INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Compsys

Compilation and Embedded Computing Systems

Rhône-Alpes

THEME 1A

Activity Report

2003

Table of contents

1. Team

The Compsys team has been created in January 2002 as part of Laboratoire de l'Informatique du Parallélisme (Lip), UMR CNRS–ENS-*Lyon–Inria 5668), and is located at* ENS-*Lyon. Compsys is also a pre-project of Inria and hopes to become a full project in 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.*

Project leader
Tanguy Risset [CR Inria]

Administrative Assistant
Isabelle Antunes [T CNRS, part-time]

Permanent Members
Alain Darte [CR CNRS]
Paul Feautrier [Pr ENS-Lyon]
Fabrice Rastello [CR Inria]

External Collaborators
Antoine Fraboulet [MC Insa-Lyon]
Anne Mignotte [Pr Insa-Lyon]

PhD Student
Antoine Scherrer [BDI CNRS and ST-Microelectronics]

2. Overall Objectives

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

An embedded computer is a digital system, part of a larger system (appliances like phones, TV sets, washing machines, game platforms, or larger systems like radars and sonars), which is not directly accessible to the user. In particular, this computer is not programmable in the usual way. Its program, if it exists, has been loaded as part of the fabrication 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).

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 forth research theme is centered on the development of this 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 category 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 performances, power consumption, die size. Our project is related to code optimization (like in the Inria project A3/ Alchemy), and to high-level architectural synthesis (like in the Inria project R2D2). It belongs to the more general theme of silicon compilation, which is today one of the challenges that we have to meet if Europe is to become "the leader for system integration on silicon chips" [1].

Recent decisions of the French government clearly show the emergence of the research field "*tools for the conception of embedded systems*". The Ministry Call for Proposals RNTL 2003 explicitly cite embedded software as a priority [2], "to design embedded, critical, and/or real time software for objects and systems". Rapid prototyping is also mentioned. 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, ST-Microelectronics, Matra, Thalès, and others.
- Since embedded systems have an increasing complexity, new problems are emerging: computer aided design, a shorter time-to-market, a better reliability, modular design, and component reuse.

In 2001, the second working group of the RNTL [3] wrote in its report "Real time, embedded systems, and codesign", that there were not enough responses to the year 2000 Call for Proposals. This fact clearly indicates a lack of research actors on this particular subject.

More recently, several tools for high-level synthesis have appeared. These tools are mostly based on C or C++ (SystemC [4], VCC, and others). The support for parallelism in these tools is minimal, but academic projects are more concerned: Flex [5] and Raw [6] at the MIT, Piperench [7] at the Carnegie-Mellon University, PiCo [8]rom the HP-Labs and at the Synfora [9] start-up, Compaan [10] 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 fabrication 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]From the description of the Medea+ program in notes from the Cisi (Comité interministériel pour la société de l'information): http://www.recherche.gouv.fr/recherche/politic/cisi/00fiche.htm#23
[2]http://www.industrie.gouv.fr/rntl
[3]http://www.industrie.gouv.fr/rntl/AAP2001/groupe2_1.htm
[4]http://www.systemc.org/
[5]http://www.flex-compiler.lcs.mit.edu/
[6]http://www.cag.lcs.mit.edu/raw/
[7]http://www.ece.cmu.edu/research/piperench/
[8]f
[9]http://www.synfora.com/
[10]http://embedded.eecs.berkeley.edu/compaan/

The recent creation of the "Architecture and Compilation" pluridisciplinary research initiative [11] by CNRS (including a "Compilation for embedded systems" subfield) is a clear evidence of the increasing interest in the French research community. In Europe, the work plan for years 2003-2004 explicitly quotes "distributed and embedded systems" [12] as needing funding in 2004, and expects the creation of an Excellence Network on systems-on-chip.

Computer-aided design of silicon systems is a wide field. The Compsys team members expertise is in regular computation parallelization and optimization. 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.

The Compsys team has almost doubled its size since it became an Inria pre-project in January 2002. The new members are Paul Feautrier (ENS-Lyon professor) and Fabrice Rastello (Inria research scientist). As a consequence, we had to slightly reshuffle our research subjects, without any modification to the central theme. Compsys will go on cooperating with Insa-Lyon (Anne Mignotte and Antoine Fraboulet are external collaborators). One of our priorities is the recruitment of several PhD students. The recruitment of Antoine Scherrer, whose co-advisors are Antoine Fraboulet and Tanguy Risset, is a first step in this direction.

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 generators of code generator. 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 super computers 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 to a few regularity constraints, one can obtain linear formulations for many of the constraints. 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 [21][20] are an outcome of this research and are ubiquitously used for handling nested loop programs (regular circuits synthesis, Kahn 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 fabrication, and the operation costs. For instance, minimizing the silicon area improves the yield and hence decreases the fabrication cost. Power consumption is an important factor for mobile systems. Computer scientists are used to a paradigm in which

[11] http://www.archi-compil.org/

[12] Journal officiel de la communauté européenne 29/08/2002: http://europa.eu.int/comm/research/fp6/index_en.html

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 level). 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 may be 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 ST-Microelectronics leads us to investigate the last solution, as implemented in the ST100 (DSP) and ST200 DSP Very Long Instruction Word processors (VLIW).

3.2.1. Optimization of Predicated 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 (i.e., ϕ functions), which are inserted when converting into SSA form (“Static Single Assignment” [32]), 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. 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 [35] generate many open problems. Our objective is to contribute to the understanding and the solution of these problems, the main tool being the SSA [33] 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 Kahn process network, (see Sect. 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 task 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 which respects dependences and meet 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 strength 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 ST-Microelectronics processors.

It seems to be difficult 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. The usual 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 Yaka software prototype (see Section 3.4.2), we are exploring a heuristics in which resource constraints are simulated by data constraints. This method is not completely satisfactory (the results may be far from the optimum) but we hope that the experience we have acquired in this way will help us to find a direct solution.

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 where 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 mains 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 fabrication cost. Similarly, if the power consumption is high, one has to include a fan with 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 caches. For these optimization, loop transformations are the most efficient tool.

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 DSP 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 resort 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. This is a generalization of the classical problem of register allocation. 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 the make-span or is related to the make-span. 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 specially 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 [23], and techniques for data allocation in scratch-pad memory. Nevertheless, the problem is still largely open. Some questions are: how to schedule a loop sequence (or even a Kahn 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 found both high-level applications (for source-to-source transforms) 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 adequation 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 Cadence, Antoine Fraboulet has the opportunity of evaluating these methods with the help of codesign tools like VCC [22]. Note also that Antoine Fraboulet has just started a collaboration with the R2D2 project, on loop compilation as a tool for the design of specialized hardware coprocessors. This will be a way to validate our theoretical models.

Alain Darte, who is cooperating on a regular basis with the PiCo project at HPLabs, has already proposed some solutions to the memory minimization problem. These ideas will be implemented in the PiCo compiler in order to find their strengths and weaknesses.

3.4. Compilation of Embedded Parallel Architectures

Participants: 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:

- he has to specify the architecture of his system; this must provide the required power, but no more than that;
- when this done, he 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 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. This operation may transform a linear program into a non-linear one, or a continuous program into an integer program. 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 [13] (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 create an interface toward tools for control-intensive applications.

Toward this goal, the team will uses the know-how that Tanguy Risset has acquired during his participation in the Cosi project (before 2001) and also the knowledge of some members of the Arénaire project (Lip). This

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

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 or Matlab or Alpha). We also hope to demonstrate an interface between our tool, which is oriented toward regular applications, and synchronous language compilers (Esterel, Syndex) which are more control oriented.

3.4.2. Scheduling Kahn Process Networks

Kahn process networks (KPN) were invented thirty years ago [27] as a notation for representing parallel programs. Such a network is made of processes which 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 semantic and to talk meaningfully of the equivalence of two implementations. As an added bonus, the circuit diagrams which are used by signal processing specialists can be translated on-the-fly into KPNs.

The problem with KPNs is that they have an asynchronous execution model, while VLIW processors and Asic are synchronous or partially synchronous. Thus, there is a need for a tool for synchronizing KPNs. This is best done by computing a schedule which has to satisfy data dependences within each process, a causality condition for each channel (a message cannot be received before being sent), and real time constraints. A prototype of such a scheduler, Yaka (Yet Another KPN Analyzer) is being developed. 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 which are similar to the 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. We expect to test these ideas in the Yaka prototype.

3.4.3. Modular Parallelization and Interfaces

The scheduling techniques of MMAlpha and Yaka 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 by Triolet and Irigoin at Ecole des Mines de Paris (especially in the framework of inter-procedural analysis), and in MMAlpha (definition of structured schedules). 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 that direction, but also to consider other solutions, like Networks on Chip and standard wrapping protocols like VCI from VSIA[14].

3.5. Federating Polyhedral Tools

Participants: 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.

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

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 Leyden, PiCo from the HPLabs, the DTSE methodology at Imec, etc.). Most of these are research projects, but the increased involvement of industry (Hewlett Packard, Phillips) is a favorable factor. Polyhedra are also used in test and certification projects (Verimag, Lande, Vertecs).

Two basic tools that have emerged from this period are Pip [19] and the Polylib [34]. They are currently the only available tools since maintenance has stopped on Omega. Their functionalities are parametric integer programming and handling 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

In the last two years, we have greatly increased the availability 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 (UPMC). These tools have been the core of new code generation tools [17] [31] which are 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 Leyden. 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, Antoine Fraboulet is working on the integration of his thesis results into the Suif platform, and aims at participating in the implementation of the compiler for the Dart reconfigurable platform in cooperation with the Inria project R2D2.
- 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. Paul Feautrier is working in this direction.

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.

5. Software

5.1. 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 Leyden. This tool is in use by many groups all over the world.

5.2. 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 cutsmethod) 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 optimization, code generation and others.

5.3. MMAlpha

Participant: Tanguy Risset.

Tanguy Risset is the main developer of MMAlpha since 1994 (http://www.irisa.fr/cosi/ALPHA/). This software has enabled several PhD thesis, and 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 is 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 ST-Microelectronics and has been a basis for Compsys participation to European and RNTL Calls for Proposals.

5.4. Participation in Dart Compiler

Participant: Antoine Fraboulet.

Antoine Fraboulet is participating in the design of the Dart compiler, whose target is reconfigurable architectures [15]. This tool is developed within the R2D2 project at Rennes and Lannion. This compiler is built over the SUIF1 infrastructure from Stanford University [16] for high level optimizations and loop transformation. The objective is to improve these parts by taking into account the specificities of the target architecture, as for instance dynamically created parallel data-paths.

5.5. Yaka

Participants: Paul Feautrier, François Thomasset.

Yaka is a Kahn process network scheduler. Its development has benefited from the help of François Thomasset (Inria-Rocquencourt). The main improvements this year were the automatization of message counting (via the

[15] http://lasti.enssat.fr/GroupeArchi/axes/axes.php?axe=2
[16] http://suif.stanford.edu/suif/suif1/

Polylib), and of the code generation module (using Cloog[17]. Yaka is still in development, and is not mature enough for open distribution.

5.6. FPGA Card

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

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

6. New Results

6.1. MMAlpha and Multidimensional Time

Participant: Tanguy Risset.

Results: One of the most appealing use of a high-level synthesis tool is design space exploration. In its present version, MMAlpha builds only simple architectures (the latency is a linear function of the size parameter). In this case, the architecture exploration is quite intuitive and can be left to the designer. When a non-linear or multi-dimensional schedule is needed, we need much more complex architectures (for instance, memory banks are needed for storing intermediate values). Design space exploration becomes more difficult, and the use of a tool similar to MMAlpha is necessary. We have defined a new design class, which is less constrained than systolic arrays, but for which the same synthesis principles can be used. This result has been presented at the ASAP 2003 conference [7]. Implementing the method in MMAlpha will allow *memory generation* beside register generation.

future prospects: Multi-dimensional scheduling in MMAlpha is only partially implemented. The usual systolic synthesis is based on affine schedules, and generates simple arrays, in which the memory elements are shift registers. Multi-dimensional schedules need real addressable memories and control signals. Hence, before tools can be constructed, the real challenge is to understand how to synthesize complex circuits including memories and implementing nested loops.

6.2. Generating SystemC from MMAlpha

Participants: Antoine Fraboulet, Tanguy Risset.

Results: As part of the participation of Compsys in the SocLib projecthttp://soclib.lip6.fr, the implementation of a SystemC generator for MMAlpha has begun. An internship student from Delhi Indian Institute of Technology (IIT), has defined the constructors that are needed to express the semantics of the AlpHard subset of Alpha, while obeying the SocLib design rules. The translation of the hardware design as given by MMAlpha is nearly operational. One still has to express the virtual component interface VCI as a SystemC template.

future prospects Thanks to our active cooperation with SocLib, we can easily relate to the IP community. We plan to integrate the design rules of SocLib into the designs that are produced by MMAlpha. The resulting IPs will use the VCI (*Virtual component interface*, a simple protocol which is becoming a standard for IP interfacing). The challenge here is to standardize our work on interfacing regular applications [18]. This work must be validated by designing a complete SoC and simulating it. This Soc must include processors and special-purpose hardware designed by MMAlpha. A better validation would be (if possible) an implementation on the WildCard FPGA platform we have bought.

6.3. Network on Chip

Participants: Antoine Fraboulet, Tanguy Risset, Antoine Scherrer.

[17]http://www.prism.uvsq.fr/~cedb/bastools/piplib.html

Recent progresses of the CMOS technology allow the integration of a complete parallel machine on a single chip. Connecting the various components of this machine is a challenge, and the most likely solution is the use of an on chip network. In cooperation with SocLib, Compsys has access to the ground-breaking work of Lip6 (the Spin network). With the help of a master student, we have studied the adequation of the usual network protocols (the network layer) to specific applications for systems on chip (mostly for multimedia applications). The performance of the system has been found to be very sensitive to the value of of some parameters like the size of the FIFO buffers. This work has been presented to SympAAA'03 [9] and will proceed on a BDI scholarship from CNRS and the suport of ST-Microelectronics.

6.4. Instruction-Shift Optimizations

Participant: Alain Darte.

In cooperation with one of our past PhD student,Guillaume Huard, we completed our work on the instruction-shift algorithms which were the subject of his thesis. Our initial aim was to build firm theoretical foundations for this transformation, especially with regard to parallelism detection. Other scholars proposed a polynomial algorithm, but there was a flaw in it, and we proved that the problem is NP-complete. We also considered the problem the problem of maximizing local references. There existed an NP-completeness result for this task, but there was an error in the proof.

We established the NP-completeness of the instruction-shift problem. This result was presented at STACS 2002 [13]. The problem is amenable to integer linear programming. This is important because its complexity is not what one would expect intuitively. Besides, it shows that some code transformations cannot be formulated as a scheduling problem.

Our expertise in instruction-shift transformations allowed us to look into an older problem from a new perspective. The problem is that of reducing the memory footprint of a sequence of loops, which is important for embedded systems. The transformation consist in contracting arrays into scalars when this is possible. This problem had exact solutions and no complexity estimate. Again we proved, using non-trivial methods, that this problem is NP-complete. We also designed an integer programming method for its solution; this result was presented to ASAP 2002 [14] and has been selected for a special issue of the Journal of VLSI Signal Processing [2]. The paper includes other NP-completeness results, including a correct proof for the maximization of local references, a result about array alignment in HPF, and a problem in the scheduling of hypermedia documents. This illustrate the fact that a good model of some problem may be used in apparently unrelated domains.

6.5. Multi-Partitionning and HPF

Participant: Alain Darte.

During a visit to Rice University (October 2000 to January 2001) we developed in cooperation with John Mellor-Crumney et. al. a new strategy for data distribution: multi-partitioning. For scientific applications that apply "waves" of computation to multi-dimensional data, this technique guarantees perfect load balancing. This technique has been implemented into the dHPF compiler from Rice. Up to now, this technique could be applied in 3 dimensions only if the number of processors was a perfect square (in higher dimensions d, to a number of processors of the form n^{d-1} for some integer n). We lifted this constraint thanks to our theory of modular allocations and gave a method for computing an optimal multi-partitioning in the general case. The design of this algorithm was the subject of the mathematics/computer science competitive exam for admission to Ecoles Normales Supérieures in 2002.

It is interesting to note that the techniques we have developed, notably the use of Hermite normal forms, make explicit links between systolic array partitioning [12], modulo allocation (see Sect. 6.6) and Latin squares, mathematical objects that are more familiar to economists than to computer scientists These results where presented at IPDPS 2002 [11], and obtained one of the best paper awards. An extended version is published as [1].

6.6. Memory Reuse

Participants: Alain Darte, Gilles Villard [project Arénaire].

Results: 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 (without using a costly cache). An example is image processing, for which we might want to store only a few lines and not the entire frame.

Reusing array memory is a recent development. The first study was by Lefebvre and Feautrier [28] for loop nests and by Francky Catthoor's team [25] in Imec, then by Quilleré and Rajopadhye [30] for recurrence equations. Usually, the relation from subscript to addresses is linear. Lefebvre and Feautrier consider modulo allocations, while scholars at Imec first linearize arrays then apply a modulo function. Quilleré and Rajopadhye use projections. In cooperation with Rob Schreiber (HP-Labs) and Gilles Villard (Lip, Arénaire project) we have reviewed all past proposals to solvethis problem. We found that all of them were using some form of modulo allocation, without clearly saying so. We developed the necessary mathematical formalism, using the concept of the critical lattice, and we found that all preceding methods were heuristics for finding a critical lattice. We were able to quantify the qualities of these techniques, finding cases where the results are good, and others in which they can be arbitrarily far from the optimum. We proved that, in practice, one can construct approximations which do not differ from the optimum more than by a multiplicative constant which depends only on the dimension of the problem. We also have shown that the problem is connected to basis reduction methods (LLL), and with the successive minima of a lattice. Lastly, we showed that the optimum can be found by a clever exhaustive enumeration. These results have been presented to CASES 2003 [6].

future prospects: In this preliminary work, we have identified the basic mathematical tools we need to solve the memory reuse problem. However, many questions are still unanswered. Can the heuristics of [6] be improved? Can one find an upper bound on the size of memory as a function of the longest lifetime? as a function of the maximum number of live variables (which is a lower bound for the memory size)? Which is, in practice, the best method? How to handle cases in which the interference set is not a convex polyhedron? All these questions concern the case where a schedule is known. But the real problem is to find a schedule which meets performance constraints in the minimum amount of memory. Our understanding of the fixed schedule case is a step in this direction.

Answering these questions would 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. We intend to build a code that search for the critical lattice on top of our present tools, Pip and Polylib. In practice, heuristics for this problem are already in use in tools like PiCo. We hope these researches will give a new interest to our cooperation with Synfora.

6.7. Converting the SSA Form to Assembly Code

Participant: Fabrice Rastello.

Results: 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, one generates a lot of useless copies; a coalescing phase is needed to eliminate them.

In cooperation with François de Ferrière and Christophe Guillon from the MCDT team at ST-Microelectronics, we found a solution for the optimal removal of ϕ functions. Our solution is an extension of the Leung and George algorithm [29] which minimizes the number of copy instructions. This is not strictly equivalent to finding the best translation of the SSA form into machine language. However, they introduce an interesting technique, pinning, for representing renaming constraints.

Our idea was to use this pinning technique for constraining the renaming of ϕ functions arguments in order to simplify coalescing. This formulation is not strictly equivalent to the initial problem, 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 ST-Microelectronics. Comparison with other approaches gave interesting results, which we explained by comparing to several hand-coded examples.

All these results are presented in an internal research report [10] and will be presented at the international conference IEEE CGO 2004 [8].

future prospect: The problem of SSA removal may be modeled as a classical problem of register coalescing as found in the register allocation phase. There, one does "conservative" coalescing, which means that it does not change the chromatic number of the interference graph. For aggressive coalescing, we had to prove new NP-completeness results.

On the other hand, our study of aggressive coalescing and its interaction with Chaitin coloring techniques has shown that we were not solving the real problem (except in the case where the number of physical registers is unlimited). This lead us to consider a more general register allocation problem, still keeping in mind that the source code we are translating is in SSA form. This is specially true in our collaboration with MCDT at ST-Microelectronics. The goal is thus to remove the SSA ϕ function, without changing the register pressure, and by using the minimum number of copy instructions.

These results are still incomplete and must be firmed up. Nevertheless, we have introduced new concepts which will allow us to design new heuristics for fixed register allocation, both in the general case and for the ST220 DSP . The next step will be to experiment with these new heuristics. LAO2, an optimizer for assembly code which is under development at ST-Microelectronics, seems to be the ideal vehicle for these experiments.

6.8. Scheduling Kahn Process Networks

Participant: Paul Feautrier.

Results: A schedule for a Kahn process Network (KPN) has to meet the following constraints:

- the classical data dependences within each process and
- messages dependences (a message cannot be received before being sent),
- size constraints, which bound the number of messages which have been sent but not received.

The last two type of constraints can be expressed by virtual numbering of the messages. We have given explicit formulas for message counting. These formulas include sums, for which closed expressions can be found using the theory of Ehrhardt polynomials as implemented in the Polylib. This counting code has been implemented in Yaka with the help of François Thomasset from Inria-Rocquencourt.

In image processing, especially for digital TV, these counting functions are linear, and scheduling can proceed using classical methods. In some cases, the counting functions are low degree polynomials, for instance when painting triangular shapes. In that case, we can resort to a heuristics method, in which the triangular loop is enlarged to a rectangular one, a schedule is computed, and the code is corrected to avoid superfluous messages and computations.

future prospects The problem here is how to handle the message constraints when the counting functions are polynomial. As said before, there is a heuristics method to handle this situation, but it entails a loss of processing power. On the other hand, it seems that in this case the schedule must be polynomial. This is linked to the problem of multidimensional time (see Sect. 6.1). We are looking for more direct solutions to the problem. A first attempt to use the test-point method (W. Weisspfenning, Passau) did not give significant results.

6.9. Locality Improvement

Participants: Paul Feautrier, Cédric Bastoul.

Results: The problem here is to optimize locality of memory references for reducing the traffic between the cache and the main memory. Our first step has been to reformulate the problem, using asymptotic estimates of the traffic instead of the exact formula's, which are too complex to be tractable. This can be done by emulating the way a scratch-pad memory is used. The program is divided in chunks. Before starting the execution of a chunk, the necessary data are copied in the scratch pad memory. In this way, all misses are concentrated at the beginning of a chunk, and counting them is easy. We can handle in this way temporal locality optimization, spatial locality optimization, self reuse and group reuse, while insuring that the data dependences are alway satisfied. These techniques are implemented in the Chunky optimizer and the Cloog code generator. Chunky is probably the only locality optimizer that tolerates arbitrary dependences in the source code.

future prospects: Chunky handles currently spatial and temporal locality, and self reuse as well as group reuse. We have not attempted to modify array layouts as this is a non-local transformation that may benefits some parts of the code, but degrades the performance of other parts.

There is, however, an important technique that is not handled by the present code: tiling. Intuitively, tiling the iteration space seems just another way of building chunks. Nevertheless, we have not yet found a seamless way of unifying chunks and tiles, and this is the most important question we have to answer now.

Our software for code generation, Cloog, has reached maturity and is widely used. However, it still needs experimenting; in code generation, there are many degree of freedom which can be exploited for better efficiency. For instance, we discovered experimentally that two loops at the same level can sometime be arbitrarily displaced with respect to each other. Separating them completely is best for a parallel machine. However, coalescing them to get the maximum overlap is best for VLIW-like machines. It gives speedups in excess of 2 on the Itanium, while the improvement is marginal on a Pentium.

In the long term, the design of chunking was guided by the way a local or scratch-pad memory should be used. While the basic techniques are now well understood, a lot of questions still need answer. What is the best programming interface for a scratch-pad memory? How to compute the layout of arrays fragments in the scratch-pad? How does one rewrite array accesses? What is the interplay between scratch-pad memories and parallelism (one should remember that the Cray II had a scratch-pad memory which was useless because the designers had not considered this point)?

Positive answers to these question will increase the usefulness of scratch-pad memories in SoC; their use will be an important step toward solving the problems of predictability for real-time systems.

6.10. Optimization for the Instruction Cache

Participant: Fabrice Rastello.

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 ST-Microelectronics, the instruction cache is direct mapped. Let L be the size of the cache. Line i of the cache can hold only instructions whose addresses are equal to i modulo L.

When a program starts executing a block which 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 (call miss), or after a conflict (conflict miss). There is a conflict when two functions share the same cache lines; each of them remove the other from the cache when their execution is 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 "C-coloring", "SHIP-BUILDING", and "edge deletion" problems.

[EXP] Minimizing the size of the code. This is equivalent to a traveling salesman problem (building an Hamiltonian circuit) on a very special graph.

Classicaly [24], the problem is solved in two steps: **COL** then **EXP**. With Florent Bouchez (a third year student of ENS-Lyon), we have solved **EXP** for a given solution of **COL**. This work is part of our cooperation with ST-Microelectronics. We have shown that the problem can be solved in polynomial time. We have also proposed a heuristics with an $O(n)$ guarantee. This allows us to integrate the cost of code expansion in the first phase, and to build a coloring which is sensitive to code expansion. We also proved the NP-completeness of **COL**, and also that it is not $(1+\epsilon)$-approximable by reduction to MAX-3-SAT. Other problems related to this first phase have been considered, so far without success.

These optimizations have been implemented in the loader of the ST220 processor. Experimentation on several representative benchmarks shows that code expansion is significantly reduced without any increase in the number of cache misses. A technical report collecting these results is in preparation.

7. Contracts and Grants with Industry

7.1. The MCDT Team of ST-Microelectronics

Participants: Alain Darte, Fabrice Rastello.

Key words: *assembly level optimizations*.

The ProCD (Programmable Consumer Devices) contract is funded by STSI convention. Its objective is the design of programs for multimedia signal processing on a VLIW architecture. Compsys contribution is to give exact solutions to combinatorial optimization problems which arise when compiling programs for VLIW processors (see Sect. 3.2). These methods are then integrated into industrial strength compilers for the ST220. We have worked on the following problems:

- Software pipeline synthesis using modulo scheduling, including register allocation.
- SSA removal.
- Code placement for instruction cache optimization.

7.2. Other Contracts with ST-Microelectronics

Participant: Tanguy Risset.

Key words: *special purpose circuit synthesis*.

There is another STSI contract between Compsys and ST-Microelectronics (Crolles plant) on experimentation with MMAlpha for synthesis of ST-Microelectronics special purpose circuits. Its duration is 8 months. Due to administrative problems, the funding is not yet available, but the work has nevertheless begun (overhauling the Alpha-VHDL translator). The contract is to be renewed next year.

8. Other Grants and Activities

8.1. European community

Participants: Alain Darte, Paul Feautrier, Tanguy Risset.

- **Network of excellence** Compsys is involved in two “expressions of interest” for the 6th PCRD of the European Union: HIPEACS and EuroSoc. HIPEACS has successfully met the first round of evaluation.
- **STREP Project** Compsys is one of the partners for the Spotlight STREP project proposal on rapid prototyping of embedded computing system performances in the field of signal processing and multimedia applications.

8.2. RNTL

Participants: Paul Feautrier, Tanguy Risset.

Compsys is a partner in two RNTL submissions. The first one, on extensions to the KPN formalism, has been rejected. The second one, which is currently being evaluated, deals with a connection between MMAlpha and the B method.

8.3. Soclib: « A Modelisation & 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.

8.4. CNRS Multidisciplinary Research Networks

Participants: Alain Darte, Paul Feautrier.

Compsys participates in the following RTPs: “Architecture and Compilation”(http://polytope.u-strasbg.fr/RTP/) – Alain Darte and Paul Feautrier are members of its steering committee –, and “SOC” (http://www.comelec.enst.fr/rtp_soc/).

8.5. CNRS Collaboration with the University of Illinois at Urbana-Champaign

Participants: Paul Feautrier, Alain Darte.

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

8.6. 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.

8.7. 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 Franky Catthoor’s team in Leuwen (Belgium) and with Ed Depreterre’s team at Leiden University (the Netherlands).
- Alain Darte has fruitful relations with the PiCo team, notably with Rob Schreiber and Bob Rau (HP-Labs, Palo-Alto, USA) – see the two patents [16][15] – and with the Synfora start-up (Mountain View, CA, USA) whose manager is Vinod Kathail.
- 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 A3/Alchemy), with François Charost and Patrice Quinton (Inria project R2D2), and with Alain Greiner and Fréderic Pétrot (Asim, LIP6).
- Compsys participates in work meetings with LETI (CEA Grenoble), preparatory to a cooperation on power minimization.

9. Dissemination

9.1. Conferences and Journals

- Alain Darte was the program chair of ASAP 2003, with Lothar Thiele and Joe Cavallero, and for track 4 (Compilers for High-Performance) of Europar-2002, with Martin Griebl.
- Alain Darte was a member of the program committee the conference CASES 2003 (ACM Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems).
- Alain Darte was a member of the steering committee of CPC 2003 and CPC 2004 (Compilers for Parallel Computer). He has recently been co-opted into the editorial board of the international review ACM TECS (Transactions on Embedded Computing System).
- Paul Feautrier is Associate Editor of reviews Parallel Computing and Int. J. of Parallel Computing.
- Tanguy Risset is a member of the editorial board of the review Integration: the VLSI Journal.

9.2. Post-Graduate Teaching

- In 2003-2004, Alain Darte is sharing this course with Yves Robert (Remap).
- Paul Feautrier is thesis advisor for Cédric Bastoul (UPMC-Paris VI), Christophe Alias (UVSQ, Denis Barthou is co-advisor) and Yosr Slama (Faculté des Sciences de Tunis, Zaher Mahjoub is co-advisor).

9.3. Other Teaching and Responsibilities

- Alain Darte is in charge of the "Computer Science" division of the entrance exam to ENS-Lyon.
- Paul Feautrier teaches the following subjects for first and second year students:
  - A guided tour of Unix (MIM1).
  - Operational Research (MIM2).
  - Compilation project (MIM2).
- Paul Feautrier is the coordinator of the "Architecture and Compiler" track of the new Master of ENS-Lyon.
- In 2003-2004, Tanguy Risset is teaching the Compilation course to second year students at ENS-Lyon (MIM2).

9.4. Animation

- Alain Darte is an elected member of the hiring committee of ENS-Lyon, for the Computer Science section.
- Alain Darte is a member, since 2000, of the PhD committee of AFIT (Association Française de l'Informatique Théorique) which gives awards to outstanding PhD dissertations.
- Alain Darte and Paul Feautrier were members of the steering committee of RTP "Compilation and Architecture".
- Paul Feautrier is a member of the Governing Board and of the PhD committee of ENS-Lyon.
- Paul Feautrier is a member of the Audit Committee of LSIIT (UMR 7005, Lois Pasteur University, Srasbourg).
- Tanguy Risset is in charge of the Polylib mailing-list. This list includes most of the actors on the polyhedral models.

9.5. Defense Committee

- Tanguy Risset is a member of the defense committee for Anne-Claire Guillou (December 19, 2003, IRISA).
- Alain Darte was a reviewer of Sid-Ahmed-Ali Touati PhD dissertation “On the influence of instruction level parallelism on register pressure”. (June 2003, advisor: Christine Eisenbeis). He also was a member of the defense committee for Youcef Bouchebaba (“Data Transfer Organization for Signal Processing: Tiling, Loop Fusion and Array Reallocation”, advisor François Irigoin).
- Paul Feautrier is reviewer for the HDR (Professorial Thesis) of Frank Delaplace, to be defended November 28, 2003) at the University of Evry Val-d’Essone.

9.6. Workshops, Seminars, and Invited Talks

- Alain Darte gave the following seminars:

  LIP6 (Paris), February 2003. Frédéric Pétrot and Alain Greiner’s team. MMAlpha and PiCo synthesis methods.

  HP-Labs (Palo Alto, USA), October 27, 2003. Memory reuse by modulo allocation, also at Synfora (Mountain View, USA), October 28, 2003.

- Tanguy Risset gave the following seminars:

  Lip6 (Paris), October 2003: Translation of Alpha to the CABA (cycle accurate, bit accurate) model of SocLib.

  Leti (Grenoble), November 2003 : Using MMAlpha for the synthesis of CEA architectures.

  – Tanguy Risset has presented the following paper: “Hardware Synthesis for Multi-Dimensional Time” at ASAP 2003.

- Alain Darte attended the following conferences:

  CPC 2003 Amsterdam, the Netherlands, from January 8 January 10, 2003: Tenth Workshop on Compilers for Parallel Computers.

  Dagstuhl Germany, from February 9 to February 15: “Emerging Technologies: Can Optimization Technology meet their Demands?”, T. Conte, C.Eisenbeis, and M-L. Soffa, organizers. Talk: “PiCo, MMAlpha, etc. Towards the Compilation of Hardware Accelerators”.

  Samos 2003 Greece, from July 21 to July 23, 2003. “Systems, Architectures, Modeling, and Simulation”, Shuvra Bhattacharyya, Stamatis Vassiliadis, and Ed F. Deprettere, organizers. Talk: “Lattice-Based Memory Allocation”.

- Paul Feautrier will give a tutorial “Automatic Memory Allocation” to JP’03 (Journées du Parallélisme 2003) at Faculté des Sciences de Tunis.

10. Bibliography

Articles in referred journals and book chapters

[1] D. CHAVARRÍA-MIRANDA, A. DARTE, R. FOWLER, J. MELLOR-CRUMMEY. *Generalized Multipartitioning of Multi-dimensional Arrays for Parallelizing Line-Sweep Computations.* in « Journal of Parallel and Distributed Computing », 2003, Numéro spécial des versions étendues des meilleurs articles d'IPDPS'02, à paraître.

[2] A. DARTE, G. HUARD. *New Complexity Results on Array Contraction and Related Problems.* in « Journal of VLSI Signal Processing », 2003, Numéro spécial des versions étendues des meilleurs articles d'ASAP'02, à paraître.

[3] S. DERRIEN, A. C. GUILLOU, P. QUINTON, T. RISSET, C. WAGNER. *Automatic Synthesis of Efficient Interfaces for Compiled Regular Architectures.* Marcel Dekker, 2003, chapter 7, pages 127-150.

[4] F. RASTELLO, A. RAO, S. PANDE. *Optimal Task Scheduling to Minimize Inter-Tile Latencies.* in « Parallel Computing », number 2, volume 29, February, 2003, pages 209-239.

[5] F. DUPONT DE DINECHIN, M. MANJUNATHAIAH, T. RISSET, M. SPIVEY. *Design of Highly Parallel Architectures with Alpha and Handel.* Kluwer, 2003.

Publications in Conferences and Workshops

[6] A. DARTE, R. SCHREIBER, G. VILLARD. *Lattice-Based Memory Allocation.* in « 6th ACM International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES 2003) », October, 2003.

[7] A. C. GUILLOU, P. QUINTON, T. RISSET. *Hardware Synthesis for Multi-Dimensionnal Time.* in « IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2003) », The Hague, The Netherlands, June, 2003.

[8] F. RASTELLO, F. DE FERRIÈRE, C. GUILLON. *Optimizing the Translation Out-of-SSA with Renaming Constraints.* in « 2004 International Symposium on Code Generation and Optimization with Special Emphasis on Feedback-Directed and Runtime Optimization (CGO 2004) », March, 2004, to appear.

[9] A. SCHERRER, A. FRABOULET. *Étude de la couche transport des réseaux sur puce.* in « Symposium en Architecture et Adéquation Algorithme Architecture (SympAAA) », La Colle sur Loup, France, October, 2003.

Internal Reports

[10] F. RASTELLO, F. DE FERRIÈRE, C. GUILLON. *Optimizing the Translation Out-of-SSA with Renaming Constraints.* Technical report, number RR2003-35, LIP, ENS Lyon, France, June, 2003.

Bibliography in notes

[11] D. CHAVARRÍA-MIRANDA, A. DARTE, R. FOWLER, J. MELLOR-CRUMMEY. *Generalized Multipartitioning*

for Multi-Dimensional Arrays. in « 16th International Parallel and Distributed Processing Symposium (IPDPS 2002) », IEEE Computer Society Press, Fort Lauderdale, Florida, April, 2002, Best paper award.

[12] A. DARTE. *Regular Partitioning for Synthesizing Fixed-Size Systolic Arrays.* in « INTEGRATION, The VLSI Journal », volume 12, December, 1991, pages 293-304.

[13] A. DARTE, G. HUARD. *Complexity of Multi-Dimensional Loop Alignment.* in « 19th International Symposium on Theoretical Aspects of Computer Science (STACS 2002) », March, 2002.

[14] A. DARTE, G. HUARD. *New Results on Array Contraction.* in « 13th International Conference on Application-specific Systems, Architectures and Processors (ASAP 2002) », IEEE Computer Society Press, July, 2002.

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

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

[17] 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.

[18] S. DERRIEN, A. C. GUILLOU, P. QUINTON, T. RISSET, C. WAGNER. *Automatic Synthesis of Efficient Interfaces for Compiled Regular Architectures.* Marcel Dekker, 2002, chapter 7, pages 127-150, à paraître.

[19] P. FEAUTRIER. *Parametric Integer Programming.* in « RAIRO Recherche Opérationnelle », volume 22, September, 1988, pages 243–268.

[20] P. FEAUTRIER. *Some Efficient Solutions to the Affine Scheduling Problem, Part II, Multidimensional Time.* in « International Journal of Parallel Programming », number 6, volume 21, December, 1992.

[21] P. FEAUTRIER. *Some Efficient Solutions to the Affine Scheduling Problem, Part I, One Dimensional Time.* in « International Journal of Parallel Programming », number 5, volume 21, October, 1992, pages 313-348.

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

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

[24] N. GLOY, T. BLACKWELL, M. D. SMITH, B. CALDER. *Procedure Placement Using Temporal Ordering Information.* in « Proceedings of the 30th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-30) », IEEE Computer Society, pages 303–313, December, 1997.

[25] E. D. GREEF, F. CATTHOOR, H. D. MAN. *Memory Size Reduction Through Storage Order Optimization for Embedded Parallel Multimedia Applications.* in « Parallel Computing », volume 23, 1997, pages 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'94 », N. HOLLAND, editor, pages 471-475, 1974.

[28] V. LEFEBVRE, P. FEAUTRIER. *Automatic Storage Management for Parallel Programs.* in « Parallel Computing », volume 24, 1998, pages 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) », pages 204–214, 1999.

[30] F. QUILLERÉ, S. RAJOPADHYE. *Optimizing Memory Usage in the Polyhedral Model.* in « ACM Transactions on Programming Languages and Systems », number 5, volume 22, 2000, pages 773-815.

[31] F. QUILLERÉ, S. RAJOPADHYE, D. WILDE. *Generation of Efficient Nested Loops from Polyhedra.* in « International Journal of Parallel Programming », number 5, volume 28, 2000, pages 469–498.

[32] V. SREEDHAR, R. JU, D. GILLIES, V. SANTHANAM. *Translating Out of Static Single Assignment Form.* in « Static Analysis Symposium, Italy », pages 194 – 204, 1999.

[33] 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.

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

[35] 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, pages 373-384, 2001.