Processor Architecture From Dataflow To Superscalar And Beyond

The refereed proceedings of the International Symposium on Parallel and Distributed Processing and Applications, ISPA 2003, held in Aizu, Japan in July 2003. The 30 revised full papers and 9 revised short papers presented together with abstracts of 4 keynotes were carefully reviewed and selected from numerous submissions. The papers are organized in topical sections on applications on Web-based and intranet systems, compiler and optimization techniques, network routing, performance evaluation of parallel systems, wireless communication and mobile computing, parallel topology, data mining and evolutionary computing, image processing and modeling, network security, and database and multimedia systems.

Processor ArchitectureFrom Dataflow to Superscalar and BeyondSpringer Science & Business Media

Media processing applications, such as three-dimensional graphics, video compression, and image processing, currently demand 10-100 billion operations per second of sustained computation. Fortunately, hundreds of arithmetic units can easily fit on a modestly sized 1cm2 chip in modern VLSI. The challenge is to provide these arithmetic units with enough data to enable them to meet the computation demands of media processing applications. Conventional storage hierarchies, which frequently include caches, are unable to bridge the data bandwidth gap between modern DRAM and tens to hundreds of arithmetic units. A data bandwidth hierarchy, however, can bridge this gap by scaling the provided bandwidth across the levels of the storage hierarchy. The stream programming model enables media processing applications to exploit a data bandwidth hierarchy effectively. Media processing applications can naturally be expressed as a sequence of computation kernels that operate on data streams. This programming model exposes the locality and concurrency inherent in these applications and enables them to be mapped efficiently to the data bandwidth hierarchy. Stream programs are able to utilize inexperience local data bandwidth when possible and consume expensive global data bandwidth only when necessary. Stream Processor Architecture presents the architecture of the Imagine streaming media processor, which delivers a peak performance of 20 billion floating-point operations per second. Imagine efficiently supports 48 arithmetic units with a three-tiered data bandwidth hierarchy. At the base of the hierarchy, the streaming memory system employs memory access scheduling to maximize the sustained bandwidth of external DRAM. At the center of the hierarchy, the global stream register file enables streams of data to be recirculated directly from one computation kernel to the next without returning data to memory. Finally, local distributed register files that directly feed the arithmetic units enable temporary data to be stored locally so that it does not need to consume costly global register bandwidth. The bandwidth hierarchy enables Imagine to achieve up to 96% of the performance of a stream processor with infinite bandwidth from memory and the global register file.

As computing devices proliferate, demand increases for an understanding of emerging computing paradigms and models based on natural phenomena. Neural networks, evolution-based models, quantum computing, and DNA-based computing and simulations are all a necessary part of modern computing analysis and systems development. Vast literature exists on these new paradigms and their implications for a wide array of applications. This comprehensive handbook, the first of its kind to address the connection between nature-inspired and traditional computational paradigms, is a repository of case studies dealing with different problems in computing and solutions to these problems based on nature-inspired paradigms. The "Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies" is an essential compilation of models, methods, and algorithms for researchers, professionals, and advanced-level students working in all areas of computer science, IT, biocomputing, and network engineering.

Effective compilers allow for a more efficient execution of application programs for a given computer architecture, while well-conceived architectural features can support more effective compiler optimization techniques. A well thought-out strategy of trade-offs between compilers and computer architectures is the key to the successful designing of highly efficient and effective computer systems. From embedded micro-controllers to large-scale multiprocessor systems, it is important to understand the interaction between compilers and computer architectures. The goal of the Annual Workshop on Interaction between Compilers and Computer Architectures (INTERACT) is to promote new ideas and to present recent developments in compiler techniques and computer architectures that enhance each other's capabilities and performance. Interaction Between Compilers and Computer Architectures is an updated and revised volume consisting of seven papers originally presented at the Fifth Workshop on Interaction between Compilers and Computer Architectures (INTERACT-5), which was held in conjunction with the IEEE HPCA-7 in Monterrey, Mexico in 2001. This volume explores recent developments and ideas for better integration of the interaction between compilers and computer architectures in designing modern processors and computer systems. Interaction Between Compilers and Computer Architectures is suitable as a secondary text for a graduate level course, and as a reference for researchers and practitioners in industry.

This book focuses on source-to-source code transformations that remove addressing-related overhead present in most multimedia or signal processing application programs. This approach is complementary to existing compiler technology. What is particularly attractive about the transformation flow pre sented here is that its behavior is nearly independent of the target processor platform and the underlying compiler. Hence, the different source code trans formations developed here lead to impressive performance improvements on most existing processor architecture styles, ranging from RISCs like ARM7 or MIPS over Superscalars like Intel-Pentium, PowerPC, DEC-Alpha, Sun and HP, to VLIW DSPs like TI C6x and Philips TriMedia. The source code did not have to be modified between processors to obtain these results. Apart from the performance improvements, the estimated energy is also significantly reduced for a given application run. These results were not obtained for academic codes but for realistic and rep resentative applications, all selected from the multimedia domain. That shows the industrial relevance and importance of this research. At the same time, the scientific novelty and quality of the contributions have lead to several excellent papers that have been published in internationally renowned conferences like e. g. DATE. This book is hence of interest for academic researchers, both because of the overall description of the methodology and related work context and for the detailed descriptions of the compilation techniques and algorithms.

The refereed proceedings of the 12th International Conference on Computer Analysis of Images and Patterns are presented in this volume. The papers cover motion detection and tracking, medical imaging, biometrics, color, curves and surfaces beyond two dimensions, reading characters, words and lines, image segmentation, shape, image registration and matching, signal decomposition and invariants, and features and classification.

This innovative book uncovers all the steps readers should follow in order to build successful software and systems With the help of numerous examples, Albin clearly shows how to incorporate Java, XML, SOAP, ebXML, and BizTalk when designing true distributed business systems Teaches how to easily integrate design patterns into software design Documents all architectures in UML and presents code in either Java or C++

This book presents a new set of embedded system design techniques called multidimensional data flow, which combine the various benefits offered by existing methodologies such as block-based system design, high-level simulation, system analysis and polyhedral optimization. It describes a novel architecture for efficient and flexible high-speed communication in hardware that can be used both in manual and automatic system design and that offers various design alternatives, balancing achievable throughput with required hardware size. This book demonstrates multidimensional data flow by showing its potential for modeling, analysis, and synthesis of complex image processing applications. These applications are presented in terms of their fundamental properties and resulting design constraints. Coverage includes a discussion of how far the latter can be met better by multidimensional data flow than alternative approaches. Based on these results, the

book explains the principles of fine-grained system level analysis and high-speed communication synthesis. Additionally, an extensive review of related techniques is given in order to show their relation to multidimensional data flow.

It gives me immense pleasure to introduce this timely handbook to the research/- velopment communities in the ?eld of signal processing systems (SPS). This is the ?rst of its kind and represents state-of-the-arts coverage of research in this ?eld. The driving force behind information technologies (IT) hinges critically upon the major advances in both component integration and system integration. The major breakthrough for the former is undoubtedly the invention of IC in the 50's by Jack S. Kilby, the Nobel Prize Laureate in Physics 2000. In an integrated circuit, all components were made of the same semiconductor material. Beginning with the pocket calculator in 1964, there have been many increasingly complex applications followed. In fact, processing gates and memory storage on a chip have since then grown at an exponential rate, following Moore's Law. (Moore himself admitted that Moore's Law had turned out to be more accurate, longer lasting and deeper in impact than he ever imagined.) With greater device integration, various signal processing systems have been realized for many killer IT applications. Further breakthroughs in computer sciences and Internet technologies have also catalyzed large-scale system integration. All these have led to today's IT revolution which has profound impacts on our lifestyle and overall prospect of humanity. (It is hard to imagine life today without mobiles or Internets!) The success of SPS requires a well-concerted integrated approach from mul- ple disciplines, such as device, design, and application.

Advances in signal and image processing together with increasing computing power are bringing mobile technology closer to applications in a variety of domains like automotive, health, telecommunication, multimedia, entertainment and many others. The development of these leading applications, involving a large diversity of algorithms (e.g. signal, image, video, 3D, communication, cryptography) is classically divided into three consecutive steps: a theoretical study of the algorithms, a study of the target architecture, and finally the implementation. Such a linear design flow is reaching its limits due to intense pressure on design cycle and strict performance constraints. The approach, called Algorithm-Architecture Matching, aims to leverage design flows with a simultaneous study of both algorithmic and architectural issues, taking into account multiple design constraints, as well as algorithm and architecture optimizations, that couldn't be achieved otherwise if considered separately. Introducing new design methodologies is mandatory when facing the new emerging applications as for example advanced mobile communication or graphics using sub-micron manufacturing technologies or 3D-Integrated Circuits. This diversity forms a driving force for the future evolutions of embedded system designs methodologies. The main expectations from system designers' point of view are related to methods, tools and architectures supporting application complexity and design cycle reduction. Advanced optimizations are essential to meet design constraints and to enable a wide acceptance of these new technologies. Algorithm-Architecture Matching for Signal and Image Processing presents a collection of selected contributions from both industry and academia, addressing different aspects of Algorithm-Architecture Matching approach ranging from sensors to architectures design. The scope of this book reflects the diversity of potential algorithms, including signal, communication, image, video, 3D-Graphics implemented onto various architectures from FPGA to multiprocessor systems. Several synthesis and resource management techniques leveraging design optimizations are also described and applied to numerous algorithms. Algorithm-Architecture Matching for Signal and Image Processing should be on each designer's and EDA tool developer's shelf, as well as on those with an interest in digital system design optimizations dealing with advanced algorithms.

The term computation gap has been defined as the difference between the computational power demanded by the application domain and the computational power of the underlying computer platform. Traditionally, closing the computation gap has been one of the major and fundamental tasks of computer architects. However, as technology advances and computers become more pervasive in the society, the domain of computer architecture has been extended. The scope of research in the computer architecture is no longer restricted to the computer hardware and organization issues. A wide spectrum of topics ranging from algorithm design to power management is becoming part of the computer architecture. Based on the aforementioned trend and to reflect recent research efforts, attempts were made to select a collection of articles that covers different aspects of contemporary computer architecture design. This volume of the Advances in Computers contains six chapters on different aspects of computer architecture. Key features: Wide range of research topics Coverage of new topics such as power management, Network on Chip, Load balancing in distributed systems, and pervasive computing Simple writing style Wide range of research topics Coverage of new topics such as power management, Network on Chip, Load balancing in distributed systems, and pervasive computing Simple writing style

Concurrency is an integral part of everyday life. The concept is so ingrained in our existence that we benefit from it without realizing. When faced with a taxing problem, we automatically involve others to solve it more easily. Such concurrent solutions to a complex problem may, however, not be quite straightforward and communication becomes crucial to ensure the successful solution of the problem.

This book offers an original and informative view of the development of fundamental concepts of computability theory. The treatment is put into historical context, emphasizing the motivation for ideas as well as their logical and formal development. In Part I the author introduces computability theory, with chapters on the foundational crisis of mathematics in the early twentieth century, and formalism; in Part II he explains classical computability theory, with chapters on the quest for formalization, the Turing Machine, and early successes such as defining incomputable problems, c.e. (computably enumerable) sets, and developing methods for proving incomputability; in Part III he explains relative computability, with chapters on computation with external help, degrees of unsolvability, the Turing hierarchy of unsolvability, the class of degrees of unsolvability, c.e. degrees and the priority method, and the arithmetical hierarchy. This is a gentle introduction from the origins of computability theory up to current research, and it will be of value as a textbook and guide for advanced undergraduate and graduate students and researchers in the domains of computability theory and theoretical computer science.

This useful text/reference describes the implementation of a varied selection of algorithms in the DataFlow paradigm, highlighting the exciting potential of DataFlow computing for applications in such areas as image understanding, biomedicine, physics simulation, and business. The mapping of additional algorithms onto the DataFlow architecture is also covered in the following Springer titles from the same team: DataFlow Supercomputing Essentials: Research, Development and Education, DataFlow Supercomputing Essentials: Algorithms, Applications and Implementations, and Guide to DataFlow Supercomputing. Topics and Features: introduces a novel method of graph partitioning for large graphs involving the construction of a skeleton graph; describes a cloud-supported web-based integrated development environment that can develop and run programs without DataFlow hardware owned by the user; showcases a new approach for the calculation of the extrema of functions in one dimension, by implementing the Golden Section Search algorithm; reviews algorithms for a DataFlow architecture that uses matrices and vectors as the underlying data structure; presents an algorithm for spherical code design, based on the variable repulsion force method; discusses the implementation of a face recognition application, using the DataFlow paradigm; proposes a method for region of interest-based image segmentation of mammogram images on high-performance reconfigurable DataFlow computers; surveys a diverse range of DataFlow applications in physics simulations, and investigates a DataFlow implementation of a Bitcoin mining algorithm. This unique volume will prove a valuable reference for researchers and programmers of DataFlow computing, and supercomputing in general. Graduate and advanced undergraduate students will also find that the book serves as an ideal supplementary text for courses on Data Mining, Microprocessor Systems, and VLSI Systems.

The term computation gap has been defined as the difference between the computational power demanded by the application domain and the computational power of the underlying computer platform. Traditionally, closing the computation gap has been one of the major and fundamental tasks of computer architects. However, as technology advances and computers become more pervasive in the society, the

domain of computer architecture has been extended. The scope of research in the computer architecture is no longer restricted to the computer hardware and organization issues. A wide spectrum of topics ranging from algorithm design to power management is becoming part of the computer architecture. Based on the aforementioned trend and to reflect recent research efforts, attempts were made to select a collection of articles that covers different aspects of contemporary computer architecture design. This volume of the Advances in Computers contains six chapters on different aspects of computer architecture. Key features: - Wide range of research topics. - Coverage of new topics such as power management, Network on Chip, Load balancing in distributed systems, and pervasive computing. - Simple writing style. · Wide range of research topics. · Coverage of new topics such as power management, Network on Chip, Load balancing in distributed systems, and pervasive computing. · Simple writing style

It is universally accepted today that parallel processing is here to stay but that software for parallel machines is still difficult to develop. However, there is little recognition of the fact that changes in processor architecture can significantly ease the development of software. In the seventies the availability of processors that could address a large name space directly, eliminated the problem of name management at one level and paved the way for the routine development of large programs. Similarly, today, processor architectures that can facilitate cheap synchronization and provide a global address space can simplify compiler development for parallel machines. If the cost of synchronization remains high, the pro gramming of parallel machines will remain significantly less abstract than programming sequential machines. In this monograph Bob Iannucci presents the design and analysis of an architecture that can be a better building block for parallel machines than any von Neumann processor. There is another very interesting motivation behind this work. It is rooted in the long and venerable history of dataflow graphs as a formalism for ex pressing parallel computation. The field has bloomed since 1974, when Dennis and Misunas proposed a truly novel architecture using dataflow graphs as the parallel machine language. The novelty and elegance of dataflow architectures has, however, also kept us from asking the real question: "What can dataflow architectures buy us that von Neumann ar chitectures can't?" In the following I explain in a round about way how Bob and I arrived at this question.

Designing VLSI systems represents a challenging task. It is a transfonnation among different specifications corresponding to different levels of design: abstraction, behavioral, stntctural and physical. The behavioral level describes the functionality of the design. It consists of two components; static and dynamic. The static component describes operations, whereas the dynamic component describes sequencing and timing. The structural level contains infonnation about components, control and connectivity. The physical level describes the constraints that should be imposed on the floor plan, the placement of components, and the geometry of the design. Constraints of area, speed and power are also applied at this level. To implement such multilevel transfonnation, a design methodology should be devised, taking into consideration the constraints, limitations and properties of each level. The mapping process between any of these domains is non-isomorphic. A single behavioral component may be transfonned into more than one structural component. Design methodologies are the most recent evolution in the design automation era, which started off with the introduction and subsequent usage of module generation especially for regular structures such as PLA's and memories. A design methodology should offer an integrated design system rather than a set of separate unrelated routines and tools. A general outline of a desired integrated design system is as follows: * Decide on a certain unified framework for all design levels. * Derive a design method based on this framework. * Create a design environment to implement this design method.

ICA3PP 2000 was an important conference that brought together researchers and practitioners from academia, industry and governments to advance the knowledge of parallel and distributed computing. The proceedings constitute a well-defined set of innovative research papers in two broad areas of parallel and distributed computing: (1) architectures, algorithms and networks; (2) systems and applications.

Abstract: "An efficient static dataflow architecture based on an argument-fetching data-driven principle has recently been proposed (as reported by Dennis and Gao [5]). This architecture opens possibilities in combining the techniques of existing high-performance conventional pipelined architectures with the strengths of the dataflow model of parallel computation. The key feature is that data never 'flows' in the new architecture even though instruction scheduling remains data-driven. The new architecture answers some speculations about the efficiency of practical dataflow architectures -- data-driven instruction scheduling need not mean higher traffic due to data token flow in the processor architecture.

This text describes the machine model designed to support parallel interface, the design of the Kleng language, the design and implementation of the parallel interface engine, the programming tools, the runtime system, and some evaluation results. The architecture of the PIE 64 is tuned specially to support parallel inference. The compiler and runtime systems proposed here are designed to reduce the overhead that inevitably incurrs when using fine granularity processing.

The 14th Iberoamerican Congress on Pattern Recognition (CIARP 2009, C-gresoIberoAmericanodeReconocimientodePatrones)formedthelatestofanow longseriesofsuccessfulmeetingsarrangedbytherapidlygrowingIberoamerican pattern recognition community. The conference was held in Guadalajara, Jalisco, Mexico and organized by the Mexican Association for Computer Vision, Neural Computing and Robotics (MACVNR). It was sponsodred by MACVNR and ?ve other Iberoamerican PR societies. CIARP 2009 was like the previous conferences in the series supported by the International Association for Pattern Recognition (IAPR). CIARP 2009 attracted participants from all over the world presenting sta- of-the-artresearchon mathematical methods and computing techniques for p- tern recognition, computer vision, image and signal analysis, robot vision, and speech recognition, as well as on a wide range of their applications. This time the conference attracted participants from 23 countries,9 in Ibe- america, and 14 from other parts of the world. The total number of submitted papers was 187, and after a serious review process 108 papers were accepted, all of them with a scienti?c quality above overall mean rating. Sixty-four were selected as oral presentations and 44 as posters. Since 2008 the conference is almost single track, and therefore there was no real grading in quality between oral and poster papers. As an acknowledgment that CIARP has established itself as a high-quality conference, its proceedings appear in the Lecture Notes in Computer Science series. Moreover, its visibility is further enhanced by a selection of a set of papers that will be published in a special issue of the journal Pattern Recognition Letters.

This book constitutes the proceedings of the International Conference on Information and Communication Technologies held in Kochi, Kerala, India in September 2010.

The structure of a computer which utilizes a data-flow program representation as its base language is described. The use of the data-flow representation allows full exploitation by the processor of the parallelism and concurrency achievable through the data-

flow form. The unique architecture of the processor avoids the usual problems of processor switching and memory/processor interconnection by the use of interconnection networks which have a great deal of inherent parallelism. The structure of the processor allows a large number of instructions to be active simultaneously. These active instructions pass through the interconnection networks concurrently and form streams of instructions for the pipelined functional units. Due to the cyclic nature of an iterative computation, the possiblity of deadlock can arise in the performance of such a computation within the dataflow architecture. A deadlock is caused by the interaction of several simultaneously active cycles of the same iterative computation. The use of a recursive rather than iterative representation of a computation avoids the deadlock problem and provides a more efficient implementation of the computation within the architecture. For this reason, a program executed by the data-flow processor is restricted to an acyclic directed graph representation. (Author).

Offering a carefully reviewed selection of over 50 papers illustrating the breadth and depth of computer architecture, this text includes insightful introductions to guide readers through the primary sources.

Network processors are the basic building blocks of today's high-speed, high-demand, quality-oriented communication networks. Designing and implementing network processors requires a new programming paradigm and an in-depth understanding of network processing requirements. This book leads the reader through the requirements and the underlying theory of networks, network processing, and network processors. It covers implementation of network processors and intergrates EZchip Microcode Development Environment so that you can gain hands-on experience in writing high-speed networking applications. By the end of the book, the reader will be able to write and test applications on a simulated network processor. Comprehensive, theoretical, and pracitical coverage of networks and high-speed networking applications Descirbes contemporary core, metro, and access networks and their processing algorithms Covers network processor architectures and programming models, enabling readers to assess the optimal network processor typer and configuration for their application Free download from http://www.cse.bgu.ac.il/npbook includes microcode development tools that provide hands-on experience with programming a network processor

The design process of embedded systems has changed substantially in recent years. One of the main reasons for this change is the pressure to shorten time-to-market when designing digital systems. To shorten the product cycles, programmable processes are used to implement more and more functionality of the embedded system. Therefore, nowadays, embedded systems are very often implemented by heterogeneous systems consisting of ASICs, processors, memories and peripherals. As a consequence, the research topic of hardware/software co-design, dealing with the problems of designing these heterogeneous systems, has gained great importance. Hardware/Software Co-design for Data Flow Dominated Embedded Systems introduces the different tasks of hardware/software co-design including system specification, hardware/software partitioning, co-synthesis and co-simulation. The book summarizes and classifies state-of-the-art co-design tools and methods for these tasks. In addition, the co-design tool COOL is presented which solves the co-design tasks for the class of data-flow dominated embedded systems. In Hardware/Software Co-design for Data Flow Dominated Embedded Systems the primary emphasis has been put on the hardware/software partitioning and the co-synthesis phase and their coupling. In contrast to many other publications in this area, a mathematical formulation of the hardware/software partitioning problem is given. This problem formulation supports target architectures consisting of multiple processors and multiple ASICs. Several novel approaches are presented and compared for solving the partitioning problem, including an MILP approach, a heuristic solution and an approach based on genetic algorithms. The co-synthesis phase is based on the idea of controlling the system by means of a static run-time scheduler implemented in hardware. New algorithms are introduced which generate a complete set of hardware and software specifications required to implement heterogeneous systems. All of these techniques are described in detail and exemplified. Hardware/Software Co-design for Data Flow Dominated Embedded Systems is intended to serve students and researchers working on hardware/software co-design. At the same time the variety of presented techniques automating the design tasks of hardware/software systems will be of interest to industrial engineers and designers of digital systems. From the foreword by Peter Marwedel: Niemann's method should be known by all persons working in the field. Hence, I recommend this book for everyone who is interested in hardware/software co-design.

Cloud Computing: Theory and Practice, Second Edition, provides students and IT professionals with an in-depth analysis of the cloud from the ground up. After an introduction to network-centric computing and network-centric content in Chapter One, the book is organized into four sections. Section One reviews basic concepts of concurrency and parallel and distributed systems. Section Two presents such critical components of the cloud ecosystem as cloud service providers, cloud access, cloud data storage, and cloud hardware and software. Section Three covers cloud applications and cloud security, while Section Four presents research topics in cloud computing. Specific topics covered include resource virtualization, resource management and scheduling, and advanced topics like the impact of scale on efficiency, cloud scheduling subject to deadlines, alternative cloud architectures, and vehicular clouds. An included glossary covers terms grouped in several categories, from general to services, virtualization, desirable attributes and security. Includes new chapters on concurrency, cloud hardware and software, challenges posed by big data and mobile applications and advanced topics Provides a new appendix that presents several cloud computing projects Presents more than 400 references in the text, including recent research results in several areas related to cloud computing

This Workshop focuses on such issues as control algorithms which are suitable for real-time use, computer architectures which are suitable for real-time control algorithms, and applications for real-time control issues in the areas of parallel algorithms, multiprocessor systems, neural networks, fault-tolerance systems, real-time robot control identification, real-time filtering algorithms, control algorithms, fuzzy control, adaptive and self-tuning control, and real-time control applications.

This book gathers the refereed proceedings of the Applied Informatics and Cybernetics in Intelligent Systems Section of

the 9th Computer Science On-line Conference 2020 (CSOC 2020), held on-line in April 2020. Modern cybernetics and computer engineering in connection with intelligent systems are an essential aspect of ongoing research. This book addresses these topics, together with automation and control theory, cybernetic applications, and the latest research trends.

A survey of architectural mechanisms and implementation techniques for exploiting fine- and coarse-grained parallelism within microprocessors. Beginning with a review of past techniques, the monograph provides a comprehensive account of state-of-the-art techniques used in microprocessors, covering both the concepts involved and implementations in sample processors. The whole is rounded off with a thorough review of the research techniques that will lead to future microprocessors. XXXXXXX Neuer Text This monograph surveys architectural mechanisms and implementation techniques for exploiting fine-grained and coarse-grained parallelism within microprocessors. It presents a comprehensive account of state-of-the-art techniques used in microprocessors that covers both the concepts involved and possible implementations. The authors also provide application-oriented methods and a thorough review of the research techniques that will lead to the development of future processors.

This monograph evolved from my Ph. D dissertation completed at the Laboratory of Computer Science, MIT, during the Summer of 1986. In my dissertation I proposed a pipelined code mapping scheme for array operations on static dataflow architectures. The main addition to this work is found in Chapter 12, reflecting new research results developed during the last three years since I joined McGill University-results based upon the principles in my dissertation. The terminology dataflow soft ware pipelining has been consistently used since publication of our 1988 paper on the argument-fetching dataflow architecture model at McGill University [43]. In the first part of this book we describe the static data flow graph model as an operational model for concurrent computation. We look at timing considerations for program graph execution on an ideal static dataflow computer, examine the notion of pipe lining, and characterize its performance. We discuss balancing techniques used to transform certain graphs into fully pipelined data flow graphs. In particular, we show how optimal balancing of an acyclic data flow graph can be formulated as a linear programming problem for which an optimal solution exists. As a major result, we show the optimal balancing problem of acyclic data flow graphs is reduceable to a class of linear programming problem, the net work flow problem, for which well-known efficient algorithms exist. This result disproves the conjecture that such problems are computationally hard.

Computing Handbook, Third Edition: Computer Science and Software Engineering mirrors the modern taxonomy of computer science and software engineering as described by the Association for Computing Machinery (ACM) and the IEEE Computer Society (IEEE-CS). Written by established leading experts and influential young researchers, the first volume of this popular handbook examines the elements involved in designing and implementing software, new areas in which computers are being used, and ways to solve computing problems. The book also explores our current understanding of software engineering and its effect on the practice of software development and the education of software professionals. Like the second volume, this first volume describes what occurs in research laboratories, educational institutions, and public and private organizations to advance the effective development and use of computers and computing in today’s world. Research-level survey articles provide deep insights into the computing discipline, enabling readers to understand the principles and practices that drive computing education, research, and development in the twenty-first century.

The 1992 Parallel Architectures and Languages Europe conference continues the tradition - of a wide and representative international meeting of specialists from academia and industry in theory, design, and application of parallel computer systems - set by the previous PARLE conferences held in Eindhoven in 1987, 1989, and 1991. This volume contains the 52 regular and 25 poster papers that were selected from 187 submitted papers for presentation and publication. In addition, five invited lectures areincluded. The regular papers are organized into sections on: implementation of parallel programs, graph theory, architecture, optimal algorithms, graph theory and performance, parallel software components, data base optimization and modeling, data parallelism, formal methods, systolic approach, functional programming, fine grain parallelism, Prolog, data flow systems, network efficiency, parallel algorithms, cache systems, implementation of parallel languages, parallel scheduling in data base systems, semantic models, parallel data base machines, and language semantics.

Motivation It is now possible to build powerful single-processor and multiprocessor systems and use them efficiently for data processing, which has seen an explosive ex pansion in many areas of computer science and engineering. One approach to meeting the performance requirements of the applications has been to utilize the most powerful single-processor system that is available. When such a system does not provide the performance requirements, pipelined and parallel process ing structures can be employed. The concept of parallel processing is a depar ture from sequential processing. In sequential computation one processor is in volved and performs one operation at a time. On the other hand, in parallel computation several processors cooperate to solve a problem, which reduces computing time because several operations can be carried out simultaneously. Using several processors that work together on a given computation illustrates a new paradigm in computer problem solving which is completely different from sequential processing. From the practical point of view, this provides sufficient justification to investigate the concept of parallel processing and related issues, such as parallel algorithms. Parallel processing involves utilizing several factors, such as parallel architectures, parallel algorithms, parallel programming lan guages and performance analysis, which are strongly interrelated. In general, four steps are involved in performing a computational problem in parallel. The first step is to understand the nature of computations in the specific application domain.

The Concise Encyclopedia of Computer Science has been adapted from the full Fourth Edition to meet the needs of students, teachers and professional computer users in science and industry. As an ideal desktop reference, it contains

shorter versions of 60% of the articles found in the Fourth Edition, putting computer knowledge at your fingertips. Organised to work for you, it has several features that make it an invaluable and accessible reference. These include: Cross references to closely related articles to ensure that you don’t miss relevant information Appendices covering abbreviations and acronyms, notation and units, and a timeline of significant milestones in computing have been included to ensure that you get the most from the book. A comprehensive index containing article titles, names of persons cited, references to sub-categories and important words in general usage, guarantees that you can easily find the information you need. Classification of articles around the following nine main themes allows you to follow a self study regime in a particular area: Hardware Computer Systems Information and Data Software Mathematics of Computing Theory of Computation Methodologies Applications Computing Milieux. Presenting a wide ranging perspective on the key concepts and developments that define the discipline, the Concise Encyclopedia of Computer Science is a valuable reference for all computer users.