INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Compsys

Compilation and Embedded Computing Systems

Rhône-Alpes

THEME COM

Table of contents

1. Team

Compsys exists since January 2002 as part of Laboratoire de l'Informatique du Parallélisme (Lip, UMR CNRS ENS-L*yon* UCB-L*yon Inria 5668), located at* ENS-L*yon, 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 leader
Tanguy Risset [CR Inria]

Administrative Assistant
Isabelle Antunes [T CNRS, part-time (up to november 2004)]
Chiraz Benamor [A CNRS, part-time (from november 2004)]

Research scientist
Alain Darte [CR CNRS]
Paul Feautrier [Pr ENS-Lyon]
Fabrice Rastello [CR Inria]

Research scientist (partners)
Antoine Fraboulet [MC Insa-Lyon]
Anne Mignotte [Pr Insa-Lyon, (up to september 2004)]

Post Doc
Fabrice Baray [Post Doc Inria]

PhD Student
Hadda Cherroun [Algerian Grant, part-time]
Nicolas Fournel [MESR Grant]
Antoine Scherrer [BDI CNRS and STMicroelectronics]
Cédric Bastoul [ATER, Clermont-Ferrand]
Philippe Grosse [Grant CEA-LETI, Grenoble]

2. Overall Objectives

Keywords: *automatic generation of vlsi chips*, *code optimization*, *compilation*, *dsp*, *fpga platforms*, *linear programming*, *memory optimization*, *parallelism*, *regular computations*, *scheduling*, *tools for polyhedra and lattices*, *vliw processors*.

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 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);
- high-level code transformations (including loop transformations for memory optimization);
- silicon compilation (with a link to micro-electronics);
- development of polyhedra manipulation tools.

These researches 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 system*. 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 much 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).

Recently a big emphasis has been put on embedded software. This priority 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, Thalès, 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. 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, the problem is a multi-criteria optimization and one has to take into account 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.

[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/

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 its 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, whether 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 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 in rationals, or, in few cases, in integers. These are well-known techniques, which are 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 implemented. Mono- and multi-dimensional scheduling techniques [20][19] 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 much 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 (increased 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 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 needs 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 to find powerful techniques towards the optimal solution. Past experience has shown that taking simultaneously all aspects of a problem into account is near 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: "High-level Code Transformations" (Section 3.3) is on top of "Optimization for Special Purpose Processors" (Section 3.2) and "Compilation of Parallel Embedded Architectures" (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 actions.

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

Embedded applications have special program profiles and dataflows. The power consumption is more than proportional to the clock frequency. Since the program is loaded in permanent memory (ROM, Flash, etc.), its compilation time is not 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 thus 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). These combinatorial problems arise mainly in the removal of the multiplexor functions (known as φ functions), which are inserted when converting into SSA form ("Static Single Assignment" [35]), in register allocation, in opcode selection, and in code placement for optimization of the instruction cache. These optimizations are mainly done in the last phases of the compiler, using an assembler-level intermediate representation. In industrial compilers, these optimizations are handled in independent phases using heuristics, in order to limit the compilation time.

One of the challenging features of today's processors is predication [26], which interferes with all optimization phases, as does the SSA form. 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, 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?

Compilation for embedded processors is difficult because the architecture and 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. 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 [17] generate many open problems. Our objective is to contribute to the understanding and the solution of these problems, the main tool being the SSA [36] representation.

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 3.4.2), 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 between 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 on many industrial compilers, is to do software pipelining using a technique known as 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. However, 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 these 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, 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. Other methods include the use of "linearized" schedules.

3.3. High-Level Code Transformations

Participants: Alain Darte, Paul Feautrier, Antoine Fraboulet, Anne Mignotte, 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 to all concerned 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, wrongly, to be less important than parallelism extraction. Today, we have realized that performance is a resultant 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. Firstly, 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 bounded 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 compiler. 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 is adverse to 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. Theoretical Models

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 [22], 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 which 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 details.

3.3.2. Experiments

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.

In the framework of a cooperation with the company Cadence, Antoine Fraboulet had the opportunity of evaluating these methods with the help of codesign tools like VCC [21]. Note also that Antoine Fraboulet had a collaboration with the R2D2 project, on loop compilation as a tool for the design of specialized hardware coprocessors.

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.

3.4. Compilation of Embedded Parallel Architectures

Participants: Fabrice Baray, Alain Darte, Paul Feautrier, 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 the scientific challenges of the subject.

3.4.1. Design of Accelerators for Signal Processing

The advent of high-level synthesis techniques allows one to create specific design for reconfigurable architectures, for instance with MMAlpha [7] (for regular architectures) or 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 uses 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. Scheduling Process Networks

Kahn process networks (KPN) were introduced thirty years ago [27] 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. This property allows one to define a semantics and to talk meaningfully of the equivalence of two implementations. As a bonus, the circuit diagrams used by signal processing specialists can be translated on-the-fly into KPNs.

The problem with KPNs is that they rely on an asynchronous execution model, while VLIW processors and Asic are synchronous or partially synchronous. Thus, there is a need for a tool 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,

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

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, we have defined another model, *Communicating Regular Processes* or 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 nearer to the expectations of system designers.

A prototype of a scheduler for CRP's, Syntol, is under development. This tool extends the scheduling techniques we developed for high-performance computers. Handling real-time constraints in this model is especially easy. For instance, if the constraint is in the form of an upper bound on the latency, one has to write that all values of the schedule are less than a maximum. This gives constraints similar to data dependence constraints and can be solved by the same tools. It is even possible to keep the clock period as an unknown, and to select the maximum value for which the problem is still feasible. Since power consumption is a decreasing function of the clock period, this is a way of reducing dissipation.

3.4.3. Modular Parallelization and Interfaces

The scheduling techniques of MMAlpha and Syntol 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 Ecole 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.

This work must be continued; one of the crucial points is the handling of off-the-shelf components (IP) in the design of embedded systems.

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 [8].

3.4.4. 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 still 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 source voltage are all techniques that decrease the power consumption.

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

- One can search for low-power algorithms. Since, for most processors, the energy consumption is proportional to the number of executed operations, this is most often synonymous to finding low complexity algorithm.
- One can act at the level of the compiler. The rule here is to classify operations in term 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 and cache.
- 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 would be to let the software or operating system be responsible for these controls.

The Compsys group has just started to work on this subject, in cooperation with CEA-LETI in Grenoble. Two PhD students, Nicolas Fournel and Philippe Grosse participate in this effort.

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 leading in this 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 Ecole 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 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 for polyhedral methods (the GCC group, Reservoir Labs in the USA).

Two basic tools that have emerged from this period are Pip [18] and the Polylib [37]. They are currently the only available tools since maintenance has stopped on Omega. 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 be the design of new tools for new scheduling techniques.

In the following, we distinguish 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 the 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. A bridge for combined use of the two tools has been created by Cédric Bastoul. These tools have been the core of new code generation tools [16] [33] 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 the Polylib for several years. Other participants are in 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 better control of the size of intermediate values; comparison with commercial tools like Cplex, for the non-parametric component of the tool.

- For the Polylib, a better handling of $\mathbb{Z}$-polyhedra used to target loops with non unit increments.
- For higher-level tools, Cédric Bastoul (a student of Paul Feautrier) 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 in experimental compilers.
- For all these tools, we want to strengthen the user community by participating in the Polylib forum and organizing meetings for all 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 3.4.2 above). 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 Clak, that can be used to derive affine mappings with modulos for memory reuse (see more details in Section 4.6).

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

Here again, our contributions in these areas will be facilitated by our leadership in polyhedral tools. We nevertheless expect that, as 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. Polylib

Participant: Tanguy Risset.

Polylib (available at http://www.irisa.fr/polylib/) is a C library for polyhedral operations. The library operates on the objects of linear algebra, like vectors, matrices, convex polyhedra, unions of convex polyhedra, lattices, $\mathbb{Z}$-polyhedra, and parametric polyhedra. Tanguy Risset has been responsible for the development of the Polylib for several years. More recently 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 in use by many groups all over the world.

4.2. Pip

Participant: Paul Feautrier.

Paul Feautrier is the main developer for 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 the following URL: 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.

4.3. 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 thesis, and MMAlpha is one of the few available tools for very high-level hardware synthesis (including the design of parallel Asic). 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.

4.4. Syntol

Participants: Paul Feautrier, Hadda Cherroun.

Syntol is a process network scheduler. Its development has benefited from the help of François Thomasset (Inria-Rocquencourt). Hadda Cherroun is responsible for the development of the back-end of the scheduler, whose aim is to generate a VHDL description, at the RTL level, of a digital circuit implementing the source algorithm.

This year has seen the change from the Kahn process network model to Communicating Regular Processes, the implementation of modular and structured scheduling techniques, and the connection to CLooG (see Section 4.8) for code generation. Syntol is still in development. We expect its front end (scheduling) will become mature enough for open distribution in 2005.

4.5. 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 they will contribute to the visibility of the project.

4.6. Clak: Algorithms on Integral Lattices

Participants: Fabrice Baray, Alain Darte.

Our recent work on memory optimizations (see Section 5.8) identified new mathematical tools useful for the automatic derivation of array mappings that enable memory reuse. We developed several algorithms and heuristics that rely on some mathematical concepts defined for integer lattices such as the notion of admissible lattice, of critical lattice, the successive minima of Minkowski, lattice basis reduction, and so on. Fabrice Baray, post-doc Inria, is currently developing a tool, called Clak (for Critical LAttice Kernel), that computes or approximates the critical lattice for a given 0-symmetric polytope (i.e., a lattice, with minimal determinant, whose intersection with the polytope is reduced to 0). This tool is a complement to the Polylib suite and we believe it 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.

4.7. Register Allocation

Participants: Alain Darte, Cédric Vincent.

Within the collaboration with the MCDT team at STMicroelectronics (see Section 6.1), Cédric Vincent, advised by Alain Darte, developed a complete register allocator in the assembly-code optimizer of STMicroelectronics. This work was part of the bachelor degree of Cédric Vincent. This was the first time a complete implementation was done with success, outside the MCDT team, in their optimizer. See more details in Section 5.2.

4.8. CLooG: Loop Generation

Participants: Paul Feautrier, Cédric Bastoul.

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][4]. The availability of CLooG on the Web as free software has been a triggering factor for a recent increase of interest for the polytope model. Beside being used in several parallelizing compilers, CLooG has found applications in some unconnected domains, as for instance in the search for optimal approximations to elementary transcendental functions.

5. New Results

5.1. Variable Coalescing under ssa Form for Assembly Code

Participant: Fabrice Rastello.

The work presented here is a joint work with François de Ferrière and Christophe Guillon from the MCDT team at STMicroelectronics.

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. Since the φ functions cannot be implemented, they must be replaced by register copy instructions when generating actual machine code. If this is done naively, too many useless copies are generated and a coalescing phase is needed to eliminate them.

Leung and George [29] use a SSA form for programs represented as native machine instructions, including the use of machine dedicated registers. For this purpose, they handle renaming constraints thanks to a pinning mechanism. Pinning φ arguments and their corresponding definition to a common resource is also a very attractive technique for coalescing variables.

Extending this idea, we proposed a method to reduce the φ-related copies during the out-of-SSA translation, thanks to a pinning-based coalescing algorithm that is aware of renaming constraints. The pinning-based coalescing formulation is not strictly equivalent to the initial problem of variable coalescing, which is still unsolved. We proved that our formalism is NP-complete, and, as a corollary, that the initial problem is also NP-complete in the size of the largest φ function. We have implemented our algorithm in the LAO assembly code optimizer from STMicroelectronics. Comparison with other approaches gives good results, which we explain by comparing to several hand-coded examples.

All these results have been presented at the international conference IEEE CGO 2004 [8].

5.2. Register Allocation and ssa Form Properties

Participants: Alain Darte, Fabrice Rastello, Cédric Vincent.

The work presented in this paragraph is done in collaboration with members of the MCDT team at STMicroelectronics. It is the logical extension of the work described in the previous paragraph.

The previously-described work deals with the minimization of the number of copy instructions. Reduction of copy instructions is performed by the copy propagation, the partial redundancy elimination, and the register coalescing phases. None of these phases minimizes the number of copy instructions. Actually, we proved several new NP-complete results related to what we call the "aggressive coalescing" problem. The term "aggressive" is in opposition to the term "conservative" used for the coalescing performed during the register allocation phase: the aggressive coalescing tries to reduce the number of copies, with no register constraints, while the conservative coalescing tries to reduce it without changing the chromatic number of the interference graph. This other problem is also trivially NP-complete. Unfortunately, the coalescing problem is highly related to the colorability of the interference graph and the two sub-problems of register allocation and register coalescing cannot be dissociated.

Hence, our study led us to consider the register allocation problem in details. We are currently exploiting the properties of the SSA form to improve register allocation, and in particular, how to spill and split variables. These results are still incomplete: they must be firmed up, and implemented. Nevertheless, we believe that

the new concepts we introduced should allow us to design new heuristics for register allocation with fixed scheduling, both in the general case and for the ST220 DSP. LAO2, an optimizer for assembly code under development at STMicroelectronics, is the ideal vehicle for these experiments. So far, we developed a complete register allocator for LAO2, based on a well-known strategy, the iterated register coalescing of George and Appel [23]. This first implementation was done by Cédric Vincent, advised by Alain Darte, during his bachelor degree and part of the funding from our contract with MCDT was used to support him. This exercise was for us a way to get used to the internal representation of the LAO2 optimizer and, for the MCDT team, this was the first success of an external implementation in their optimizer. We will use this (not so) elementary register allocator as a basis for comparison for our future developments.

Among other problems, our current work is the design of a heuristic for the restricted problem of conservative register coalescing during the translation out of SSA form. This is part of the contract, currently under negotiation, with the MCDT team at STMicroelectronics.

5.3. Instruction Cache Optimization

Participants: Fabrice Rastello, Éric Thierry [Lip, Trio Project].

Results: The work presented in this paragraph is a joint work with Christophe Guillon and Thierry Bidault from the MCDT team at STMicroelectronics.

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 lines; 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 two objective functions:

COL Minimizing the number of conflicts for a given execution trace. This is equivalent to the *Max-K-Cut* and *Ship-Building* problems. We have proved that, for n functions, *COL* is not $n^{1-\epsilon}$-approximable.

EXP Minimizing the size of the code. This is equivalent to a traveling salesman problem (building an Hamiltonian circuit) on a very special graph said *Cyclic-Metric*. With Florent Bouchez (a third year student of ENS-Lyon), we have shown that, for a given solution of *COL*, *EXP* can be solved optimally in $O(n \min(n, L))$.

Classically [24], the problem is solved in two steps: *COL* then *EXP*. In the light of our theoretical study, we have a) taken final program size into consideration during the first phase (*COL*) of Gloy and Smith algorithm, b) replaced the second phase (*EXP*) by our optimal solution. These optimizations have been implemented in the linker of the ST220 processor. Experiments on several representative benchmark suites show that code expansion is significantly reduced from 177% for the initial Gloy and Smith algorithm to 8%, while the cache miss reduction is nearly the same as the Gloy and Smith solution with 35% cache miss reduction. Practical results have been presented at the international conference IEEE CASES 2004 [7]. Part of the theoretical results will appear in the Journal of Embedded Computing [3].

Current work: Experiments have shown the importance of a new objective function:

NBH The neighborhood objective function is due to the cache line granularity (larger than the size of an instruction) and to the presence of a prefetch buffer. Indeed, memory slots just after the end of a procedure have a high probability to be fetched whenever the end of that procedure has to be executed (symmetric situation for the beginning of a procedure). The objective of *NBH* problem is to put together procedure extremities that have a high affinity in terms of time locality. This is equivalent to the *Traveling Salesman problem*.

We have proved *NBH* to be NP-complete. So far, we have implemented a classical "branch-and-bound" heuristic in the STMicroelectronics instruction cache optimizer tool which provides improvements compared to the initial algorithm.

Perspectives: Experiments have also shown that the *COL* phase still contains many degrees of freedom. Hence, in practice, it might be possible to optimize together both the *NBH* and *COL* objective functions without producing any code size expansion. Our idea is to first regroup procedures with high neighboring and conflict affinity within groups that fit in the cache. This is a kind of *Min-Cut* problem that can be solved using a randomized algorithm. Then, within each group, we can find several ordering of procedures with good affinity. We will use for that the previously-used branch-and-bound heuristic. Finally, we will merge all groups together using a method similar to the Gloy and Smith's one. This project is part of the contract, currently under negotiation, with the MCDT team at STMicroelectronics.

5.4. Generating SystemC Simulation Models from High-Level Synthesis Tools

Participants: Antoine Fraboulet, Tanguy Risset.

As part of the participation of Compsys in the SocLib project (http://soclib.lip6.fr), the implementation of a SystemC generator for MMAlpha has been provided. This has raised the problem of generating efficient simulation models from high level design tools of digital circuits. The classical decomposition of hardware into three automata (Moore, Mealy, and Transition automata) proposed by the SocLib group has a major drawback: it duplicates code making it slower and more difficult to debug.

We have proposed a method to solve this problem by changing slightly the way the different automata are built. We have explained precisely how to build the new automata from a register transfer level representation of the circuit in linear time. We have experienced a 40% simulation time improvement on a simple example. This work has been published at the Samos Workshop [9].

Simulation models are now a critical issue in SoC design. The integration of several simulation model generators in any high-level synthesis tool is now mandatory to integrate the generated circuits in a SoC simulation environment. We believe that our approach provides very efficient cycle-accurate simulation models.

5.5. Integration of $\mathbb{Z}$-polyhedra into MMAlpha

Participants: Antoine Fraboulet, Tanguy Risset.

$\mathbb{Z}$-polyhedra computations have been available in the Polylib for a few years [30]. The extension of the Alpha language to handle union of $\mathbb{Z}$-polyhedra as domains (instead of simply unions of polyhedra) has also been studied [34], however it was impossible to implement this extension in the MMAlpha software because of a technical problem: the library linking C and Mathematica were not compatible in the Windows operating system. We solved this problem this year and were able to finally extend many of the transformations of the MMAlpha tools to handle $\mathbb{Z}$-polyhedra. These are: analysis of the program correctness, change of basis (used to reflect the space-time transformations), normalization (used to have a readable program after many transformations), and all basic operations on polyhedra, $\mathbb{Z}$-polyhedra, and matrices. Not all transformations have been extended however; in particular, the scheduler and the code generators (C, SystemC, or VHDL code) have not been extended to handle $\mathbb{Z}$-polyhedra yet.

Some manual attempts have been made to use this new tool to express partitioning. However, further experiments are needed, as it is not clear whether the best way of expressing partitioning and resource constraints is to use $\mathbb{Z}$-polyhedra or to keep polyhedra with additional dimensions (the translation to $\mathbb{Z}$-polyhedra being performed at the very last time, during code generation).

5.6. On Chip Traffic Analysis

Participants: Antoine Fraboulet, Tanguy Risset, Antoine Scherrer.

This work is being done in a PhD co-funded by the CNRS and STMicroelectronics.

Recent progress of the CMOS technology allows the integration of a complete parallel machine on a single chip. Connecting the various components of this machine 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.

Our approach is to (semi-)automatically generate a traffic generator from an execution trace of the component we want to replace. We work on this topic in close relation with STMicroelectronics. We are also working in collaboration with Patrice Abry, physicist at 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).

Prior to this work, we provided a hardware wrappers classification, which was presented at the SCS conference [10], proposing different levels of complexity depending on the wrappers needed by the application. We are now working on the set-up of the flow for generating traffic generators, and the first results should be submitted in early 2005. A publication of the theoretical part of this work is also scheduled for early 2005.

5.7. Data-Flow IP Interface Generator

Participants: Antoine Fraboulet, Tanguy Risset, Antoine Scherrer.

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 IP. This interface allows us to efficiently map burst-mode communications generated by the processor to data-flow architectures. This interface has been presented at the SocLib university booth during the DATE'04 conference [13]. We have started a collaboration with the Lester laboratory (http://web.univ-ubs.fr/lester/) to add support for architectures generated by their tool Gaut. 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 have been presented at the ASAP'04 conference [6].

5.8. Memory Reuse and Modular Mappings

Participants: 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 appear 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. A possibility to reduce the memory size is to reuse the memory locations assigned to array elements, following, among others, the work of Francky Catthoor's team [25], of Lefebvre and Feautrier [28], and of Quilleré and Rajopadhye [32].

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 work. We found that they are all particular cases of more general heuristics for finding a critical lattice. We also proved that, in practice, we can build approximations that do not differ from the optimum more than by a multiplicative constant, which depends only on the dimension of the problem.

In 2004, we continued our study, analyzing in more details the strengths and weaknesses of previous approaches for array reuse, revealing similarities and differences with early 70's and 80's work on data layouts allowing parallel accesses – results that were forgotten in the meantime –, exploring more deeply the properties of linear mappings with, in particular, a new result concerning the minimal dimensions of these mappings, etc. This work has been submitted to 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. All details are available in the research report [12].

In practice, heuristics for the memory reuse problem are already in use in tools like PiCo. We hope these researches will give a new interest to a possible cooperation with Synfora. We are also currently developing the algorithms on lattices, successive minima, and critical lattices, needed to implement our memory reuse strategies (see Section 4.6). This tool should have an impact both for the practical problem of designing hardware accelerators and for the mathematical problem of finding the critical lattice of an object. This is a fundamental algorithmic problem, which has not attracted much attention from mathematicians. Clak, built on top of our present tools Pip and Polylib, is a perfect extension for our tool suite on polyhedral/lattice manipulations.

5.9. 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. Rob Schreiber has been working on the high-level synthesis of hardware accelerators in the past (PiCo project) but 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 the 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 [31], 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. All details of this work are available in the research report [11], a shorter version has been submitted to PPoPP'05.

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). This research, if continued, would be part of a contract, currently under negotiation, with HP-Labs.

5.10. 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.

It 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.

All in all, the method improves the scheduler performance, facilitates modular scheduling, and promotes reuse. The subject has already been presented to the Samos workshop [5]. An extended version is in preparation. The method is implemented in the prototype scheduler Syntol, and is the basis for further work on high-level synthesis of digital systems.

5.11. Locality Optimization

Participants: Paul Feautrier, Cédric Bastoul [ATER, Clermont-Ferrand].

Both general-purpose processors and embedded systems use a memory hierarchy, because DRAM memories are much slower than processors. While for general-purpose processors this hierarchy takes the form of a succession of caches (whose management is entirely under control of the hardware), the tendency in embedded systems is to use scratch-pad or local memory, which are managed by software. The reason is that for local memories, one can still hope to provide real-time guarantees, which is impossible for caches.

We have started, four years ago, the design of a method for the automatic management of a local memory, called *chunking*. The method has been emulated for a classical cache, with very good results. CLooG has been developed as part of this effort. This has resulted in the PhD thesis of Cédric Bastoul (defended December 7, 2004) and in several papers, including one of the distinguished papers at Europar 2004.

6. Contracts and Grants with Industry

6.1. Contracts with stmicroelectronics on Assembly Code Optimizations

Participants: Alain Darte, Fabrice Rastello, Cédric Vincent.

The ProCD (Programmable Consumer Devices) contract was funded by STSI (i.e., contract between STMicroelectronics and the Ministry of Industry). Its objective was the design of programs for multimedia signal processing on a VLIW (Very Long Instruction Word) architecture. Compsys contribution was to work on combinatorial optimization problems that arise when compiling programs for VLIW processors and, in particular SSA removal (see Section 5.1), register allocation (see Section 5.2), and code placement for instruction cache optimization (see Section 5.3).

Some of the funds from this contract were used as a support for Cédric Vincent who developed, as part of his bachelor degree, a full iterated register allocator in LAO2, the code optimizer of the MCDT team.

6.2. Contracts with stmicroelectronics on Special-Purpose Circuit Synthesis

Participant: Tanguy Risset.

There is another STSI contract between Compsys and STMicroelectronics (Crolles plant) on experimentation with MMAlpha for synthesis of STMicroelectronics special purpose circuits. Due to administrative problems, the funding is not yet available, but the work has nevertheless begun (overhauling the Alpha-VHDL translator). The contract is expected to be executed next year.

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), 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 proposal which has now the official ITEA label.

7.2. Soclib: « A Modeling & Simulation Platform for Systems on Chip »

Participants: Antoine Fraboulet, Tanguy Risset.

Tanguy Risset and Antoine Fraboulet are members of the SocLib project (http://soclib.lip6.fr). Its aim is to develop a library of simulation models for virtual components (IP cores) for Systems on Chip.

7.3. CNRS Collaboration with the University of Illinois at Urbana-Champaign (USA)

Participants: Paul Feautrier, Alain Darte.

A convention between UIUC and CNRS supports visits and cooperation between David Padua's team and Compsys.

7.4. Procope Convention with Passau University (Germany)

Participant: Paul Feautrier.

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

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 [15][14] 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), with François Charot and Patrice Quinton (Inria project R2D2), and with Alain Greiner and Fréderic Pétrot (Asim, LIP6).
- Compsys participates in the EmSoc project with LETI (CEA Grenoble) on software techniques for power minimization.

8. Dissemination

8.1. Conferences and Journals

- Alain Darte was one of the 3 program chairs of the IEEE 14th International Conference on Application-specific Systems, Architectures and Processors (ASAP 2003) and he was (and will be) member of the program committees of ASAP 2003, ASAP 2004, and ASAP 2005. He was member of the program committees of CASES 2003 and CASES 2004 (ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems) and he was publicity chair for CASES 2004. He was one of the 6 members (3 Americans, 3 Europeans) of the technical program committee for the new topic S2 (Compilers, architectures, and software synthesis for embedded systems) of the conference DATE 2005 (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. He was the general chair of the ACM International Conference on Supercomputing (Saint-Malo, June 26–July 1, 2004), and a member of the program committee of the conference Compiler Construction 2004.
- Tanguy Risset is a member of the editorial board of Integration: the VLSI Journal. He will be a member of the program committee of the SAMOS conference in 2005.

8.2. Post-Graduate Teaching

- In 2004-2005, Alain Darte and Paul Feautrier are sharing a master-degree course on advanced compilation and program transformations.
- Paul Feautrier is thesis advisor for Cédric Bastoul (now ATER, Clermont-Ferrand), Christophe Alias (UVSQ, co-advisor Denis Barthou), 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 untitled “Design of embedded computing systems” at Insa-Lyon.

8.3. 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 guided tour of Unix (L3IF).
  - Operational Research (M1).
  - Compilation project (M1).
- Paul Feautrier is the coordinator of the “Architecture and Compiler” track of the new Master of ENS-Lyon.
- In 2004-2005, Tanguy Risset and Fabrice Rastello are in charge of the Compilation course to Master 1 students at ENS-Lyon.

8.4. 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 evaluation commission (CE) of INRIA.

8.5. Defense Committee

- Tanguy Risset was a member of the defense committee for François Donnet (January 20, 2004, LIP6, Paris).
- Paul Feautrier is reviewer for the HDR (Professorial Thesis) of Olivier Beaumont (Bordeaux I), and for the PhD thesis of Benoît Meister (ULP), and a member of the defense committee for the HDRs of Mohammed Jemni (Tunis), J.-L. Lamotte (UPMC), and for the PhDs of Pierre Amiranoff (CNAM) and Cédric Bastoul (UPMC).

8.6. Workshops, Seminars, and Invited Talks

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

- Tanguy Risset gave a talk at the SAMOS 2004 workshop. He was invited to give a talk to the Flemish network PA3CT untitled "Some trends in High Level Synthesis Tools" (http://www.elis.rug.ac.be/wog/).
- Tanguy Risset and Antoine Fraboulet presented a demonstration at the university booth of DATE'04 untitled "HW/SW Fast and Accurate Prototyping" (demo available at http://soclib.lip6.fr/)
- Alain Darte gave a talk at the 11th Workshop on Compilers for Parallel Computing (CPC 2004), in Seeon (Germany). He attended CASES 2004 as a member of the organization.
- Paul Feautrier gave one of the invited lectures at FORMATS+FTRTFT'04, September 22–24, 2004, in Grenoble, and gave a talk at the SAMOS 2004 workshop.
- Antoine Fraboulet, Antoine Scherrer, and Tanguy Risset attended the annual RTP SoC workshop (May 16–19, 2004) and participated to the presentation of the SocLib project.
- Antoine Scherrer made a presentation at the PhD student meeting organized by the Architecture, Network, and Parallelism CNRS research group (GDR ARP), September 29, 2004 in Antibes.
- Antoine Scherrer attended the Summer School WAMA: Wavelet And Multifractal Analysis 2004, Corsica, France, July 19–31, 2004 - http://wama2004.org/

9. Bibliography

Articles in referred journals and book chapters

[1] C. BASTOUL, P. FEAUTRIER. *Adjusting a Transformation for Legality*, in "Parallel Processing Letters", to appear.

[2] A. DARTE, G. HUARD. *New Complexity Results on Array Contraction and Related Problems*, in "Journal of VLSI Signal Processing", to appear, vol. 40, nº 1, May .

[3] C. GUILLON, F. RASTELLO, T. BIDAULT, F. BOUCHEZ. *Procedure Placement using Temporal-Ordering Information: Dealing with Code Size Expansion*, in "Journal of Embedded Computing", to appear, 2004.

Publications in Conferences and Workshops

[4] C. BASTOUL, P. FEAUTRIER. *More Legal Transformations for Locality*, in "Euro-Par'04", Distinguished Paper Award, vol. LNCS 3149, Springer Verlag, 2004, p. 272–283.

[5] P. FEAUTRIER. *Scalable and Modular Scheduling*, in "Computer Systems: Architectures, Modeling and Simulation (SAMOS 2004)", A. D. PIMENTEL, S. VASSILIADIS (editors)., vol. LNCS 3133, Springer Verlag, July 2004, p. 433–442.

[6] A. FRABOULET, T. RISSET. *Efficient On-Chip Communications for Data-Flow IPs*, in "Application Specific Array Processors (ASAP'04)", IEEE Computer Society Press, 2004, p. 293-303.

[7] C. GUILLON, F. RASTELLO, T. BIDAULT, F. BOUCHEZ. *Procedure Placement using Temporal-Ordering Information: Dealing with Code Size Expansion*, in "International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'04)", ACM Press, 2004, p. 268–279.

[8] 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.

[9] A. SCHERRER, A. FRABOULET, T. RISSET. *Hardware-Software Fast and Accurate Prototyping with Soclib & MMAlpha*, in "Computer Systems: Architecture, Modeling, and Simulation (SAMOS 2004)", A. D. PIMENTEL, S. VASSILIADIS (editors)., LNCS, vol. 3133, Springer Verlag, July 2004, p. 453–462.

[10] A. SCHERRER, T. RISSET, A. FRABOULET. *Hardware Wrapper Classification and Requirements for On-Chip Interconnects*, in "Signaux, Circuits et Systèmes 2004, Monastir, Tunisie", March 2004, p. 31-34.

Internal Reports

[11] A. DARTE, R. SCHREIBER. *Nested Circular Arc Families: A Model for Barrier Placement in Single-Program, Multiple-Data Codes with Nested Loops*, Technical report, nº RR2004-57, LIP, ENS-Lyon, December 2004.

[12] A. DARTE, R. SCHREIBER, G. VILLARD. *Lattice-Based Memory Allocation*, Technical report, nº RR2004-23, LIP, ENS-Lyon, April 2004.

Miscellaneous

[13] A. SCHERRER, A. FRABOULET, T. RISSET. *Hardware-Software Fast and Accurate Prototyping with Soclib & MMAlpha*, February 2004, Design, Automation and Test in Europe (DATE'04), University Booth Demonstration.

Bibliography in notes

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

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

[16] 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.

[17] 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.

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

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

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

[21] A. FRABOULET. *Optimisation de la mémoire et de la consommation des systèmes multimédia embarqués*, Ph. D. Thesis, INSA de Lyon, November 2001.

[22] 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.

[23] L. GEORGE, A. W. APPEL. *Iterated Register Coalescing*, in "23rd ACM Symposium on Principles of Programming Languages, St. Petersburg Beach, FL, USA", January 1996, p. 208-218.

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

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

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

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

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

[29] A. L. LEUNG, L. GEORGE. *Static Single Assignment Form for Machine Code*, in "ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)", 1999, p. 204–214.

[30] S. P. K. NOOKALA, T. RISSET. *A Library for Z-polyhedral Operations*, Technical report, n° 1330, Irisa, Rennes, 2000.

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

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

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

[34] P. QUINTON, S. V. RAJOPADHYE, T. RISSET. *Extension of the Alpha language to recurrences on sparse periodic domains*, in "Int. Conf. on Application Specific Array Processors", 1996.

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

[36] 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.

[37] D. WILDE. *A library for doing polyhedral operations*, Technical report, n° 785, Irisa, Rennes, France, 1993.