INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Compsys

Compilation and Embedded Computing Systems

Grenoble - Rhône-Alpes

THEME COM

Table of contents

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. The team 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 became a full Inria project in January 2004. It has been evaluated in Spring 2007, positively, and will continue 4 more years.*

1. Team

Research Scientist

Alain Darte [Team Leader, Research Director (DR) CNRS, HdR]
Fabrice Rastello [Research Associate (CR) Inria]

Faculty Member

Paul Feautrier [Professor (Pr) ENS-Lyon, HdR]

Technical Staff

Quentin Colombet [Contract within Minalogic Sceptre]

PhD Student

Benoit Boissinot [ENS-Lyon grant, 2006-...]
Florent Bouchez [ENS-Lyon grant, 2005-...]
Alexandru Plesco [MESR grant, 2006-...]
Clément Quinson [BDI CNRS and STMicroelectronics, 2005-...]

Post-Doctoral Fellow

Laure Gonnord [ATER UCB-Lyon, September 2008-...]
Ouassila Labbani [December 2006-September 2008]

Administrative Assistant

Caroline Suter [Part-time]

2. Overall Objectives

2.1. Introduction

Compsys started as an Inria project in 2004, after 2 years of maturation, and was positively evaluated in Spring 2007 after its first 4 years period (2004-2007). It will continue for 4 more years with updated research directions. The next section defines the general context of Compsys activities. The second section specifies the research objectives targeted during the first 4 years, the main achievements over this period, and the new research directions Compsys will follow in the upcoming years. The last section highlights the main results of 2008.

2.2. General Presentation

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*.

The objectives of Compsys are twofold: to increase our knowledge of embedded computing systems and to adapt/extend code optimization techniques, primarily designed for high performance computing, to the special case of embedded computing systems.

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. Today, as the embedded systems market grows and evolves, this view of embedded systems tend to be too restrictive. Many aspects of general-purpose computers apply to embedded platforms. Nevertheless, embedded systems remain characterized by application types, constraints (cost, power, efficiency, heterogeneity), market.

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). Compsys is primarily concerned with the second type of embedded systems, which is now referred to as *embedded computing systems*. The objective is to develop design and compilation methods for compute-intensive processing with large data sets 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. Our priority towards embedded software is motivated by the following observations: a) the embedded system market is expanding. Among many factors, one can quote pervasive digitalization, low cost products, appliances, etc. b) software engineering for embedded systems is not well developed in France, especially if one considers the importance of actors like Alcatel, STMicroelectronics, Matra, Thales, ..., c) 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.

The aims of Compsys are to develop new compilation and optimization techniques for the field of embedded computing system design. This field 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. The specificity of Compsys is the study of code transformations (at source level or at assembly level) 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 some of the research in the Inria projects Alchemy and Caps) and to high-level architectural synthesis (like the Inria project Cairn).

As for high-level synthesis, several compilers/systems have appeared, after some first unsuccessful industrial attempts. These tools are mostly based on C or C++ (SystemC, VCC, CatapultC, Altera C2H, and others). The support for parallelism in these tools is minimal, but academic projects are more concerned: Flex and Raw at MIT, Piperench at Carnegie-Mellon University, PiCo from HP Labs and now at the Synfora start-up, Compaan at the University of Leiden, Ugh/Disydent at LIP6 (Paris), Gaut at Lester (Bretagne), 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 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. The main expertise in Compsys for this aspect 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.

Another fundamental aspect of embedded computing systems is its use of various kinds of processors, with many specificities (instruction sets, registers, data and instruction caches) and constraints (code size, performance, storage). The development of compilers is central for this industry, as selling a platform without its programming environment and compiler would not be acceptable. To cope with such a range of different processors, the development of robust, generic (retargetable), though efficient, compilers is mandatory. But unlike “classical” compilation for general-purpose processors, compilers can be more aggressive (i.e.,

take more time to optimize) for optimizing some important parts of applications. This opens a new range of optimizations for compilers. Another interesting aspect is the introduction of intermediate platform-independent languages (Java bytecode-type) for which, on the contrary, lighter compilation mechanisms (i.e., faster and less memory-consuming) must be developed for this dynamic/just-in-time compilation. Our objective here, for compilation for embedded computing systems, is thus to revisit past compilation techniques, to deconstruct them and improve them in this new context.

2.3. Highlights of the first 4-years period

In its first period (2004-2007) Compsys had four research directions, centered on compilation methods for simple or nested loops. These directions were:

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

These research activities were primarily supported by a marked investment in polyhedra manipulation tools and, more generally, solid mathematical and algorithmic studies, with the aim of constructing operational software tools, not just theoretical results. Hence the fourth research theme was centered on the development of these tools.

The main event in 2007 was the evaluation of Compsys in April. The evaluation, conducted by Erik Hagersted (Uppsala University), Vinod Kathail (Synfora, inc), Ramanujam (Baton Rouge University) was positive. Compsys will thus continue for 4 years, with Inria support, but in a new configuration as Tanguy Risset and Antoine Fraboulet leave the project to follow research directions closer to their host laboratory at Insa-Lyon. The main achievements of Compsys, for this period, were the following:

- The development of a strong collaboration with the compilation group at STMicroelectronics, with important results on instruction cache optimization and register allocation.
- New results on the foundation of high-level program transformations, including scheduling techniques for process networks and lattice-based memory reuse techniques.
- Many original contributions with partners closer to hardware constraints, including CEA, related to SoC simulation, hardware/software interfaces, power models and simulators.

Due to the size reduction of Compsys (from 5 permanent researchers to 3 in 2008), the team will now focus on two research directions only:

- Code generation for embedded processors, on the two opposite, though connected, aspects: aggressive compilation and just-in-time compilation.
- High-level program analysis and transformations for high-level synthesis tools.

2.4. Highlights of 2008

Compsys has continued its activities on static single assignment (SSA) and register allocation, in collaboration with STMicroelectronics, but working more deeply on just-in-time compilation, in particular on the developments of code optimizations algorithms that take into account speed and memory footprint. This work has led to two important developments:

- An algorithm for fast liveness checking for SSA-form programs. This algorithm, developed by Benoit Boissinot, Sebastian Hack, Fabrice Rastello, and other colleagues outside Compsys, is the base for avoiding the computation and storage of liveness analysis and of an interference graph. It was presented at the IEEE/ACM International Symposium on Code Generation and Otpimization (CGO 2008) and received the best paper award (second year in a row for the Compsys team).

- An algorithm of out-of-SSA conversion that improves previous approaches on all aspects: it is provably correct, more general, easier to implement, faster, and less memory-consuming. This algorithm, developed by Benoit Boissinot, Alain Darte, Fabrice Rastello, and other colleagues from STMicro, will be presented at CGO 2009. In particular, it uses our fast liveness checking algorithm.

The Sceptre Minalogic project entered its third year. Developments in register coalescing were completed, which gave rise to a publication at CASES 2008. The first implementation of a full SSA-based register allocator, with two phases (spilling + coloring/coalescing), was done by Quentin Colombet. A tutorial was co-organized by Fabrice Rastello at CASES 2008 on this topic. A book is planned. Compsys, through Alain Darte, was also in the heart of the signature (Nov. 2008) of a global Inria/ST cooperation that should, among others, help to organize research on embedded systems, for both partners.

In high-level synthesis, high-level transformations are being explored for improving performances and optimizing communications, in particular analyzing the industrial tool Altera C2H. Also, a joint project with the Cairn Inria project has been settled, in collaboration with STMicroelectronics, on this topic. We hope that this topic will be able to grow in 2009 thanks to the additional force of Christophe Alias who was just hired as an Inria researcher.

The ITEA Martes project was completed in September 2008. It focused on interoperability of the different partner's tools for embedded system design. The Compsys team (Ouassila Labbani and Paul Feautrier) implemented a parallelization tool, based on the Syntol scheduler, as a front end to the SPEAR Development Environment of Thales Reasearch. Martes as a whole won a silver award at the last ITEA symposium.

Compsys work on power optimization for Systems on Chip led to the defense of Nicolas Fournel and Philippe Grosse PhD theses (December 2007). In cooperation with Yves Durand (CEA-LETI), this work has been reported in a journal paper, which has recently been accepted by the Transactions on the Design Automation of Electronic Systems (TODAES) and will probably appear in 2009.

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). The first constraint is of course that the meaning (or semantics) of the source program must not be altered. Depending on the context, some parallelism must be exploited to cope with the 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, in the 90s, 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, and linear programming optimizations can be used, 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* that is often affixed to the method. Based on this theory, efficient software tools have been developed. One-dimensional and multi-dimensional scheduling techniques [12], [13] 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 more complex. 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. In other words, computer scientists are used to a paradigm in which the architecture is fixed and the only free variable is the program. But this is no longer true in embedded systems. The system itself is to be designed and is part of the "compilation" process. The critical problem is thus to adapt and extend traditional optimization compilation methods to handle many more free variables.

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). Today, some tools start to appear again, but either they cannot exploit loops (they unroll them), or they use limited loop optimization (simple pipeline for example). At some point, these tools will anyway have to face the same problems as for parallelizing compilers: language issues, loop transformations, memory optimizations, communication optimizations.

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. Applying past techniques is not enough. 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. As previously mentioned, these four directions were developed during the first 4 years of the project. Due to limited size, Compsys will continue to work on all aspects but will concentrate its forces on two main goals:

- Code optimization and JIT compilation;
- Front-end optimizations for high-level synthesis.

This document however still covers the four aspects.

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) family.

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, and because the amount of resources is strictly limited. For instance, 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, of predication, but also the special form of applications [10] generate many open problems. Our goal is to contribute to their understanding and their solutions.

Compilation for embedded processors is either aggressive or just in time (JIT). Aggressive compilation consists in allowing more time to implement costly solutions (so, looking for complete, even expensive, studies is mandatory): the compiled program is loaded in permanent memory (ROM, flash, etc.) and its compilation time is not significant; also, for embedded systems, code size and energy consumption usually have a critical impact on the cost and the quality of the final product. Hence, the application is cross-compiled, in other words, compiled on a powerful platform distinct from the target processor. Just-in-time compilation corresponds to compiling applets on demand on the target processor. For compatibility and compactness, the source languages are CLI or Java. The code can be uploaded or sold separately on a flash memory. Compilation is performed at load time and even dynamically during execution. Used heuristics, constrained by time and limited resources, are far from being aggressive. They must be fast but smart enough.

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 multiplexer functions (known as ϕ functions) that are inserted when converting into SSA form ("Static Single Assignment" [18]). These optimizations are usually done in the last phases of the compiler, using an assembly-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 a more global understanding of these optimization problems to derive both aggressive heuristics and JIT techniques, the main tool being the SSA representation.

In particular, 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 [15], 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 trade-off 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 [19], 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.

3.3. Platform-Independent Code Transformations

Participants: Christophe Alias, Alain Darte, Paul Feautrier.

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 [16] 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 multi-dimensional 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 global 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 second and third 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 [1].

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 [14], 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 relevance 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.

3.4. Hardware and Software System Integration

Participants: Christophe Alias, Alain Darte, Paul Feautrier.

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. The models of an embedded accelerator and of a compute-intensive program may be similar, but one may have to use different solution methods because the unknowns are no longer the same.

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

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 [2] (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 was using 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). As Tanguy Risset left the Compsys project, this activity related to MMAlpha will not be continued.

Another important issue is to understand what are the needs in program transformations to be able to use, in practice, high-level tools for synthesizing hardware accelerators. All such tools, including MMAlpha but not only, require that the input program respects some strong constraints on the code shape, array accesses, memory accesses, communication protocols, etc. Furthermore, to get the tool do what the user wants requires a lot of program tuning, i.e., of program rewriting. What can be automated in this rewriting process? Semi-automated? Our partnership with STMicroelectronics (synthesis) should help us answer such a question, considering both industrial applications and industrial HLS tools.

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, Compsys has proposed 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.

This work on traffic generators was achieved in the first four 4 years of the project and will not be continued. However, the problem of connecting accelerators together, or with the host processor, remains extremely challenging, on several aspects: how to optimize communications so that the accelerator is not limited by memory bandwidth, how to organize memories so as to optimize communications and computations, what language features need to be introduced to make such optimizations possible? These questions need to be addressed.

The first step to answering such questions was to adapt the MMAlpha tool to generate simulation models that are compatible with the SoCLib (http://soclib.lip6.fr) environment. A challenge is to develop a 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 data-flow hardware accelerator is needed for compute-intensive treatments. As Tanguy Risset left the Compsys project, these developments will not be continued in Compsys. However, the questions remain and will be examined in the light of other tools, possibly industrial ones such as Altera C2H.

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.

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

Low power can be achieved in four ways. 1) 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. 2) 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. 3) 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. 4) 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.

This work was done in cooperation with CEA-LETI in Grenoble and was highly successful. Two PhDs were defended and several papers were accepted by conferences and journals.

3.5. Federating Polyhedral Tools

Participants: Christophe Alias, Alain Darte, Paul Feautrier.

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, now Cairn and Alchemy, 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). Recently, several compiler groups have shown their interest in polyhedral methods (the GCC group, Reservoir Labs in the USA).

Two basic tools that have emerged from this early period are Pip [11] and Polylib [20]. 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

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 [9], [17] 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. Since Tanguy Risset left Compsys, Polylib is now just a tool Compsys uses but does not maintain.

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 modulo operations for memory reuse (see more details in Section 5.4).

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. Application Domains

4.1. Application Domains

Keywords: *Embedded computing systems*, *compilation*, *compilation*, *high-level synthesis*, *program optimizations*.

The previous sections describe our main activities in terms of research directions, but also places Compsys within the embedded computing systems domain, especially in Europe. We will therefore not come back here to the importance, for industry, of compilation and embedded computing systems design.

In terms of application domain, the embedded computing systems we consider are mostly used for multimedia: phones, TV sets, washing machines, game platforms, etc. But, more than the final applications developed as programs, our main application is the computer itself: how the system is organized (architecture) and designed, how it is programmed (software), how programs are mapped to it (compilation).

The industry that can be impacted by our research is thus all the companies that develop embedded systems and processors, and those (the same plus other) than need software tools to map applications to these platforms, i.e., that need to use or even develop programming languages, program optimization techniques, compilers, operating systems. Compsys do not focus on all these critical parts, but our activities are connected to them.

5. Software

5.1. Introduction

This section lists and briefly describes the software developments conducted within Compsys. Most are tools that we extend and maintain over the years. The previous reports contained descriptions of the Polylib tool, a C library of polyhedral operations and of MMAlpha a circuit synthesis tool for systolic arrays. Both were developed, with other colleagues outside Compsys, by Tanguy Risset who left Compsys. They are thus not maintained anymore in Compsys.

5.2. 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.piplib.org. 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).

5.3. Syntol

Participants: Paul Feautrier, Hadda Cherroun [Former member of Compsys].

Syntol is a modular process network scheduler. The source language is C augmented with specific constructs for representing Communicating Regular Process systems (CRP). The present version features a syntax analyzer, a semantic analyzer which is able to identify DO loops in C code, a dependence computer, a modular scheduler, and interfaces for CLooG (loop generator developed by C. Bastoul) and Cl@k (see 5.4). The dependence computer has been extended to handle casts, records (`structures`) and the modulo operator in subscripts and conditional expressions. A system for the automatic generation of XML reports has recently been implemented. XML is the preferred format for information exchange between tools.

The conversion of Syntol into Java is now complete. This has resulted in a much faster scheduler. The MuPAD version is no longer maintained. The next developments are a) a new code generator, based on the ideas of Boulet and Feautrier [7], which should be able to generate either C code or VHDL at the RTL level, b) tools for the construction of bounded parallelism schedules ,– virtual dependences, allocation functions, pseudo arrays, and c) a general strengthening of the analysis phase, with the long-term objective of handling as much of C as possible.

5.4. Algorithms on Integer Lattices and Memory Reuse Module: Cl@k+Bee

Participants: Christophe Alias, Fabrice Baray [Mentor, Former Post-Doc in Compsys], Alain Darte.

A few years ago, we identified new mathematical tools useful for the automatic derivation of array mappings that enable memory reuse, in particular the notions of admissible lattice and of modular allocation (linear mapping plus modulo operations). Fabrice Baray, post-doc Inria, developed a tool in 2005-2006, 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.) So far, Cl@k was a stand-alone optimization software tool, with no connections with programs or memory reuse. It has now been plugged by Christophe Alias (also Post-Doc Inria) into ROSE, a source-to-source program transformer, thanks to the development of a lifetime analyzer called Bee. Bee uses ROSE as a high-level parser, analyzes the lifetime of elements of the arrays to be compressed, and builds the necessary input for Cl@k, i.e., the 0-symmetric polytope of conflicting differences. See previous reports for more details.

Cl@k can be viewed as a complement to the Polylib suite, enabling yet another kind of optimizations on polyhedra. Bee is the complement of Cl@k in terms of its application to memory reuse. We believe that Cl@k 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. As for Bee, it is our first development into ROSE, which may become our future reference platform for implementations of high-level program transformations.

5.5. Register Allocation

Participants: Benoit Boissinot, Florent Bouchez, Quentin Colombet, Alain Darte, Sebastian Hack, Fabrice Rastello, Cédric Vincent [Former student in Compsys].

Our developments on register allocation with the STMicroelectronics compiler started when 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 (now CEC) team, in their optimizer. Since then, new developments are constantly done, in particular by Florent Bouchez, advised by Alain Darte and Fabrice Rastello, as part of his master internship and PhD thesis. During this year, Quentin Colombet has developed and integrated into the main trunk of LAO a full implementation of a two-phases register allocation. This includes a decoupled spilling phase as described by Sebastian Hack's PhD thesis and an up to date graph based coalescing. Current efforts also focuses on developing a tree-scan register allocator for the JIT part of the compiler. See more details in Sections 6.2, and 6.3.

6. New Results

6.1. Introduction

This section presents the results obtained by Compsys in 2008. For clarity, some earlier work is also recalled, when results were continued or extended in 2008.

6.2. Improvements to Conservative and Optimistic Register Coalescing

Participants: Florent Bouchez, Alain Darte, Fabrice Rastello.

This work is part of the contract (see Section 7.1) with the CEC team at STMicroelectronics.

In the context of embedded systems, it is crucial to minimize memory transfers to reduce latency and power consumption. Stack access due to variable spilling during register allocation can be significantly reduced using aggressive live-range splitting. However, without a good (conservative) coalescing technique, the benefit of a good spilling may be lost due to the many register-to-register moves introduced. The challenge is thus to find a good trade-off between a too aggressive strategy that could make the interference graph uncolorable without more spilling, and a too conservative strategy that preserves colorability but leaves unnecessary moves. Two main approaches are "iterated register coalescing" by George and Appel and "optimistic coalescing" by Park and Moon. The first coalesces moves one by one conservatively. The second coalesces moves regardless of the colorability, then undo coalescings to reduce spilling.

Restricting to greedy-k-colorable graphs (obtained after all spill decisions and, possibly, some split decisions), we show how these two approaches can be improved, optimistic coalescing with a more involved de-coalescing phase, incremental coalescing with a less pessimistic conservative technique. Unlike previous experiments, our results show that optimistic strategies do not outperform conservative ones. Our incremental conservative coalescing performs even better than our improved de-coalescing scheme and leads to about 15% improvement compared to the state-of-the-art optimistic coalescing. These results have been presented at CASES'08 [5]. Future work will aim at exploiting these results to vector register allocation (for STxP processor).

6.3. Revisiting Out-of-SSA Translation for Correctness, Efficiency, and Speed

Participants: Benoit Boissinot, Alain Darte, Benoit Dupont-de-Dinechin [STMicroelectronics], Christophe Guillon [STMicroelectronics], Fabrice Rastello.

This work is part of the contract (see Section 7.1) with the CEC team at STMicroelectronics. It is a first step towards the development of a fast and efficient SSA-based register allocator.

Static single assignment (SSA) form is an intermediate program representation in which many code optimizations can be performed with fast and easy to implement algorithms. However, some of these optimizations create situations where SSA variables that correspond to the same original variable now have overlapping live ranges. This makes the translation from SSA code to standard code more complicated. There are three issues: correctness, optimization (elimination of useless copies), speed (of algorithms). Briggs et al. addressed mainly the first point and proposed patches to correct the original approach of Cytron et al. For correctness, a simpler approach was then proposed by Sreedhar et al. who also developed techniques to reduce the number of generated copies. We go one step further. We propose a conceptually simpler approach, based on coalescing and a more precise view of interferences, where correctness and optimization are separated. First, it reduces significantly the number of generated copies compared to previous approaches. Second, it allows us to develop algorithms that are simpler to implement, with no patches or particular cases as in previous solutions, and that are fast enough to be suitable for just-in-time compilation.

These results will be presented at CGO'09 [3].

6.4. Fast Liveness Checking for SSA-Form Programs

Participants: Benoit Boissinot, Benoit Dupont-de-Dinechin [STMicroelectronics], Daniel Grund [Saarland University], Sebastian Hack, Fabrice Rastello.

This work is part of the contract (see Section 7.1) with the CEC team at STMicroelectronics.

Liveness analysis is an important analysis in optimizing compilers. Liveness information is used in several optimizations and is mandatory during the code generation phase. Two drawbacks of conventional liveness analyses are that their computations are fairly expensive and their results are easily invalidated by program transformations.

We proposed a method to check liveness of variables that overcomes both obstacles. The major advantage of our technique is that the analysis result survives all program transformations except for changes in the control-flow graph. For common program sizes our technique is faster and consumes less memory than conventional data-flow approaches. Thereby, we heavily make use of SSA-form properties, which allow us to completely circumvent data-flow equation solving.

We evaluated the competitiveness of our approach in the *LAO* code assembly optimizer. Our measurements use the integer part of the SPEC2000 benchmarks and investigate the liveness analysis used by the SSA destruction pass. We compared the time spent in liveness computations of our implementation against the one provided by that compiler. The results show that, in the vast majority of cases, our algorithm, while providing the same quality of information, is less time-consuming resulting in an average speed-up of 16%.

These results have been presented at CGO'08 [4] and were acknowledged by the **best paper award**.

6.5. Split Register Allocation: Linear Complexity Without Performance Penalty

Participants: Albert Cohen [Inria, Alchemy], Boubacar Diouf [Université Paris Sud, Alchemy], Fabrice Rastello.

This work is part of a collaboration with the Alchemy Inria project.

Just-in-time compilers are catching up with ahead-of-time frameworks, stirring the design of more efficient algorithms and more elaborate intermediate representations. They rely on continuous, feedback-directed (re-)compilation frameworks to adaptively select a limited set of hot functions for aggressive optimization. Leaving the hottest functions aside, (quasi-)linear complexity remains the driving force structuring the design of just-in-time optimizers.

This work addresses the (spill-everywhere) register allocation problem, showing that linear complexity does not imply lower code quality. We present a split compiler design, where more expensive ahead-of-time analyses guide lightweight just-in-time optimizations. A split register allocator can be very aggressive in its offline stage (even optimal), producing a semantical digest through bytecode annotations that can be processed by a lightweight online stage. The algorithmic challenges are threefold: (sub-)linear-size annotation, linear-time online stage, minimal loss of code quality. In most cases, portability of the annotation is an important fourth challenge.

We propose a split register allocator meeting those four challenges, where a compact annotation derived from an optimal integer linear program drives a linear-time algorithm near optimality. We study the robustness of this algorithm to variations in the number of physical registers and to variations in the target instruction set. Our method is implemented in JikesRVM and evaluated on standard benchmarks. The split register allocator achieves wall-clock improvements reaching 4.2% over the baseline allocator, with annotations spanning a fraction of the bytecode size.

6.6. Loop Transformations for High Level Synthesis and Communication Optimizations

Participants: Alain Darte, Alexandru Plesco, Tanguy Risset.

We have started a study on the use of a loop transformation front-end to high-level synthesis (HLS) tools. The study is based on the Wrapit loop transformation tools developed by the Alchemy team project and integrated into the ORC open-source compiler. This tool allows the user to identify part of C programs which are loops with static control, to indicate many loop transformations (fusion, code motion, strip mining, etc.) and to generate back a C program where the loops have been transformed. The Wrapit tool has been applied to C code, synthesized by the Spark HLS tool, showing important performance improvements of the resulting design (in terms of silicon area and communication optimization). This work was done by Alexandru Plesco and presented at SYMPA'08 [6].

These results also confirm that there is a strong need for data communication/storage mechanisms between the host processor and the hardware accelerator. Alexandru Plesco is currently investigating how to design a hardware/software interface model for enabling communication optimizations, such as burst communications. This may imply the design of a memory controller, the development of a performance model, as well as compilation techniques, possibly with compilation directives. The integration with existing HLS tools is tricky, current investigations are made with the Altera C2H code generator.

6.7. Loop Transformations for High Level Synthesis Tools

Participants: Christophe Alias, Alain Darte, Clément Quinson.

The topic of using compiling techniques as front-end optimizations to HLS tools is currently very hot. It is also important to identify what program transformations are required on source codes so that they can be accepted as input of HLS tools and achieve a good enough *Quality of Result*, i.e., good results both in terms of area and performance when compared to hand-written HDL. These manually-applied transformations could in the near future be automated or, at least, semi-automated, guided by the user. The "painful" part that can be automated would then be done by a compiler or, more precisely, a high-level source-to-source transformer.

One of the useful program transformations that we identified, both from a structural point of view and for the evaluation of computation-time (WCET, worst case execution time), is the transformation of WHILE loops into DO loops, i.e., the transformation of a loop whose iteration count (or even worse whose termination) is unknown into a loop with bounded number of iterations. This is of course not always possible but we are currently exploring a new strategy, based on both abstract interpretation and scheduling techniques, to extend the range of cases that can be treated. We plan to use, among others, a software tool developed by Nicolas Halbwachs and Laure Gonnord.

Due to the departure of Christophe Alias in the US for a post-doc, this work has unfortunately not progressed so far, but is still to be done. A project will start in 2009 on these different high-level program transformations, in collaboration with STMicroelectronics and the Cairn Inria project.

7. Contracts and Grants with Industry

7.1. Minalogic SCEPTRE project with stmicroelectronics on SSA, Register Allocation, and JIT Compilation

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

This contract started in October 2006 as part of the "pôle de compétitivité" MINALOGIC. The collaboration deals with the possible applications of the SSA form for program optimizations, including JIT and dynamic compilation for media processors. It concerns predication, register allocation, and instruction selection. Related work on register allocation are described in Sections 6.2, 6.3, and 6.5. Related work on JIT constraints are described in Sections 6.3, 6.4, and 6.5.

8. Other Grants and Activities

8.1. ITEA Project

Participants: Paul Feautrier, Antoine Fraboulet.

Compsys, mainly through Paul Feautrier, was involved in the Martes (Model driven Approach to Real-Time Embedded Systems development) ITEA project, which was completed in September 2008. The french partners of the project have focused their work on interoperability of their respective tools using a common UML meta-model. Ouassila Labbani has been hired as a postdoctoral engineer. The Compsys participation to MARTES was focused on the interaction between Syntol and the parallel design environment SPEAR of Thales Research. A gateway, partially based on the Eclipse framework, was implemented by Ouassila Labbani and has been successfully demonstrated at several review meetings. The MARTES project won a Silver Award at the 2008 ITEA Symposium.

8.2. Informal Cooperations

- Fabrice Rastello and Alain Darte have regular contacts with Jens Palsberg at UCLA (Los Angeles, USA), Sebastian Hack at at Saarland University (Saarbrücken, Germany) and with Philip Brisk at EPFL (Lausanne, Suisse).
- Compsys is in contact with Francky Catthoor's team in Leuwen (Belgium), and with Ed Depreterre's team at Leiden University (the Netherlands). They participate to joint discussions in the workshop Map2MPSoC of the network of excellence Artist2.
- Alain Darte has fruitful relations with Rob Schreiber at HP Labs, with three joint patents and many publications. The last patent was accepted in 2008 [8].
- Compsys is in regular contact with Christine Eisenbeis, Albert Cohen and Sid-Ahmed Touati (Inria project Alchemy, Paris), with Steven Derrien and Patrice Quinton (Inria project Cairn, Rennes), with Alain Greiner (Asim, LIP6, Paris), and Frédéric Pétrot (TIMA, Grenoble).
- Compsys participates in the EmSoC research project, which is part of the new research clusters of the Rhône-Alpes region.
- Compsys, as some other Inria projects, is involved in the network of excellence HIPEAC (High-Performance Embedded Architecture and Compilation http://www.hipeac.net/). Compsys is also partner of the network of excellence Artist2 to keep an eye on the developments of MPSoC and disseminate past work on automatic parallelization.

9. Dissemination

9.1. Introduction

This section lists the various scientific and teaching activities of the team in 2008.

9.2. Conferences and Journals

- In 2008, Alain Darte was member of the program committees of CC'08 (International Conference on Compiler Construction) and PLDI'08 (ACM SIGPLAN Conference on Programming Language Design and Implementation). He will be part of the program committee for Scopes'09 and Euro-Par'09. He is member of the steering committee of the workshop series CPC (Compilers for Parallel Computing), which will take place, in January 2009, in Zurich. He is member of the editorial board of the international journal ACM Transactions on Embedded Computing Systems (ACM TECS).
- In 2008, Fabrice Rastello was member of the program committees of the conferences CASES'08 (ACM/IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems) and CGO'09 (ACM/IEEE Symposium on Code Generation and Optimization). He is the main organizer of a seminar on static single assignment (SSA) that will regroup in April 2009 more than 40 people (international audience) during 4 days (see http://www.prog.uni-saarland.de/ssasem/). He will also be the local chair of CGO'11 that will be held in Chamonix, France.
- Paul Feautrier is associate editor of Parallel Computing and the International Journal of Parallel Computing.

9.3. Teaching and Thesis Advising

- In 2007-2008, Fabrice Rastello and Alain Darte have been teaching a Master 2 course on advanced compilation.
- In 2008-2009, Paul Feautrier will teach a Master 2 course on "Automatic Parallelization", a Master 1 course on "Compilation", and will be responsible for the "Compilation" project at the L3 level.
- Alain Darte and Fabrice Rastello are thesis co-advisors of Florent Bouchez. Fabrice Rastello is thesis advisor of Benoit Boissinot. Alain Darte and Tanguy Risset are thesis co-advisors of Alexandru Plesco. Alain Darte is thesis advisor of Clément Quinson.

9.4. Teaching Responsibilities

- Alain Darte is the vice-president of the admission exam to ENS-Lyon, responsible for the "Computer Science" part. He was also the creator of the exam on mathematics and computer science, whose topic was this year on convex optimizations [2].

9.5. Animation

- Fabrice Rastello is member of the evaluation commission (CS) of LIP.
- Paul Feautrier is a member of the PhD committee of ENS-Lyon, of the hiring committees of ENS-Lyon.
- Alain Darte was member of the national evaluation commission (CE) of INRIA until Spring 2008.
- Alain Darte was member of the hiring committees of Inria for junior researchers (CR2 Rocquencourt) and senior researchers (DR2).
- Alain Darte is the scientific expert, with Alain Girault, in the Inria group in charge of coordinating the joint research efforts of Inria and STMicroelectronics. This work led to the signature, in November 2008, of a global agreement between STMicroelectronics and Inria for joint projects.

9.6. Defense Committees

- Alain Darte was reviewer for the HDR (professorial thesis) of Ivan Augé (Paris VI) entitled “High-level synthesis and integration of hardware/software systems”, in Paris, December 2008.

9.7. Workshops, Seminars, and Invited Talks

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

- Fabrice Rastello gave a tutorial on SSA-based Register Allocation during ESWeek’08 (Embedded System Week).

10. Bibliography

Year Publications

Articles in International Peer-Reviewed Journal

[1] P. GROSSE, Y. DURAND, P. FEAUTRIER. *Methods for Power Optimization in SoC-based Data Flow Systems*, in "TODAES", to appear, 2009.

Articles in Non Peer-Reviewed Journal

[2] A. DARTE. *Quelques propriétés mathématiques et algorithmiques des ensembles convexes. Énoncé et corrigé de l’épreuve de mathématiques et informatique, concours d’entrée aux ENS de Cachan, Lyon et Ulm, session 2008*, in "Revue de Mathématiques Spéciales", vol. 119, n^{o} 1, 2008.

International Peer-Reviewed Conference/Proceedings

[3] B. BOISSINOT, A. DARTE, B. DUPONT DE DINECHIN, C. GUILLON, F. RASTELLO. *Revisiting Out-of-SSA Translation for Correctness, Efficiency, and Speed*, in "International Symposium on Code Generation and Optimization (CGO’09)", IEEE Computer Society Press, March 2009.

[4] B. BOISSINOT, S. HACK, D. GRUND, B. DUPONT DE DINECHIN, F. RASTELLO. *Fast Liveness Checking for SSA-Form Programs*, in "Sixth Annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO’08), Boston, MA, USA", Best paper award, ACM, 2008, p. 35–44, http://doi.acm.org/10.1145/1356058.1356064.

[5] F. BOUCHEZ, A. DARTE, F. RASTELLO. *Advanced Conservative and Optimistic Register Coalescing*, in "International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES’08), Atlanta, GA, USA", ACM, 2008, p. 147–156, http://doi.acm.org/10.1145/1450095.1450119.

Workshops without Proceedings

[6] A. PLESCO, T. RISSET. *Coupling Loop Transformations and High-Level Synthesis*, in "SYMPosium en Architectures nouvelles de machines (SYMPA’08)", February 2008.

References in notes

[7] P. BOULET, P. FEAUTRIER. *Scanning Polyhedra without DO loops*, in "PACT’98", October 1998.

[8] A. DARTE, R. SCHREIBER. *System and Method of Optimizing Memory Usage with Data Lifetimes*, April 2008, US patent number 7363459.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.

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

[19] 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.

[20] D. WILDE. *A Library for Doing Polyhedral Operations*, Technical report, n^{o} 785, Irisa, Rennes, France, 1993, http://hal.inria.fr/inria-00074515.