INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Compsys

Compilation and Embedded Computing Systems

Rhône-Alpes

THEME COM

Activity Report

2005

Table of contents

1. Team **1**
2. Overall Objectives **1**
2.1. Overall Objectives 1
3. Scientific Foundations **3**
3.1. Introduction 3
3.2. Optimization for Special Purpose Processors 4
3.2.1. Optimization of Assembly-Level Code 4
3.2.2. Scheduling under Resource Constraints 5
3.3. Platform-Independent Code Transformations 6
3.3.1. Modular Scheduling and Process Networks 6
3.3.2. Theoretical Models for Scheduling and Memory Optimizations 7
3.4. Hardware and Software System Integration 8
3.4.1. Design of Accelerators for Compute-Intensive Applications 8
3.4.2. Hardware Interfaces and On-Chip Traffic Analysis 8
3.4.3. Optimization for Low Power 9
3.5. Federating Polyhedral Tools 10
3.5.1. Developing and Distributing the Polyhedral Tools 10
3.5.2. New Models 11
4. Software **11**
4.1. Introduction 11
4.2. Polylib 11
4.3. Pip 11
4.4. MMAlpha 11
4.5. Syntol 12
4.6. Software Developments for FPGA 12
4.7. Cl@k: Algorithms on Integral Lattices 12
4.8. CLooG: Loop Generation 12
4.9. Register Allocation 13
5. New Results **13**
5.1. Introduction 13
5.2. Variable Coalescing and ssa Form 13
5.3. Register Allocation and ssa Form Properties 14
5.4. Instruction Cache Optimization 14
5.5. On-Chip Traffic Analysis 16
5.6. Data-Flow IP Interface 16
5.7. Memory Reuse and Modular Mappings 16
5.8. Optimal Barrier Placement in Nested Loops 17
5.9. Modular Scheduling 17
5.10. Scheduling for Synthesis 18
6. Contracts and Grants with Industry **18**
6.1. Contract with stmicroelectronics on Register Allocation and Instruction Cache Optimizations 18
6.2. Contract with stmicroelectronics on Special-Purpose Circuit Synthesis 18
6.3. Grant from hp labs 19
7. Other Grants and Activities **19**
7.1. European Community 19
7.2. CNRS Convention with the University of Illinois at Urbana-Champaign (USA) 19
7.3. Procope Convention with Passau University (Germany) 19

7.4. Inria Support for Collaboration with Federal University, Rio Grande Do Sul (Brasil) 19
7.5. Informal Cooperations 19
8. Dissemination **20**
8.1. Introduction 20
8.2. Conferences and Journals 20
8.3. Post-Graduate Teaching 20
8.4. Other Teaching and Responsibilities 21
8.5. Animation 21
8.6. Defense Committees 21
8.7. Workshops, Seminars, and Invited Talks 21
9. Bibliography **22**

1. Team

Compsys exists since January 2002 as part of Laboratoire de l'Informatique du Parallélisme (Lip, UMR CNRS ENS-Lyon UCB-Lyon Inria 5668), located at ENS-Lyon, and as an Inria pre-project. It is now a full Inria project since January 2004. The objective of Compsys is to adapt and extend optimization techniques, primarily designed for high performance computing, to the special case of embedded computing systems.

Team leaders

Alain Darte [DR CNRS, leads the project since September 2005]
Tanguy Risset [CR Inria, has led the project since its start, up to September 2005]

Administrative assistants

Isabelle Antunes [T CNRS, part-time]
Chiraz Benamor [A CNRS, part-time (from November 2004 to March 2005)]

Research scientists

Paul Feautrier [Pr ENS-Lyon]
Fabrice Rastello [CR Inria]
Antoine Fraboulet [MC Insa-Lyon, Inria delegation from September 2005]

Post doc

Fabrice Baray [Post doc Inria, September 2004-August 2005]

Engineer

Édouard Bechetoille [Engineer, March 2005-September 2005]

PhD Students

Florent Bouchez [ENS grant, 2005-...]
Hadda Cherroun [Algerian grant, part-time, 2004-2005]
Nicolas Fournel [MESR grant, 2004-...]
Philippe Grosse [Grant CEA-LETI, Grenoble, 2004-...]
Clément Quinson [BDI CNRS and STMicroelectronics, 2005-...]
Antoine Scherrer [BDI CNRS and STMicroelectronics, 2003-...]

Internships

Benoit Boissinot [First year master degree internship, Spring 2005]
Florent Bouchez [Bachelor and master degree internship, Spring 2005]
Clément Quinson [Master degree internship, Spring-Summer 2005]
Oriol Vidal-Perez [Erasmus Spanish internship, Winter 2005]
Mohit Mangal [IIT Indian internship, Spring 2005]
Kapil Modi [IIT Indian internship, Spring 2005]

2. Overall Objectives

2.1. Overall Objectives

Keywords: *DSP*, *FPGA platforms*, *VLIW processors*, *automatic generation of VLSI chips*, *code optimization*, *compilation*, *linear programming*, *memory optimization*, *parallelism*, *regular computations*, *scheduling*, *tools for polyhedra and lattices*.

An embedded computer is a digital system, part of a larger system (appliances like phones, TV sets, washing machines, game platforms, or larger systems like radars and sonars), which is not directly accessible to the user. In particular, this computer is not programmable in the usual way. Its program, if it exists, has been loaded as part of the manufacturing process, and is seldom (or never) modified.

The objective of Compsys is to adapt and extend code optimization techniques, primarily designed for high performance computing, to the special case of embedded computing systems. Compsys has four research directions, centered on compilation methods for simple or nested loops. These directions are:

- code optimization for specific processors (mainly DSP and VLIW processors);
- platform-independent transformations (including loop transformations for memory optimization);
- silicon compilation and hardware software integration
- development of polyhedra manipulation tools.

These research activities are supported by a marked investment in polyhedra manipulation tools, with the aim of constructing operational software tools, not just theoretical results. Hence the fourth research theme is centered on the development of these tools. We expect that the Compsys experience on key problems in the design of parallel programs (scheduling, loop transformations) and the support of our respective parent organizations (Inria, CNRS, Ministry of Education) will allow us to contribute significantly to the European research on embedded computing systems.

The term *embedded system* has been used for naming a wide variety of objects. More precisely, there are two categories of so-called *embedded systems*: (1) control-oriented and hard real-time embedded systems (automotive, nuclear plants, airplanes, etc.) and (2) compute-intensive embedded systems (signal processing, multi-media, stream processing). The Compsys team is primarily concerned with the second type of embedded systems, which is now referred as *embedded computing systems*. The design and compilation methods proposed by the team will be efficient on compute intensive processing with big sets of data processed in a pipelined way.

Today, the industry sells many more embedded processors than general purpose processors; the field of embedded systems is one of the few segments of the computer market where the European industry still has a substantial share, hence the importance of embedded system research in the European research initiatives.

Compsys' aims are to develop new compilation and optimization techniques for embedded systems. The field of embedded computing system design is large, and Compsys does not intend to cover it in its entirety. We are mostly interested in the automatic design of accelerators, for example optimizing a piece of (regular) code for a DSP or designing a VLSI chip for a digital filter. Compsys' specificity is the study of code transformations intended for optimization of features that are specific to embedded systems, like time performances, power consumption, die size. Our project is related to code optimization (like the Inria project Alchemy), and to high-level architectural synthesis (like the Inria project R2D2).

Our priority towards embedded software is motivated by the following observations:

- The embedded system market is expanding. Among many factors, one can quote pervasive digitalization, low cost products, appliances, etc.
- Software engineering for embedded systems is not well developed in France, especially if one considers the importance of actors like Alcatel, STMicroelectronics, Matra, Thales, and others.
- Since embedded systems have an increasing complexity, new problems are emerging: computer aided design, shorter time-to-market, better reliability, modular design, and component reuse.

Recently, several tools for high-level synthesis have appeared in the synthesis/compiler community. These tools are mostly based on C or C++ (SystemC [1], VCC, and others). The support for parallelism in these tools is minimal, but academic projects are more concerned: Flex [2] and Raw [3] at the MIT, Piperench [4] at the Carnegie-Mellon University, PiCo from the HP Labs and now at the Synfora [5] start-up, Compaan [6] at the University of Leiden, and others.

The basic problem that these projects have to face is that the definition of *performance* is more complex than in classical systems. In fact, it is a multi-criteria optimization problem and one has to take into account

[1]http://www.systemc.org/
[2]http://www.flex-compiler.lcs.mit.edu/
[3]http://www.cag.lcs.mit.edu/raw/
[4]http://www.ece.cmu.edu/research/piperench/
[5]http://www.synfora.com/
[6]http://embedded.eecs.berkeley.edu/compaan/

the execution time, the size of the program, the size of the data structures, the power consumption, the manufacturing cost, etc. The incidence of the compiler on these costs is difficult to assess and control. Success will be the consequence of a detailed knowledge of all steps of the design process, from a high-level specification to the chip layout. A strong cooperation between the compilation and chip design communities is needed.

Computer aided design of silicon systems is a wide field. The main expertise in Compsys is in the *parallelization* and optimization of *regular computations*. Hence, we will target applications with a large potential parallelism, but we will attempt to integrate our solutions into the big picture of CAD environments for embedded systems. This is an essential part of Compsys activities and will be a test of the project success.

3. Scientific Foundations

3.1. Introduction

Twenty years ago, the subject of compilation was considered to be mature enough to become an industry, using tools like Lex and Yacc for syntax analysis, and Graham-Glanville code-generator generators. The subject was reactivated by the emergence of parallel systems and the need for automatic parallelizers. The hot topic is now the intermediate phase between syntax analysis and code generation, where one can apply optimizations, particularly those that exploit parallelism, either in an autonomous way or with the help of the programmer. In fact, there is parallelism in all types of digital systems, from supercomputers to PCs to embedded systems.

Compilation consists in a succession of code transformations. These transformations are applied to an intermediate representation that may be very similar to the source code (high-level optimization), or very similar to machine code (assembly code and even register transfer level (RTL) for circuit specification). Almost always, the main constraint is that the meaning (or semantics) of the source program must not be altered. Depending on the context, one may have to express the fact that the degree of parallelism must not exceed the number of available resources (processors, functional units, registers, memories). Finally, the specification of the system may enforce other constraints, like latency, bandwidth, and others. In the case of a complex transformation, one tries to express it as a constrained optimization problem.

For instance, in automatic parallelization, the French community has mainly targeted loop optimization. If the source program obeys a few regularity constraints, one can obtain linear formulations for many of the constraints. In this way, the optimization problem is reduced to a linear program to be solved either over the rationals, or, in few cases, over the integers. These are well-known techniques, based on the theory of convex polyhedra – hence the name *polyhedral model* which is often affixed to the method. Based on this theory, efficient software tools have been developed. One- and multi-dimensional scheduling techniques [21], [22] are an outcome of this research and are ubiquitously used for handling nested loop programs (regular circuit synthesis, process networks for instance).

Extending these methods to embedded systems is difficult because the objective function is complex to express. Performance, for instance, is no longer an objective but a constraint, the goal being to minimize the "cost" of the system, which may be a complex mixture of the design, the manufacturing, and the operation costs. For instance, minimizing the silicon area improves the yield and hence decreases the manufacturing cost. Power consumption is an important factor for mobile systems. Computer scientists are used to a paradigm in which the architecture is fixed and the only free variable is the program. The critical problem is thus to extend our optimization methods to handle many more free variables, mostly of a discrete nature.

In parallel with compiler research, the circuit design community has developed its own design procedures. These techniques have as input a structural specification of the target architecture, and use many heavy-weight tools for synthesis, placement, and routing. These tools mainly use sophisticated techniques for boolean optimization and do not consider loops. When trying to raise the level of abstraction, circuit designers have introduced the terms *architectural synthesis* and *behavioral synthesis*, but the tools did not follow, due to the above mentioned problems (increasing complexity of the constraints, increasing number of free variables).

Technological advances in digital electronics have motivated the emergence of standards for design specifications and design methodologies. Languages like VHDL, Verilog, and SystemC have been widely accepted. The concepts of off-the-shelf components (intellectual property or IP) and of platform-based design are gaining importance. However, the problem remains the same: how to transform a manual design process into a compilation process?

The first proposal was to use several tools together. For instance, the hardware-software partitioning problem is handled by architecture explorations, which rely on rough performance estimates, and the degree of automation is low. But since the complexity of systems on chip still increases according to Moore's law, there is a pressing need to improve the design process and to target other architectures, like DSP, or reconfigurable FPGA platforms. The next generation of systems on chip will probably mix all the basic blocks of today's technology (DSP, Asic, FPGA, network, and a memory hierarchy with many levels). We intend to participate in the design and programming of such platforms.

Our vision of the challenges raised by these new possibilities is the following: one has to *understand* the technological constraints and the existing tools in order to *propose* innovative, efficient, and realistic compilation techniques for such systems. Our approach consists in modeling the optimization process as precisely as possible and then in finding powerful techniques to get an optimal solution. Past experience has shown that taking simultaneously all aspects of a problem into account is nearly impossible.

Compsys has four research directions, each of which is a strong point in the project. These directions are clearly not independent. Their interactions are as follows: "Platform-Independent Code Transformations" (Section 3.3) is on top of "Optimization for Special Purpose Processors" (Section 3.2) and "Hardware and Software System Integration" (Section 3.4), since its aim is to propose architecture-independent transformations. "Federating Polyhedral Tools" (Section 3.5) is transversal because these tools are useful in all other research axes.

3.2. Optimization for Special Purpose Processors

Participants: Alain Darte, Fabrice Rastello, Paul Feautrier.

Applications for embedded computing systems generate complex programs and need more and more processing power. This evolution is driven, among others, by the increasing impact of digital television, the first instances of UMTS networks, and the increasing size of digital supports, like recordable DVD. Furthermore, standards are evolving very rapidly (see for instance the successive versions of MPEG). As a consequence, the industry has rediscovered the interest of programmable structures, whose flexibility more than compensates for their larger size and power consumption. The appliance provider has a choice between hard-wired structures (Asic), special purpose processors (Asip), or quasi-general purpose processors (DSP for multimedia applications). Our cooperation with STMicroelectronics leads us to investigate the last solution, as implemented in the ST100 (DSP processor) and the ST200 (VLIW DSP processor).

3.2.1. Optimization of Assembly-Level Code

Compilation for embedded processors is difficult because the architecture and the operations are specially tailored to the task at hand, because the amount of resources is strictly limited and, lastly, because one would like to take the time to implement costly solutions (so, looking for complete, even expensive, studies is mandatory). For instance, predication, the potential for instruction level parallelism (SIMD, MMX), the limited number of registers and the small size of the memory, the use of direct-mapped instruction caches, but also the special form of applications [19] generate many open problems. Our goal is to contribute to the understanding and the solution of these problems. Since the program is loaded in permanent memory (ROM, Flash, etc.), its compilation time is not so significant. In these conditions, it is interesting to use aggressive and costly compilation techniques, including the use of exact solutions to NP-hard problems.

Our aim is, in particular, to find exact or heuristic solutions to *combinatorial* problems that arise in compilation for VLIW and DSP processors, and to integrate these methods into industrial compilers for DSP processors (mainly the ST100 and ST200). Such combinatorial problems can be found for example in register allocation, in opcode selection, or in code placement for optimization of the instruction cache. Another example is the problem of removing the multiplexor functions (known as φ functions) that are inserted when converting into SSA form (“Static Single Assignment” [33]). These optimizations are usually done in the last phases of the compiler, using an assembler-level intermediate representation. In industrial compilers, they are handled in independent phases using heuristics, in order to limit the compilation time. We want to develop new aggressive techniques based on a more global understanding, the main tool being the SSA representation.

More generally, we want to investigate the interaction of register allocation, coalescing, and spilling, with the different code representations, such as SSA. One of the challenging features of today’s processors is predication [25], which interferes with all optimization phases, as the SSA form does. Many classical algorithms become inefficient for predicated code. This is especially surprising, since, besides giving a better tradeoff between the number of conditional branches and the length of the critical path, converting control dependences into data dependences increases the size of basic blocks and hence creates new opportunities for local optimization algorithms. One has first to adapt classical algorithms to predicated code [34], but also to study the impact of predicated code on the whole compilation process. What is the best time and place to do the if conversion? Which intermediate representation is the best one? Is there a universal framework for the various styles of predication, as found in VLIW and DSP processors?

3.2.2. Scheduling under Resource Constraints

The degree of parallelism of an application and the degree of parallelism of the target architecture do not usually coincide. Furthermore, most applications have several levels of parallelism: coarse-grained parallelism as expressed, for instance, in a process network (see Section 3.3.1), loop-level parallelism, which can be expressed by vector statements or parallel loops, instruction-level parallelism as in “bundles” for Epic or VLIW processors. One of the tasks of the compiler is to match the degree of parallelism of the application and the architecture, in order to get maximum efficiency. This is equivalent to finding a schedule that respects dependences and meets resource constraints. This problem has several variants, depending on the level of parallelism and the target architecture.

For instruction-level parallelism, the classical solution, which is found in many industrial compilers, is to do software pipelining using heuristics like modulo scheduling. This can be applied to the innermost loop of a nest and, typically, generates code for an Epic, VLIW, or super-scalar processor. The problem of optimal software pipelining can be exactly formulated as an integer linear program, and recent research has allowed many constraints to be taken into account, as for instance register constraints. However the codes amenable to these techniques are not fully general (at most one loop) and the complexity of the algorithm is still quite high. Several phenomena are still not perfectly taken into account. Some examples are register spilling and loops with a small number of iterations. One of our aims is to improve these techniques and to adapt them to the STMicroelectronics processors.

It is not straightforward to extend the software pipelining method to loop nests. In fact, embedded computing systems, especially those concerned with image processing, are two-dimensional or more. Parallelization methods for loop nests are well known, especially in tools for automatic parallelization, but they do not take resource constraints into account. A possible method consists in finding totally parallel loops, for which the degree of parallelism is equal to the number of iterations. The iterations of these loops are then distributed among the available processors, either statically or dynamically. Most of the time, this distribution is the responsibility of the underlying runtime system (consider for instance the “directives” of the OpenMP library). This method is efficient only because the processors in a supercomputer are identical. It is difficult to adapt it to heterogeneous processors executing programs with variable execution time. One of today’s challenges is to extend and merge these techniques into some kind of multi-dimensional software pipelining or resource-constrained scheduling. In the Syntol research prototype (see Section 4.5), we are exploring several heuristics for this problem. One of them, which has already been used for reducing register pressure in software pipelining, consists in adding virtual dependences to reduce parallelism. Another method consists in exhibiting a large set of schedules that satisfy the resource constraints, and then selecting in this set those that satisfy the dependence constraints.

3.3. Platform-Independent Code Transformations

Participants: Fabrice Baray, Alain Darte, Paul Feautrier, Antoine Fraboulet, Tanguy Risset.

Embedded systems generate new problems in high-level code optimization, especially in the case of loop optimization. During the last 20 years, with the advent of parallelism in supercomputers, the bulk of research in code transformation was mainly concerned with parallelism extraction from loop nests. This resulted in automatic or semi-automatic parallelization. It was clear that there were other factors governing performance, as for instance the optimization of locality or a better use of registers, but these factors were considered to be less important than parallelism extraction, at least to understand the foundations of automatic parallelization. Today, we have realized that performance is a consequence of many factors, and, especially in embedded systems, everything that has to do with data storage is of prime importance, as it impacts power consumption and chip size.

In this respect, embedded systems have two main characteristics. First, they are mass produced. This means that the balance between design costs and production costs has shifted, giving more importance to production costs. For instance, each transformation that reduces the physical size of the chip has the side-effect of increasing the yield, hence reducing the manufacturing cost. Similarly, if the power consumption is high, one has to include a fan, which is costly, noisy, and unreliable. Another point is that many embedded systems are powered from batteries with limited capacity. Architects have proposed purely hardware solutions, in which unused parts of the circuits are put to sleep, either by gating the clock or by cutting off the power. It seems that the efficient use of these new features needs help from the operating system. However, power reduction can be obtained also when compiling, e.g., by making better use of the processors or of the caches. For these optimizations, loop transformations are the most efficient techniques.

As the size of the needed working memory may change by orders of magnitude, high-level code optimization also has much influence on the size of the resulting circuit. If the system includes high performance blocks like DSPs or Asics, the memory bandwidth must match the requirements of these blocks. The classical solution is to provide a cache, but this goes against the predictability of latencies, and the resulting throughput may not be sufficient. In that case, one resorts to the use of scratch-pad memories, which are simpler than a cache but require help from the programmer and/or compiler to work efficiently. The compiler is a natural choice for this task. One then has to solve a scheduling problem under the constraint that the memory size is severely limited. Loop transformations reorder the computations, hence change the lifetime of intermediate values, and have an influence on the size of the scratch-pad memories.

The theory of scheduling is mature for cases where the objective function is, or is related to, the execution time. For other, non-local objective functions (i.e., when the cost cannot be directly allocated to a task), there are still many interesting open problems. This is especially true for memory-linked problems.

3.3.1. Modular Scheduling and Process Networks

Kahn process networks (KPN) were introduced thirty years ago [26] as a notation for representing parallel programs. Such a network is built from processes that communicate via perfect FIFO channels. One can prove that, under very general constraints, the channel histories are deterministic. Thanks to this property, one can define a semantics and talk meaningfully about the equivalence of two implementations. As a bonus, the dataflow diagrams used by signal processing specialists can be translated on-the-fly into process networks.

The problem with KPNs is that they rely on an asynchronous execution model, while VLIW processors and Asics are synchronous or partially synchronous. Thus, there is a need for a tool for synchronizing KPNs. This is best done by computing a schedule that has to satisfy data dependences within each process, a causality condition for each channel (a message cannot be received before it is sent), and real-time constraints. However, there is a difficulty in writing the channel constraints because one has to count messages in order to establish the send/receive correspondence and, in multidimensional loop nests, the counting functions may not be affine.

In order to bypass this difficulty, one can define another model, *communicating regular processes* (CRP), in which channels are represented as write-once/read-many arrays. One can then dispense with counting functions. One can prove that the determinacy property still holds. As an added benefit, a communication system in which the receive operation is not destructive is closer to the expectations of system designers.

The challenge with this model is that a modicum of control is necessary for complex applications like wireless networks or software radio. There is an easy conservative solution for intra-process control and channel reads. Conditional channel writes, on the other hand, raise difficult analysis and design problems, which sometimes verge on the undecidable.

The scheduling techniques of MMAlpha and Syntol (tools that we develop) are complex and need powerful solvers using methods from operational research. One may argue that compilation for embedded systems can tolerate much longer compilation times than ordinary programming, and also that Moore's law will help in tackling more complex problems. However, these arguments are invalidated by the empirical fact that the size and complexity of embedded applications increase at a higher rate than Moore's law. Hence, an industrial use of our techniques requires a better scalability, and in particular, techniques for modular scheduling. Some preliminary results have been obtained at École des Mines de Paris (especially in the framework of inter-procedural analysis), and in MMAlpha (definition of structured schedules). The use of process networks is another way of tackling the problem.

The scheduling of a process network can be done in three steps:

- In the first step, which is done one process at a time, one deduces the constraints on the channel schedules that are induced by the data dependences inside the process.
- In the second step, one gathers these constraints and solves them for the channel schedules.
- Lastly, the scheduling problem for each process is solved using the previous results.

This method has several advantages: each of the scheduling problems to be solved is much smaller than the global problem. If one modifies a process, one only has to redo step one for this process, and then redo the first and second steps completely. Lastly, this method promotes good programming discipline, allows reuse, and is a basic tool for the construction of libraries.

Off-the-shelf components pose another problem: one has to design interfaces between them and the rest of the system. This is compounded by the fact that a design may be the result of cooperation between different tools; one has to design interfaces, this time between elements of different design flows. Part of this work has been done inside MMAlpha; it takes the form of a generic interface for all linear systolic arrays. Our intention is to continue in this direction, but also to consider other solutions, like Networks on Chip and standard wrapping protocols such as VCI from VSIA [7].

3.3.2. Theoretical Models for Scheduling and Memory Optimizations

Many local memory optimization problems have already been solved theoretically. Some examples are loop fusion and loop alignment for array contraction and for minimizing the length of the reuse vector [23], and techniques for data allocation in scratch-pad memory. Nevertheless, the problem is still largely open. Some questions are: how to schedule a loop sequence (or even a process network) for minimal scratch-pad memory size? How is the problem modified when one introduces unlimited and/or bounded parallelism? How does one take into account latency or throughput constraints, or bandwidth constraints for input and output channels?

Theoretical studies here search for new scheduling techniques, with objective functions that are no longer linear. These techniques may be applied to both high-level applications (for source-to-source transformations) and low-level applications (e.g., in the design of a hardware accelerator). Both cases share the same computation model, but objective functions may differ in detail.

One should keep in mind that theory will not be sufficient to solve these problems. Experiments are required to check the pertinence of the various models (computation model, memory model, power consumption model) and to select the most important factors according to the architecture. Besides, optimizations do interact: for instance reducing memory size and increasing parallelism are often antagonistic. Experiments will be needed to find a global compromise between local optimizations.

[7] http://www.vsia.org

Alain Darte, who was cooperating on a regular basis with the PiCo project at HP Labs (now in Synfora), has already proposed some solutions to the memory minimization problem. These ideas may be implemented in the PiCo compiler in order to find their strengths and weaknesses. The interaction between Compsys and companies such as STMicroelectronics, Philips, or Thales on high-level synthesis will also be important to make sure that we focus on the right practical problems.

3.4. Hardware and Software System Integration

Participants: Alain Darte, Paul Feautrier, Antoine Fraboulet, Tanguy Risset.

Embedded systems have a very wide range of power and complexity. A circuit for a game gadget or a pocket calculator is very simple. On the other hand, a processor for digital TV needs a lot of computing power and bandwidth. Such performances can only be obtained by aggressive use of parallelism.

The designer of an embedded system must meet two challenges:

- one has to specify the architecture of the system, which should deliver the required performance, but no more than that;
- when this is done, one has to write the required software.

These two activities are clearly dependent, and the problem is how to handle their interactions.

The members of Compsys have a long experience in compilation for parallel systems, high-performance computers, and systolic arrays. In the design of embedded computing systems, one has to optimize new objective functions, but most of the work done in the polyhedral model can be reinvested. Our first aim is thus to adapt the polyhedral model to embedded computing systems, but this is not a routine effort. As we will see below, a typical change is to transform an objective function into a constraint or vice-versa. The models of an embedded accelerator and of a compute-intensive program may be similar, but one may have to use very different solution methods because the unknowns are no longer the same, and this is why this topic is challenging.

3.4.1. Design of Accelerators for Compute-Intensive Applications

The advent of high-level synthesis techniques allows one to create specific design for reconfigurable architectures, for instance with MMAlpha [8] (for regular architectures) or with lower-level tools such as HandelC, SiliconC, and others. Validating MMAlpha as a rapid prototyping tool for systolic arrays on FPGA will allow designers to use it with a full knowledge of its possibilities. To reach this goal, one has first to firm up the underlying methodology and then to try to interface it with tools for control-intensive applications.

Towards this goal, the team will use the know-how that Tanguy Risset has acquired during his participation in the Cosi Inria project (before 2001) and also the knowledge of some members of the Arénaire Inria project (Lip). This work is a natural extension of the “high-level synthesis” action in the Inria project Cosi. We want to show that, for some applications, we can propose, in less than 10 minutes, a correct and flexible design (including the interfaces) from a high-level specification (in C, Matlab, or Alpha). We also hope to demonstrate an interface between our tool, which is oriented towards regular applications, and synchronous language compilers (Esterel, Syndex), which are more control oriented.

3.4.2. Hardware Interfaces and On-Chip Traffic Analysis

Connecting the various components of a machine on the same interconnect is a challenge, and the most probable solution is the use of an on-chip network instead of the classical on-chip bus. In order to set the parameters of this on-chip network as soon as possible, fast simulation of the interconnection network is needed early in the design flow. To achieve this, we propose to replace some components by stochastic traffic generators. The design of the traffic generators has to be as fast as possible, in order to prototype rapidly different parameters of the network on chip.

[8] http://www.irisa.fr/cosi/ALPHA/

We are actively working in the SoCLib group (http://soclib.lip6.fr). We have developed a deep understanding of SoCLib simulation models and we have started collaborations with hardware designers in the LIP6 laboratory (Paris) and Lester laboratory (Lorient). Our aim is to adapt the MMAlpha tool to generate simulation models that are compatible with SoCLib. We will particularly concentrate on the data-flow interface generator, which should be adapted to IPs produced by the Gaut high-level synthesis tool (Lester). These developments will allow fast prototyping of SoC in SoCLib, particularly when a dataflow hardware accelerator is needed for compute-intensive treatments.

3.4.3. Optimization for Low Power

Present-day general-purpose processors need much more power than was usual a few years ago: about 150W for the latest models, or more than twice the consumption of an ordinary TV set. The next generation will need even more power, because leakage currents, which are negligible at present, will increase exponentially as the feature size decreases.

At the other end of the spectrum, for portable appliances, a lower power consumption translates into extended battery life. But the main tendency is the advent of power scavenging devices, which have no external power source, and extract power from the outside world, in the form of light, heat, or vibrations. Here the power budget is more of the order of milliwatts than hundreds of watts. Hence the present-day insistence on low-power digital design.

Low power can be achieved in four ways:

- One can search for low-power technologies and low-power architectures. Reducing the size of the die, or lowering the clock frequency or supply voltage are all techniques that decrease the power consumption.
- One can search for low-power algorithms. Since, for most processors, the energy consumption is proportional to the number of executed operations, this amounts, most often, to find low complexity algorithms.
- One can act at the level of the compiler. The rule here is to classify operations in terms of their power need, and to avoid, as far as possible, those with the highest need. For instance, an external memory access costs much more than a cache access, hence the need for maximizing the hit ratio of the cache. The same reasoning applies to registers.
- Lastly, one can combine the hardware and software approaches. The latest generation of processors and custom devices for embedded systems gives the software some degree of control on power consumption, either by controlling the clock frequency and source voltage, or by disconnecting unused blocks. The best solution would be to let the software or operating system be responsible for these controls.

The Compsys group works in cooperation with CEA-LETI in Grenoble in the field of hardware and software power modeling and optimization.

As far as hardware power modeling is concerned, Philippe Grosse has already demonstrated that the power consumption induced by the Network on Chip (NOC), in a 4G mobile phone custom chip, is not the most significant part of the power budget. Memories still represent the major part of the consumption. The power model built for this demonstration allows us now to explore frequency and voltage scaling solutions, where memory size is an important parameter.

For the software point of view, Nicolas Fournel is building an operating system power model. The aim of this model is to help software developers to predict with more accuracy the power consumption of their application by taking into account the OS consumption. Another goal is to give clues on the most power consuming part of an operating system, and to propose power aware alternatives.

3.5. Federating Polyhedral Tools

Participants: Fabrice Baray, Alain Darte, Antoine Fraboulet, Paul Feautrier, Tanguy Risset.

Present-day tools for embedded system design have trouble handling loops. This is particularly true for logic synthesis systems, where loops are systematically unrolled (or considered as sequential) before synthesis. An efficient treatment of loops needs the polyhedral model. This is where past results from the automatic parallelization community are useful. The French community is a leader in the field, mainly as one of the long-term results of the C^3 cooperative research program.

The polyhedral model is now widely accepted (Inria projects Cosi and A3, PIPS at École des Mines de Paris, Suif from Stanford University, Compaan at Berkeley and Leiden, PiCo from the HP Labs, the DTSE methodology at Imec, etc.). Most of these groups are research projects, but the increased involvement of industry (Hewlett Packard, Philips) is a favorable factor. Polyhedra are also used in test and certification projects (Verimag, Lande, Vertecs). A very recent development is the interest, shown by several compiler groups, in polyhedral methods (the GCC group, Reservoir Labs in the USA).

Two basic tools that have emerged from this early period are Pip [20] and Polylib [35]. They are currently the only available tools since maintenance has stopped on Omega (Maryland). Their functionalities are parametric integer programming and manipulations of unions of polyhedra. Granting that the showroom effect is important for us (these tools are used in many foreign laboratories), we nevertheless think that maintaining, improving, and extending these tools is a proper research activity. One of our goals must also be the design of new tools for new scheduling/mapping techniques.

In the following, we distinguish between the development of existing tools and the conception and implementation of new tools. These tasks are nevertheless strongly related. We anticipate that most of the new techniques will be evolutions of the present day tools rather than revolutionary developments.

3.5.1. Developing and Distributing the Polyhedral Tools

Recently, we have greatly increased the software quality of Pip and Polylib. Both tools can now use exact arithmetic. A CVS archive has been created for cooperative development. The availability for one year of an ODL software engineer has greatly improved the Polylib code. An interface bridge for combined use of the two tools has been created by Cédric Bastoul (former PhD student of Paul Feautrier). These tools have been the core of new code generation tools [18], [30] widely used in prototyping compilers. Paul Feautrier is the main developer of Pip, while Tanguy Risset has been in charge of coordinating the development of Polylib for several years. Other participants are at Irisa (Rennes) and ICPS (Strasbourg), and also in Lyon and Leiden. In the near future, we contemplate the following actions:

- For Pip, algorithmic techniques for a better control of the size of intermediate values; comparison with commercial tools like Cplex, for the non-parametric part.
- For Polylib, a better handling of $\mathbb{Z}$-polyhedra, which are needed for handling loops with non unit increments.
- For higher-level tools, Cédric Bastoul has developed CLooG, an improved loop generation tool along the lines of Fabien Quilleré's system, which is now available on the Web and is being incorporated into experimental compilers.
- For all these tools, we want to strengthen the user community by participating in the Polylib forum and organizing meetings for interested parties.

3.5.2. New Models

Industry is now conscious of the need for special programming models for embedded systems. Scholars from the University of Berkeley have proposed new models (process networks, SDL, etc.). This has culminated in the use of Kahn process networks, for which a complete overhaul of parallelization techniques is necessary (see Section 3.3.1). Optimizations for memory reduction are also very important. We are developing a tool, based on operations on integral lattices (including Minkowski's successive minima), named Cl@k, that can be used to derive affine mappings with modulos for memory reuse (see more details in Section 4.7).

Besides, our community has focused its attention on linear programming tools. For embedded systems, the multi-criteria aspect is pervasive, and this might require the use of more sophisticated optimization techniques (non-linear methods, constraint satisfaction techniques, "pareto-optimal" solutions).

Here again, our leadership in polyhedral tools will make our contributions in these areas easier. We nevertheless expect that, as sometimes in the past, the methods we need have already been invented in other fields like operational research, combinatorial optimization, or constraint satisfaction programming, and that our contribution will be in the selection and adaptation (and possibly the implementation) of the relevant tools.

4. Software

4.1. Introduction

This section lists and briefly describes the software developments conducted within Compsys. Most are tools that we extend and maintain over the years.

4.2. Polylib

Participant: Tanguy Risset.

Polylib (available at http://www.irisa.fr/polylib/) is a C library for polyhedral operations. The library operates on objects such as vectors, matrices, (convex) polyhedra, unions of polyhedra, lattices, $\mathbb{Z}$-polyhedra, and parametric polyhedra. Tanguy Risset has been responsible for the development of Polylib for several years. An ODL software engineer has firmed up the basic infrastructure of the library. The development is now shared between Compsys, the Inria project R2D2 in Rennes, the ICPS team in Strasbourg, and the University of Leiden. This tool is being used by many groups all over the world.

4.3. Pip

Participants: Paul Feautrier, Cédric Bastoul [MCF, IUT d'Orsay].

Paul Feautrier is the main developer of Pip (Parametric Integer Programming) since its inception in 1988. Basically, Pip is an "all integer" implementation of the Simplex, augmented for solving integer programming problems (the Gomory cuts method), which also accepts parameters in the non-homogeneous term. Most of the recent work on Pip has been devoted to solving integer overflow problems by using better algorithms. This has culminated in the implementation of an exact arithmetic version over the GMP library.

Pip is freely available under the GPL at http://www.prism.uvsq.fr/~cedb/bastools/piplib.html. Pip is widely used in the automatic parallelization community for testing dependences, scheduling, several kind of optimizations, code generation, and others. Beside being used in several parallelizing compilers, Pip has found applications in some unconnected domains, as for instance in the search for optimal polynomial approximations of elementary functions (see the Inria project Arénaire).

4.4. MMAlpha

Participant: Tanguy Risset.

Tanguy Risset is the main developer of MMAlpha since 1994 (http://www.irisa.fr/cosi/ALPHA/). The design and development of this software tool was the heart of several PhD theses, and MMAlpha is one of the few available tools for very high-level hardware synthesis (including the design of parallel Asics).

This tool is now in the public domain and has been used in many places (England, Canada, India, USA). Its development is shared between Compsys in Lyon and R2D2 in Rennes. MMAlpha is being evaluated by STMicroelectronics and has been a basis for Compsys participation to European and RNTL calls for proposals.

4.5. Syntol

Participants: Paul Feautrier, Hadda Cherroun.

Syntol is a process network scheduler. Its development has benefited from the help of François Thomasset (Inria Rocquencourt). The present version features a modular scheduler for Communicating Regular Processes. The next developments should be:

- An extension of the input parser and the scheduler in order to handle a larger subset of C. Cases in point are records and casts. An extension for fixed-point computations is also contemplated.
- An extension of the scheduler for handling conditionals.
- An implementation a virtual dependence heuristics for the generation of bounded parallelism schedules.

4.6. Software Developments for FPGA

Participants: Tanguy Risset, Antoine Scherrer, Paul Feautrier, Antoine Fraboulet.

Compsys has bought two FPGA boards for rapid prototyping. The boards are WildCard II (from Annapolis Inc), using the Xilinx XCV3000 FPGA circuit. These cards can be plugged into the PCMCIA slot of any laptop. We use them as a demonstrator for the design tools from Compsys. We hope that they will contribute to the visibility of the project.

4.7. Cl@k: Algorithms on Integral Lattices

Participants: Fabrice Baray, Alain Darte.

Our recent work on memory optimizations (see Section 5.7) identified new mathematical tools useful for the automatic derivation of array mappings that enable memory reuse. We have developed several algorithms and heuristics that rely on some mathematical concepts defined for integer lattices such as the notions of admissible lattice, of critical lattice, the successive minima of Minkowski, lattice basis reduction, and so on.

Fabrice Baray, post-doc Inria, has developed a tool, called Cl@k (for Critical LAttice Kernel), that computes or approximates the critical lattice for a given 0-symmetric polytope. An admissible lattice is a lattice whose intersection with the polytope is reduced to 0; a critical lattice is an admissible lattice with minimal determinant. Cl@k is still under development. So far, Cl@k is a stand-alone optimization program. A lot of work has to be done not only to improve its internal heuristics but also to use its power for memory optimizations in a compiler.

This tool is a complement to the Polylib suite. Our implementation of successive minima computations has already been used by Achill Schürmann's group (University of Magdeburg, Germany) to test some mathematical conjectures on Ehrhart polynomials. We believe that this tool is going to be used for other not-yet-identified problems, in particular problems for which finding a form of "bounding box" for a polytope is important.

4.8. CLooG: Loop Generation

Participants: Paul Feautrier, Cédric Bastoul [MCF, IUT d'Orsay].

The aim of CLooG is to generate a system of loops that visit once and only once the integer points in the union of several $\mathbb{Z}$-polyhedra. The algorithm is an improved version of a previous effort by Fabien Quilleré (past Inria project Cosi). The code generated by CLooG is compact and quite efficient [1]. The availability of CLooG on the Web as a free software (http://www.cloog.org) has been a triggering factor for a recent increase of interest for the polytope model.

4.9. Register Allocation

Participants: Florent Bouchez, Alain Darte, Fabrice Rastello, Cédric Vincent.

Within the collaboration with the MCDT team at STMicroelectronics, Cédric Vincent (bachelor degree, under Alain Darte supervision) developed a complete register allocator in the assembly-code optimizer of STMicroelectronics. This was the first time a complete implementation was done with success, outside the MCDT team, in their optimizer. New developments are in progress, in particular by Florent Bouchez, advised by Alain Darte and Fabrice Rastello, as part of his master internship and PhD thesis. See more details in Section 5.3.

5. New Results

5.1. Introduction

This section presents the results obtained by Compsys in 2005. For clarity, some earlier work is also recalled, when results were continued or extended in 2005.

5.2. Variable Coalescing and ssa Form

Participants: Florent Bouchez, Alain Darte, François de Ferrière [STMicroelectronics], Christophe Guillon [STMicroelectronics], Fabrice Rastello.

The SSA form (Static Single Assignment) is an intermediate representation in which multiplexers (called φ functions) are used to merge values at a "join" point in the control graph. The SSA form has very interesting properties that make optimizations like partial redundancy elimination straightforward and efficient. Unfortunately, the SSA form is not machine code and φ functions have to be replaced, at the end of the process, by register-to-register copy instructions on control flow edges. In practice naive methods for translating out of SSA will generate many useless copies (live-range splitting) and additional goto instructions (edge splitting).

Adding copies roughly corresponds to split the live ranges of variables. The reverse transformation is called variable coalescing: coalescing different variables into a common representative variable constrains those variables to be allocated to the same register. Hence, coalescing constrains register allocation and splitting may relax it: in practice, the reduction of register-to-register copies is performed during the register allocation phase. However, we are convinced that coalescing should be performed during the out of SSA translation: many more copies can be removed this way with lower complexity; we can keep trace of coalesced variables and un-coalesce them when necessary during the register allocator phase.

Coalescing during the out of SSA translation has been studied in different contexts. Rastello et al. [31] (*PINNING*) studied the problem for programs represented as native machine instructions, including the use of machine dedicated registers: the heuristic is sophisticated and coalesces variables very aggressively without splitting edges. Budimlić et al. [14] (*D-FOREST*) studied the problem in the context of just in time (JIT) compilers: their algorithm is very fast but requires edge splitting. The coalescing obtained is probably not the best possible. The last two heuristics use a common approach in the sense that they both perform a non-conservative coalescing followed by a repairing phase. Sreedhar et al. [32] (*C*SSA) provided another interesting approach without the above constraints: it is a conservative approach in one pass with basic but efficient ideas that do not require edge splitting.

The context of our work is the same as for *D-FOREST*: we have designed a linear heuristic to aggressively coalesce without splitting edges. For that purpose, we generalized in [11] ideas used in all three approaches.

Our solution remains to be implemented in the *LAO* code assembly optimizer from STMicroelectronics and compared with other possible approaches.

We also developed a more accurate way of defining life-range interferences, based on a classical lattice-based fixed-point program analysis, which allows many more variables to be coalesced. For more details, see the master thesis of Florent Bouchez [8].

5.3. Register Allocation and ssa Form Properties

Participants: Florent Bouchez, Alain Darte, Fabrice Rastello.

The work presented in this section is a joint work with members of the MCDT team at STMicroelectronics. It is the natural extension of the work described in Section 5.2.

It deals with the following problems related with register allocation: spilling, coalescing, and splitting. Register allocation consists in allocating abstract variables to physical registers, spilling chooses which variables to store in memory when there are too few registers to store all of them, coalescing is used to amalgamate two variables into one when possible, hence reducing the number of variables, and splitting does the converse. Understanding the interactions between these three techniques is the heart of our work.

Minimizing the amount of spilled variables is a highly studied problem in compiler design; it is an NP-complete problem in general. Meanwhile, a program under SSA form has the interesting property that for cases where spilling is unnecessary, register allocation is a polynomial problem whereas it is NP-complete in general. Hence, among other problems, our current work is the design of a register allocator scheme that would first spill variables under SSA and that would then perform register coalescing and allocation during the translation out of SSA form. This approach poses several new questions that we are currently investigating: is the spilling problem simpler under SSA form? A classical approach expresses the spilling problem as a node-deletion problem on the so-called interference graph. On a basic block, this graph is an intersection graph of intervals called interval graph, and the node deletion problem can be optimally solved in polynomial time (for simple cost models, not in general though); under SSA form, the interference graph is an intersection graph of subtrees of a tree, what is called a triangulated graph; also, the linear program used for basic blocks can easily be generalized to an intermediate form of intersection graph called path graph. Unfortunately, at least in theory, we cannot go further: we proved the NP-completeness of this spilling formulation for triangulated graphs as well as many other NP-completeness and polynomiality results related to the spilling problem [8]. Even though we believe that spilling should be performed under SSA (or a similar form leading to an easy graph coloring), we have not designed any smart heuristic yet.

The next question is: suppose we have spilled variables and have performed allocation under SSA. Is it possible to translate this allocated code out of SSA form? An easy way is to split edges and to add as many copies as necessary. But then several problems arise: first, in practice, we probably do not want to split edges; second, this approach would lead to too many copy instructions; lastly, more than k registers might be required to permute a variable assignment on k registers. Fortunately, we have solutions (that will be described soon in a research report) to the first and last difficulties.

To finish with, the last question is related to the coalescing problem: how can we minimize or at least reduce the number of generated copies? Suppose again that spilling has been performed under SSA. Suppose that we want to perform allocation under SSA form such that the number of copies generated by the translation out of SSA form is minimized. Again, we proved several NP-complete and polynomiality results related to this type of questions. These results will be described in a research report in 2006.

This work is part of the current contract (see Section 6.1) with the MCDT team at STMicroelectronics. Florent Bouchez' master thesis [8] describes our first results on the spilling problem. A more exhaustive research report on the interactions of spilling, splitting, and coalescing is planned.

5.4. Instruction Cache Optimization

Participants: Thierry Bidault [STMicroelectronics], Benoit Boissinot, Christophe Guillon [STMicroelectronics], Fabrice Rastello, Éric Thierry [Lip, MC2 team].

The instruction cache is a small memory with fast access. All binary instructions of a program are executed from it. In the ST220 processor from STMicroelectronics, the instruction cache is direct mapped: let L be the size of the cache; the cache line i can hold only instructions whose addresses are equal to i modulo L. When a program starts executing a block that is not in the cache, one must load it from main memory; this is called a cache miss. This happens either at the first use of a function (cold miss), or after a conflict (conflict miss). There is a conflict when two functions share the same cache line; each of them removes the other from the cache when their executions are interleaved. The cost of a cache miss is of the order of 150 cycles for the ST220, hence the interest of minimizing the number of conflicts by avoiding line sharing when two functions are executed in the same time slot. This problem has in fact three objective functions:

COL Minimizing the number of conflicts for a given execution trace. This can be reduced to the *Max-K-Cut* and *Ship-Building* problems.

EXP Minimizing the size of the code. This is equivalent to a traveling salesman problem (building an Hamiltonian circuit) on a very special graph called *Cyclic-Metric*.

NBH Leaving spaces or uncalled procedures between successively-called procedures yields cache misses: if two successively-called procedures share the same cache line, a cache miss is saved when the second procedure is called, since it is already in the cache. Another similar phenomenon is instruction prefetch, where instructions are loaded from the memory before the processor needs them, according to a branch prediction policy. Therefore, increasing code locality is good in order to avoid cache misses. Gloy and Smith in [24] have addressed the problem of spatial locality for a paged virtual memory system. However, contrarily to Gloy and Smith who work on virtual pages (*locality* problem), we have to work on cache line granularity (*neighboring* problem, or *NBH* problem for short), which is smaller than the size of most procedures. The objective of the *NBH* problem is to put together procedure extremities that have a high affinity in terms of time locality. This can be reduced to the traveling salesman problem.

We have proved several non-approximability and polynomiality results related to the *COL*, *EXP*, and *NBH* problems. For the *COL* problem, Gloy and Smith (GS) proposed a greedy heuristic based on a conflict graph called the temporal relationship graph (TRG). We proposed several improvements to this approach: we have lowered the complexity by an order of magnitude and implemented an algorithm that takes *EXP* into account. We have implemented several variations that provide a better tradeoff between the *COL* and *EXP* optimization problems.

We also worked on a completely different approach whose goal is to provide a procedure placement that yields *no* code size expansion and takes *NBH* into account. Our algorithm still has to be implemented. Finally, we currently work on the following problem, but with no satisfactory answer yet:

Profiling vs static The GS algorithm is based on a conflict graph built using profiling. In practice, it leads to a much better conflict reduction than any other known static based approach. But a profiling technique has several drawbacks: the compilation process of profiling feedback is too complex for most application developers. Also, using good representative data sets for profiling is an important issue. The problem is then to be able to build a TRG using fully static, fully dynamic, or hybrid techniques.

This work is part of the current contract with the MCDT team at STMicroelectronics, see Section 6.1. Some of the results presented in this section will appear in [4].

5.5. On-Chip Traffic Analysis

Participants: Patrice Abry [ENS-Lyon, Physics Lab], Antoine Fraboulet, Tanguy Risset, Antoine Scherrer.

This work is part of the PhD of Antoine Scherrer, co-funded by the CNRS and STMicroelectronics.

Our approach is to (semi-)automatically generate a traffic generator from an execution trace of the component we want to replace (for simulation). We work on this topic in relation with STMicroelectronics. We are also working in collaboration with Patrice Abry, from the Physics laboratory of ENS-Lyon. Patrice Abry (http://perso.ens-lyon.fr/patrice.abry/) is a worldwide expert in traffic modeling with self-similar processes. We are currently investigating the use of these advanced probabilistic models to model non-stationary behaviors of the communications such as burst transactions (uninterrupted communication of a block of data between two IPs). A short article has appeared in the Inria *Inedit* newspaper (November 2005). We are now working on the set-up of the flow for generating traffic generators, and the first results will be submitted in early 2006. A research report detailing this work is available [12].

5.6. Data-Flow IP Interface

Participants: Alain Darte, Steven Derrien [R2D2 Inria project], Antoine Fraboulet, Tanguy Risset, Antoine Scherrer, Oriol Vidal-Perez [Erasmus internship].

As IP models generated by MMAlpha use a systolic-like computational model, we have designed and developed a hardware interface generator that can bridge the communication gap between bus or network on chip interconnection mechanisms and our generated IPs (one-dimensional arrays). This interface allows us to efficiently map burst-mode communications generated by the processor to data-flow architectures. Aside from the interface, we have developed a generic DMA engine used to generate burst transactions and a terminal type that can process input and output events within the SoCLib simulation environment. The interface generator and its associated communication mechanism and performance using CPU or DMA driven transfers were presented at the ASAP'04 conference.

This year, we extended the principle of the data-flow IP interface to multidimensional arrays. For that, we have modeled the access to the bus as a classical resource constraint problem and we have explained how to solve it using existing techniques developed for the PiCo tool. The key idea is to determine, at compile-time, the number of registers (delays) that need to be inserted between the processors and the FIFO that accesses the bus, so that the processors never access the FIFO at the same time. This work was presented at the ASAP'05 [6] conference.

5.7. Memory Reuse and Modular Mappings

Participants: Fabrice Baray, Alain Darte, Rob Schreiber [HP Labs], Gilles Villard [Lip, Arénaire Project].

When designing hardware accelerators, one has to solve both scheduling problems (when is a computation done?) and memory allocation problems (where is the result stored?). This is especially important because most of these designs use pipelines between functional units (this appears in the code as successive loop nests), or between external memory and the hardware accelerator. To decrease the amount of memory, the compiler must be able to reuse it. An example is image processing, for which we might want to store only a few lines and not the entire frame, data being consumed quickly after they are produced. A possibility to reduce memory size is to reuse the memory locations assigned to array elements, following, among others, the work of Francky Catthoor's team [17], of Lefebvre and Feautrier [27], and of Quilleré and Rajopadhye [29].

In our previous work (published at CASES'03), we introduced a new mathematical formalism for array reuse, using the concept of critical lattice, so as to understand and analyze previous approaches. We found that they are all particular cases of more general heuristics for finding a critical lattice. We also proved that we can compute approximations that do not differ from the optimum more than by a multiplicative constant that depends only on the dimension of the problem.

In 2004-2005, we continued our study. We analyzed in more detail the strengths and weaknesses of the previous approaches for array reuse. We revealed similarities and differences with early 70's and 80's work on data layouts allowing parallel accesses, results that were forgotten in the meantime. We explored more deeply the properties of linear mappings with, in particular, a new result concerning the minimal dimensions of these mappings, etc. This work is published in IEEE Transactions on Computers, for a special issue on Embedded Systems, Microarchitecture, and Compilation Techniques in memory of Bob Rau, leader of the PiCo project at HP Labs, in which our work grew [3].

In practice, (simpler) heuristics for the memory reuse problem are already in use in tools like PiCo or at IMEC. We hope these researches will give a new interest to a possible cooperation with Synfora and IMEC. In 2005, we developed the algorithms on lattices, successive minima, and critical lattices, needed to implement our memory reuse strategies (see Section 4.7). This tool (developed by Fabrice Baray) should impact both the practical problem of designing hardware accelerators and the mathematical problem of finding the critical lattice of an object. Cl@k, built on top of our present tools Pip and Polylib, is a perfect extension for our tool suite on polyhedral/lattice manipulations.

The optimization problem addressed by Cl@k is a fundamental algorithmic problem, which has surprisingly not attracted much attention from mathematicians. However, we point out that Cl@k (actually the part that computes the successive minima) has been used by Achill Schürmann, from the University of Magdeburg, to find a counter-example to a conjecture on Ehrhart polynomials. Alain Darte has also been invited at the "Mathematisches Forschungsinstitut Oberwolfach" in a one-week seminar on sphere packings to present the lattice-based memory allocation problem.

5.8. Optimal Barrier Placement in Nested Loops

Participants: Alain Darte, Rob Schreiber [HP Labs].

The results presented in this section are, *a priori*, a bit out of the mainstream of Compsys research. Rob Schreiber, who has pursued a long collaboration since the mid 90's with Alain Darte, came in May 2004, as an invited professor at ENS-Lyon in the Compsys team. In the past, Rob Schreiber has been working on the high-level synthesis of hardware accelerators (PiCo project) but he is now moving back to high performance computing, which explains the topic of this joint work. Nevertheless, as embedded systems are incorporating more and more features from high-performance computing, this research may find its applications in the future development of languages for high-performance embedded systems.

Rob Schreiber's team is currently working on recent parallel languages for high-performance computing, such as Co-Array Fortran, Titanium, OpenMP, UPC, etc. In these languages, of SPMD type (Single-Program Multiple-Data programs), one of the issues to ease programming while ensuring performance is to let the compiler optimize the necessary synchronizations. We started to study the literature on how to place synchronization barriers in SPMD codes and we realized that, although well studied in the past and apparently simple, this problem was actually not solved! The best contribution for barrier placement in nested loops was by O'Boyle and Stöhr [28], who proposed an optimal (and complex) algorithm but only for some restricted cases of loops (loops containing at most one other loop) and a heuristic in the general case. We developed an algorithm, which is optimal for any form of nested loops for SPMD codes with reasonable assumptions on synchronizations (same as for the Titanium language). The simplest (to understand) version of our algorithm is of quadratic complexity (already better than O'Boyle and Stöhr's algorithm) and we even proposed a linear-time algorithm. This work has been presented at the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'05) [7].

Several open questions remain, for example how to reorganize the code for minimizing synchronization barrier requirements, in a simplified BSP (Bulk Synchronous Parallel) processing model (where the number of barriers is the objective function), or how to place barriers in a finer BSP processing model, where threads may have unbalanced workload (in which case adding barriers may actually reduce the execution time).

5.9. Modular Scheduling

Participant: Paul Feautrier.

Scheduling of ordinary programs is a highly non scalable process. We estimate that the scheduling time grows as the sixth power of the program size. We have designed a new scheduling method, which uses projections instead of linear programming, and which is both scalable and structured.

Our method is scalable because the scheduling proceeds by successive elimination of statements from the relevant subset of the dependence graph. Hence, it is almost linear in the program size. It is still exponential in the loop nest depths, but these factors are very small integers. It is structured because the application can be split in a hierarchical process network, and because each process can be scheduled independently of the others.

The prototype modular scheduler Syntol is now completed and may be used for experiments. It has been interfaced with CLooG (see Section 4.8) for code generation. We are using Syntol for the following tasks:

- Writing kernels for real-life applications, especially in the field of digital TV. The present day parser needs to be extended, especially as regards fixed-point arithmetics and structures.
- Investigating the question of real-time constraints, in the case of non-unit execution time operations and multidimensional schedules.
- Investigating heuristics for handling resource constraints.
- Imposing additional constraints on schedules in order to improve code generation.

5.10. Scheduling for Synthesis

Participants: Hadda Cherroun, Alain Darte, Paul Feautrier.

While scheduling for high-performance computations can be done at a somewhat abstract level (the back-end compiler takes care of detail), in circuit synthesis all elementary steps (e.g., address computations and memory accesses) must be taken into account. The resulting scheduling problems are typically too large to be solved even by the methods of Section 5.9. Hence the idea of a two-levels scheduling, in which the *fronts* generated by the first-level schedule are refined using combinatorial optimization methods.

We have designed and compared several Branch & Bound algorithms, and also a very fast but approximate greedy scheduling algorithm. Our fastest exact Branch & Bound algorithm is based on variants of Bellman-Ford and Dijkstra's algorithm, using a reweighting technique similar in the spirit to Johnson's algorithm. We point out that this technique is also nothing but a "retiming" technique (in the sense of Leiserson and Saxe), technique that we explored in the past for loop optimizations [2].

The results of this work will be presented at DATE'06, in spring 2006 [5].

6. Contracts and Grants with Industry

6.1. Contract with stmicroelectronics on Register Allocation and Instruction Cache Optimizations

Participants: Alain Darte, Fabrice Rastello, Florent Bouchez, Benoit Boissinot.

This currently finishing contract is funded by STMicroelectronics. Its objective is the improvement of existing techniques for the optimization of the instruction cache use and for the optimization of the register allocation phase. The study targets ST200 architectures, for which the MCDT team develops a compiler tool chain. Work on instruction cache optimizations are described in Section 5.4, studies on register allocation in Sections 5.2 and 5.3.

6.2. Contract with stmicroelectronics on Special-Purpose Circuit Synthesis

Participants: Tanguy Risset, Édouard Bechetoille.

Compsys had a STSI contract (i.e., contract between STMicroelectronics and the Ministry of Industry) with STMicroelectronics (Crolles plant) on experimentation with MMAlpha for the synthesis of STMicroelectronics special purpose circuits. Due to administrative problems, the funding was extremely late and we could not really achieve the implementation of tiling in MMAlpha. We were finally able to hire Édouard Bechetoille in 2005 as an engineer and to develop a methodology to handle different types of rounding in the VHDL produced by MMAlpha. These developments are now integrated in MMAlpha.

6.3. Grant from hp labs

Participant: Alain Darte.

To help maintaining alive the collaboration between Rob Schreiber (HP Labs) and Alain Darte, despite the growing gap between their current main research interests (see explanations in Section 5.8), the HP Labs have offered to increase their donation to ENS-Lyon, a part dedicated for Compsys. This funding is to be used for visits between the two labs and trips in conferences such as PPoPP'05 [7], where our recent results have been published.

7. Other Grants and Activities

7.1. European Community

Participants: Alain Darte, Paul Feautrier, Tanguy Risset.

Network of excellence Compsys is involved in the network of excellence HIPEAC (High-Performance Embedded Architecture and Compilation http://escher.elis.ugent.be/hipeac/), which has been granted by the European community.

ITEA project Compsys is involved in the Martes (Model driven Approach to Real-Time embedded Systems development) ITEA project. The project has been accepted by the EC authority in 2004, and the French Ministry has decided to fund it, starting in 4Q 2005. The French partners have already held several technical meetings. It seems probable that the cooperation with Thales will focus on an assessment of process networks as a method for describing compute-intensive embedded systems.

7.2. CNRS Convention with the University of Illinois at Urbana-Champaign (USA)

Participant: Paul Feautrier.

A convention between UIUC and CNRS supports some visits and cooperation between David Padua's team, Compsys and members of the Alchemy project.

7.3. Procope Convention with Passau University (Germany)

Participant: Paul Feautrier.

Paul Feautrier's cooperation with the University of Passau is supported by a Procope convention, which has been recently renewed.

7.4. Inria Support for Collaboration with Federal University, Rio Grande Do Sul (Brasil)

Participant: Antoine Fraboulet.

An Inria grant supported this year a cooperation visit between Luigi Carro's team and Compsys.

7.5. Informal Cooperations

- Tanguy Risset is in regular contact with the University of Québec at Trois-Rivières (Canada), where MMAlpha is in use.

- Compsys is in regular contact with Sanjay Rajopadhye's team at Colorado State University (USA).
- Compsys is in regular contact with Francky Catthoor's team in Leuwen (Belgium) and with Ed Depreterre's team at Leiden University (the Netherlands).
- Alain Darte has fruitful relations with the HP Labs and Rob Schreiber's group – with several joint patents [16], [15] and publications – and the past members of the PiCo team (Vinod Kathail at Synfora, Scott Mahlke at the University of Michigan).
- Paul Feautrier has regular contact with Zaher Mahjoub's team in the Faculté des Sciences de Tunis, notably as the co-advisor of a PhD student.
- Compsys is in regular contact with Christine Eisenbeis's team (Inria project Alchemy, Paris), with François Charot and Patrice Quinton (Inria project R2D2, Rennes), with Alain Greiner (Asim, LIP6, Paris) and Frédéric Pétrot (TIMA, Grenoble).
- Compsys participates in the center of microelectronics design Inria/CEA with LETI (CEA Grenoble) on software techniques for power minimization.
- Compsys participates in the EmSoC research project, which is part of the new research clusters of the Rhône-Alpes region.
- Compsys is in regular contact with Luigi Carro's team (UFRGS, Porto Alegre, Brasil).

8. Dissemination

8.1. Introduction

This section lists the various scientific and teaching activities of the team in 2005.

8.2. Conferences and Journals

- In 2005, Alain Darte was member of the program committees of ASAP 2005 (IEEE International Conference on Application-Specific Systems, Architectures and Processors), CASES 2005 (ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems), DATE 2005 and DATE 2006 (Design, Automation and Test in Europe). He is member of the steering committee of the workshop series CPC (Compilers for Parallel Computing). He is member of the editorial board of the international journal ACM Transactions on Embedded Computing Systems (ACM TECS).
- Paul Feautrier is associate editor of Parallel Computing and the International Journal of Parallel Computing. In 2005, he was a member of the program committee of POPL'06 to be held in Charleston, USA in the spring of 2006.
- Tanguy Risset is a member of the editorial board of Integration: the VLSI Journal. He was a member of the program committee of the SAMOS workshop in 2005.

8.3. Post-Graduate Teaching

- In 2005-2006, Alain Darte and Paul Feautrier are sharing a Master 2 course on advanced compilation and program transformations.
- Paul Feautrier is thesis advisor for Christophe Alias (UVSQ, co-advisor Denis Barthou, defended in December 2005), Yosr Slama (Faculté des Sciences de Tunis, co-advisor Mohamed Jemni), Nicolas Fournel (co-advisor Antoine Fraboulet) and Philippe Grosse (co-advisor Yves Durand, CEA-LETI). Hadda Cherroun has received a French-Algerian grant for an internship of 18 months with Compsys, in preparation for an Algerian PhD.
- In 2004-2005, Tanguy Risset and Antoine Fraboulet started a new Master 2 course entitled "Design of embedded computing systems" at Insa-Lyon.
- Antoine Fraboulet and Tanguy Risset are thesis co-advisors of Antoine Scherrer.

8.4. Other Teaching and Responsibilities

- Alain Darte is in charge of the “Computer Science” division of the admission exam to ENS-Lyon.
- Paul Feautrier teaches the following subjects for first and second year students: a) A guided tour of Unix (L3IF), b) Operational Research (M1), c) Compilation project (M1).
- Paul Feautrier is responsible for the second year of the Computer Science Master of ENS-Lyon (DEA).
- In 2004-2005, Tanguy Risset and Fabrice Rastello (also in 2005-2006) are in charge of the Compilation course to Master 1 students at ENS-Lyon.
- Tanguy Risset took a professor position at Insa-Lyon in 2005, he is in charge of the algorithmic and programming course, he is also currently setting up with Antoine Fraboulet and some other people a new course on embedded systems targeted to engineers from different departments: electrical engineering, network, and computer science.

8.5. Animation

- Paul Feautrier is a member of the governing board and the PhD committee of ENS-Lyon, and of the hiring committees of ENS-Lyon and Université Joseph Fourier. He is a member of the expert group in charge of “Prime d’encadrement Doctoral et de Recherche”.
- Tanguy Risset is in charge of the Polylib mailing-list. This list includes most of the actors on the polyhedral models.
- Alain Darte is member of the national evaluation commission (CE) of INRIA.
- Antoine Fraboulet is a member of the hiring committees of Insa-Lyon.

8.6. Defense Committees

- Tanguy Risset was a member of the defense committee for Madeleine Nyamsy (November 22, 2005, Irisa, Rennes) and Lionel Lelong (December 7, 2005, LTSI, Saint-Étienne).
- Alain Darte was a member of the defense committee of Sven Verdoolaege (April 2005, Katholieke Universiteit Leuven, Belgium) entitled “Incremental loop transformations and enumeration of parametric sets”.
- Paul Feautrier is reviewer for the HDR (professorial thesis) of Bernard Pottier (Université de Bretagne Occidentale), and a member of the defense committee for the PhD of Christophe Alias (UVSQ).

8.7. Workshops, Seminars, and Invited Talks

(For conferences with published proceedings, see the bibliography.)

- Alain Darte was invited in April 2005 to give a talk to the Flemish network ACES on “Lattice-based memory allocation” (http://www.elis.ugent.be/aces/activity1.html) and to the “Mathematisches Forschungsinstitut Oberwolfach” (Germany), as part of a seminar on Sphere Packings (3rd week of November 2005, http://fma2.math.uni-magdeburg.de/~achill/mfo).
- Paul Feautrier gave a talk on modular scheduling at ETH Zurich in May 2005.
- Hadda Cherroun, Alain Darte, Nicolas Fournel, and Antoine Scherrer attended the CNRS School on “Architectures des systèmes matériels enfouis et méthodes de conception associées”, Autrans, France, March 21-25, 2005 (http://www.ens-lyon.fr/ARCHI05/).
- Nicolas Fournel attended the 2005 ARM Connected Community Technical Symposium, Paris, October 28, 2005.

9. Bibliography

Articles in refereed journals and book chapters

[1] C. BASTOUL, P. FEAUTRIER. *Adjusting a Program Transformation for Legality*, in "Parallel Processing Letters", vol. 15, n^o 1-2, March-June 2005, p. 3-17.

[2] A. DARTE, G. HUARD. *New Complexity Results on Array Contraction and Related Problems*, in "Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology", vol. 40, n^o 1, May 2005, p. 35-55.

[3] A. DARTE, R. SCHREIBER, G. VILLARD. *Lattice-Based Memory Allocation*, in "IEEE Transactions on Computers", Special Issue: Tribute to B. Ramakrishna (Bob) Rau, vol. 54, n^o 10, October 2005, p. 1242-1257.

[4] C. GUILLON, F. RASTELLO, T. BIDAULT, F. BOUCHEZ. *Procedure Placement using Temporal-Ordering Information: Dealing with Code Size Expansion*, in "Journal of Embedded Computing", vol. 1, n^o 4, 2005.

Publications in Conferences and Workshops

[5] H. CHERROUN, A. DARTE, P. FEAUTRIER. *Scheduling under Resource Constraints using Dis-Equations*, in "Design, Automation, and Test in Europe (DATE'06), Munich, Germany", to appear, 2006.

[6] A. DARTE, S. DERRIEN, T. RISSET. *Hardware/Software Interface for Multi-Dimensional Processor Arrays*, in "IEEE International Conference on Application-Specific Systems, Architecture, and Processors (ASAP'05)", IEEE Computer Society Press, 2005, p. 28–35.

[7] A. DARTE, R. SCHREIBER. *A Linear-Time Algorithm for Optimal Barrier Placement*, in "ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'05), Chicago, IL, USA", June 2005, p. 26-35.

Internal Reports

[8] F. BOUCHEZ, A. DARTE, C. GUILLON, F. RASTELLO. *Register Allocation and Spill Complexity under SSA*, Technical report, n^o 2005-33, LIP, ENS-Lyon, August 2005.

[9] H. CHERROUN, A. DARTE, P. FEAUTRIER. *Scheduling with Resource Constraints using Dis-Equations*, Technical report, n^o 2005-40, LIP, ENS-Lyon, September 2005.

[10] A. DARTE, S. DERRIEN, T. RISSET. *Hardware/Software Interface for Multi-Dimensional Processor Arrays*, Technical report, n^o 2005-15, LIP, ENS-Lyon, April 2005.

[11] F. RASTELLO, F. DE FERRIÈRE, C. GUILLON. *Optimizing the Translation Out-of-SSA with Renaming Constraints*, Technical report, n^o 2005-34, LIP, ENS-Lyon, August 2005.

[12] A. SCHERRER, A. FRABOULET, T. RISSET. *Analysis and Synthesis of Cycle-Accurate On-Chip Traffic with Long-Range Dependence*, Technical report, n^o 2005-53, LIP, ENS-Lyon, December 2005.

[13] A. SCHERRER, N. LARRIEU, P. OWEZARSKI, P. ABRY. *Non Gaussian and Long Memory Statistical Char-*

acterisations for Internet Traffic with Anomalies, Technical report, n^o 2005-35, LIP, ENS-Lyon, September 2005.

Bibliography in notes

[14] Z. BUDIMLIĆ, K. D. COOPER, T. J. HARVEY, K. KENNEDY, T. S. OBERG, S. W. REEVES. *Fast Copy Coalescing and Live-Range Identification*, in "PLDI '02: Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation, New York, NY, USA", ACM Press, 2002, p. 25–32.

[15] A. DARTE, B. R. RAU, R. SCHREIBER. *Programmatic Iteration Scheduling for Parallel Processors*, August 2002, US patent number 6438747.

[16] A. DARTE, R. SCHREIBER. *Programmatic Method For Reducing Cost Of Control In Parallel Processes*, April 2002, US patent number 6374403.

[17] E. DE GREEF, F. CATTHOOR, H. DE MAN. *Memory Size Reduction Through Storage Order Optimization for Embedded Parallel Multimedia Applications*, in "Parallel Computing", vol. 23, 1997, p. 1811-1837.

[18] E. F. DEPRETTERE, E. RIJPKEMA, P. LIEVERSE, B. KIENHUIS. *Compaan: Deriving Process Networks from Matlab for Embedded Signal Processing Architectures*, in "8th International Workshop on Hardware/Software Codesign (CODES'2000), San Diego, CA", May 2000.

[19] BENOÎT. DUPONT DE DINECHIN, C. MONAT, F. RASTELLO. *Parallel Execution of the Saturated Reductions*, in "Workshop on Signal Processing Systems (SIPS 2001)", IEEE Computer Society Press, 2001, p. 373-384.

[20] P. FEAUTRIER. *Parametric Integer Programming*, in "RAIRO Recherche Opérationnelle", vol. 22, September 1988, p. 243–268.

[21] P. FEAUTRIER. *Some Efficient Solutions to the Affine Scheduling Problem, Part I, One Dimensional Time*, in "International Journal of Parallel Programming", vol. 21, n^o 5, October 1992, p. 313-348.

[22] P. FEAUTRIER. *Some Efficient Solutions to the Affine Scheduling Problem, Part II, Multidimensional Time*, in "International Journal of Parallel Programming", vol. 21, n^o 6, December 1992.

[23] A. FRABOULET, K. GODARY, A. MIGNOTTE. *Loop Fusion for Memory Space Optimization*, in "IEEE International Symposium on System Synthesis, Montréal, Canada", IEEE Press, October 2001, p. 95–100.

[24] N. GLOY, M. D. SMITH. *Procedure Placement Using Temporal-Ordering Information*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", vol. 21, n^o 5, 1999, p. 977-1027.

[25] R. JOHNSON, M. SCHLANSKER. *Analysis of Predicated Code*, in "Micro-29, International Workshop on Microprogramming and Microarchitecture", 1996.

[26] G. KAHN. *The Semantics of a Simple Language for Parallel Programming*, in "IFIP'74", N. HOLLAND (editor). , 1974, p. 471-475.

[27] V. LEFEBVRE, P. FEAUTRIER. *Automatic Storage Management for Parallel Programs*, in "Parallel Computing", vol. 24, 1998, p. 649-671.

[28] M. O'BOYLE, E. STÖHR. *Compile Time Barrier Synchronization Minimization*, in "IEEE Transactions on Parallel and Distributed Systems", vol. 13, n^o^ 6, 2002, p. 529–543.

[29] F. QUILLERÉ, S. RAJOPADHYE. *Optimizing Memory Usage in the Polyhedral Model*, in "ACM Transactions on Programming Languages and Systems", vol. 22, n^o^ 5, 2000, p. 773-815.

[30] F. QUILLERÉ, S. RAJOPADHYE, D. WILDE. *Generation of Efficient Nested Loops from Polyhedra*, in "International Journal of Parallel Programming", vol. 28, n^o^ 5, 2000, p. 469–498.

[31] F. RASTELLO, F. DE FERRIÈRE, C. GUILLON. *Optimizing Translation Out of SSA using Renaming Constraints*, in "International Symposium on Code Generation and Optimization (CGO'04)", IEEE Computer Society Press, March 2004, p. 265-278.

[32] V. C. SREEDHAR, R. D.-C. JU, D. M. GILLIES, V. SANTHANAM. *Translating Out of Static Single Assignment Form*, in "Static Analysis", A. CORTESI, G. FILÉ (editors). , Lecture Notes in Computer Science, vol. 1694, Springer, 1999, p. 194–210.

[33] V. SREEDHAR, R. JU, D. GILLIES, V. SANTHANAM. *Translating Out of Static Single Assignment Form*, in "Static Analysis Symposium, Italy", 1999, p. 194 – 204.

[34] A. STOUTCHININ, F. DE FERRIÈRE. *Efficient Static Single Assignment Form for Predication*, in "International Symposium on Microarchitecture", ACM SIGMICRO and IEEE Computer Society TC-MICRO, 2001.

[35] D. WILDE. *A library for doing polyhedral operations*, Technical report, n^o^ 785, Irisa, Rennes, France, 1993, http://www.inria.fr/rrrt/rr-2157.html.