DESIGNING AND PROGRAMMING MODERN COMPUTER SYSTEMS

Volume II

SUPERCOMPUTING SYSTEMS: RECONFIGURABLE ARCHITECTURES

Svetlana P. Kartashev

Steven I. Kartashev

DESIGNING AND PROGRAMMING MODERN COMPUTER SYSTEMS

Volume II

SUPERCOMPUTING SYSTEMS: RECONFIGURABLE ARCHITECTURES

Svetlana P. Kartashev

International Supercomputing Institute

PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data
(Revised for vol. 2)

Designing and programming modern computers and systems.

Vol. 2— has title: Designing and programming modern computer systems.
Includes bibliographies and indexes.
Contents: v. 1. LSI modular computer systems— v. 2 Supercomputing systems.
1. Computer engineering. 2. Computer architecture. I. Kartashev, Svetlana. II. Kartashev, Steven I. III. Title: Designing and programming modern computer systems.
TK7885.D474 1982 004 81-21078

Editorial/production supervision: *Raeia Maes*
Manufacturing buyer: *Mary Ann Gloriande*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-201435-1

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Foreword

BY

GENE MYRON AMDAHL

This is the second volume in the series, *Designing and Programming Modern Computer Systems*, initiated by Prentice Hall in 1982.

This volume deals with supercomputing in the context of reconfigurable architectures. The important class of these architectures, called dynamic architectures and invented by Svetlana P. Kartashev and Steven I. Kartashev in the 1970s, allows dynamic partitioning of the resources into different sets of computers with varied word sizes and automatic assumption of various computer architectures under the software control.

The authors' vision is to define a dynamic architecture as given hardware box assembled from processor and memory units that can be formed into differing computing structures under the program control. These structures are: multicomputers/multiprocessors, arrays, pipelines, networks, and mixed. Each structure is characterized by parameters variability extended to the architectural type of the structure (multicomputer, array, pipeline, network, mixed), the word sizes of its units (dynamic computers, processor elements, pipeline stages, network nodes), the number of units included in the structure, and the interconnections between concurrent structures formed dynamically.

The major objective of this approach is to improve performance through more extensive utilization of the available hardware resources than is possible to achieve in modern computer systems. Currently, available supercomputers achieve peak performance only for particular portions of computations being underutilized a significant portion of their time.

There are many reasons for this performance degradation, but the major one is associated with the **mismatch** between the hardware computing structure and the task being computed.

Sources of this mismatch are:

- **Bit size mismatch:** The bit size requirements of the program are smaller or larger than that of the hardware resources that compute this program. If these requirements are smaller, the difference in bit sizes extended to the processor and memory resources becomes unused and leads to **performance degradation** caused by the failure of the computer hardware to utilize it. If these requirements are larger, the computation should proceed in a much slower and less precise floating-point form, leading again to a performance degradation as a result.
- **Concurrency mismatch:** The program requirements on the number of parallel instruction and data streams are smaller or larger than those of the hardware resources. If these requirements are smaller, the idleness of the unused portion of the hardware leads to slowing the time of the execution of other concurrent programs (tasks) in the mix. If these requirements are larger, parallel information streams (instruction and/or data) must be computed sequentially, again leading to an increase in the overall computation time.
- **Interconnection mismatch:** The program requirements on the interconnection of the used resource units (processor and memory resources involved in computations) mismatch the available interconnections in a computer system.

As a result, the data exchanges among engaged resources take much longer than if these resources were connected optimally.

The only way to overcome mismatches that create a nonoptimal use of the hardware resources is through reconfigurable interconnections when the resources are partitioned dynamically into a varied number of computing structures each of which has the bit size also varied via software.

Another problem that must be solved is the development of very fast reconfiguration from one computing structure to another in order to eliminate any reconfiguration overhead from the total computation time. This radical departure from the conventional computation process requires development of program preprocessing techniques aimed at finding the optimal architectural structures that can be used in computations. Thereafter, these structures will be assumed automatically via software with the use of developed reconfiguration methodology.

What will happen as a result is that the same hardware resource will perform automatic switches from one architectural configuration to another in order to achieve a significant performance improvement because of the creation of tightly fitted computer structures and the release of redundant resources into computations of new programs.

Therefore, development of the software which, on the one hand, accomplishes actual system reconfigurations and which, on the other hand, finds a sequence of matching hardware computing structures that must be assumed in computations becomes the cornerstone of this approach.

The authors' solution to this formidable task involves developing the following:

- **Comprehensive reconfiguration methodology** which allows the authors to perform very fast reconfigurations in dynamic multicomputer systems/multicomputer networks and fault-tolerant reconfigurations. The latter are aimed at turning off the faulty modules from computations. The structure used previously is preserved. Its performance is organized on a reduced level with faulty modules being turned off.
- **Program preprocessing techniques** for dynamic multicomputer systems and multicomputer networks. The algorithms are developed to allow automatic construction of the sequence of architectural states that can be assumed during computations. The authors call this sequence the **reconfiguration flow chart**. Each state of the reconfiguration flow chart is understood as a set of concurrent dynamic computers with the word sizes selected by the programmer via software. The authors present techniques to accommodate both static programs and those arriving during computations (dynamic programs). For the dynamic programs, the flow chart constructed for static programs is modified by inserting dynamically created states that take into account the resource requirements of dynamic programs.

All in all, this volume contains highly original research material on supercomputing systems with dynamic architecture for use by hardware and software engineers in designing such systems and in performing their extensive software development.

The end result is to take advantage of the reconfigurability of the hardware for unlocking a new and heretofore unused source of performance improvement for the applications with very demanding requirements on supercomputer power, of which mission critical computations is a particular case.

Preface

This book is dedicated to the description of the principal software tools for dynamic architectures, which are called **reconfiguration software**. We will discuss the two categories of reconfiguration software:

I. reconfiguration methodology
II. reconfiguration flow chart

Another major subject of this book is associated with extensive algorithm development aimed at performing comprehensive comparison computation made by dynamic architectures with those performed by conventional systems having similar resource complexity. The results of these comparisons are either in concrete speed-up figures expressed in percentages or in other valid demonstrations of superior computations shown by dynamic architectures, if concrete numerical percentages cannot be obtained due to the multiplicity of alternative ways computations can be exhibited by conventional systems.

The composition of this book is as follows. Chapter I, **Motivation**:

a. Introduces the problem of mission-critical Supercomputing systems which can be resolved only with the use of dynamic architectures.
b. Finds desirable characteristics for dynamic architectures in mission-critical applications.
c. Shows that dynamic architectures possess powerful capabilities for implementing most useful architectural features for reconfigurable architectures in mission-critical applications.

Chapter II, **Reconfiguration Software**, discusses the two categories of reconfiguration software introduced above. It gives a comprehensive description of the reconfiguration methodology applicable for architectural and fault-tolerant reconfigurations. General problems of these reconfigurations are presented in Sec. 1. For the architectural reconfigurations introduced in Sec. 2, the following techniques are introduced:

a. Multicomputer reconfigurations
b. Network reconfigurations

For fault-tolerant reconfiguration (Secs. 5 through 7), we discuss the formation of gracefully degraded reconfigurable binary trees with the use of reconfigurations.

Chapter III introduces the software techniques leading to an automatic construction of the **reconfiguration flow chart** for two types of systems; (a) **dynamic multicomputer systems** (Secs. 3 and 5) and (b) **dynamic networks organized as reconfigurable binary trees** (Sec. 4).

For dynamic multicomputer systems, the construction of a reconfiguration flow chart leads toward the minimization of program delays and the total time of executing all programs from a given program mix. For reconfigurable binary trees, the construction of such a flow chart leads to a significant data-exchange optimization, because it is possible to select such tree configurations as consecutive states in this flow chart in which each pair of nodes with large data exchanges can be connected with the minimal communication path of length 1 (i.e., these two nodes become adjacent in a selected tree configuration).

Also, for dynamic multicomputer systems, we discuss two types of reconfiguration flow charts: **static**, in Sec. 3, and **dynamic**, in Sec. 5. A **static reconfiguration flow chart** takes into account the reconfiguration requests of those programs called **static** that are available in the system before the beginning of the procedures aimed at automatic construction of this flow chart.

A **dynamic reconfiguration flow chart** considers reconfiguration requests of those programs called **dynamic** that arrive at the system when it is executing a reconfiguration flow chart. Thus a dynamic reconfiguration flow chart can be conceived as a modified static flow chart with newly added architectural states created dynamically that take into account reconfiguration requests of dynamic programs.

The objectives of Chapter IV are:

1. To outline the effect of dynamic reconfiguration on some popular ADA constructs
2. To give memory management techniques for data structures in relational data bases that are created dynamically

Section 1 is dedicated to handling, in dynamic architectures, such well-known ADA constructs as:

a. ADA packages (Sec. 1.1)

b. Task rendezvous (Sec. 1.2)
c. Exceptions handling (Sec. 1.3)

Section 2 introduces the topic of memory allocation in a relational data base implemented as a dynamically reconfigurable multiprocessor system including content-addressable memories. The description of such a relational data base is made in Sec. 2.2. Section 2.3 performs the classification of all allocation schemes. Section 2.4 addresses the problem of file interference and Secs. 2.5 and 2.6 devise optimal allocations for various types of files introduced in Sec. 2.3.

Chapter V, **Algorithm Development**, discusses the organization of computations in dynamic architectures for the following classes of algorithms:

a. General-purpose program mix made of concurrent programs with no data dependencies (Sec. 2.1)
b. Parallel construct fork–join (Sec. 2.2)
c. Producer–consumer algorithms (Sec. 3)
d. Array computations encountered in the solution of relaxation equations and pulse deinterleaving algorithms (Sec. 4)
e. Data-base management in tree data bases (Sec. 5)
f. Real-time sorting (Sec. 6)
g. Tree-structured algorithms (Sec. 7)

Finally, following Chapter V we present the conclusions to this volume. Overall conclusions are divided into the following topical areas presented in respective sections:

a. Conclusions on the reconfiguration flow chart and reconfiguration methodology
b. Conclusions on algorithm development

Svetlana P. Kartashev
Steven I. Kartashev

Contents