of network controllers for error detection and fault-tolerant message routing is discussed. Error detection algorithms which run concurrently with typical application examples, *viz.*, matrix multiplication and multinode broadcast have also been discussed.

10. MOHAN KUMAR, J., PATNAIK, L. M. and PRASAD, D. K., Transputer-based extended hypercube, *Microprocessing and Microprogramming*, 1990, **29**, 225–236. An extended version of the above hypercube architecture, its transputer-based implementation and performance studies are reported in this paper.

11. MOHAN KUMAR, J. and PATNAIK, L. M., Performance studies of a transputer-based extended hypercube, *This Journal, This Issue*. In this paper, implementation of an extended hypercube (EH) on a multitransputer system is discussed. A comparative study of the performance of EH and the hypercube in executing communication-intensive tasks is carried out.

12. RAMANI, K., SRIKANTA, S., VENKATESH, Y. V. and RAYMOND, J. W., A parallel fast convolver for computer vision, *This Journal, This Issue*. The design and fabrication of an add-on board to an IBM PC/AT that performs the two-dimensional convolution of images is described here. The mathematical preliminaries are given. A comprehensive amount of literature survey on this topic is presented in this context. A hardware algorithm for convolution is developed which uses a *Product look-up table* and computes partial products with both row and column offsets. This scheme is eminently suitable for parallel implementation. The board-level architecture of the system is described thereafter. An Intel MCS-48 single-chip microprocessor coordinates the internal operations and communications with the host which is an IBM PC/AT. The convolver is compatible with the I/O expansion slot of the PC/AT. It has an on-board input buffer of size $256 \times 256 \times 8$ bits and an on-board output buffer of $256 \times 256 \times 16$ bits. Address generation for these buffers and the look-up table is done by counters. There are opcodes for different operations of the convolver which can be downloaded to the MCS-48 from the host. After the operation is completed, a return code is sent back to the host. Coordination between the host and the convolver is done by mutual interrupts. The paper concludes with reports on the status of high-level language support for the convolver among other activities. How the convolver board can be used for performing morphological operations such as erosion and dilation is pointed out.

13. MUKHERJEE, A. and VENKATESH, Y. V., Implementation of a prototype cellular logic array processor, *Defence Science Journal*, 1985, **35**, 353–359. Interactive processing of image data cannot be efficiently done using conventional computers. This paper describes a prototype array processor CLAP-4 that was implemented as an eight-by-four rectangular grid that runs as an attached processor to an HP 1000 minicomputer. Each processing element has a four-bit ALU and operand and result registers. A processing element can do *Boolean, simple near-neighborhood* and *recursive near-neighborhood* operations. There is a control unit that performs data transfer between the host and the array processor, command word decoding, detection of errors in processors and

other operations. Different implementation details of CLAP-4 are given. Future directions of work are pointed out at the end.

7.2. *Special-purpose parallel computers*

14. SASTRY, A. V. S. and PATNAIK, L. M., OR-Parallel execution of logic programs on modified Manchester dataflow architectures, *Proceedings of the SEARCC*, 1988, Tata McGraw Hill Publishing Company Ltd.

15. SASTRY, A. V. S. and PATNAIK, L. M., A dataflow architecture for OR-parallel execution of logic programs, *Proceedings of the ICPP*, 1988.

16. SASTRY, A. V. S. and PATNAIK, L. M., OR-Parallel execution of logic programs on a multi-ring dataflow machine, *New Generation Computing*, 1991, **9**(2). Studies on the OR-parallel execution of logic programs on extensions of the Manchester machine have been made in these three papers. A new type of a functional unit, called the *definition search unit*, has been incorporated. It stores all the definitions of a program. When a goal arrives, this unit spawns the execution of all the clauses that are there in the definition of the goal and thus OR-parallelism is exploited. One variant of the extension interconnects multiple matching units, multiple processors and multiple memory modules through a shared bus. The other architecture employs a multistage interconnection network to interconnect three types of rings called *processor ring, memory ring* and *definition ring*. One starts from the basic premise of the computation model of logic programs on dataflow systems and then shows how to handle multiple binding environments efficiently. Details of the simulation are given. Speedup is plotted against the number of processors employed.

17. RAVIKUMAR, C. P. and PATNAIK, L. M., An architecture for CSP and its simulation, *Proceedings of the International Conference on Parallel Processing*, 1987, pp 874–881. The simulation of an architecture for efficient execution of CSP programs is presented here. The requirements of a programming language for denoting distributed processes are brought out. A distributed architecture for a CSP environment is proposed next. Here, there are a number of processing elements connected to a central processor by a star-connected message-passing network. Each processor has local memory and a communication interface unit. Several schemes are discussed to implement CSP. The instruction set of each processor is formulated. Details of the simulation are given.

18. MURTHY, C. S. R. and RAJARAMAN, V., An architecture of a Navier-Stokes computer and its implementation, *Proceedings of the Sixth IMACS International Symposium on Computer Methods for PDEs*, 1987. The importance of the Navier-Stokes equation is emphasized. The requirement of high-speed computation to solve these equations is pointed out. A survey of parallel computer architectures is done. An architecture for solving the Navier-Stokes equations is proposed. It has a two-dimensional broadcast bus network with a round robin non-gated sequential service access scheme to ensure collision-free transmission of data. The system architecture is described thereafter. The instruction set for interprocessor communication is explained. A typical system of Navier-Stokes equations is shown as a case study. The finite-difference

formulae for this system is derived along with a convergence criteria. How to map the problem on to hardware is shown. Complexity analysis is done.

19. GHOSHAL, S. K. and RAJARAMAN, V., A parallel digital differential analyzer, *Proceedings of the Indo-US Workshop on Spectral Analysis in One and Two Dimensions*, 1989, Oxford and IBH. A systolic architecture is proposed here for solving ordinary differential equations using parallel predictor-corrector methods. The task-partitioning scheme is described among the processors. Requirements of the interprocessor communication link are brought out. The process view of parallel numerical integration of a system of ordinary differential equation is encoded in the notation of CSP.

7.3. *Architectural proposals – Simulation and analysis*

20. BISWAS, N. N., SRINIVAS, S. and DHARANENDRA, T., A centrally controlled multistage shuffle network for reconfigurable and fault tolerant architecture, *ACM SIGARCH, Computer Architecture Newsletter*, 1987, 81–87. Conventional routing methods for shuffle networks are sequential ones. They are prone to collisions and conflict of messages. The scheme described in this paper is a centrally controlled one. Here a control code uniquely determines the route of a message without any conflict. This scheme enhances fault tolerance also. However, a centrally controlled scheme like this one has its own problems too.

21. BISWAS, N. N. and SRINIVAS, S., Fault tolerance in multiprocessing systems, *Sadhana*, 1987, **11**, 93–110. How the fault tolerance of message-passing multiprocessors using shuffle exchange networks can be enhanced is described here. An improvement in the routing algorithm which makes the routing conflict free and enhances the reliability of the network is elaborated.

22. BISWAS, N. N. and SRINIVAS, S., Simple methods for the calculations of root of reconfigurable binary tree structure, *Proceedings of the IEEE Letters*, 1987, 690–692. How to find the root of a reconfigured binary tree architecture uniquely from its configuration code is given in this paper. The algorithm can be adapted to identify other nodes as well in any tree architecture. The amenability of the root to be identified easily in a reconfigured tree architecture using multistage shuffle interchange networks leads to the ability of the architecture to recover from faults.

23. BISWAS, N. N. and SRINIVAS, S., Novel reconfigurable tree machine, *Electronics Letters*, 1987, 1144–1145.

24. BISWAS, N. N. and SRINIVAS, S., A reconfigurable tree architecture with multistage interconnection network, *IEEE Transactions on Computers*, 1990, **39**, 1481–1485. A new approach to design reconfigurable tree architectures is presented in the above two papers. A binary encoding scheme for the numbering of the nodes is used, which enables easy reconfigurability in the event of a failure. A multistage shuffle-exchange network is used that allows the multiprocessor to rapidly switch from one configuration to another (there being as many configurations as the number of processing elements). Results about the connectivity and uniqueness of the encoding scheme are stated and

proved. An example is given with eight processing elements. The advantages of this interconnection methodology are explained.

25. RAVIKANTH, K., SASTRY, P. S. and VENKATESH, Y. V., A reduction architecture for the optimal scheduling of binary trees, *Future Generation Computing Systems*, 1988, **4**, 225–233. An interconnection network that allows for scheduling of binary trees of arbitrary depth on reduction architectures for applicative languages is designed in this paper. The applicative semantic model of the languages to be implemented on this architecture is discussed. Issues in designing the interconnection network are discussed. A scheduling strategy is designed thereafter. Results are stated and proved in this regard. The strategy is illustrated by partitioning a very large binary tree on an eight-processor machine. An interconnection scheme for trees with arbitrary arity is formulated thereafter. Analytical formulae are derived for speedup. Scheduling and load-balancing strategies are given. The overall design is compared with AMPS, REDIFLOW and ZAPP for the intended application and is demonstrated to be better than these. The paper concludes by pointing out the need to investigate dynamic scheduling strategies, among other directions of future research.

26. BANERJEE, K., SASTRY, P. S. and VENKATESH, Y. V., An SIMD machine for low-level vision, *International Journal of Information Science*, 1988, **44**, 19–50. This paper pesents an SIMD machine which has been tuned to execute low-level vision algorithms employing the relaxation labelling paradigm. Novel features of the design include: 1) A communication scheme capable of window-accessing under a single instruction; 2) Flexible I/O instructions to load overlapped data segments; and 3) Data-conditional instructions which can be nested to an arbitrary degree. A time analysis of the stereo-correspondence problem, as implemented on a simulated version of the machine using probabilistic relaxation techniques, shows a speedup of almost N^2 for an $N \times N$ array of processing elements.

27. RAVIKUMAR, C. P. and PATNAIK, L. M., High-level simulation of parallel VLSI architectures, *Proceedings of the Second International Workshop on VLSI Design*, 1988, pp 367–381. Several strategies are discussed here to simulate SIMD, systolic and MIMD architectures. With so many VLSI architectures being proposed for different applications, there is a need for *software bread-boarding*. Issues regarding the simulation of parallel computer hardware are discussed. Mapping between physical entities in a parallel computer architecture and the corresponding software objects is tabulated. Next a processor is characterized in terms of its size and capabilities and also its principal classes of operations. The design and implementation of software objects that emulate these operations are given in detail. An example of simulating a systolic architecture using this methodology is given. It is followed by another one, of simulating an SIMD architecture. Conway's *Game of Life* is played on an $N \times N$ array of cells being driven by a control processor.

28. SHAH, N. R. and PATNAIK, L. M., SPARCS: A system for parallel architecture simulation, *Proceedings of the ICPP*, 1990, pp 605–606. A software tool featuring a built-in architecture compiler is described here which can be used to simulate parallel architectures. An extension of standard Pascal, called X-Pascal, is introduced as a

high-level language for specifying concurrency as a directed acyclic graph. Mechansims for interprocess communication and synchronization are discussed thereafter. An architecture model is introduced as a quadruple consisting of sets of processors, routers, memory modules and an irreflexive relationship denoting connectivity. MIMD systems can be directly mapped on this architecture model. A programmable communication processor is introduced next, which can be used as a router or an intermediate message processor. Each router can be programmed to adopt a first-in first-out strategy for the messages it processes. The simulator core is described next. It consists of seven components. They are: architecture builder, algorithm reader, scheduler, memory manager, simulation driver, communication handler, and statistics collector, respectively. The simulator has been used to study a number of proposed novel architectures.

29. KRISHNAN, D. and PATNAIK, L. M., Systolic architecture for Boolean operations on polygons and polyhedra, *Computer Graphics Forum*, 1987, **6**, 203–210. A systolic architecture for computing the union, intersection and difference of two polygons and polyhedra is described here. The different stages of a scan-grid algorithm are implemented using corresponding hardware units such that the data denoting the edges of the polygon flow through the system in a systolic fashion.

30. MATHIAS, P. C. and PATNAIK, L. M., A systolic evaluator for linear, quadratic and cubic expressions, *Journal of Parallel and Distributed Computing*, 1988, **5**, 729–740. A systolic evaluator for linear, quadratic and cubic expressions is given here. First, a simple and regular interconnection structure on a square grid of processing elements is proposed for evaluating linear expressions. It is developed further within the body of the paper to evaluate quadratic and cubic expressions.

31. MATHIAS, P. C. and PATNAIK, L. M., Systolic evaluation of polynomial expressions, *IEEE Transactions on Computers*, 1990, **39**, 653–665. The approach adopted earlier[30] is extended to cover polynomial expressions in this paper by deriving and using recurrence relationships between the results computed by multiple wavefronts.

32. AJJANAGADDE, V. G. and PATNAIK, L. M., Design and performance evaluation of a systolic architecture for hidden-surface removal, *Computers and Graphics*, 1988, **12**, 71–74. A systolic architecture for hidden surface removal is described here. Input data format, systolic scheme for hidden surface removal and architectural details are given. Performance evaluation of the same is done by simulation studies.

33. VAIDYA, N. H., DAS, S. R., MATHIAS, P. C. and PATNAIK, L. M., A systolic algorithm for scanline-based hidden-surface removal, *Proceedings of the Third International Conference on Supercomputing*, 1988, **2**, 239–246. An algorithm for hidden surface removal is given here. A clever use is made of the fact that a segment which is visible to the observer at the start of an interval retains its visibility until the end of that interval, provided that an appropriate preprocessing step has already been carried out to remove polygon intersections in the original scene by suitably fragmenting the intersecting polygons.

34. MATHIAS, P. C., PATNAIK, L. M. and SUDHA, R., Systolic architectures in curve generation, *Computers and Graphics*, 1989, **13**, 561–567. Systolic architectures are

proposed here for B-spline generation and inversion. The hardware implementation of such a systolic array is described in detail along with the interface with a graphics display processor.

35. LIGINLAL, D., NARASIMHA MURTY, M. and PATNAIK, L. M., A systolic approach to pattern clustering, *Proceedings of the International Symposium on Computer Architecture and Digital Signal Processing*, 1989. A systolic approach to pattern clustering is developed here. A cluster operator notation is developed. Its signal flow graph is analyzed. The graph is then localized and mapped on to a two-dimensional wavefront array. Performance analyses and a case study for remote sensing are presented.

36. PATNAIK, L. M., GOVINDARAJAN, R. and RAMADOSS, N. S., Design and performance evaluation of EXMAN: An EXtended MANchester dataflow computer, *IEEE Transactions on Computers*, 1986, **35**, 229–244. The Manchester dataflow machine execution model was appropriately upgraded here to support array access and procedure call efficiently. The description of a simulator of EXMAN, which is an extension of the Manchester dataflow computer, is given in this paper along with the design and performance evaluation of EXMAN. EXMAN efficiently supports arrays by following a hybrid approach that uses both dynamic pointer scheme and random access structure. The first feature avoids copying operations, which in turn increases the available memory area for other uses than storing arrays while the second one increases the effective access speed. EXMAN has multiple matching units. Thus, tokens with different labels, procedure invocation numbers and depths of iteration can be matched in parallel. EXMAN incorporates a runtime garbage collection scheme. The flowgraph representation for arrays as implemented in EXMAN facilitates the detection of nodes that will no longer be referenced and this aids the garbage collection scheme. An ingenious scheme is developed to support the concurrent execution of the different iterations of reentrant procedures by incorporating a procedure processing unit. On each invocation of a procedure, a record called 'procedure map information' is created. This enables the result tokens of the procedure to be properly returned. A similar mechanism is developed to support the concurrent execution of different loops in an iteration. A detailed analysis of the performance of EXMAN is done by simulation studies. The different parameters of EXMAN assumed by the simulator are clearly tabulated. The speedup obtained is plotted against the number of multiple matching units employed for the cases of executing dataflow graphs for matrix multiplication and two graphics-related programs. A detailed flow analysis is done to assess the resource balance achieved and the load balance.

37. PATNAIK, L. M. and MOHAN KUMAR, J., Neural network simulators on multiprocessor systems, *Proceedings of the Workshop on Parallel Processing*, Bombay, 1990, pp SEA-23 to SEA-32. A discussion on mapping neural networks on to multiprocessor systems can be found here. The memory requirements for storing the connection weights are estimated. Comments are made about the requirements of connectivity among the various processing elements. Different practical computer architectures are compared to evaluate their utility for this particular application.

38. MURTHY, C. S. R. and RAJARAMAN, V., A multi-microprocessor architecture for solving partial differential equations, *Microprocessing and Microprogramming*, 1987, **20**, 113–118.

39. MURTHY, C. S. R. and RAJARAMAN, V., A multiprocessor architecture for solving non-linear partial differential equations, *Mathematics and Computers in Simulation*, 1988, **30**, 453–464.

40. MURTHY, C. S. R. and RAJARAMAN, V., Multiprocessor architectures for solving PDEs, *Journal of the Institution of Electronics and Telecommunication Engineers*, 1988, **34**, 172–184.

41. MURTHY, C. S. R. and RAJARAMAN, V., Analytical and simulation studies of multiprocessor systems for high-speed numerical computation, *Mathematics and Computers in Simulation*, 1990, **32**, 393–401. Applications such as heat transfer and fluid-flow simulation require high-speed numerical computation. An architecture called 'broadcast cube system' has been proposed in the above four papers. It is a message-passing architecture with its processing elements interconnected by high-speed buses in different dimensions. The buses are capable of multicasting. Hardware modules called broadcast control units route the data among different dimensions appropriately. The three previous papers discuss the various issues related to solving partial differential equations on this architecture. Strategies are developed in this paper for analyzing the system on practical applications. Simulation results are presented.

7.4. *Performance analysis of parallel computers*

42. NARAHARI, Y. and VISWANADHAM, N., Performance modelling of local area distributed systems using stochastic Petri Nets, *Proceedings of the IEEE Conference on Systems and Signal Processing*, 1986, pp 304–308. In the design of parallel processing systems, analytical modelling plays an important role and entails powerful modelling tools to be used, such as stochastic Petri nets (SPNs) and queueing networks (QNs). In this paper, SPNs have been used to model a local area distributed system LOCUS comprising several VAX-11/750 computer systems, to derive performance measures such as response time and to study issues of scalability.

43. NARAHARI, Y. and VISWANADHAM, N., Performance modelling of a fault-tolerant real-time multiprocessor using stochastic Petri nets, *Sadhana*, 1987, **11**, 187–208. Here, the fault-tolerant multiprocessor (FTMP) built by NASA, USA, is considered and an SPN model is developed to capture the real-time operation of this system. The SPN model used here is more realistic and accurate compared to existing QN models.

44. NARAHARI, Y. and VISWANADHAM, N., Performance evaluation of computer systems using Petri nets with deterministic and stochastic timed transitions, *Proceedings of the IEEE TENCON*, 1987. A generalized class of SPNs has been proposed as a new modelling tool for parallel computing systems in this paper.

45. NARAHARI, Y., SURIYANARAYANAN, K. and SUBBA REDDY, N. V., Discrete event simulation of distributed systems using stochastic Petri nets, *Proceedings of the*

IEEE-TENCON, 1989, pp 31.4.1–31.4.5. Here SPNs are proposed as a convenient modelling paradigm for discrete event simulation of parallel and distributed computing systems. Efficient algorithms have been developed for simulation using the SPN paradigm.

46. RAVIKANTH, K., SASTRY, P. S. and VENKATESH, Y. V., Simulation studies on the performance of an organizational model for graph reduction, *Future Generation Computer Systems*, 1990, **6**, 163–180. This paper deals with the reduction of graphs in parallel. A set of processors interconnected as a binary de Brujin graph is used. Each processor has an *execution unit* and a *communication unit*. Shared memory interconnection is provided between neighboring processors. Performance analysis is done for both static and dynamic scheduling strategies. Load balancing is done in the latter case based on a set of criteria defined in the paper. Random binary and ternary graphs are generated and used in simulation studies. Details of simulation experiments, including task creation and execution models, are provided. A wealth of information regarding speedup and other performance-related issues is given in neatly organized graphs for various classes and depths of trees, scheduling policies and number of nodes. Simulation results are analyzed in detail.

47. NARAYAN, R. and RAJARAMAN, V., Performance analysis of a multiprocessor machine based on dataflow principles, *Microprocessing and Microprogramming*, 1990, **30**, 601–608. The performance analysis of a multiprocessor system based on static dataflow principles is given here. An execution model of the machine is given. A mathematical analysis of the same is presented. An idealized fine-grain dataflow graph is used as a starting example for deriving performance measures. The model is used to study the machine behaviour and the execution time of the example program, when the program is represented by dataflow graphs (again idealized) of different grain sizes. Thus the optimal grain size of the dataflow graph for a class of application programs is derived, so as to execute the programs in minimal time on a practical configuration of the machine.

48. NARAYAN, R. and RAJARAMAN, V., A method to evaluate the performance of a multiprocessor machine based on dataflow principles, *IEEE TENCON*, 1989. Further, the model used earlier[47] is tuned to accommodate practical problems. A coalescing algorithm is developed to combine nodes of a dataflow graph to a desired size for optimal execution of the program and optimal utilization of the processing elements. An extensive study leading to design decisions on building such multiprocessor systems is reported.

49. NARAYAN, R. and RAJARAMAN, V., A method to model a multicomputer system, *Proceedings of the Twenty first Annual Pittsburgh Conference on Modelling and Simulation*, 1990. The methodology and practice (which have been developed in the context of the studies reported earlier[47,48]) of the mathematical modelling of the multiprocessor system is given in this paper. It is further described how the results of performance analysis of application-specific programs can be used to bring out the design criteria of a scheduler for such multiprocessing systems. The study reveals certain key factors in the design of multiprocessor systems, *viz.*, reasonable speedup for larger

programs (of the order of 10,000 instructions with eight processing elements (PEs)) can be achieved with a dynamic scheduling strategy, thereby imposing demands for a high-speed scheduling unit and communication media. In contrast, however, the eight-PE machine using static scheduling achieves comparable speedup with slower components. It is further established through the simulation study that the eight-PE machine using quasi-dynamic scheduling strategy is better than using static policy by permitting slower scheduling units, which are easily realizable in hardware.

7.5. *Benchmarking an experimental multiprocessor built in hardware*

50. GHOSAL, D. and PATNAIK, L. M., Parallel polygon scan conversion algorithms: Performance evaluation on a shared bus architecture, *Computers and Graphics*, 1986, **10**, 7–25. Three parallel polygon scan-conversion algorithms were encoded and executed on SHAMP[2] to evaluate the architecture. One of them (PA1) uses only scan coherence and the other two (PA2 and PA3) take advantage of both scanline and edge coherence. Both simulation, using a tool called SIMPAR, and actual implementation on SHAMP were used. The results are given in this paper. Parallel algorithms are compared for different aspects of their execution behavior on practical multiprocessing architectures. The improvement of the performance of the algorithms is studied with different architectural enhancements. A modified multiprocessing architecture is proposed thereafter which uses a crossbar interconnection network between the processors and the refresh buffers. Simulation studies are done for this architecture executing the first algorithm (PA1).

7.6. *Languages for parallel computers*

51. DAVE, M. A. and SRIKANT, Y. N., A parallelizing compiler for Pascal, *This Journal, This Issue.* A Pascal compiler that can extract parallelism from sequential programs is designed and implemented on a real shared memory multiprocessor. Instead of using a flowgraph analysis, boxgraphs are used. Interprocedural dataflow and array subscript analysis are done efficiently. Issues in multiprocessing and compilation for parallelism are discussed. The rationale for preferring boxgraphs over flowgraphs is elaborated. The intermediate representation used in this compiler is described in detail. The nature of different types of dependence and their analysis are given. Techniques for detection of parallelism and implementation details are given. Results are tabulated. Conclusion identifies future direction of work, among other details.

52. SRIKANT, Y. N., Parallel parsing of arithmetic expressions, *IEEE Transactions on Computers*, 1990, **39**, 130–132. Erstwhile parallel algorithms for parsing arithmetic expressions have mostly used an idealized PRAM model of parallel program execution. This paper presents a parsing algorithm that uses practical interconnection between processors, *e.g.*, mesh, shuffle, cubes and cube-connected cycles. The algorithm has four steps. How each step can be executed in parallel on mesh-connected and other practical machines is elaborated. Complexity analysis for each step is done.

53. SRIKANT, Y. N. and SHANKAR, P., A new parallel algorithm for parsing arithmetic

infix expressions, *Parallel Computing*, 1987, **4**, 291–304. A parallel algorithm for converting an infix expression into a parse tree is given here. It uses an SIMD shared memory machine. Certain symbols are introduced and a few basic results are stated and proved first. Then the algorithm for constructing the parse tree is described. How the code generation is done for this algorithm to execute on an SIMD machine is described next. Vector quadruples are used for this purpose. Hardware and time complexities of the algorithm are derived.

54. SRIKANT, Y. N. and SHANKAR, P., Parallel parsing of programming languages, *Information Sciences*, 1987, **43**, 55–83. A new method called *hierarchical language specifications* (*HLS*) is introduced for specifying the syntax of programming languages. Efficient parallel algorithms executing on an exclusive-read exclusive-write PRAM machine are presented. An important point to note here is that the parallel algorithms do not use a stack. Complexity analyses are done to determine the hardware and time complexities of the algorithms.

55. VISWANATHAN, N. and SRIKANT, Y. N., Parallel attribute evaluation, *Supercomputing Symposium*, 1991. Tree contraction is applied on a CREW PRAM machine for evaluating attribute expressions in parallel. Other workers' contributions in parallel evaluation of attribute grammars and their models of parallel program execution are discussed. Details of the algorithm developed in this paper are explained with examples.

56. SRIKANT, Y. N., A parallel algorithm for the minimization of finite state automata, *International Journal of Computer Mathematics*, 1990, **32**, 1–11. A parallel algorithm for solving the set-partitioning problem is developed. The algorithm uses a CREW PRAM model of parallel computer. Hardware and time complexities are given. The algorithm can be used to minimize finite state automata. The parallel algorithm is explained in detail and results are stated and proved in this context.

57. GIBBONS, A. M. and SRIKANT, Y. N., A class of problems efficiently solvable on mesh-connected computers including dynamic expression evaluation, *Information Processing Letters*, 1989, **32**, 305–311. The class of efficiently solvable problems using a PRAM is discussed with an introductory example of a general polynomial expression evaluation. Different problem-solving strategies like divide and conquer are analyzed for their complexity. Finally, the problem of dynamic expression evaluation is defined and discussed in detail. How the different operations can be carried out in parallel is stated and the required hardware and time complexities are derived.

58. SHEKHAR, K. H. and SRIKANT, Y. N., Linda subsystem on transputers, *Proceedings of Transputing* 1991. Linda is introduced to the reader. The different tuple space operators in Linda are described. Issues in implementing Linda on distributed memory MIMD computers are discussed. Different ways to organize and partition the tuple space are compared. The format of a tuple used in this system is shown. Implementation details of the Linda kernel are given. This implementation supports recursions which many other Linda systems do not. How recursion is accommodated within the framework of Linda is shown. Application programming examples using this Linda system are given. Plots of speedup are given This Linda system is likely to be

implemented on a 64-transputer machine built at the Centre for Development of Advanced Computing (CDAC).

59. PATNAIK, L. M. and BASU, J., Two tools for interprocess communication in distributed dataflow systems, *The Computer Journal*, 1986, **29**, 506–521. Two programming languages are proposed here for representing distributed processing in dataflow systems. Essentially the paradigms of Hoare's CSP and Brinch Hansen's distributed processes were used to enhance the expressive powers of dataflow languages to arrive at two languages, *viz.*, DDFC and CDFC, respectively. A number of new features have been incorporated in both of these to make their execution more efficient. Two simulators were written to test the effectiveness of the implementation. Details of the simulated dataflow processor and the data structures used by the simulators are given.

60. GOSWAMI, A. K. and PATNAIK, L. M., A functional style of programming with CSP-like communication mechanisms, *New Generation Computing*, 1990, **7**, 341–364. Communication mechanisms similar to Hoare's CSP have been built into Backus' FP and other enhancements are made to FP in this paper. An algebra of programs with non-determinism is developed. The axiomatic semantics of communication constructs are presented. It is demonstrated how to transform communicating FP programs into equivalent non-communicating ones by using the axioms of communication to facilitate reasoning about the execution behaviour of the former. Feasibility of the inverse transformation is demonstrated as well.

7.7. Algorithms for parallel computers

61. MOHAN, T. S. and GANESAN, V. S., A higher order parallel abstraction for fixed degree divide and conquer problem solving strategy, *Proceedings of the IEEE ACE-90*, pp 217–221. An algorithmic framework for the *fixed degree divide and conquer* (FDDC) problem-solving strategy is presented. This adapts the divide and conquer algorithm for parallel computation. The notion of a thread of execution is captured in an active *task* object which are the instances of a data type called the *tasktype*. A higher order tasking abstraction called *maptask* is proposed. C is extended with *tasktype* and *maptask* in a package called the *CT Kernel* that currently executes tasks on a real distributed environment. Using the above abstractions, FDDC algorithms have been derived and implemented for a number of practical application problems. Performance results are presented and analyzed.

62. KRISHNA, S. and MOHAN, T. S., Partitioning bibliographic databases for parallel hypertext retrieval, *Proceedings of the COMAD-90*, 1990, INSDOC. A description of the problem is given. A multicomputing system model is formulated. The organization of the bibliographic data is elaborated. The cost of different types of search operations is analyzed. A sample bibliographic abstract is shown as a case study. The complexity of a partition assignment is derived. A heuristic partitioning algorithm is described.

63. RAVIKUMAR, C. P. and PATNAIK, L. M., Performance improvement of simulated annealing algorithms, *International Journal of Computer Systems, Science and Engineering*, 1990, **5**, 111–115. A pipelined task-partitioning scheme has been suggested here for

simulated annealing. Logic placement problem has been used as an example problem to solve by simulated annealing. The execution zone of the annealing algorithm has been divided into two zones: high- and low-temperature zones. Pipelined processing is used to accelerate the execution in the high-temperature zone. A rejectionless variant of the simulated annealing algorithm has been implemented in parallel to handle the low-temperature case properly.

64. RAVIKUMAR, C. P., SASTRY, S., and PATNAIK, L. M., Parallel circuit partitioning on a reduced array architecture, *Computer Aided Design*, 1989, **21**, 447–455. A reduced array architecture has been developed here in the context of partitioning large VLSI circuits. Kernighan-Lin algorithm for partitioning a VLSI layout is described. An SIMD architecture with a two-dimensional arrangement of memory and a linear organization of processing elements is proposed next. The internal organization of each processing element is described. Six most time-consuming steps in the Kernighan-Lin method are identified and described thereafter. How each of these can be executed in parallel on the proposed architecture is elaborated. Data and control structures are described thereafter for a practical implementation. Analytical results are stated and proved next about the speedup and optimality of the parallel algorithm on the proposed architecture.

65. GHOSHAL, S. K., GUPTA, M. and RAJARAMAN, V., A parallel multistep predictor corrector algorithm for solving ordinary differential equations, *Journal of Parallel and Distributed Computing*, 1989, **6**, 636–648. A new algorithm is developed in this paper for solving ordinary differential equations on parallel computers. Miranker and Liniger's parallel predictor corrector algorithm is enhanced to accommodate three correctors executing in parallel with a predictor approximation. The convergence and consistency of the algorithm is established. A methodology is implemented to determine the region of stability of this algorithm by numerical iterations. Task-partitioning schemes are developed for implementing the parallel predictor-corrector algorithm on practical multiprocessors. Numerical examples are conducted to plot the errors incurred by the algorithm for a number of test problems. The superiority of this algorithm over other sequential and parallel algorithms is demonstrated.

66. GHOSHAL, S. K. and RAJARAMAN, V., Increasing stability interval of parallel multistep predictor corrector methods, *Proceedings of the National Seminar on Parallel Processing Systems and Their Applications*, Institution of Engineers, Calcutta, India, December 9–11, 1988. A nonlinear optimization scheme is executed to maximize the region of absolute stability of the parallel predictor-corrector algorithm developed by Ghoshal *et al*[65]. A vector representation is developed to denote the errors incurred by the algorithm at each step. Data dependency to be honoured in cases of implementation on real multiprocessors is taken into account. The transformation to the vector of errors at each step is formulated as a matrix, whose eigenvalues must lie within the unit circle for the parallel algorithm to be stable. Rosenbrock's nonlinear optimization algorithm is used to alter the coefficients of the three correctors so as to endow the parallel predictor-corrector algorithm with the maximum region of stability. Numerical experiments are presented to demonstrate the effectiveness of the optimization methodology over a number of test problems. Results are presented graphically.

67. GHOSHAL, S. K. and RAJARAMAN, V., Optimally stable parallel techniques for handling variable steps in initial value integration, *Computer Science and Informatics*, 1987, **17**, 22–38. In order that they can be used in a practical initial value integrator, approximation formulae should be able to change their step size. In this paper, appropriate parallel techniques are developed for the predictor-corrector algorithms described earlier[65,66]. Task-partitioning schemes for these techniques are elaborated using a practical computer architecture as a model of parallel computer. Techniques similar to what had been applied to the approximation formulae[65,66] are used to maximize the interval stability for these techniques so that they withstand frequent changes in step size. This enables an efficient initial value integrator implemented on a practical multiprocessing architecture to integrate moderately stiff equations with a step size that is larger than what is possible with sequential methods. Thus the parallelism unfolded by the algorithm and exploited by the architecture can be effectively used on a large class of practical problems. Numerical results are plotted to substantiate this claim.

68. PAI, M. A., GHOSHAL, S. K. and KULKARNI, A., A predictor corrector algorithm in the parallel solution of power system dynamics, *Proceedings of the North American Power Systems Symposium*, 1990. No parallel algorithm can be called effective unless it solves a practical problem on a real multiprocessor. What has been developed earlier[65–67] is used here to solve systems of ordinary differential equations arising out of a real power system application using the four-processor-shared memory multiprocessor CRAY-2. The transient stability analysis of a fifty-generator power system spanning the United States and Canada is done to simulate the clearing of a fault in one of the buses. Various models of tasking that can be used on the CRAY-2 to harness parallelism present in a program, *viz. auto-tasking, micro-tasking,* and *macro-tasking* are compared. The issues in designing and profiling parallel programs for running on the CRAY-2 are discussed and distinguished between. The need for developing *quasi-parallel* programs in benchmarking is brought out. Sequential, *quasi-parallel* and parallel programs (using different tasking models available on the CRAY-2) are developed. Different user-controllable architectural and operating systems parameters are varied. Various compiler directives (to control the operation of the vectorizer and to do other compiler optimizations) are used in an organized way that makes sense to a power systems engineer. A suite of programs are prepared this way and executed in the dedicated mode on the CRAY-2. The results are compared with what is given by an equivalent program using the IMSL library and are found to tally. They agree with other industry standard programs (*e.g.*, the EPRI program which was run on the same data) too. The speedups obtained are tabulated and analyzed. The speedup and processor utilization are found to be satisfactory. Scope of further work in this area is pointed out.

69. HALDAR, C. and PATNAIK, L. M., Three algorithms on hypercube architecture and their applications to geometry problems, *Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers and Applications*, March 1989. Guidelines have been developed here for developing efficient hypercube algorithms on the basis of analyzing the execution of three basic algorithmic operations. First, the hypercube

nodes are labelled and other tagging primitives are performed on the architecture so as to set the context of the mathematical abstraction of a few routing and partial ordering operations. Then the syntax and semantics of BROADCAST and SORT are defined which are more or less the same as their names. Subsequently, the notion of dimension-sort is introduced, and a recursive divide and conquer algorithm is developed for finding the maximal elements of a set of points. Results are stated and proved about the complexity of the various steps of this algorithm. Then a set of geometric intersection problems are taken up for finding their optimal mapping on the hypercube and determining the resultant complexity. Finally, a search algorithm is derived and demonstrated by solving the problem of finding the Convex Hull of a set of points on a plane. Conclusions are drawn about the design of efficient geometric algorithms and their partitioning on hypercubes.

70. RAMAN, S., PATNAIK, L. M. and MALL, R., Parallel implementation of circuit simulation, *to appear in International Journal of High-speed Computing*. This paper presents a synthesizing overview of current trends in parallel algorithms and architectures for the simulation of VLSI circuits based on SPICE and relaxation techniques.

71. MALL, R., PATNAIK, L. M. and RAMAN, S., Simulated annealing-based channel routing on hypercube computers, *Proceedings of the Fourth CSI/IEEE International Symposium on VLSI Design*, New Delhi, January 1991, pp 75–81.

72. MALL, R. and PATNAIK, L. M., PMAZE: A parallel maze routing for hypercube computers, *Proceedings of the Third International Workshop in VLSI Design*, 1990, pp 124–132. An efficient algorithm for solving the routing problem in VLSI layout design has been proposed in the above two papers to execute on hypercubes. The three phases of Lee's maze-routing algorithm, *viz.*, wavefront expansion, backtrace and cleanup are made parallel. The resulting algorithm, called PMAZE, is then mapped on to a hypercube. A scheme is developed to route information about the different nets that do not overlap in parallel. Theorems are stated and proved in this context to derive the maximal set of such nets from a given netlist. The execution of PMAZE is studied on a simulated hypercube. Details of the netlist used are given, along with plots of the speedups for different routing lengths and number of processors.

73. RAVIKUMAR, C. P., SASTRY, S. and PATNAIK, L. M., A data parallel approach to local search placement algorithms, *Proceedings of the Second International Workshop on VLSI Design*, 1988, pp 3–27. A data parallel approach is developed here to arrive at an SIMD algorithm for the circuit placement problem in VLSI. The details of a typical VLSI chip in its number of grids, slots, channels and nets, etc., are tabulated first to get an idea of the complexity of a typical practical routing problem. Previous approaches to accelerate the execution of a router are elaborated next. An iterative improvement algorithm is presented thereafter for the placement of modules. It is analyzed next for finding out ways to execute it in parallel. A taxonomical study is made next of the different parallel iterative placement methods. The appropriate computational model to suit the execution of each of these is decided thereafter. The hardware requirement is estimated. Theorems are stated and proved after that to show that the algorithm that was developed to generate the random placement is appropriate

and provides a good speedup. Similar studies are undertaken thereafter for placement perturbation and cost difference evaluation. That sets the context for describing the overall parallel algorithm for parallel placement. Analytical results are presented on the performance evaluation of the algorithm. Simulation studies are done then to study the execution time of key steps in the algorithm, the overall running time, the behaviour of the cost improvement function and other parameters. A wealth of information obtained by simulation studies is presented graphically.

74. RAMAN, S. and PATNAIK, L. M., An annealing-based circuit partitioner for hypercube architectures: Design and performance evaluation, *International Journal of High-speed Computing*, 1990, **2**, 69–84.

75. RAMAN, S. and PATNAIK, L. M., HIRECS: Hypercube implementation of relaxation-based circuit simulation, *International Journal of High-speed Computing*, 1989, **1**, 399–432. Hypercube-related parallel circuit simulation algorithms and partitioning strategies have been reported in these two papers.

76. CHOWDHURY, S. and NARASIMHA MURTY, M., A parallel algorithm for generation of minimal spanning tree in Euclidean space and its implementation using systolic arrays, *Proceedings of the IEEE TENCON*, Seoul, 1987. Other workers' contribution towards developing parallel algorithms for generation of minimal spanning trees is reviewed. An algorithm that takes advantage of the fact that not all the distance computations are necessary is proposed next. This algorithm generates Bentley's $K-d$ trees which are complete k-ary trees. A systolic architecture for executing this algorithm efficiently is described next. It has four basic building blocks: the differencer, the squarer, the accumulator and the square-rooter. An extensive simulation is done and the results are presented.

77. NANDY, S. K., Geometric design rule check of VLSI layouts in distributed computing environment, *to appear in International Journal of Computer Aided VLSI Design*. The paper provides a distributed system to perform VLSI layout verification. Design rule checking (DRC) can be performed in parallel by exploiting either spatial independence or layer independence in layout data on a network of workstations. It is shown that the former approach performs better than the latter one for large layouts. An algorithm to optimally partition a layout and a scheme to allocate DRC tasks to idle processors in a distributed computing environment to attain load balancing is also described.

78. NANDY, S. K. and PANWAR, R. B., Geometric design check of VLSI layouts in mesh-connected processors, *to appear in International Journal of Computer Aided VLSI Design*. This paper presents a parallel algorithm to perform design rule check (DRC) of layout geometries in a VLSI layout. The algorithm is based on a linear quadtree representation and works on a mesh-connected interconnection of processors. DRC is performed by doing a parallel boundary following of the layout geometries in the VLSI layout. Through a complexity analysis, it is shown that a linear speedup with respect to the number of processors can be achieved for very large layouts. Further, the algorithm can be readily adapted to other interconnection topologies of higher connectivity, *viz.*, higher dimensional meshes or hypercubes by embedding 2-dimensional meshes on to these interconnection topologies.

79. NANDY, S. K., MOONA, R. and RAJAGOPALAN, S., Linear quadtree algorithms on the hypercube, *Proceedings of the ICPP*, 1988. Two algorithms of importance to image processing, VLSI design and computer graphics are developed for executing on hypercubes in this paper. Images are represented as quadtrees. A scheme is developed to encode the quadtrees in a linear array. A boundary following algorithm and the related neighbor finding algorithms are described then. Ways of embedding the quadtrees on the hypercube are described thereafter. A complexity analysis of the neighbor finding algorithm is performed. Analytical expressions for speedup are given.

7.8. *Neural networks*

80. RAO, D. S. and PATNAIK, L. M., Neural network-based approach to standard cell placement, *Electronics Letters*, 1989, **25**, 208–209. An algorithm for placement of standard cells in VLSI circuits based on an analogy with neural networks is given in this paper. A measure for the stimulus to a cell is formulated. Hebb's rule is then used to arrive at a connectivity matrix between the standard cells. The connectivity information given by the user is presented too. The process is repeated until each cell has been perturbed at least once. After the connection strengths between all the cells is reinforced, a simple partitioning is done. This procedure is applied recursively. The performance of this algorithm has been found to be better than the standard Kernighan-Lin method.

81. SRIRAM, K. B. and PATNAIK, L. M., Neural network approach for the two-dimensional assignment problem, *Electronics Letters*, 1990, **26**, 809–810. The approach used earlier[80] has been extended to cover the two-dimensional assignment problem in this paper.

82. RAO, D. S., PROVENCE, J. D. and PATNAIK, L. M., A neural network-based method for dynamic reconfigurability of array processors, *Proceedings of the Symposium on Parallel Processing*, 1990, Fullerton, CA, USA. A neural network-based method for dynamic reconfiguration of array processors has been reported here.

83. DESHPANDE, V. and DASGUPTA, C., Neural network models of associative memory, *Proceedings of the DAE Symposium on Solid State Physics*, 1990. Modelling of associative memory using neural networks is done here. The paper also gives the history of the subject along with the shortcomings of the classical Hopfield model. Three improvements are studied over the Dotsenko model for the storage and associative recall of strongly correlated memories. A neural network in which computations are performed with limit cycles is proposed. Analytical and numerical treatment of the same is summarized.

7.9. *Task scheduling in parallel computers*

84. MURTHY, C. S. R. and RAJARAMAN, V., Task assignment in a multiprocessor system, *Microprocessors and Microprogramming*, 1989, **26**, 63–71.

85. MURTHY, C. S. R. and RAJARAMAN, V., On the assignment of precedence and communication constrained tasks in a multiprocessor system, *Computer Science and*

Informatics, 1989, **19**, 17–22. Issues in task assignment in a multiprocessing context are discussed in the above two papers. A survey is made of the work done by others in this subject. A message-based multiple-bus multiprocessor is described next. The set of tasks to be assigned on this multiprocessor is posed as a directed acyclic graph. Terminology is developed for assignment configurations and the precedence and the constraints involved in this context Certain combinatorial results are stated and proved. A heuristic task assignment algorithm is proposed thereafter. Its complexity is estimated. The implementation details of the algorithm are given. The efficiency of the algorithm is demonstrated over a series of task-assignment problems using graphs.

86. RAJARAMAN, V., Conversion of decision tables to programs in multiprocessor system, *Computer Science and Informatics*, 1989, **19**, 30–35. Two heuristic methods are developed in this paper for converting decision tables to computers programs that execute in parallel on a multiprocessor. The methods are so designed that the programs execute in minimum time. Simulation studies on randomly generated decision tables show that resulting programs execute in time close to the lower bound for ideal execution. The two algorithms are described in detail within the body of the paper. Results of the simulation studies are tabulated. Conclusions point out the maximum number of processors usable under this scheme for generating near-optimal programs among other inferences.

87. SADAYAPPAN, P. and VISVANATHAN, V., Modeling and optimal scheduling of parallel sparse Gaussian elimination, *to appear in IEEE Transactions on Computers*. Task graphs called *minimally constrained task graphs* (MCTG) are proposed here to model parallel Gaussian elimination and its scheduling on multiprocessors. MCTGs contain both directed edges and undirected edges. Their optimality of scheduling is proved on a CREW multiprocessor with unbounded number of processors. The suboptimality of DAGs in this respect is demonstrated. The Gaussian elimination algorithm is explained and a taskgraph corresponding to it is used as an example in this regard. An MCTG is formally defined thereafter for this problem. An algorithm for greedy-level assignment for such a graph is explained. Comparisons between the number of levels using the DAG and the MCTG approaches are made and tabulated for a number of sparse matrices arising out of circuit simulation. Certain results which are used in this paper are stated and proved in the Appendix.

88. MOHAN, T. S., A tasking abstraction for message-passing architectures, *Proceedings of the PARCOM-90*, 1990, pp 21–29, Narosa. A semantically sound linguistic construct called *Abstract Tasktype* has been proposed. It is formulated for C. An implementation is made on a real distributed computing environment. Performance evaluation for a large number of application problems is made.

7.10. *Books*

89. RAJARAMAN, V., *Elements of parallel computing*, 1990, Prentice-Hall of India. This book is a basic introduction to the design of parallel computers and how to solve problems on parallel computers. It has an annotated bibliography for further reading.

90. GOSWAMI, A. K. and PATNAIK, L. M., *Non-determinism and communication in functional programming systems*, Lecture Notes in Computer Science, Springer-Verlag (*in press*). Non-determinism and inter-program communication are introduced in functional programming systems. Several new program-forming operations are developed in this context which simplify the ways of reasoning about the programs algebraically. Algebraic methods for development of programs from their specifications are described. Illustrative examples are given to explain the various programming concepts.