Int. J. Engng Ed. Vol. 11, Nos 4 and 5, pp. 329–335, 1995
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 1995 TEMPUS Publications.

HIPP: An Honors Program in Parallel Processing

BOB P. WEEMS
KRISHNA M. KAVI
BEHROOZ SHIRAZI
Department of Computer Science and Engineering, The University of Texas at Arlington, POB 19015, Arlington, TX 76019, USA

The Honors Program in Parallel Processing (HIPP) was established to provide undergraduate education in parallel processing. We present the goals of parallel processing education, and outline the supporting curriculum and laboratories. We emphasize certain courses that include relevant parallel processing material. The Programming Language Concepts course introduces parallel languages, while the Algorithms course introduces algorithm analysis concepts needed for later examination of parallel algorithms. The two semester Computer Systems Architecture course provides foundation in parallel architectures. The final related course, Parallel Processing, brings together these foundations in the study of parallel algorithms. In addition, we describe some recent undergraduate theses (a HIPP graduation requirement) and a NSF Research Experience for Undergraduates (grant no. CDA-9300252) program. A NSF-ILI grant (no. CDA-9052136)has made possible the purchase of the parallel processing equipment for use with the HIPP program.

THE CURRENT STATE OF PARALLEL PROCESSING

UNDERGRADUATE honors programs accelerate the learning pace for motivated students and provide experiences outside the usual curriculum. The goal of our honors program is to cultivate parallel processing skills along with a high-quality computer science and engineering education. An eight-processor Sequent Symmetry system to support the honors program has been purchased with an NSF-ILI grant and UTA matching funds.

The need for increasingly powerful computing systems has existed ever since computers were invented. This trend is expected to continue, resulting in multiprocessor systems with large numbers of processing units. The number of people capable of using parallel systems, however, is still very small. Even though more parallel processors, parallel and vectorizing compilers, and debugging tools are becoming available, we believe that without the scientific base, creative talent, understanding and demanding users, the potential of such configurations is wasted.

Typical computer science and computer engineering programs provide few opportunities to specialize in parallel processing, often only at the graduate level [1]. Some programs offer courses on parallel architectures, concurrent programming languages or senior design projects involving parallel processing systems. Parallel programming requires a clear understanding of both the target machine architecture and the exploitable parallelism within the application. Generating parallel code involves non-trivial operations that have no analog in code generation for uniprocessor architectures. These include the partitioning of work into schedulable units, optimal assignment of the partitions on the available processors, and the additional code needed for interprocessor communication and synchronization. The performance of the code on parallel processors depends significantly on these operations.

The curriculum and courses must be designed to provide the student with a background in parallel architecture, interconnection networks, memory systems, parallelization of algorithms, concurrent programming methods and operating systems, before experiments and design projects can be conducted. Courses providing necessary foundation in these areas must be taught early in the program. Only then can carefully planned and well-designed capstone projects utilizing parallel systems for the solution of non-trivial engineering problems be conducted.

THE HONORS IN PARALLEL PROCESSING CURRICULUM

Curriculum overview

We believe that the principles of parallel processing cannot be taught in a single semester. Nor can they be mastered in a small number of laboratory exercises. They require a carefully planned introduction to the underlying principles, techniques and mechanisms. Moreover, the foundation should be laid very early in the program.

The program in the UTA Computer Science Engineering Department is accredited by ABET and thus emphasizes the traditional engineering education. The CSE curriculum provides an extensive background in computer science and computer engineering, including architecture, operating systems, software engineering, data structures, algorithms, numerical methods and microprocessors. Students use elective courses to concentrate in an area of specialization.

While maintaining the emphasis on traditional engineering, the HIPP curriculum differs from the existing undergraduate program by emphasizing theory and practice of parallel programming at several levels. First, in selected courses, HIPP students perform additional activities that emphasize parallel processing. For example, students in the Algorithms course (CSE 2320) study approaches to parallel sorting, and students in the Programming Languages course (CSE 3302) learn concurrent programming languages such as Ada, SR [2], Occam [3], Concurrent FORTRAN, Concurrent C and SISAL (a dataflow language).

Secondly, all students in HIPP are required to take courses on parallel processor architecture (CSE 4323) and parallel processing (CSE 4351). These courses provide the theory and practical relevance to the underlying models of parallel computations. These courses are available to regular CSE students as technical electives.

Thirdly, the HIPP curriculum requires an individual honors thesis. The thesis is initiated by an Honors Seminar (CSE 4155) taken concurrently with the Parallel Processing course (CSE 4351). Students develop project proposals, collect the relevant literature, and make a presentation to the HIPP faculty advisers. The project is then implemented during two semesters of honors thesis (CSE 4356/4357). The thesis sequence replaces the current Senior Design Project sequence (CSE 4316/4317). The honors projects provide the HIPP student with depth and experience in exploring parallel solutions to scientific and engineering problems. These problems challenge the student with trade-offs related to scalability, maximizing parallelism, minimizing communication and synchronization overheads, memory configurations, interconnection topologies and data organizations. The student must analyze the problem carefully, study possible parallel implementations (either existing or new approaches), select an optimal solution, and evaluate the performance, efficiency and accuracy of the chosen solution. The thesis report must document all design decisions, and reflect the scientific and engineering nature of the experiment. It is hoped that these projects will encourage students to pursue graduate studies in computer science and engineering. The thesis sequence is available only to the HIPP students. The courses directly relevant to HIPP are diagrammed in Fig. 1.

Another difference between HIPP and the current curricula is the timing of required courses (see Appendix A for a suggested course sequence). We anticipate that many HIPP students will be exempt from the introductory Pascal course and Analytic Geometry, to allow for a rapid progression through the prerequisite courses. Courses which do not directly impact the parallel processing education are delayed. Even though the HIPP curriculum is more structured and contains more required courses than the non-HIPP undergraduate program, it still allows two technical electives. Students are encouraged to use these electives during their junior year to gain depth in an application area for cultivating parallel processing projects of the honors thesis.

Supporting courses

The HIPP curriculum is designed to follow the spirit of the ACM curriculum task force [3] recommendations. The major problem with our previous CSE curricula stemmed from a programming emphasis in early courses, without a firm foundation in theory and abstractions. As is common in CSE (and CS) departments, our students were required to take Discrete Mathematics in the Mathematics Department. Typically, students take this course in their junior year, after they have already been exposed to some of the material in freshman and sophomore CSE courses.

Previously, students enter the CSE curriculum with a two-hour course, Introduction to Computing (CSE 1241), which teaches elementary algorithm design and the use of computing resources. Following CSE 1241, students took a Pascal course (CSE 2304), and then a course in assembler language and C programming (CSE 2310). Few students used the option of taking Data Structures (CSE 3306), concurrent with or before the assembler course. Thus, the students gain three semesters of programming with little theory. Inevitably, these programming courses tend to impinge on material from the data structures course. In addition, a large number of demanding CSE courses (e.g. Architecture, Software Engineering, Operating Systems, Compilers) are pushed into the senior year. These courses and the Senior Design Project (CSE 4316/4317) place undue demands on students' time and resources. This initial course sequence also did not consider exceptional freshman who can be exempt from the first two introductory courses (CSE 1241/2304).

In support of the HIPP curriculum, we have redesigned the introductory sequence. A suggested HIPP course sequence, that also reflects these changes in course content, is included in Appendix A. The problem of mathematical foundations is addressed in the first year. Also note that HIPP students complete courses related to parallel processing (e.g. CSE 2441 and CSE 3322) early in the program. All HIPP students are required to take courses on parallel processor architecture (CSE 4323), and parallel processing (CSE 4351). We now elaborate on some courses.

Mathematical Principles (CSE 1442). Given the widespread use of computers, we can assume that

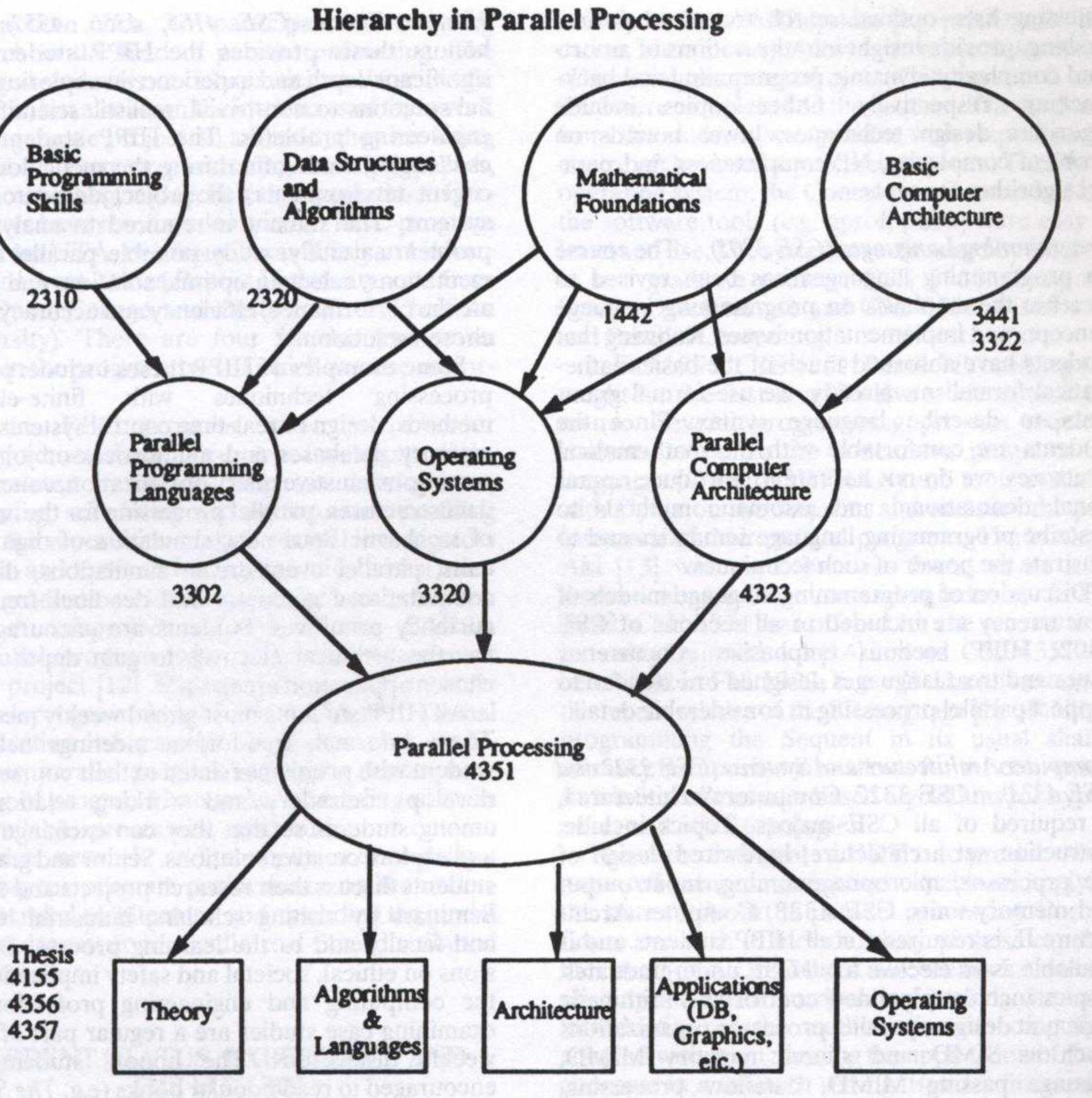

Fig. 1. HIPP curriculum structure.

students entering our program will have a modest background in programming and some experience with computer-based tools. (A remedial course on Pascal programming is still available.) This allows a freshman-level course to introduce the theory and abstractions that are the foundation of the computer science. Part of this theory is normally taught in the junior year under the rubric of Discrete Mathematics. Other aspects of the theory, such as propositional and predicate calculus, are not included in most programs. Teaching logic and encouraging use of formal proofs will make it possible to teach formal specification and verification in other courses.

CSE 1442 is designed with this motivation. This course introduces the mathematical formalisms that form a basis for later courses. The use of logic proofs is encouraged throughout the course. Algorithm design and some data structures are introduced as needed, thus building on their programming knowledge. Because of the introduction of these mathematical structures, it is possible to improve other courses such as CSE 2441 (Digital Logic) and CSE 2320 (Algorithms/Data Structures). This, in turn, makes it possible to bring courses on architecture, compilers and operating systems to the sophomore and junior years of the curriculum.

Design and Analysis of Algorithms (CSE 2320). Even though introductory courses emphasize top-down decomposition of a programming task, the refinements at lower levels may still present a simply-described problem for which a straightforward solution will not provide sufficient performance. In CSE 2320, such problems are introduced, analyzed and efficient designs are examined. CSE 2320 requires a background in the elementary data structures learned in programming courses and the mathematical maturity from CSE 1442. This course introduces complex data structures through examples. For instance, we believe that balanced trees and hashing deserve early attention. Specialized structures, such as selfadjusting lists, optimal search trees and perfect hashing, provide insight into the notions of amortized complexity, dynamic programming and backtracking, respectively. Other topics include algorithm design techniques, lower bounds on problem complexity, NP-completeness and parallel algorithm concepts.

Programming Languages (CSE 3302). The course on programming languages has been revised to increase the emphasis on programming language concepts and implementation issues. Knowing that students have absorbed much of the basic mathematical formalisms already, we use formal grammars to describe language syntax. Since the students are comfortable with the mathematical structures, we do not hesitate to introduce operational, denotational and axiomatic methods to describe programming language semantics and to illustrate the power of such techniques.

Discussion of programming language models of concurrency are included in all sections of CSE 3302. HIPP sections emphasize concurrency issues and treat languages designed or extended to support parallel processing in considerable detail.

Computer Architecture and Systems (CSE 3322 and CSE 4323). CSE 3322, Computer Architecture I, is required of all CSE majors. Topics include: instruction set architecture, hardwired design of the processor, microprogramming, input/output and memory units. CSE 4323, Computer Architecture II, is required for all HIPP students and is available as an elective to all CSE undergraduates. Topics include: pipelined control and arithmetic logic unit designs, parallel processor organizations (such as SIMD and shared memory MIMD, message passing MIMD, dataflow processing, cache memory design and processor-memory interconnections.

Parallel Processing (CSE 4351). A new course called Parallel Processing is designed mainly for HIPP students. Through this course, students obtain basic knowledge in programming parallel processors in preparation for their honors theses. More specifically, the course (i) integrates the concepts and knowledge of parallel processing from previous courses (i.e. issues related to parallel architecture covered in CSE 4323, operating systems including interprocess communication and synchronization covered in CSE 3320, and programming languages and compilers covered in CSE 3302 and CSE 4305); (ii) teaches parallel algorithms (e.g. sorting, searching, matrix manipulation and graph algorithms) and analyzes them for efficiency and speed-up; and (iii) discusses the issues in utilizing parallel systems for applications. Both topology-independent algorithms based on shared memory abstractions (more commonly called P-RAM algorithms, Parallel Random Access Memory [5]) and topology-dependent algorithms [6] are introduced.

Honors Thesis (CSE 4155, 4356, 4357). The honors thesis provides the HIPP student with significant depth and experience in exploring parallel solutions to non-trivial, realistic scientific and engineering problems. The HIPP student faces challenging trade-offs during the meticulous and cogent development of the project ideas into actual systems. The student is required to analyze the problem carefully, study possible parallel implementations, select an optimal solution, and evaluate the performance, efficiency and accuracy of the chosen solution.

Some examples of HIPP theses include: parallel processing techniques with finite-element methods, design of real-time control systems, main memory databases and multiprocessor join processing, exhaustive query optimization, concurrent data structures, parallel processing for the analysis of stochastic Petri nets, simulation of digital circuits, parallel event-driven simulations, discrete computational geometry and deadlock-free concurrency primitives. Students are encouraged to use the technical electives to gain depth in the chosen application area.

All HIPP students must attend weekly meetings. These informal, free-format meetings help the student with problems related to their courses, and develop comradery and working relationships among students so that they can exchange ideas and explore creative solutions. Senior and graduate students discuss their research projects and theses. Seminars by visiting scholars, industrial leaders and faculty add to the learning process. Discussions on ethical, societal and safety implications of the computing and engineering professions by examining case studies are a regular part of these weekly discussions. The honors students are encouraged to read popular books (e.g. *The Soul of a New Machine* [7], *Fumbling the Future* [8], *The Cuckoo's Nest* [9]), along with various texts providing advanced knowledge not specifically covered in the courses.

THE RESEARCH EXPERIENCES

Three honors theses have been completed by undergraduates in the HIPP program. One student studied the MAISIE environment for developing distributed simulation [10] software. Another student implemented and evaluated concurrent approaches to maintaining a binary search tree [11]. In a recent ambitious project, a student developed a simulator for the MONSOON dataflow architecture. We anticipate the completion of numerous theses in the next few years.

The Research Experience for Undergraduates for Undergraduates in Software Tools for Parallel Program Development and Assessment project provides opportunities for highly talented and motivated women, minority, and disabled undergraduate students, especially from institutions lacking research facilities in the Dallas/Fort Worth metropolitan area, to participate in on-going research in the field of parallel processing. The recruited students not only participate in scientific research, but also learn the intricacies of teamwork in a large-scale project. In addition, they are being trained in literature searching, reading papers, technical writing, and technical presentations. Currently we have seven students in our REU program (two from UTA, two from Texas Weslyan University, one from Dallas Baptist University, one from Paul Quinn College, and one from Texas Christian University). There are four females and seven males in the program, two of which are also minorities.

In this REU program, seven junior/senior undergraduate students per year investigate problems related to parallel program development, debugging, scheduling and performance profiling under the supervision of the principal investigators (Shirazi and Kavi) and in collaboration with a number of graduate students. The students can choose a research subproject as part of the PARSA (PARallel program Scheduling and Assessment tools) project [12]. Examples of projects include: parallel program design and verification, formal specification of parallel programs and architectures, parallel program debugging, static partitioning of programs into tasks, scheduling of tasks on available processors, distribution of data to minimize network and memory access delays, profiling the program performance on the underlying parallel architecture, and extending the model and environment to handle real-time distributed applications.

CURRENT STATUS, EXPERIENCE AND THE FUTURE

During the Spring 1991 semester, the Sequent system was used by the students in CSE 4323 (Architecture of Parallel Processors). Several simple programming problems were assigned to introduce the MIMD system (e.g. matrix multiplication, bitonic sort). A few term projects involving substantial programming in Concurrent C and Sisal were completed on the Sequent. The feedback from the students was positive with respect to the choice of the parallel processing system. The Dynix operating system, the Concurrent C compiler, and the software tools (e.g. gprof, pdbx) were easy to learn and use. The system was also used by students in two graduate courses, CSE 5350: Computer Systems Architecture and CSE 6351: Distributed and Parallel Computing.

In Fall 1991, CSE 4351: Parallel Processing was taught for the first time. Twelve students took the course. Several programming projects required students to utilize the concurrent programming languages (C and FORTRAN) and tools available with Dynix on the Sequent, and measure the performance of their parallel programs. The book by Akl [13] was used, but was found to be a bit theoretical and distant to undergraduates. In Fall 1992, CSE 4351 was offered for the second time, but using Quinn's book [14] instead. CSE 4351 will be offered in Fall 1993, but will take advantage of the recent release of the SR language [2]. SR allows programming the Sequent in its usual shared memory fashion, but also allows for the convenient simulation of message-based algorithms for hypercubes and other topologies.

The immediate goal of HIPP is to produce well-prepared computer scientists capable of applying the state-of-the-art in parallel processing. For the future, we wish to include many of these capabilities in all of our graduates. The means for accomplishing these is a curriculum featuring the early introduction of the mathematics of computing, coupled with strengthening the application of these concepts in a research setting. Besides fine-tuning the HIPP curriculum, we are anxious to provide other parallel processing experiences to HIPP students (e.g. vector processing, message passing MIMD, functional languages, and distributed processing via a network of workstations).

REFERENCES

1. R. Miller, The Status of Parallel Processing Education: 1993, Technical Report, Department of Computer Science, State University of New York at Buffalo (August 4, 1993). Available by anonymous ftp from cs.buffalo.edu.
2. G. R. Andrews and R. A. Olsson, *The SR Programming Language: Concurrency in Practice*, Benjamin/Cummings, Redwood City, CA (1993).
3. INMOS Limited, *Occam Programming Manual*, Prentice Hall, Englewood Cliffs, NJ (1984).
4. P. J. Denning *et al.*, Computing as a discipline: final report of the ACM Task Force on the Core of Computer Science (February 1988).
5. J. H. Reif (ed.), *Synthesis of Parallel Algorithms*, Morgan Kaufmann, San Mateo, CA (1993).
6. F. Thomson Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, San Mateo, CA (1992).
7. Tracy Kidder, *The Soul of a New Machine*, Little, Brown, Boston, MA (1981).
8. D. K. Smith and R. C. Alexander, *Fumbling the Future: How Xerox Invented, then Ignored, the First Personal Computer*, W. Morrow, New York (1988).
9. C. Stoll, *The Cuckoo's Nest: Tracking a Spy through the Maze of Computer Espionage*, Doubleday, New York (1989).
10. D. A. Reed and R. M. Fujimoto, *Multicomputer Networks: Message-Based Parallel Processing*, MIT Press, Cambridge, MA (1987).

11. U. Manber and R. E. Ladner, Concurrency control in a dynamic search structure, *ACM Trans. Database Syst.*, **9** (3), 439–455 (1984).
12. Behrooz Shirazi, PARSA: A PARallel program Scheduling and Assessment environment, Technical Report, Department of Computer Science and Engineering, University of Texas at Arlington (January 1992).
13. S. G. Akl, *The Design and Analysis of Parallel Algorithms*, Prentice Hall, Englewood Cliffs, NJ (1989).
14. M. J. Quinn, *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill, New York (1987).

APPENDIX A: SAMPLE FOUR-YEAR PROGRAM

Freshman year
Fall 1 (15)
CSE 1442: Computer Science: A Mathematical Introduction
MATH 1426: Calculus I
PHYS 1443: General Technical Physics I
CHEM 1301: General Chemistry
Spring 1 (18)
CSE 2310: Assembly and C Languages
CSE 2441: Computer Organization
PHYS 1444: General Technical Physics II
MATH 2425: Calculus II
ENGL 1301: Critical Thinking, Reading and Writing I

Sophomore year
Fall 2 (16)
CE 2312: Statics and Dynamics
EE 2315/2181: Circuit Analysis I/Lab
CSE 2320: Design and Analysis of Algorithms
ENGL 1302: Critical Thinking, Reading and Writing II
MATH 2326: Calculus III
Spring 2 (15)
CSE 3302: Algorithmic Languages
CSE 3322: Computer Architecture I
MATH 3318: Differential Equations
MATH 3330: Matrix Algebra
ENGL: Literature Elective

Junior year
Fall 3 (18)
EE 2321: Electronics for Engineers
CSE 4323: Computer Architecture II
CSE 3320: Operating Systems
CSE 4305: Compilers
IE 3301: Engineering Probability
HIST/POL: History/Political Science Elective
Spring 3 (14)
CSE 3442: Microcomputer Systems Design
CSE 3310: Software Engineering
CSE 4351: Parallel Processing
CSE 4155: Honors Seminar
HIST/POL: History/Political Science Elective

Senior year
Fall 4 (15)
CSE 4356: Honors Thesis I
CSE: Elective
SPCH 3302: Professional and Technical Communication
HIST/POL: History/Political Science Elective
HUMA: Humanities Elective
Spring 4 (15)
CSE 4357: Honors Thesis II
CSE: Elective
IE 3312: Engineering Economy
HIST/POL: History/Political Science Elective
HUMA: Humanities Elective

Total: 126 hours (Analytic Geometry and Pascal Programming are assumed to be waived via Advanced Placement Examinations)

APPENDIX B: SUPPORTING HARDWARE

- Sequent S27 Building Block
- Four dual processor boards, each with two Intel 80386/80387/MMU/64KB cache
- 40 MBytes ECC memory
- 1/4 in. cartridge tape
- 6250/1600 BPI tape drive
- Two 540 MByte disk drives
- Two 792 MByte disk drives

Bob P. Weems received the Ph.D. from Northwestern University in 1985. Since that time he has been with the Department of Computer Science Engineering in the College of Engineering at the University of Texas at Arlington. He has taught courses on databases, algorithms, data structures and parallel processing. His research emphasizes topics in these areas, including databases security, engineering databases, computational geometry and resolution theorem proving. He has served as an associate undergraduate advisor and currently serves as an associate graduate advisor. Finally, he has advised student organizations and has coached the UTA Programming Team since 1986.

Krishna Kavi is currently a professor of Computer Science and Engineering at the University of Texas at Arlington. His research interests are in dataflow architecture, performance

analysis, parallelizing compilers and formal specification of concurrent processing systems. He is also interested in developing courses and curicula in the area of parallel processing. He serves as a CSAB visitor, and he is an editor of the *IEEE Transactions on Computers*. He was an IEEE CS Distinguished visitor and served as an editor of the IEEE CS Press. He received his MS and Ph.D. from Southern Methodist University.

Dr **Behrooz Shirazi** is an associate professor of Computer Science and Engineering at the University of Texas at Arlington. Dr Shirazi's research interests include parallel and distributed systems, task partitioning and scheduling, and computer architecture. He has published over 60 technical papers in these areas. Dr Shirazi's research has been sponsored by grants from NSF, DARPA, AFOSR, TI, and the State of Texas ATP. He has been a guest-editor of a special issue of the *Journal of Parallel and Distributed Computing* and a track coordinator of the HICSS '93 Conference, both on 'Scheduling and Load Balancing Issues'. Dr Shirazi is the principle founder of the IEEE Symposium on Parallel and Distributed Processing and has served on the program committee of many international conferences. He is currently an IEEE Distinguished Visitor as well as an ACM Lecturer.