Christian Lengauer, Patrice Quinton,
Yves Robert, Lothar Thiele (editors):

Parallelization Techniques for Uniform Algorithms

Dagstuhl-Seminar-Report; 66
21.06.-25.06.93 (9325)

ISSN 0940-1121

Tel.: +49-6871 - 2458
Fax: +49-6871 - 5942

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Bezugsadresse: Geschäftsstelle Schloss Dagstuhl
Universität des Saarlandes
Postfach 15 11 50
D-66041 Saarbrücken, Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397
e-mail: office@dag.uni-sb.de

Dagtuhl Workshop
on
Parallelization of Regular Algorithms

Organized by

Christian Lengauer (Universität Passau)
Patrice Quinton (INRIA, Rennes),
Yves Robert (ENSL, Lyon),
Lothar Thiele (Universität des Saarlandes)

Schloß Dagstuhl 21.6. – 25.6.1993

Contents

1 Preface

Recently the two research areas of parallelizing compilation and regular array design have found common ground when, with the help of researchers in both areas, the parallelization of nested loops could be put on a theoretical basis in linear programming, linear algebra, convex polyhedra and formal semantics. At the same time, the relevance of loop parallelization has increased due to advances in parallel (particularly massively parallel) processor technology.

This workshop aimed to bring the two research communities even closer together and provide a forum for the discussion of questions like:

- Combine research areas of parallelizing compilation and regular array design. In particular, mathematical models and methods have been compared, i.e. linear (integer) programming, linear (integer) algebra, convex polyhedra, formal semantic and automated proving for construction and verification of parallelizing program transformations.
- What techniques in either area are relevant for loop parallelization? Constraints implied by different target architectures have been taken into account.
- What is the impact of recent developments in parallel hardware on loop parallelization (and vice versa)?
- What degree of automation can be expected in loop parallelization and what shape do (should) loop parallelization tools take?

Some of the presentations will be published within a special issue of "Parallel Processing Letters" in Spring 1994.

The 40 participants of this workshop came from 8 countries, i.e. 11 from Germany, 17 from other European countries, 1 from Canada and 11 from US. The organizers would like to thank everyone who has helped to make the seminar a great success.

Christian Lengauer Patrice Quinton Yves Robert Lothar Thiele

2 Abstracts

Optimal Tiling of Dynamic Programming Recurrences on a Ring of Processors

Rumen Adonov and Sanjay Rajopadhye
INRIA, Rennes

We discuss the implementation of a class of dynamic programming algorithms on a ring. Systolic solutions for these algorithms are known, but are not efficient on general purpose processors because of high communication costs. We develop a tiling approach for the corresponding class of recurrences and present an optimization method for reducing synchronization costs and communication overhead. First we analytically determine the optimal size of the tile which minimize the parallel execution time as a function of problem size and number of processors. Then we show that we are able to also find the optimal number of processors. Since the results are analytic, they can easily by used in parallelizing compilers to determine tile sizes automatically.

The Complexity of analyzing Data Dependencies in Systems of Bounded Uniform Recurrence Equations

Wolfgang Backes
Universität des Saarlandes

My talk is concerned with the determination of longest paths in periodic graphs. Periodic graphs are infinite graphs with a regular structure. They are described by a finite directed graph, usually called the static graph. I do stress the analysis of the structure of longest paths, i.e. we show that the regular topology of periodic graphs is reflected in a regularity of longest paths. This regularity allows a finite representation of infinitely many longest paths. The major work consists is exactly describing what this periodicity looks like. The analysis of longest paths in periodic graphs is completely based on an investigation of the corresponding static graph.

Transformation of Nested Loops with Modulo Indexing to Affine Recurrences

Florin Balasa, Frank H.M. Franssen, Francky V.M. Catthoor and Hugo J. De Man
IMEC VSDM, Leuven

For multi-dimensional (M-D) signal and data processing systems, transformation of algorithmic specifications is a major instrument both in code optimization and code generation for parallelizing compilers and in control flow optimization for (application specific) architecture synthesis. State-of-the-art transformation techniques are limited to affine index expressions. This is however not sufficient for many important applications in image, speech, and numerical processing. In this paper, a novel transformation method is introduced, oriented to the subclass of algorithm specifications that contains modulo expressions of affine functions to indexed M-D signals. The method employs extensively the concept of Hermite normal form. The computational complexity is polynomial and only integer arithmetic needs to be applied.

Piece-Wise Linear Program Transformation

Young-il Choo
Yale University

Data fields are functions defined over finite index domains that represent computations over distributed data structures. The index domains embody the shape of the computation. As functions, data fields can be defined recursively and transformed according to simple algebraic rules. Mappings between index domains can be used to specify various types of program optimizations for distributed memory machines. Within this framework, Warshall's algorithm for computing the transitive closure can be succinctly specified, but its naive implementation is very inefficient. After applying a standard method to eliminate broadcasting, we showed how to transform the program into a new one with a more efficient piece-wise linear schedule. The key insight is in formulating the piece-wise linear domain morphism. Next, the inverse needs to be defined. Then, the new program can be described in a purely algebraic way.

A Graph Theoretic Approach to the Alignment Problem

Alain Darte and Yves Robert
ENSL, Lyon

This talk deals with the problem of aligning data and computations when mapping uniform or affine loop nests onto SPMD distributed memory parallel computers. For affine loop nests we formulate the problem by introducing the communication graph which can be viewed as the counterpart for the mapping problem of the dependence graph for scheduling. We illustrate the approach with several examples to show the difficulty of the problem. In the simplest case, that of perfect loop nests with uniform dependences, we show that minimizing the number of communications is NP-complete, although we are able to derive a good allocation matrix in most practical cases.

Delay Estimation and Clocking for Structured Circuits

Jean-Marc Delosme
Yale University

Digital circuits often consist of several interconnected arrays consisting of translations of identical cells. Closed form expressions -in terms of cell delays and arrays dimensions- that approximate the delay between circuit inputs and outputs enable quick performance evaluation for such circuits. These expressions may be used to compare different circuit structures and guide the design of cells. The critical paths within the arrays are found by first enumerating all possible paths, without introducing extraneous ones, assuming unbounded arrays. Then the constraints on these paths due to the physical boundaries of the arrays are introduced as well as the time at which the inputs are available. Under reasonable modeling assumptions the delay estimation amounts to solving many small, parameterized, linear programs. This set up enables the interactions between arrays to be handled via a simple relaxation technique. The technique suggests an approach for obtaining near-optimal pipelined or bit-serial implementations of such circuits, by combining a search over tes within the cells to roughly equalize their delays (and approximating the delay of these groups by their upper bound) with the application of the delay estimation procedure. The resulting approximated computation wavefronts specify where the registers should be inserted. The regularity (piecewise

linearity) of these schedules facilitates the layout of pipelined systems and is essential to the derivation of bit-serial implementations.

Estimating Data Locality in Loops

F. Bodin, C. Eisenbeis, W. Jally and D. Windheiser
INRIA, Rocquencourt and Rennes

One major point in loop restructuring for data locality optimization is the choice and the evaluation of data locality criteria. In this talk, we show how to compute approximations of window sets defined by Garron, Jalky and Gallivan. The window associated with an iteration i describes the "active" portion of an array: elements that have already been referenced before iteration i and that will be referenced after iteration i. Such a notion is extremely useful for data localization because it identifies the portion of arrays that are worth keeping in local memory because they are going to be referenced later. The computation of these window approximations can be performed symbolically at compile time and generates a simple geometrical shape that simplifies the management of the data transfers. This strategy allows derivation of a global strategy of data management for local memories which may be combined efficiently with various parallelization and/or vectorization optimizations. Indeed, the effects of loop transformations fit naturally into the geometrical framework we use for the calculations. The determination of window approximations is studied both from a theoretical and a computational point of view and examples of applications are given.

Toward Automatic Distribution

Paul Feautrier
Laboratoire MASI

This paper addresses the problem of distributing data and code among the processors of a distributed memory supercomputer. Provided that the source program is amenable to data flow analysis, one may identify potential communication and set up the problem of removing as many exchanges as possible by clever placement of the data. Such a placement is specified by a function which gives the identity of the virtual processor on which each elementary calculation is executed. One has then to "realize" the virtual processors on the PE. The placement function is obtained by solving a system of over-determined linear homogeneous equations by a process analogous to Gaussian

elimination. The resulting program satisfies the "owner computes rule" and is reminiscent of two level distribution schemes like HPF's ALIGN and DISTRIBUTE, or the CM-2 virtual processor system

On the Parallelization of While-Loops

Martin Griebel
Universität Passau

WHILE-loops can be interpreted as FOR-loops with a varying upper bound which has to be evaluated after every iteration. The index domain of a perfect loop nest containing WHILE-loops is a polyhedron (not a polytope) of which at run time only some -in general not convex- subspace, called the execution space is executed. The loop nest is parallelized via an affine space-time-mapping. Then the transformed execution space is converted into a nest of target loops. Since this is not always possible without enumerating holes a criterion for the transformation matrix is presented to allow a precise scanning.

Analysis of Uniform Dependence Structures

Franz Höfting
Universität GH Paderborn

A formal modelling of the dependencies between variable instances in terms of dependence graphs and reduced dependence graphs is given and restated in terms of matrices. Thereby only uniform unconditional systems without parameters are considered. The matrices representing these form of dependencies are then used to formulate questions on properties of dependence structures as systems of linear equations,

$$\begin{aligned} Ax &= b \\ x &\neq \vec{0} \\ x & \quad \text{nonnegative integer} \end{aligned}$$

where A is defined by the dependency structure, b is some integer vector and the solution x must observe a certain connectivity property. Properties of interest are, for instance, computability, parallelizability of two variable instances and determining the best possible computation time of the dependency system. By means of the system of equations it is shown that the problem mentioned above can be solved in nondeterministic polynomial time. It is also shown that in several restricted versions the problem

remain NP-hard. For the practically relevant restriction where the dimension of the problem is a constant and the index dependencies are given in unary the best possible value for a schedule of the dependency system can be computed in polynomial time. The results have been published in the proceedings of SODA 93.

Dynamic Loop Based Scheduling for Hardware Description

Ahmed Amine Jerraya
INPG/TIMA, Grénoble

We present an algorithm that can effectively schedule large control-flow graphs. This algorithm is an improvement of the trace scheduling developed by Fisher (1981). The new algorithm allows the inclusion of loop feed back edges in the control flow graphs. It also interrupts the generation of traces on the fly. The latter eliminates the generation of false paths thereby avoiding the path explosion problem. These improvements make the algorithm able to handle complex descriptions with an acceptable response time.

The OMEGA Project

William Pugh, Wayne Kelly, Vadim Maslov and Dave Wommacott
University of Maryland

We first give a summary of our research, including:

- The Omega test: algorithms dealing with linear constraints over integer variables.
- Standard array reference alias detection.
- Analysis of value-based array data dependences.
- Narrowing the search for unexplored parallelism.
- Unified framework for reordering transformations.

We then describe our system for transforming programs which has two parts:

1. *THE FRAMEWORK:* We present a framework for unifying iteration reordering transformations such as loop interchange, loop distribution, skewing, tiling, index set splitting and statement reordering. The framework is based on the idea that a transformation can be represented as a schedule which maps the original iteration space to a new iteration space. The framework is designed to provide a uniform way to represent and reason about transformations. As part of the framework, we provide algorithms to assist in the building and use of schedules. In particular, we provide algorithms to test the legality of schedules, to align schedules and to generate optimized code.

2. *DECIDING WHICH TRANSFORMATION TO APPLY:* We show how we can estimate the performance of a program by considering only the schedule from which it was produced. We also show how to produce an estimate of the maximum performance obtainable by extending a partially specified schedule. Our ability to estimate performance directly from schedules and to do so even for partially specified schedules allows us to efficiently find schedules which will produce good code.

Representing Lattice Domains in ALPHA

Hervé le Verge and Patrice Quinton
Universität Passau and IRISA, Rennes

The synthesis of regular iterative algorithms relies basically upon the expression and transformation of systems of recurrence equations. Another equivalent formalism is based upon a language of functions mapping integral polyhedra (their domains) onto some value type. This approach has been taken in Alpha, Crystal, and more recently, in PEI. One problem which remains largely unsolved is the possibility of expressing functions on more general domains, such as for example lattices. This talk deals with this problem. We present the Alpha language, its properties. We illustrate this problem called "lattice extension" on the description of a multi-rate system. We then presents solution to this problem based on the possibility to describe lattices as projections of higher dimensional dense lattices. This extension can be used to model efficiency deficient systolic arrays, two-level pipelined arrays, and multi-rate systems.

Formal Extraction of Concurrent Systems

Brian McConnell
University of Edinburgh

We present an overview of the FECS (Formal Extraction of Concurrent Systems) project. FECS aims to analyze formally strategies for developing parallel systems from abstract specifications. In this talk we present an approach for deriving synchronous systems from specifications in a language based on first order equational logic. The approach we follow is a "proofs to programs" approach based on translating a proof of correctness of a specification into a function scheme in the form of a collection of simultaneous primitive recursive equations. We argue that simultaneous primitive recursive equations are a model of a certain class of parallel algorithms, known as Synchronous Concurrent Algorithms (SCAs). Finally, we observe the similarity between generalized simultaneous primitive recursive equations and recurrence equations

(and their standard models) and speculate on the application of transformation techniques for recurrence equations to systems of equations derived from the "proof to programs" approach.

Mapping a Class of Run-Time Dependencies onto Regular Arrays

Graham Megson
University of Newcastle

The production of regular computations using algorithmic engineering techniques is beginning to play an important role in the synthesis of massively parallel and VLSI processor arrays. In this paper we widen the class of algorithms that can be formally synthesized by introducing a mapping theorem for a class of algorithms with run-time dependencies. The technique is illustrated by deriving uniform recurrences for the so called knapsack problem, the resulting systolic array is known to be optimal.

An Approach to Systematic Design of a Variety of Processor Arrays

A. Schubert, R. Merker and H. Schreiber
Technische Universität Dresden

A design method is considered that is aimed at designing a variety of processor arrays for one initial algorithm specification. The initial algorithm specification of our approach is a single assignment code, which may contain global data with non-localized data dependencies in the several sets of statements on several index spaces. The single assignment code is associated with a dependence graph on the iteration-level since it realizes only data dependencies between different iterations. Therefore, the graph holds both local data dependencies and global data references. To handle the inhomogeneity of the graph, control data are used. During the design process different transformations are selected; optimization strategies are used to determine localization of data dependencies. With this approach we provide the possibility to choose from among a variety of "good" processor arrays (near the minimal number of processors and the minimal time) the one best suited to the appropriate application.

Data Compiling from Systems of Recurrence Equations

Catherine Mongenet
Université Lois Pasteur, Straßburg

The objective is to compile systems of recurrence equations in order to get efficient implementations on distributed memory machines. One issue when focusing on distributed memory architectures is to minimize the number of communications, i.e. to distribute the data in such a way that the virtual processors use as much as possible local data. Automatic parallelization of such systems of equations are well-known techniques due to the systolization studies. They are founded on a dependency analysis from which one determines a space/time mapping. This mapping usually describes an affine timing or scheduling function and a linear allocation function. The timing function indicates at which time step an elementary computation of the system is executed while the allocation function specifies on which virtual processor it is executed. Since the number of communications of a parallel solution is related to the allocation, we first

show how to determine an allocation which minimizes the number of communications and then we present how to compile these communications and the virtual processors code. These techniques are applied to systems of uniform recurrence equations and then generalized to systems of affine recurrence equations. They are based on the dependency analysis, more precisely the notion of utilization sets. For a given result item $Y(z_0)$ we call utilization set related to this item, the set of all the point using it as a data for their own computation. The notion of band is then introduced and used to determine an optimal allocation. Once an optimal allocation has been computed and a schedule has been chosen, one can generate the processor code. The data dependencies are compiled either in terms of communications or in terms of local registers. This compiling process uses the dependency information to automatically synthesize the code, in particular the send/receive instructions and the register management.

Reasoning About Parallel Loops With Scan Functions -or- Parallelizing Functional Loops

John T. O'Donell
University of Glasgow

The family of scan functions provides a general mechanism for expressing iteration over arrays. By restricting the function being scanned, we can often transform a general iteration into a more efficient one targeted for an appropriate parallel machine. Depending on the function being scanned, the result may be

1. The scan is inherently sequential because of data dependencies.
2. The scanned function is associative, allowing a log time parallel implementation using a tree machine.
3. The scanned function performs near-neighbor communication, leading to a transformed version that does unit-time communication followed by a parallel map - this may be time $\Theta(1)$.

This approach exploits the expressiveness of functional languages: the original general iteration says nothing about the implementation, while the final result may be a parallel program where the communication and the complexity are made explicit.

Nano-Threads

Constanine D. Polychronopoulos
University of Illinois

In this presentation we address new directions in the development of powerful compilers and operating systems for high-performance multiprocessors, and in particular we will overview the design and characteristics of a new compiler-based threads model. The resulting environment - called auto-scheduling - becomes possible due to recent progress in control dependence analysis and optimization, which will also be reviewed in this presentation. Our work is based on the position that the problem of parallel programming is too complex to be entirely solved by the user, although at the same time, the environment must provide the necessary mechanisms for user feedback/interaction, or in case of expert users, the ability to overwrite the compiler. Full automation of the process of thread packaging, creation and scheduling is achieved via "intelligent" code embedded in user code by the compiler, following extensive data and control dependence analysis.

Rewriting Loops With Parametric Integer Programming

Jean Francois Collard, Paul Feautrier and Tanguy Risset
ENSL, Lyon and Laboratoire MASI

Most parallelization techniques for DO-loop nests are based on re-indexation. Re-indexation yields a new iteration space, which is a convex integer polyhedron defined by a set of affine constraints. Parallel code generation needs this to scan all the integer points of this convex, thereby requiring the construction of a new DO-loop nest. We detail an algorithm for this purpose, which relies on a parameterized version of the dual simplex algorithm. We show how the resulting loop nest and especially the loop bounds can be kept simple and streamlined, thus reducing the control overhead of parallelization to a minimum.

LACS: A Language for Affine Communication Structures

Sanjay Rajopadhye
IRISA, Rennes

We propose a language which provides a unified notation for specifying parallel assignment, communication and reduction operations in massively parallel programs. It is designed around a Parallel Assignment Statement (PAS) and Atomic Communication Events (ACE's), and can be used to describe most common communication libraries such as one-to-one transfer, broadcast, personalized communication structures and scans as APL-style one-liners. LACS uses convex polyhedra and affine transformations. This allows works from linear algebra to be used for compile-time analysis. We can automatically detect whether a PAS is well-formed, has write conflicts etc.. We can also detect the presence of communication schemes like scatters, gathers, reduces, scans and their generalizations.

Verification of Computational Networks

Ivan Sadinsky
Delft University of Technology

The verification objective is set to compare two networks for input-output equivalence. A network communicates with its environment via sequential inputs and outputs and is seen as an asynchronous multi-rate dynamical system. An input-output representation is suggested that is a recurrence which unrolls to an infinite output process of symbolic expressions. An expression maps input token values to an output token value by depth first tree evaluation semantics. For two networks under comparison, the input-output representation establishes formal correspondence between a finite number of expressions over a common alphabet of prime function symbols. The verification process starts with transforming each network to an equivalent sequential program, i.e. to a single node. Next, the sequential program is compiled into the input-output representation by applying transformation primitives such as sequential composition of basic blocks and branch unification, as well as more complex loop transformations. Finally, the corresponding expressions from the input-output representation of two systems are compared. The comparison is a maximal common subgraph problem, which is known

to be polynomial for trees. Eventually, a resulting difference is output as a function equation that becomes a condition for the equivalence of two networks which may be asserted as a mathematical identity by the user.

Accurate Data Dependence Test

Zhaoyun Xing and Weijia Shang
University of Southwestern Louisiana

To test if there is a dependence between different iterations in a loop can be converted to checking if there exist integral points in a polyhedron described by a set of linear equations and inequalities. In this paper, two transformations of data dependence test problems with any number of linear equations to equivalent test problems with only one linear equation are presented. Also, three dependence test methods, interval test, tent function method and sequence test, are proposed. The interval test is motivated by I-test, and is applicable only to test problems with one equation. In the tent function method, a nonnegative and piecewise linear function, called tent function, is defined on the polyhedron whose minimum value is zero iff the polyhedron contains integers. Then a piecewise linear program is used to find the minimum value of this tent function. By using the modified simplex method for linear programming, a polynomial average time method is proposed to test dependences accurately. In the sequence test, data dependence test problems are first transformed to test problem with only one linear equation. Then, a sequence of efficient test methods applicable only to test problems with one equation are used in order of their time complexities. If one of those methods succeeds, then stop. If not, a more expensive method is tried. If all those methods do not work, then the tent function method is applied which always gives accurate results with polynomial average time. The methods reported improve some previously proposed methods in aspects of accuracy, applicability and efficiency. This research was supported in part by Louisiana Education Quality Support Fund under contract number LEQSF(191-93)-RD-A-42 and by the National Science Foundation under Grant MIP-9110940.

Scheduling to Parallelize Loop with Loop-Carried Dependencies

Barbara Simons
IBM, Palo Alto

Loop parallelization, an important tool for automatic parallelization of code written in a sequential language such as Fortran, assigns separate invocations of a loop to different

processors. It is often necessary to delay the start time of loops to avoid violating data dependence. We study a scheduling problem, called the Delay Problem, that approximates the problem of minimizing the delay in the start time of the loops. Our major result is a fast polynomial time algorithm for the case in which the precedence constraints (obtained from the "backwards" loop carried dependences) are a forest of in-trees or a forest of out-trees. Since most of the dependence graphs derived from loop-carried dependences for the Delay Problem are sparse, the algorithm can be used as a heuristic for solving the general Delay Problem as it arises in practice. We prove that the Delay Problem becomes NP-complete when the precedence constraints are a set of arbitrary trees. We also prove that the Delay Problem becomes NP-complete for precedence constraints of independent chains when it is generalized to allow either non-unit execution times or release times and deadlines.

Loop Transformation Using Non-Singular Matrices

Miguel Valero-Garcia
Universitat Politecnica de Catalunya, Barcelona

In this presentation we describe two pieces of work under development at our Department. Both works are related to loop transformation using non-singular matrices. The first work is a solution to the problem of rewriting a loop nest to implement a given transformation represented by a matrix T. Previous methods are restricted to unimodular transformations. Our solution is more general and works even when the transformation matrix is non-unimodular. The second part of the presentation describes a proposal to include loop alignment in the transformation framework based on non-singular matrices. Loop alignment allows to reduce loop carried dependences. This type of transformation requires to treat separately the different statements of the loop body. In the presentation, a methods is described to apply loop alignment combined with an other loop transformation represented by a non-singular matrix T. The method produces the loop nest which implements the combined transformation.

Design of a Co-processor for Recurrent Computations

Benjamin W. Wah
University of Illinois

In this talk we present the design of an application-specific co-processor for algorithms expressed as uniform recurrences or nested loops with constant dependencies. The

co-processor is simple with a regular array of processors connected to an access unit for intermediate storage of data. Our approach is based on a parameter-based method for synthesizing optimal arrays of lower dimension from a general uniform recurrence. Constraints on parameters are derived to avoid data collisions in lower dimensional arrays. Our parameter-based method also allows tradeoffs between the completion time of evaluating a recurrence and the number of processors required. We present results on tradeoffs between reduction in clock rate and area of a chip for implementing a design.

The Complexity of Analyzing Data Dependencies in Systems of Bounded Uniform Recurrence Equations

Egon Wanke
GMD, St. Augustin

We consider systems of uniform recurrence equations

$$U(x) := f(U(I_1(x)), \ldots, U(I_k(x))) \qquad \text{if} \quad x \in D$$

whose index domains D are finite, whenever the computation functions f are constant. We analyze the complexity of the following three problems: Given a system S of bounded uniform recurrence equations and two variable instances $I(x)$, $U(y)$. Has the dependence graph G of S a cycle? Has G a path from $U(x)$ to $U(y)$? How long is a longest path from $U(x)$ to a variable instance that does not depend on further variable instances? It is shown that all three questions are PSPACE-complete even (1.) for 2-dimensional systems in that all equations with non-constant computation functions are defined for index domain $\{x \in \mathbf{Z}^2 \mid \vec{0} \leq x \leq m\}$ for some $m \in \mathbf{Z}^2$, (2.) for systems of the form

$$U(x) := \begin{cases} f(U(I_1(x)), \ldots, U(I_k(x))) & \text{if} \quad x \in \{0,1\}^d \\ 1 & \text{else} \end{cases}$$

and (3.) for 4-dimensional systems of the form

$$U(x) := \begin{cases} f(U(I_1(x)), \ldots, U(I_k(x))) & \text{if} \quad x \in D \\ 1 & \text{else} \end{cases}$$

where $\{x \in \mathbf{Z}^4 \mid \vec{0} \leq x \leq m\}$ for some $m \in \mathbf{Z}^4$.

An Experimental Parallelizing Systolic Compiler for Regular Programs

Friedrich Wichmann
Universität Paderborn

Systolic techniques have been applied to the generation of parallel code from regular loop programs in an experimental work to study the practical problems. The compiler uses unimodular index transformations yielding linear mappings to define an execution time $\Pi(\vec{i})$ and an executing processors $S(\vec{i})$ for each iteration $\vec{i} \in D \subseteq \mathbf{Z}^d$ of the loop nest. Its input language allows nested loop programs to be specified in an imperative or functional style and is analyzed by a front-end developed with help of the compiler generation tool-set ELI. Different techniques have been used to derive the systolic mappings: an enumeration method called "vector-guessing", a simplified techniques from [Huang/Lengauer 89] and the method from [Moldovan/Fortes 86]. The selection of "good" solutions can be changed to study the effectiveness of the generated parallel C code for a simulator. This is shown by the convolution example, which are evaluated with respect to different hardware characteristics. Such experimental system is a good basis for development of systolic parallelizing compilers for regular programs or program parts.

Compiling Distributed Loops onto SPMD Code

Vincent Van Dongen
CRIM, Montreal

Given a loop written in a sequential language and a distribution, we present some compiler techniques for generating SPMD code to run on distributed memory machines. Three cases are being considered. First, the loop is parallel and with only one statement. The distribution model to start with is a block distribution. We present a first compilation technique that generates a SPMD code made of two parts: (i) the communication part and (ii) the computation part. We show that each of these parts involves the scanning of parameterized polyhedra that can be written at compile-time as a loop. This technique is then applied to other distribution models such as block-cyclic, alignment etc.

We also present a way of distributing the loop in terms of a folding factor, which defines the number of blocks per processor, and the manner to compile it with scanning polyhedra also. We present a second compilation techniques which performs for each parallel loop the computation first, and then the communications. The communication step is such that it sends all the data produced in the loop to all the processors that will consume them. The motivation behind this is to reduce the communications. The

way to compile the program is by using the producer-consumers relation given by the array data flow analysis. We then consider parallel loops that contain several sequential statements. The first way to compile such a loop is to transform it so that the parallel loop becomes internal. The compiler techniques presented in the previous model are then applicable. Although this transformation can always be achieved, it introduces a lot of communication steps. We show that, under some distribution conditions, it is not necessary and that all the communications can be done in one step. In order to achieve this, we use array data flow analysis. We also present another distribution model which distribute the computations instead of the data. The motivation for this model is that a parallel loop will always be able to be executed with only one communication step. The compiler technique relies on the array data flow analysis and the scanning of polyhedra again. In the third loop model, the parallelism is hidden. That is the loop must be re-indexed in order to make it parallel. Under this more general model, we first consider a compilation strategy that preserves the order of execution of the loop within each processor. However, each processor scans the computations that it owns only. Under a data distribution model, this scanning can be quite expensive if the data to be produced have different distributions. A computation distribution is suggested to improve this. Finally, we consider a compilation strategy that internally performs a loop transformation so that it becomes parallel. Because there is a linear relation between the new indices and the original ones, the compilation strategies developed for parallel loops can now be used. The data distribution model may however be too restrictive because it distributes the variables of the original program. We explain why a computation distribution seems more promising.

Programming Multi-computers

Jan L.A. van de Snepscheut
California Institute of Technology, Pasadena

We describe how the architecture of a fine-grain multi-computer influences software design. The Mosaic C multi-computer is used as an example. Its relevant characteristics are that each node has limited storage capacity and that communication in the form of message passing is fast compared with computation. A consequence is that our programs are organized as a large collection of small processes. We illustrate the development of such programs as a sequence of transformation steps that start with a sequential program and then introduce variables to avoid reevaluation of expressions; next, introduce more variables to reduce interference; partition the program into processes that share variables; and finally modify the processes to communicate via message passing. As an example, a distributed sorting algorithm is derived. Finally, we describe an editor whose purpose is to support the most tedious parts of this transformation process.

The derivation of the same algorithm is redone, as well as a functional programming example. The programmer selects the transformation and the editor carries it out after determining the substitutions needed to make the transformation applicable.

A Mathematical Theory and its Environment for Parallel Programming

Eric Violard
University of Franche-Comté

A lot of programming models have been proposed to deal with parallelism in order to express program transformations and refinements. This justifies to introduce an unifying theory to abstract different notions. This paper presents the main concepts of such a theory called PEI. It includes the definitions of problems, programs and transformation rules. It is founded on the simple mathematical concepts of multi-set and of an equivalence between their representations as data fields. Program transformations are based on this equivalence and defined from a refinement relation. The mathematical basis of this theory leads to an elegant software environment in CENTAUR using MAPLE whose purpose is to transform parallel programs. It is illustrated by two examples: the convolution sum and the Dirichlet product. The second one uses non-affine dependencies that can be easily treated using PEI.

Dagstuhl-Seminar 9325:

List of Participants

Rumen **Andonov**
Université de Rennes
IRISA
Campus de Beaulieu
Avenue du Général Leclerc
F-35042 Rennes Cedex
France
andonov@irisa.fr
tel.: +33-99.84.72.07

Mike **Barnett**
The University of Idaho
Depart. of Computer Science
Laboratory for Applied Logic
Moscow ID 83844-1010
USA
mbarnett@cs.uidaho.edu
tel.: +1-208-885-5524

Francky **Catthoor**
IMEC VSDM
Kapeldreef 75
B- 3001 Leuven
Belgium
catthoor@imec.be
tel.: +32-16-281201

Young-il **Choo**
Yale University
Department of Computer Science
P. O. Box 21 58
Yale Station
New Haven CT 06520-2158
USA
tel.: +1 203-432-6400
choo-young-il@cs.yale.edu

Alain **Darte**
Ecole Normale Superieure de Lyon
Laboratoire LIP/IMAG
46 allee d'Italie
F - 69364 Lyon Cedex 07
France
darte@lip.ens-lyon.fr

Jean Marc **Delosme**
Yale University
Department of Electrical Engineering
Yale Station
New Haven CT 06520
USA
delosme-jean-marc@cs.yale.edu

Ed **Deprettere**
Delft University of Technology
Department of Electrical Engineering
P.O. Box 5031
NL-2600 GA Delft
The Netherlands
ed@dutentb.et.tudelft.nl

Vincent **van Dongen**
CRIM
Bureau 8oo
1801 Av. McGill College
Montreal PQ H3A 2N4
Canada
vandonge@crim.ca
tel.: +1-514-398-1234

Christine **Eisenbeis**
INRIA
Domaine de Voluceau
Rocquencourt
BP 105
F-78153 Le Chesnay Cedex
France
christine.eisenbeis@inria.fr
tel.: +33-1-139 63 55 82

Paul **Feautrier**
Laboratoire MASI
45 Avenue des Etats-Unis
F-78035 Versailles Cedex
France
feautrier@masi.ibp.fr
tel.: +033-1-39254066

José A. B. **Fortes**
Purdue University
School of Electrical Engineering
West Lafayette IN 47944
USA
fortes@ecn.purdue.edu

Martin **Griebl**
Universität Passau
Fachbereich Mathematik / Informatik
Postfach 2540
D-94030 Passau
griebl@fmi.uni-passau.de
tel.: +49-851-509-710

Franz **Höfting**
Universität GH Paderborn
FB 17 - Mathematik/Informatik
Warburgerstr. 100
W-4790 Paderborn
fh@uni-paderborn.de
tel.: +49-228-550-2 18 (an der Uni Bo

Christian **Heckler**
Universität des Saarlandes
Fachbereich 16 - Elektrotechnik
Im Stadtwald 13
W-6600 Saarbrücken 11
heckler@ee.uni-sb.de
tel.: +49-6801-302-3574

Ahmed Amine **Jerraya**
System-Level Synthesis Group
INPG/TIMA
46 Avenue Felix Viallet
F-38031 Grenoble Cedex
France
jerraya@imag.fr
tel.: +33-76-57 47 59 / +33-76-87 61 74

Wayne **Kelly**
University of Maryland
Department of Computer Science
College Park MD 20742
USA
wak@cs.umd.edu
tel.: +1-301-405-2726

Hervé **Le Verge**
Université de Rennes
IRISA
Campus de Beaulieu
Avenue du Général Leclerc
F-35042 Rennes Cedex
France

Christian **Lengauer**
Universität Passau
Fachbereich Mathematik / Informatik
Postfach 2540
D-94030 Passau
lengauer@fmi.uni-passau.de
tel.: +49 851 509-347

Brian **McConnell**
University of Edinburgh
Department of Computer Science
King's Buildings
Mayfield Road
Edinburgh EH9 3JZ
Great Britain
BM@DCS.ED.AC.UK
tel.: +44--31-650 5124

Graham **Megson**
University of Newcastle
Department of Computer Science
Claremont Tower
Claremont Road
Newcastle-upon-Tyne NE1 7RU
Great Britain
graham.megson@newcastle.ac.uk
tel.: +44-91-222-76 53

Renate **Merker**
TU Dresden
Fakultät Elektronik
Institut IEE / Lehrstuhl Systemtheorie
Mommsenstr. 13
O-8027 Dresden
merker@e-technik.tu-dresden.dbp.de
tel.: +49-351-463-3108

Catherine **Mongenet**
Université Straßburg
Departement d'Informatique
7 rue René Descartes
F-67084 Strasbourg
France
mongenet@dpt-info.u-strasbg.fr
tel.: +33-8841-6344

Thomas **Noll**
RWTH Aachen
Fachbereich Informatik
Ahornstr. 55
W-5100 Aachen
noll@zeus.informatik.rwth-aachen.de

John T. **O'Donnell**
University of Glasgow
Department of Computing Science
17 Lilybank Gardens
Glasgow G12 8QQ
Great Britain
jtod@dcs.glasgow.ac.uk

Constantine D. **Polychronopoulos**
The University of Illinois
Center for Supercomputing
Research and Development
1308 West Main Street
Urbana IL 61801
USA
cdp@csrd.uiuc.edu
tel.: 217-244-4144

Patrice **Quinton**
Université de Rennes
IRISA
Campus de Beaulieu
Avenue du Général Leclerc
F-35042 Rennes Cedex
France
quinton@irisa.fr
tel.: +33-99 36 20 00

Sanjay **Rajopadhye**
Université de Rennes
IRISA
Campus de Beaulieu
Avenue du Général Leclerc
F-35042 Rennes Cedex
France
rajopadhye@irisa.irisa.fr

Tanguy **Risset**
Ecole Normale Superieure de Lyon
Laboratoire LIP/IMAG
46 allee d'Italie
F - 69364 Lyon Cedex 07
France
risset@lip.ens-lyon.fr

Yves **Robert**
Ecole Normale Superieure de Lyon
Laboratoire LIP/IMAG
46 allee d'Italie
F - 69364 Lyon Cedex 07
France
yrobert@lip.ens-lyon.fr
tel.: + 33 72 72 83 89

Georg **Sander**
Universität des Saarlandes
Fachbereich 14 - Informatik
Postfach 1150
W-6600 Saarbrücken 11
Germany
sander@cs.uni-sb.de
tel.: +49-681-302 3054

Yvan **Sandinsky**
University of Delft
Department of Electrical Engineering
P. O. Box 50 31
NL-2600 GA Delft
The Netherlands

Weijia **Shang**
University of Southwestern Louisiana
Center for Advanced Computer Studies
P.O. Box 44330
Lafayette LA 70504-4330
USA
sw@cacs.usl.edu

Barbara **Simons**
IBM Corp.
MS/17
1510 Page Mill RD
Palo Alto CA 94304
USA
simons@paloalto.vnet.ibm.com
tel.: +1-415-855-41 75

Jan **van de Snepscheut**
California Institute of Technology
Computer Science 256-80
Pasadena CA 91125
USA
jan@vlsi.cs.caltech.edu
tel.: +1-818-356-42 69

Jürgen **Teich**
Universität des Saarlandes
Fachbereich 16 - Elektrotechnik
Postfach 1150
W-6600 Saarbrücken 11
teich@ee.uni-sb.de
tel.: +49-681-302 3230

Lothar **Thiele**
Universität des Saarlandes
Fachbereich 16 - Elektrotechnik
Postfach 1150
W-6600 Saarbrücken 11
thiele@ee.uni-sb.de
tel.: +49-681-302-3584

Miguel **Valero-Garcia**
Universidad Politécnica de Catalu\~na
Dept. Arquitectura de Computadores
Gran Capita s/n
E-08028 Barcelona
Spain
miguel@ac.upc.es
tel.: +34-3-4016995

Eric **Violard**
Université de Franche-Comte
Laboratoire d'Informatique
Route de Gray
F-25030 Besancon Cedex
France
perin@comte.uucp

Ben **Wah**
The University of Illinois
Coordinated Science Laboratory
1308 West Main Street
Urbana IL 61801
USA
wah@manip.crhc.uiuc.edu
tel.: +1-217-333-35 16

Egon **Wanke**
Gesellschaft für Mathematik und
Datenverarbeitung mbH
Schloß Birlinghoven
Postfach 1316
W-5205 St. Augustin 1
wanke@gmd.de
tel.: +49-2241-142783

Friedrich **Wichmann**
Universität GH Paderborn
FB 17 - Mathematik/Informatik
Warburger Str. 100
33098 Paderborn
fwich@uni-paderborn.de
tel.: +49-5251-60-26 51

Zuletzt erschienene und geplante Titel:

C.A. Ellis, M. Jarke (editors):
Distributed Cooperation in Integrated Information Systems; Dagstuhl-Seminar-Report; 38; 5.4.-9.4.92 (9215)

J. Buchmann, H. Niederreiter, A.M. Odlyzko, H.G. Zimmer (editors):
Algorithms and Number Theory, Dagstuhl-Seminar-Report; 39; 22.06.-26.06.92 (9226)

E. Börger, Y. Gurevich, H. Kleine-Büning, M.M. Richter (editors):
Computer Science Logic, Dagstuhl-Seminar-Report; 40; 13.07.-17.07.92 (9229)

J. von zur Gathen, M. Karpinski, D. Kozen (editors):
Algebraic Complexity and Parallelism, Dagstuhl-Seminar-Report; 41; 20.07.-24.07.92 (9230)

F. Baader, J. Siekmann, W. Snyder (editors):
6th International Workshop on Unification, Dagstuhl-Seminar-Report; 42; 29.07.-31.07.92 (9231)

J.W. Davenport, F. Krückeberg, R.E. Moore, S. Rump (editors):
Symbolic, algebraic and validated numerical Computation, Dagstuhl-Seminar-Report; 43; 03.08.-07.08.92 (9232)

R. Cohen, R. Kass, C. Paris, W. Wahlster (editors):
Third International Workshop on User Modeling (UM'92), Dagstuhl-Seminar-Report; 44; 10.-13.8.92 (9233)

R. Reischuk, D. Uhlig (editors):
Complexity and Realization of Boolean Functions, Dagstuhl-Seminar-Report; 45; 24.08.-28.08.92 (9235)

Th. Lengauer, D. Schomburg, M.S. Waterman (editors):
Molecular Bioinformatics, Dagstuhl-Seminar-Report; 46; 07.09.-11.09.92 (9237)

V.R. Basili, H.D. Rombach, R.W. Selby (editors):
Experimental Software Engineering Issues, Dagstuhl-Seminar-Report; 47; 14.-18.09.92 (9238)

Y. Dittrich, H. Hastedt, P. Schefe (editors):
Computer Science and Philosophy, Dagstuhl-Seminar-Report; 48; 21.09.-25.09.92 (9239)

R.P. Daley, U. Furbach, K.P. Jantke (editors):
Analogical and Inductive Inference 1992 , Dagstuhl-Seminar-Report; 49; 05.10.-09.10.92 (9241)

E. Novak, St. Smale, J.F. Traub (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 50; 12.10.-16.10.92 (9242)

J. Encarnação, J. Foley (editors):
Multimedia - System Architectures and Applications, Dagstuhl-Seminar-Report; 51; 02.11.-06.11.92 (9245)

F.J. Rammig, J. Staunstrup, G. Zimmermann (editors):
Self-Timed Design, Dagstuhl-Seminar-Report; 52; 30.11.-04.12.92 (9249)

B. Courcelle, H. Ehrig, G. Rozenberg, H.J. Schneider (editors):
Graph-Transformations in Computer Science, Dagstuhl-Seminar-Report; 53; 04.01.-08.01.93 (9301)

A. Arnold, L. Priese, R. Vollmar (editors):
Automata Theory: Distributed Models, Dagstuhl-Seminar-Report; 54; 11.01.-15.01.93 (9302)

W. Cellary, K. Vidyasankar, G. Vossen (editors):
Versioning in Database Management Systems, Dagstuhl-Seminar-Report; 55; 01.02.-05.02.93 (9305)

B. Becker, R. Bryant, Ch. Meinel (editors):
Computer Aided Design and Test , Dagstuhl-Seminar-Report; 56; 15.02.-19.02.93 (9307)

M. Pinkal, R. Scha, L. Schubert (editors):
Semantic Formalisms in Natural Language Processing, Dagstuhl-Seminar-Report; 57; 23.02.-26.02.93 (9308)

W. Bibel, K. Furukawa, M. Stickel (editors):
Deduction , Dagstuhl-Seminar-Report; 58; 08.03.-12.03.93 (9310)

H. Alt, B. Chazelle, E. Welzl (editors):
Computational Geometry, Dagstuhl-Seminar-Report; 59; 22.03.-26.03.93 (9312)

H. Kamp, J. Pustejovsky (editors):
Universals in the Lexicon: At the Intersection of Lexical Semantic Theories, Dagstuhl-Seminar-Report; 60; 29.03.-02.04.93 (9313)

W. Strasser, F. Wahl (editors):
Graphics & Robotics, Dagstuhl-Seminar-Report; 61; 19.04.-22.04.93 (9316)

C. Beeri, A. Heuer, G. Saake, S. Urban (editors):
Formal Aspects of Object Base Dynamics , Dagstuhl-Seminar-Report; 62; 26.04.-30.04.93 (9317)

R. V. Book, E. Pednault, D. Wotschke (editors):
Descriptional Complexity, Dagstuhl-Seminar-Report; 63; 03.05.-07.05.93 (9318)

H.-D. Ehrig, F. von Henke, J. Meseguer, M. Wirsing (editors):
Specification and Semantics, Dagstuhl-Seminar-Report; 64; 24.05.-28.05.93 (9321)

M. Droste, Y. Gurevich (editors):
Semantics of Programming Languages and Algebra, Dagstuhl-Seminar-Report; 65; 07.06.-11.06.93 (9323)

Ch. Lengauer, P. Quinton, Y. Robert, L. Thiele (editors):
Parallelization Techniques for Uniform Algorithms, Dagstuhl-Seminar-Report; 66; 21.06.-25.06.93 (9325)

G. Farin, H. Hagen, H. Noltemeier (editors):
Geometric Modelling, Dagstuhl-Seminar-Report; 67; 28.06.-02.07.93 (9326)

Ph. Flajolet, R. Kemp, H. Prodinger (editors):
"Average-Case"-Analysis of Algorithms, Dagstuhl-Seminar-Report; 68; 12.07.-16.07.93 (9328)

J.W. Gray, A.M. Pitts, K. Sieber (editors):
Interactions between Category Theory and Computer Science, Dagstuhl-Seminar-Report; 69; 19.07.-23.07.93 (9329)

D. Gabbay, H.-J. Ohlbach (editors):
Automated Practical Reasoning and Argumentation, Dagstuhl-Seminar-Report; 70; 23.08.-27.08.93 (9334)

A. Danthine , W. Effelsberg, O. Spaniol, (editors):
Architecture and Protocols for High-Speed Networks, Dagstuhl-Seminar-Report; 71; 30.08.-03.09.93 (9335)

R. Cole, E. W. Mayr, F. Meyer a.d.Heide (editors):
Parallel and Distributed Algorithms, Dagstuhl-Seminar-Report; 72; 13.09.-17.09.93 (9337)

V. Marek, A. Nerode, P.H. Schmitt (editors):
Non-Classical Logics in Computer Science, Dagstuhl-Seminar-Report; 73; 20.-24.09.93 (9338)

A. Odlyzko, C.P. Schnorr, A. Shamir (editors):
Cryptography, Dagstuhl-Seminar-Report; 74; 27.09.-01.10.93 (9339)

J. Angeles, G. Hommel, P. Kovács (editors):
Computational Kinematics, Dagstuhl-Seminar-Report; 75; 11.10.-15.10.93 (9341)

T. Lengauer, M. Sarrafzadeh, D. Wagner (editors):
Combinatorial Methods for Integrated Circuit Design, Dagstuhl-Seminar-Report; 76; 18.10.-22.10.93 (9342)