INRIA, Evaluation of Theme Architecture and Compiling

Project-team Compsys

Evaluation: March 21-22, 2012

Project-team title: Compsys

Scientific leader: Alain Darte

Research center: Inria Rhône-Alpes

Common project-team with: LIP, ENS-Lyon, CNRS, UCBL

1 Team members

Team members (Previous evaluation: April 25-26, 2007)

	Insa	Inria	CNRS	ENS-Lyon	**Total**
DR[1] / Professors	Risset		Darte	Feautrier	**3**
CR[2] / Assistant Professor	Fraboulet	Rastello			**2**
Permanent Engineers[3]					
Temporary Engineer[4]					
PhD Students			1	4	**5**
Post-Doc.		3			**3**
Total	**2**	**4**	**2**	**5**	**13**
External Collaborators					
Visitors ($>$ 1 month)					

[1] *"Senior Research Scientist (Directeur de Recherche)"*

[2] *"Junior Research Scientist (Chargé de Recherche)"*

[3] *"Civil servant (CNRS, INRIA, ...)"*

[4] *"Associated with a contract (Ingénieur Expert or Ingénieur Associé)"*

Team members (Evaluation: March 21-22, 2012)

	Misc.	INRIA	CNRS	ENS-Lyon	**Total**
DR / Professors			Darte	Feautrier	**2**
CR / Assistant Professor		Alias, Rastello			**2**
Permanent Engineer					
Temporary Engineer					
PhD Students		Colombet		Iooss	**2**
Post-Doc.					
Total		**3**	**1**	**2**	**6**
External Collaborators	Gonnord	Plesco			**2**
Visitors ($>$ 1 month)					

+ Laetitia Lecot: administrative assistant.

Changes in staff (since 2007)

DR / Professors CR / Assistant Professors	Misc.	INRIA	CNRS	ENS-Lyon	total
Arrival		1			1
Leaving	2				2

Comments: As was announced at the end of Compsys I (period 2004-2007), Antoine Fraboulet and Tanguy Risset, who moved at Insa-Lyon, left the team, so as to explore research areas closer to the activities at Insa-Lyon (namely sensors and software radio). Compsys II (2007-2012) thus started with 3 permanent researchers (A. Darte, P. Feautrier, F. Rastello). In January 2009, Christophe Alias integrated Compsys as an Inria research scientist (after a post-doc in Compsys, then another post-doc in the USA). Paul Feautrier, who had the possibility to retire, obtained an Emeritus status to continue in Compsys II and will continue, if the project is renewed, in Compsys III (2012-2016). Laure Gonnord, after a post-doctoral year in Compsys in 2008-2009, is now assistant professor in Lille but part-time in Compsys as external collaborator. Compsys remains a small team, with still the same difficulties at hiring good collaborators in program analysis, in compilation, and even harder in high-level synthesis.

Current composition of the project-team: March 2012

Permanent members

- Christophe Alias, Inria junior research scientist.
- Alain Darte, CNRS senior research scientist.
- Paul Feautrier, ENS-Lyon professor.
- Fabrice Rastello, Inria junior research scientist.

External collaborators

- Laure Gonnord, Lille University assistant professor.
- Alexandru Plesco, Inria engineer transfer and innovation, Zettice start-up.

PhD student

- Quentin Colombet, Mediacom project.
- Guillaume Iooss, joint PhD with Colorado State University (S. Rajopadhye).

Current position of former project-team members (including PhD students) during the 2007-2012 period:

Post-doctoral fellows

Christophe Alias post-doc in Compsys from February 2006 to December 2007, now in Compsys as Inria junior research scientist.

Sebastian Hack post-doc in Compsys from December 2006 to December 2007, now full professor of computer science at Saarland University, Germany.

Ouassila Labbani post-doc in Compsys from December 2006 to September 2008, now assistant professor in University of Bourgogne, Dijon, France.

Laure Gonnord ATER (non-permanent research/teaching position) from September 2008 to June 2009, now assistant professor at Lille University, France.

Florian Brandner post-doc in Compsys from December 2009 to October 2011, now post-doc in Danmarks Tekniske Universitet, Kongens Lyngby, Denmark.

PhD students

Nicolas Fournel PhD student (Ministry of Research grant) from 2004 to 2007 graduated in November 2007 ("Estimation and Optimization of Time and Energy Performances for the Design of Embedded Systems" [p4]), now assistant professor at Université Joseph Fourier, TimA, Grenoble.

Philippe Grosse PhD student (CEA-Leti grant) from 2004 to 2007, graduated in December 2007 ("Dynamic Task Management in an Integrated Micro-Architecture Targeting Low Power" [p5]), now research engineer at the Fraunhofer Institute, Erlangen, Germany.

Florent Bouchez PhD student (ENS-Lyon grant) from 2005 to 2008, graduated in May 2009 ("A Study of Spilling and Coalescing in Register Allocation as Two Separate Phases" [p2]), now research engineer at Kalray, Grenoble.

Clément Quinson PhD student (CNRS/STMicroelectronics grant) from 2005 to 2008. Did not graduate. Now engineer at Lumeneo, Ecquevilly, France.

Alexandru Plesco PhD student (Ministry of Research grant) from 2006 to 2010, graduated in September 2010 ("Program Transformations and Memory Architecture Optimizations for High-Level Synthesis of Hardware Accelerators" [p6]), now Inria ITI (engineer transfer & innovation), Zettice start-up incubation.

Benoit Boissinot PhD student (ENS-Lyon grant) from 2006 to 2010, graduated in September 2010 ("Towards an SSA-based Compiler Back-end: Some Interesting Properties of SSA and Its Extensions" [p1]), now engineer at Google, Zurich.

Engineers

Quentin Colombet Engineer within Minalogic Sceptre contract from 2007 to 2009, now PhD student in Compsys.

Last INRIA enlistments

- Christophe Alias was hired at CR2 junior research assistant in 2008 and took his position in January 2009, after he finished his post-doc at Ohio State University (P. Sadayappan). He was promoted CR1 at the end of 2011.

2 Work progress

2.1 Keywords

Embedded systems, DSP, VLIW, FPGA, hardware accelerators, compilation, code & memory optimization, program analysis, high-level synthesis, parallelism, scheduling, polyhedra, graphs, regular computations.

2.2 Context and overall goal of the project

Before its creation, all members of Compsys have been working, more or less, in the field of automatic parallelization and high-level program transformations. Paul Feautrier was the initiator of the polytope model for program transformations in the 90s and, before coming to Lyon, started to be more interested in programming models and optimizations for embedded applications, in particular through collaborations with Philips. Alain Darte worked on mathematical tools and algorithmic issues for parallelism extraction in programs. He became interested in the automatic generation of hardware accelerators, thanks to his stay at HP Labs in the PiCo project in Spring 2001. Antoine Fraboulet did a PhD on code and memory optimizations for embedded applications. Fabrice Rastello did a PhD on tiling transformations for parallel machines, then was hired by STMicroelectronics where he worked on assembly code optimizations for embedded processors [1]. Tanguy Risset worked for a long time on the synthesis of systolic arrays, being the main architect of the HLS tool MMAlpha. Finally, Christophe Alias, who joined Compsys with a permanent position in 2009, brought his expertise on source-to-source program analysis and optimizations as well as software development.

At this time – end of the 90s – most researchers in France working on high-performance computing (automatic parallelization, languages, operating systems, networks) moved to grid computing. On the contrary, we all thought that applications, industrial needs, and research problems were more important in the design of embedded platforms. Also, we were convinced that our expertise on high-level code transformations could be more useful in this field. We fully shared and still share the vision of compilation and architecture given by Bob Rau and his colleagues (IEEE Computer, sept. 2002):

> *"Engineering disciplines tend to go through fairly predictable phases: ad hoc, formal and rigorous, and automation. When the discipline is in its infancy and designers do not yet fully understand its potential problems and solutions, a rich diversity of poorly understood design techniques tends to flourish. As understanding grows, designers sacrifice the flexibility of wild and woolly design for more stylized and restrictive methodologies that have underpinnings in formalism and rigorous theory. Once the formalism and theory mature, the designers can automate the design process. This life cycle has played itself out in disciplines as diverse as PC board and chip layout and routing, machine language parsing, and logic synthesis. We believe that the computer architecture discipline is ready to enter the automation phase. Although the gratification of inventing brave new architectures will always tempt us, for the most part the focus will shift to the automatic and speedy design of highly customized computer systems using well-understood architecture and compiler technologies."*

With this view in mind, we were convinced of two complementary facts:

- The mathematical tools developed in the past for manipulating programs in automatic parallelization were lacking in high-level synthesis and embedded computing optimizations. Even more, they started to be rediscovered frequently under less mature forms. But they also needed to be extended to become more robust, more general, and to address new challenges. Similarly, back-end code optimizations needed to be revisited in the light of embedded processors features and objectives.
- Before being able to really use these techniques in HLS and embedded program optimizations, we needed to learn from the application, the electrical engineering, and the embedded architecture sides. We did in Compsys I – with work on traffic generators, SoC simulation, power issues [c33, c34, j11] – but failed to hire students

[1] This stay at STMicroelectronics is also one of the reasons why back-end code optimizations became a major research direction of Compsys, while the initial goal was more focused on high-level synthesis.

or researchers with strong electrical engineering background. With the departure of T. Risset and A. Fraboulet, who were the main SoC experts in Compsys, this pushed us to a stronger focus again on compiler optimizations and "fundamental" research.

Compsys specificity remains to tackle combinatorial optimization problems (graphs, linear programming, polyhedra) arising from actual compilation problems (register allocation, cache optimization, memory allocation, scheduling, consumption, generation of software/hardware interfaces, etc.) and to validate these developments in compiler tools. To address relevant problems and to have more impact, we believe our research efforts should be combined with strong industrial collaborations. Our work is at the frontier between languages and architectures, trying to identify the concepts and techniques that make the automation, from codes to machines, possible by compilation techniques.

2.3 Objectives for the evaluation period

Compsys I (the initial proposal) had four research directions, centered on compilation methods for embedded applications, both for software and accelerators design:

- code optimization for specific processors (mainly DSP and VLIW processors);
- platform-independent loop transformations (including memory optimization);
- silicon compilation and hardware/software codesign;
- development of polyhedral (but not only) optimization tools.

These research activities were supported by a marked investment in solid mathematical and algorithmic studies, with the aim of constructing operational software tools, not just theoretical results. Hence the 4th research theme, centered on the development of these tools. In Compsys II, these four objectives were recentered into the following 3 objectives:

- back-end code optimization for both aggressive and just-in-time (JIT) compilation;
- program analysis and transformations for high-level synthesis (HLS);
- development of polyhedral tools.

2.4 Objective 1: aggressive and JIT back-end code optimizations

2.4.1 Staff

Permanent researchers Alain Darte, Fabrice Rastello.
PhD students Benoit Boissinot, Florent Bouchez, Quentin Colombet.
Post-docs Florian Brandner, Sebastian Hack.

2.4.2 Project-team positioning

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 executable code is loaded in permanent memory and the compilation time to obtain it is not so significant. In particular, for embedded systems, where code size and energy consumption usually have a critical impact on the cost and the quality of the final product, the application is cross-compiled, i.e., compiled on a powerful platform distinct from the target processor. JIT compilation, on the other hand, corresponds to compiling bytecode on demand on the target processor. The code can be uploaded or sold separately on a flash memory. Compilation is performed at load time or even dynamically during execution. The heuristics, constrained by time and limited resources, cannot be too aggressive: they must be fast enough.

In this context, our goal was to contribute to the understanding of combinatorial problems that arise in compilation for embedded processors (e.g., in opcode selection, SSA conversion, register allocation, code placement in the instruction cache) to derive

both aggressive heuristics and JIT techniques. A first specificity of our work is that we always aim at adding a theoretical value on the problems we address (using graph theory, NP-completeness), even for problems that can appear "old" (such as register allocation). The second specificity is that, thanks to the collaboration with STMicroelectronics (and more recently with Kalray), we can implement and test our techniques directly within an industrial compiler. After clarifying, debunking, understanding the key issues that make the addressed problem hard, we first develop potentially-costly solutions (e.g., using integer linear programing) for aggressive compilation. This process allows us to confront the theory with the practice and provides a basis for designing and evaluating JIT solutions.

In Compsys II, all our activities were centered on the development of SSA-based code optimizations. Static single assignment (SSA) is an intermediate code representation or code property where each scalar variable is defined, textually, only once. It is becoming more and more popular in retargetable compilers as it leads to simpler analysis and optimization algorithms, easier to design, debug, and maintain (a feature increasingly important in compilers), and is more suitable to JIT techniques, without compromising performances too much. In Compsys I, we were the first to point out that the interference graph of variables in SSA is chordal and to advocate, based on this property, a decoupled register allocator that first spills (assignment to memory, optimizing load/stores), then colors (allocation to registers with coalescing to reduce register-to-register copies), and finally repairs (lowering using available instructions, that possibly inserts shuffle code and basic blocks). Compsys II was devoted to a deeper understanding of such a register allocation strategy and, more generally, of properties of SSA (and related intermediate representations) linked to dominance, out-of-SSA conversion, liveness analysis, etc.

2.4.3 Scientific achievements

Going out of SSA In SSA, multiplexers (called ϕ functions) are used to merge values at a "join" point in the control flow graph. To generate machine code, ϕ functions have to be replaced, at the end of the process, by register-to-register copy instructions on control flow edges. Naive methods for destructing SSA, when correct, generate many useless copies (live-range splitting), but also relies on the ability of disambiguating indirect jumps for splitting edges. We addressed three issues: correctness, code quality (elimination of useless copies), algorithm efficiency (speed and memory footprint). Our method, best paper at CGO'09 [c8], separates the issues of correctness and optimization, which makes it conceptually simpler and more robust than previous approaches that were often based on "patches". This correctness issue was, for a long time, a slowing factor to the development of SSA (e.g., bugs in GCC and Jikes). Also, by exploiting SSA properties (in particular with a liveness *check* algorithm, see hereafter), our algorithm outperforms the speed of the best algorithm so far (Sreedhar) by 2x and reduces the memory footprint by 10x.

Liveness analysis in SSA One important source of error of prior out-of-SSA implementations is a bad understanding of the liveness of ϕ-related variables. We proposed at CGO'08 [c9] (also best paper) a method to check the liveness of a variable at a given program point. Our method is specialized to SSA and survives all program transformations other than changes of the control-flow graph structure. As a bonus, it is less memory consuming and, depending on the client, usually faster. The SSA properties we identified for this liveness check allowed us to revisit the problem of computing liveness sets, too. By exploiting the dominance property of (strict) SSA form and the concept of loop-nesting forest, we designed a two-phase data-flow algorithm. Compared to traditional iterative data-flow approaches, which perform updates until a fixed point is reached, our algorithm, presented at APLAS'11 [c7], is twice faster on average than the fastest algorithm (Cooper).

Structure of interferences in SSA & SSI As previously mentioned, one of the important properties of SSA is that graph coloring under SSA is polynomial because the corresponding interference graph is chordal. What first attracted our curiosity to SSI (static single information, a variant of SSA with bi-directional properties) was the claim that the interference graph in SSI is an interval graph. Our debunking paper [o5, j1] clarifies a number of mistakes on SSI and provides a proof (much harder than the initial proof, which was completely wrong) for this interval graph property. We also revisited SSI for its theoretical ability to perform both forward and backward sparse data flow analysis. This work, still under review, aims at organizing the zoo of existing program representations (SSA, SSI, e-SSA, SSU, etc.) that exploit live-range splitting (e.g., with ϕ and σ functions) to enforce a static single information property (i.e., valid on a whole live-range).

Spilling The fact that the interference graph in SSA is chordal enables the design of a decoupled allocator: the spilling phase that stores variables to memory to lower the register pressure can be done *before* the coloring phase that assigns the other variables to registers. This decoupling opened the door for new spilling strategies. We first made an exhaustive study on how SSA impacts the complexity of "spill everywhere" (i.e., the whole live-range of a spilled variable is in memory). Contrarily to our initial hopes, most problems remain NP-complete [c12]. However, the fact they are polynomial for a fixed number of registers suggested spill-everywhere heuristics that incrementally solve, in "polynomial" time, the allocation problem with few registers, then "stack" the solutions. We applied this principle for split compilation [c26] (with Alchemy team): an ahead-of-time stacking algorithm drives, through portable bytecode annotations, the decisions of a light online JIT algorithm that adapts the allocation to the right target. We are also currently designing a purely JIT "stacking" solution. In parallel, to better understand spilling in its generality (not just spill everywhere), we developed an integer linear programming formulation, more accurate and expressive than previous approaches, that exploits the decoupling between spilling and coalescing (CASES'11 [c21]). The experimental comparison, in the STMicroelectronics compiler, of various heuristics to this "optimal" solution draws, among others, the following conclusions: a) significant savings can still be obtained in terms of static spill costs, cache misses, and dynamic instruction counts; b) rematerialization is extremely important and SSA can pay off here; c) SSA complicates the formulation of optimal spilling, because of memory coalescing of interfering variables; c) micro-architectural features are significant and thus should be accounted for in the model (but it is never the case). This deep study is still the first step for designing new aggressive and JIT spilling strategies.

Coalescing The effectiveness of the decoupled approach depends on the ability to cope efficiently, during the coloring phase (coalescing), with the shuffle code (register-to-register copies, edge splitting) introduced by the repairing phase (ϕ-functions replacement, register constraints handling, etc.). Our first results (best paper at CGO'07 [c11]) were devoted to the complexity of coalescing problems (aggressive, conservative, incremental, and optimistic), discussing also on the structure of the interference graph (arbitrary, chordal, or k-colorable in a greedy fashion). This study was extremely useful to point out where the complexity comes from. In [c13] – a more practical paper – we improved the de-coalescing phase of an optimistic approach and designed an advanced incremental conservative approach, which, contradicting the common belief, turned out to be simple to implement and close to optimal. A good context to stress its performances was to apply it in the context of register aliasing, which we address in [c46] with the introduction of a "semi-elementary form", generalizing the "puzzle" approach of Pereira and Palsberg. With the democratization of SIMD instruction set architectures, handling register aliasing will become critical

even though current compilers are not mature enough to fully expose it. The last step towards the design of a practical SSA-based "coloring" algorithm – generalization of linear scan – was the efficient handling of register constraints [c20]. Thanks to the concept of post-repairing of violated register constraint, the spirit of decoupled register allocation can be kept, i.e., with spilling and coloring as simple as possible, without tricky patches to handle special cases of the instruction set architecture. The cost of repairing (as for register-to-register copies used to get rid of ϕ-functions) is encapsulated in the coloring objective function, through affinities and dislikes (negative weight affinities). We applied this method to develop both a graph-based approach (extension of conservative coalescing to handle register dislikes) and a scan-based decoupled approach (new tree scan coalescer).

Parallel copies All decoupled approaches, and even out-of-SSA translation, rely on shuffle code represented as parallel copies (involving registers but also memory slots) in basic blocks and also, implicitly, on critical edges, i.e., edges that flow from a block with several successors to a block with several predecessors. To optimize such copies, we proposed a new back-end optimization called parallel copy motion [c10]. The technique is to move copy instructions in a register-allocated code from a program point, possibly an edge, to another. In contrast with traditional schedulers that must preserve data dependences, our copy motion can permute register assignments so that a copy can "traverse" all instructions of a basic block, except those with conflicting register constraints. As the interplay of this optimization with the scheduler is high, we pushed this idea further to perform code motion (of copies) on register-allocated data dependence graphs. This technique [c15] can eliminate useless copies and reorder instructions, while preserving a valid register assignment. It is a step forward the design of register-pressure aware schedulers.

2.4.4 Collaborations

Our work on back-end code optimizations were done in tight collaboration with colleagues from STMicroelectronics (Benoît Dupont de Dinechin, Christophe Guillon, François de Ferrière). After we proposed SSA-based decoupled register allocation, Sebastian Hack (Karlsruhe) got the same idea independently and became a close collaborator. Then, Jens Palsberg, Fernando Pereira, Philip Brisk (UCLA) started to explore this area too and became partners on many aspects (students, tutorials, PhD defenses, joint papers, etc.). The work on split compilation, one of the topics of the Mediacom project, was the outcome of a collaboration with the Alchemy Inria team (Albert Cohen, Boubacar Diouf).

2.4.5 External support

- SCEPTRE project, collaboration with STMicroelectronics (compilation team).
- MEDIACOM project, collaboration with STMicroelectronics (compilation team) and Alchemy Inria team (A. Cohen).
- FAPEMIG-INRIA (Brazil-France funding mechanism), collaboration with Federal University of Minas Gerais, Brazil (F. Pereira).
- PROCOPE (Germany-France funding mechanism), collaboration with Saarland University (S. Hack).

2.4.6 Self assessment

Our activities with STMicroelectronics were a huge success for us, not only for the contracts we get. Being able to work within a complete industrial compiler made our study more relevant and more than just theoretical. Based, among others, on this success story, Inria and STMicroelectronics established a general agreement for joint research projects

(Alain Darte was part of the corresponding committee). Also, through this activity, Compsys succeeded to attract young researchers: two PhD students from ENS-Lyon (Florent Bouchez [p2] and Benoit Boissinot [p1]), two top-level post-docs (Sebastian Hack and Florian Brandner), one engineer from STMicroelectronics (Quentin Colombet) join Compsys for a PhD, a past student (Cédric Vincent) was hired by STMicroelectronics.

As for scientific results, the comments we got from other researchers were very encouraging. For example, Keith Cooper (specialist of register allocation since he advised Preston Briggs PhD thesis) concluded his analysis of the PhD manuscript of Florent Bouchez by "Taken together, the chapters of Florent Bouchez's Ph.D. thesis form the most complete exploration of the theory of register allocation that I have seen". We received three consecutive best paper awards at CGO, a conference with mainly practical contributions, which was also, for us, a sign that many researchers think that tightening theory and practice is fundamental. We were also asked to give tutorials (CASES'08 [o13], CGO'09 [o12], LCPC'09 [o7]) on the new view we proposed on register allocation. This recognition helped us organizing the very first international workshop on SSA (see hereafter).

This topic is not yet finished, but low-hanging fruits are rare. Beating 30 years of heuristics, even if we improved the theoretical understanding, is not obvious. In 2010-2011, Fabrice Rastello left Compsys for a sabbatical, renewing his centers of interest, all Nano2012 projects were cancelled, and manpower on back-end code optimizations has been reduced at STMicroelectronics (due to strategic options). For all these reasons, this activity will now slow down, unless Compsys can hire new researchers on this topic.

2.5 Objective 2: program analysis and transformations for HLS

2.5.1 Staff

Permanent researchers Christophe Alias, Alain Darte, Paul Feautrier.
PhD students Hadda Cherroun, Alexandru Plesco, Clément Quinson.
Post-docs Ouassila Labbani.

2.5.2 Project-team positioning

With the advent of parallelism in supercomputers, the bulk of research in code transformation resulted in (semi-)automatic parallelization, with many techniques based on the description and manipulation of nested loops with polyhedra. Embedded systems generated new problems in high-level code optimization, especially for loops, both for optimizing embedded applications and transforming programs for high-level synthesis (HLS) (where loop unrolling and basic block scheduling of the loop body have been, for a long time, the only loop optimizations). Everything that has to do with data storage is of prime importance as it impacts power consumption, performance, and chip area.

On the application side, multimedia applications often make intensive use of multi-dimensional arrays, in sequences of (nested) loops, which make them good targets for static program analysis. In practice, the applications are rewritten several times, by the compiler or developer, to go from a high-level algorithmic description down to an optimized and customized version. But for memory optimizations, the high-level description is where the largest gain can be obtained because global program analysis and transformations can be done: analyzing multimedia applications at the source level is thus important. On the architecture side, the hardware, in particular memories, can be customized. When designing/optimizing a programmable embedded system, adequate parameters can be selected for cache and scratch-pad memories to achieve the smallest cost for the right performance for a given application or set of applications. In HLS, memories (size, topology, connections between processing elements) can even be fully customized for a given application.

Embedded systems are thus good targets for memory optimizations. But powerful compile-time program and memory analysis are needed to (semi-)automatically generate a fully-customized and optimized circuit from a high-level C-like description. Also, new specification languages or compilation directives are needed to express communicating processes and their communication media: processes communicating through FIFOs or shared memories are a good target. Our objective in this topic was to adapt and extend high-level transformations, previously developed for automatic parallelization, to the context of HLS and embedded computing optimizations. Such techniques started to be rediscovered under various forms and we thought our previous expertise could be useful both for the dissemination of already-known techniques and the development of new ones.

2.5.3 Scientific achievements

Static memory allocation and reduction When designing hardware accelerators, one has to solve both scheduling (when is a computation done?) and memory allocation (where is the result stored?). This is important to exploit pipelines between functional units, or between external memories and hardware accelerators, and save memory space. An example is image processing, for which it is preferable to store only a few lines and not the entire frame, data being consumed quickly after being produced. Reducing memory size can be done by re-mapping each array so that it reuses its memory locations when they contain a dead value (intra-array reuse). In Compsys I, in a collaboration with HP Labs (Rob Schreiber, PiCo project), we showed that all previous approaches are particular cases of a general technique based on the construction of admissible lattices (see patent [o15]). A stand-alone tool was developed for generating such a lattice and a corresponding linear mapping with modulo (see the software tool Cl@k and Section 2.6.3).

After the theory, we developed the algorithms needed to implement these memory reuse strategies, i.e., the interface with programs and the required program analysis (analysis of the lifetimes of individual array elements). The resulting tool Bee was implemented thanks to the source-to-source transformer ROSE, developed by D. Quinlan (Livermore). The technique was demonstrated on benchmarks borrowed from IMEC, thanks to P. Clauss, S. Verdoolaege, and F. Balasa [o1, c1]. The combination Cl@k+Bee was the first implementation for array contraction based on modular mappings. As a side effect, this provides the most aggressive algorithm for converting arrays into scalars. This technique was also used in Chuba (see hereafter in this section) for defining the local memory required by its underlying "double-buffering"-like execution. Nevertheless, more work remains to be done for deriving faster and better heuristics, for speeding-up the program analysis, and for coupling scheduling, memory reuse, and memory size constraints.

First studies in high-level synthesis At the end of Compsys I and beginning of Compsys II, our preliminary activities on HLS focused on identifying how compiler techniques – back-end scheduling and high-level transformations – could be integrated in existing HLS tools. This research thus also required an important effort in analyzing these tools.

The thesis of Hadda Cherroun [p3] focused on scheduling iterations of loops, extracted from multi-dimensional techniques, taking into account resource constraints. The idea was a two-levels scheduling, in which the "fronts" generated by the first-level schedule are refined using combinatorial optimization methods. Several branch-and-bound algorithms, incorporating reweighting techniques similar in spirit to Johnson's algorithm, were designed [j2] as well as scheduling techniques using graph coloring [c19].

As part of the PhD work of Clément Quinson, we analyzed the instruction scheduling approach used in UGH, the HLS tool developed at LIP6. To guide the synthesis, the user constrains the scheduler with a pre-allocation (draft data path) of some scalar variables

to physical registers. To increase its freedom, the scheduler relaxes output dependences between different writes and adds anti dependences on the fly to preserve the program semantics. We proved that deciding if a deadlock will appear with this strategy is NP-complete, which implies a need for backtracking or register duplication [c24]. These issues are related to Bernstein's conditions [j7], the scheduling of register-allocated codes, and the phase-ordering problem (for register allocation & scheduling) in standard compilers.

Finally, in the PhD work of Clément Quinson and Alexandru Plesco, we studied the impact of high-level transformations, applied at source level before HLS tools (namely Catapult-C, Pico, and C2H). These studies [c27, c43], hard to "sell" by publications, were nevertheless the starting point on our more advanced work on communication optimizations for HLS (see hereafter in this section) and on the analysis of while loops (see Section 2.6.3), for termination and iterations counting.

Program analysis for Array-OL Array-OL is a formalism that combines the streaming paradigm with template programming. It was introduced to ease the implementation of data-intensive signal processing as found in sonar and radar software. Based on the work of Alain Demeure at Thales in the 1980s, it has been developed both by Thales' TRT and the University of Lille LIFL. Array-OL is not a programming language, but a development tool in which a design is specified by annotations on a graphic interface. In the context of the Martes ITEA project (see Section 3.3), Ouassila Labbani and Paul Feautrier developed a tool for the extraction of Array-OL specifications from legacy C code [c39]. The tool reused part of the Syntol scheduler for parsing and semantic analysis as well as new program analysis to make data movements explicit, as required in Array-OL [o16].

Program analysis and communication optimization for HLS Today, HLS tools are clearly becoming more mature for generating hardware accelerators with an optimized internal structure, thanks to efficient instruction scheduling techniques, resource sharing, and finite-state machines generation. However, interfacing them with the outside world, i.e., integrating the automatically-generated hardware accelerators within the complete design, with optimized communications, so that they achieve the best throughput, remains a very hard task, reserved to expert designers. In a previous work [j10] (at the end of Compsys I), we focused on how to feed non-programmable accelerators – systolic arrays generated by the HLS tool MMAlpha – with external data, thanks to a customized communication module. Following our preliminary study on source-to-source transformations for HLS, our goal in Compsys II was to improve the design of these interfaces, trying to consider the HLS tool as a back-end for more advanced front-end transformations.

Using the C2H HLS tool from Altera, which can synthesize hardware accelerators communicating to an external DDR-SDRAM memory, we showed that it is possible to restructure the application code, to generate adequate communication processes entirely in C, and to compile all of them with C2H, so that the resulting application makes full usage of the memory bandwidth [c3]. These transformations and optimizations combine, in an interleaved manner, techniques such as double buffering, array contraction, loop tiling, software pipelining, among others. We showed how to perform the required analysis and optimizations automatically using polyhedral techniques [c4, c5]. A unique feature is that we can pipeline tiles, exploit inter-tile data reuse, and space locality to improve DDR accesses, and even cope with approximations [o2, o3]. These techniques were incorporated in an automatic source-to-source transformation tool, called Chuba (see Section 3.2), which is the core of Alexandru Plesco PhD [p6]. This study shows that HLS tools can indeed be used as back-end optimizers for front-end optimizations, as in standard compilation where high-level transformations can be developed on top of assembly-code optimizers.

High-level synthesis with pipelined arithmetic In HLS, the target circuit must not only be efficient, but also produce quality results, thanks to specific arithmetic operators. Producing such operators is the specialty of the Arenaire Inria project, which develops FloPoCo, an open-source FPGA-specific generator that converts functional descriptions into pipelined floating-point arithmetic operators. These pipelined operators need a fine optimization of the data and control paths to deliver performances. As current HLS tools usually provide an abstraction that hides the back-end details, a purely source-to-source approach was not enough in this case. We developed an algorithm to generate, from a C program, an hardware accelerator that efficiently uses these pipelined operators, rescheduling the initial program execution to keep the operator's pipeline as busy as possible, while minimizing the memory accesses. This new schedule is then used to generate the VHDL code of finite state machines (FSM) controlling the data-flow through the arithmetic operator [c6]. We also addressed the problem of generating control FSMs of multiple parallel computing cores accelerating the same application [o4]. A startup, Zettice, is currently in incubation around the technologies developed in this section and the previous one.

2.5.4 Collaborations

Our research activities on HLS involved collaborations and informal contacts with the Inria project Cairn (S. Derrien), the Inria project Arénaire (F. Dupont de Dinechin, B. Pasca), STMicroelectronics (HLS team), Thales (through the Martes project), colleagues from the french HLS community (in particular F. Pétrot at TimA and A. Greiner at LIP6).

2.5.5 External support

- PhD support CNRS/STMicroelectronics.
- S2S4HLS project, with STMicro (HLS team) and Cairn Inria team (S. Derrien).

2.5.6 Self assessment

On the positive side, we think that we made slow but strong progress on program analysis and transformations for HLS. In particular, the automation of optimized communications with inter-tile reuse and automated "double-buffering"-like execution, which generalizes some work on scratch-pad memory optimizations and kernel offloading for GPUs, offers interesting perspectives, beyond HLS. But, before getting any fundamental results, such research requires a lot of efforts and time due to the interaction with HLS tools and FPGA platforms (tough developments and use).

On the negative side, except for Alexandru Plesco, we never succeeded to find students and hire researchers with a strong architecture/synthesis background and who can, at the same time, understand mathematics and computer science. In 2010 and 2011, all actors of the french HLS community tried to structure their efforts through two ANR proposals but, despite the coordinating and writing efforts, these proposals were rejected. It seems in retrospect that the HLS community has yet to find a clearer balance between new research and industrial development. The same is true for our unfruitful collaboration with the HLS team of STMicroelectronics. Finally, we also regret to have missed the opportunity to participate in the network of excellence ArtistDesign. We were affiliate members at its start (in 2007) but, at this time, we were more involved in back-end code optimizations with STMicroelectronics. It is only at the end of Compsys II that our activities started to be connected to the topics addressed in ArtistDesign. Also, the small size of Compsys unfortunately does not allow us to be everywhere.

2.6 Objective 3: development of polyhedral tools

2.6.1 Staff

Research staff Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord
PhD students Alexandru Plesco
Post-docs Fabrice Baray (in Compsys I).

2.6.2 Project-team positioning

Since the times of PIP and of the POLYLIB, Compsys has been active in the implementation of basic mathematical tools for program analysis and synthesis. PIP is still developed by Paul Feautrier and Cédric Bastoul, while the POLYLIB is now taken care of by the Inria Camus project, which introduced Ehrhart polynomials. These tools are still in use worldwide, but it is interesting to observe that they have been reimplemented many times with (sometimes slight) improvements: consider for instance the Parma Polylib, Sven Verdoolaege's ISL and barvinok libraries, or the Jollylib of Reservoir Labs. More recently, other groups also made a lot of efforts towards the democratization of the use of polyhedral techniques, in particular the Alchemy Inria project, with the development of Graphite in GCC, and Sadayappan's group in the USA, with the development of Pluto.

Compsys II has continued in its tradition, focusing on the introduction of new concepts and techniques to extend the polytope model, with a shift toward tools that may prepare the future of parallel computing. For instance, POCO and C2FSM are able to parse general programs, not just SCoPs (static control programs), while the efficient handling of Boolean affine formulas is a prerequisite for the construction of non-convex approximations. CL@K is the first step towards memory optimization in stream languages and may be useful in all kind of situations. Our work on CHUBA introduces new analysis related to the lifetimes of array elements and the possibility of handling approximations. Finally, our work on the analysis of while loops is both an extension of the polytope model itself (i.e., beyond SCoPs) and of its applications (interest in program termination and possibly WCET tools).

2.6.3 Scientific achievements

Critical and admissible lattices At the end of Compsys I, as a result of our formalization, in a common model, of all intra-array reuse techniques, we developed CL@K, a stand-alone combinatorial optimization tool 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 has minimal determinant.) It also computes the successive minima of such polytopes. CL@K completes the POLYLIB suite, enabling yet another kind of optimizations on polyhedra, with modulo operations. Its initial application was the automatic derivation, in a sequential program, of array mappings that enable memory reuse thanks to modulo operations. In Compsys II, we used it with no change for designing memory mappings for our "double-buffering"-style kernel offloading in CHUBA. Other applications are foreseen, which will require some extensions.

Polyhedral compilation framework Usually, implementing a polyhedral analysis is a long, difficult, and bug-prone task. The programmer must be familiar with many different libraries (POLYLIB/ISL, PIPLIB, Cloog to quote a few), each with its own format and API, which complicates the programming and lengthens the development time. There is a need for a common interface to these tools as well as for an higher level of abstraction. POCO provides such an interface and enables the manipulation of high-level objects (e.g. polyhedra, operations, schedules), rather than vectors and matrices. These features, which can

seem simple at first glance, allow the programmer to quickly prototype polyhedral tools. PoCo, developed by Christophe Alias, is the result of a systematic effort to restructure the code into reusable functions and objects along 6 years of development. The tools Bee, Rank, and Chuba were built thanks to PoCo's high level interface, which made their development more robust and stable.

Analysis of while loops: towards irregular programs The polyhedral model is inherently limited to programs with static control: only DO loops, no while loops, tests on loop counters only. The only data structures are scalars and arrays, and the index functions must be affine in the surrounding loop counters. Many computational kernels fail to fit in this model by a small margin: see for instance Gaussian elimination with pivoting, or the Singular Value Decomposition, which uses the `break` statement. There is a time-honored approach for handling irregular problems, which dates back to the pioneering work of Floyd and Manna. The program is first converted into an affine interpreted automaton – combination of a finite state machine (FSM) and a list of variables – whose behavior is a superset of the program behavior. A transition can be fired only if an affine constraint on the variables is satisfied, and its effect is an affine assignment of new values to the variables (or more generally an affine relation between input and output variables). One can compute *invariants* for each state, i.e., constraints on the variables that are satisfied whenever the control reaches this state. In general, the invariant cannot be computed exactly and must be over-approximated, using techniques from abstract interpretation.

In program termination, a *ranking function* is a function to a well-founded set that decreases at each transition. By establishing a strong link – exposed in the survey papers [c23, j4] – between the work of Karp, Miller, and Winograd on recurrence equations, the multi-dimensional scheduling techniques we invented in the 1990s for loops, and the concept of ranking functions, we have been able to bring to the program termination community many results developed for the detection of parallel loops. Our technique for generating multi-dimensional affine ranking functions subsumes several algorithms proposed by this community. In addition, coupled with counting methods in polytopes, it can provide upper bounds on the number of iterations, which can be of interest for the WCET (worst-case execution time) community. This work was presented at SAS'10 [c2]. A complete software suite was developed, which first uses c2fsm to convert the C source into an interpreted automaton. Aspic, developed by Laure Gonnord, is then responsible for computing invariants as polyhedral approximations. Finally, Rank builds a ranking (if any) using Pip and computes upper bounds on the number of iterations using the Ehrhart polynomial module of the Polylib. The first two stages of this tool chain (c2fsm and Aspic) were presented at TAPAS'10 [c29].

Simplification of Boolean affine formulas Up to now, in the polyhedral model, one has only considered conjunctions of affine inequalities, or, equivalently, convex polyhedra. The expressive power of these formulas is enough to handle simple abstractions like DO loop iteration domains or elementary dependences. However, as soon as one wants to handle more complex objects, like value-based dependences or conditionals in loop nests, one must consider unions of polyhedra or even formulas using the full range of Boolean operators, including negations and disjunctions. Algorithms on such formulas have a tendency of generating highly redundant formulas of exponentially increasing size. Paul Feautrier developed an algorithm and a tool for the elimination of redundancies [o18], which has already found a use in the simplification of FSMs for high-level synthesis.

Transitive closure of Boolean affine relations Since the seminal paper of Bill Pugh et al., it has been noticed that the transitive closure operator (a.k.a. Kleene star) is an important tool for program analysis. For instance, to analyze and summarize the effect of an inner loop in a set of nested loops, one needs to compute the transitive closure of the inner loop body. Since the transitive closure of a Boolean affine relation is not necessarily affine, one must resort to over- or under- approximations, depending on the context. All known algorithms, for instance by Bill Pugh, by François Irigoin, or by Sriram Sankaranarayanam, reduce to the construction and summation of the distance polyhedron of the given relation. Paul Feautrier has proposed a new algorithm, based on the decomposition of a preorder into an equivalence relation and a partial order, which reduces to Pugh and Irigoin algorithms in simple cases, but can be extended to more complex formulations. This work in progress has been presented at IMPACT'12 [c28].

2.6.4 Collaborations

The development of our software tools (excluding those related to back-end code optimizations and developed in STMicroelectronics compilers) is not supported by any formal collaboration. However, we have regular discussions with colleagues from the Inria teams Alchemy (A. Cohen, C. Bastoul, S. Verdoolaege), Cairn (S. Derrien), Camus (V. Loechner, P. Clauss), the PIPS group (F. Irigoin, B. Creusillet, R. Keryell), people at Reservoir Labs (B. Meister, N. Vasilache), the Pluto group (Sadayappan, Ramanujam, U. Bondhugula), the team of S. Rajopadhye, etc.

2.6.5 External support

No specific support.

2.6.6 Self assessment

We believe that our effort to convert abstract theorems and algorithms into practical software is worthwhile and on the whole successful. A crucial point is that while abstract complexity results are important, they should not deter us to attack NP-complete problems, which may have feasible solutions in practical cases. The recent success of SAT and SMT solvers, due to improved algorithms and processors, is a case in point.

In its developments, Compsys is using a wide choice of languages, including OCaml, C++, and Java. This is both an advantage and a drawback. When associating tools, it forces communication through files (or Unix pipes), at some cost in performance. On the other hand, having independent tools simplifies maintenance and evolution. Having our tools converge to a common language would be a major effort, which cannot be considered unless additional manpower is made available, possibly by INRIA.

All our tools are "free software", but Compsys has no fixed policy on distribution and licensing. This is a point that should be settled in the near future.

3 Knowledge dissemination

3.1 Publications

The following table summarizes all international conferences with a selection process, but with no distinction between top conferences and smaller workshops. See details in the bibliography section. We point out that publication in journals is usually not the target for us, unless we want to publish results with a more theoretical depth, with all details, and when speed of publication is not an issue. We therefore in general submit to conferences,

targeting top conferences first. However, recently, we experienced a surprising change in the review process: acceptance is much more random, with very arguable reviews, while reviews for journals are more balanced and even often less demanding.

	2007	2008	2009	2010	2011	2012	Total
PhD Thesis	3		1	2			6
Journal	3	1	1	1	1		7
Conference (*)	13	4	4	9	12	4	46
Book chapter	1				5		6
Book (edited)						1	1
Patent	1						1
Technical report	5		1		6		12
Deliverable	2	2	3	2	1		10

Major journals for our field include ACM TECS, TOPLAS & TODAES, IEEE Transactions on Computers, the International Journal of Parallel Programming (IJPP), the Journal of VLSI Signal Processing, operation research journals (e.g., RAIRO-OR):

- IEEE Transactions on Embedded Computing Systems: 1
- RAIRO Operations Research: 1
- Journal of VLSI Signal Processing: 1
- ACM Transactions on Design Automation of Electronic Systems: 1

Major conferences in the field are CGO, CASES, DATE, DAC, ASAP, SAS, CC, CODES, LCTES, plus some conferences on parallelism such as PACT, PPoPP, IPDPS.

- ASAP, International Conference on Application-Specific Systems, Architectures, and Processors: 4 (including one short paper).
- CGO, International Symposium on Code Generation and Optimization: 3 (and three consecutive best paper awards in 2007, 2008, 2009).
- LCTES, Languages, Compilers, and Tools for Embedded Systems: 3.
- CASES, International Conference on Compilers, Architecture, and Synthesis of Embedded Systems: 3.
- PPoPP, Symposium on Principles and Practice of Parallel Programming: 1 (short paper/poster).
- PACT, International Conf. on Parallel Architectures and Compilation Techniques: 1.
- SAS, Static Analysis Symposium: 2.
- CC, Compiler Construction: 1 (best paper award for S. Hack, post-doc in Compsys).

3.2 Software

We develop three kinds of software tools (see also our 2011 activity report).

- Compiler-like research tools that are used internally to implement, validate, and improve ideas presented in our papers. These tools are usually intended to be progressively extended (e.g., Syntol, PoCo, Chuba).
- Stand-alone software tools that solve particular polyhedral problems and whose goal is to contribute to the polyhedral community (e.g., Pip, Cl@k, Simplifiers).
- Developments in external tools (e.g., LAO, Open64, UGH) to incorporate specific algorithms designed by Compsys (e.g., register allocation, liveness analysis).

PIP Tool for parametric integer programming (`www.piplib.org`), developed by Paul Feautrier, then slightly improved in collaboration with Cédric Bastoul and Sven Verdoolaege. Freely available under the GPL and widely used (worldwide) in the polyhedral community.

Syntol Research tool developed by Paul Feautrier, Hadda Cherroun, Ouassila Labbani, for studying communicating regular processes (CRP) and their scheduling in a modular fashion. Not distributed. Used in the Martes project [o16, c39].

Cl@k Stand-alone tool, developed within Compsys I by Fabrice Baray and Alain Darte for computing an admissible lattice (with reduced determinant) for a 0-symmetric polytope. Used to derive array mappings (linear mappings plus modulo operations) that enable the reuse of array cells (kind of sliding windows). Available on demand (see also `http://www.ens-lyon.fr/LIP/COMPSYS/clak/`).

PoCo Polyhedral compilation framework, used by Bee, Chuba, and Rank, that provides many features to quickly prototype polyhedral analysis and optimizations. Front-end based on EDG (via Rose). Roughly 20000 lines of C++. Registration at APP ("agence de protection des programmes") in progress (since May 2011).

Bee Source-to-source optimizer for array contraction, with analysis of the lifetime of array elements and memory mapping based on Cl@k. Roughly 2500 lines of C++. Binary of Bee+Cl@k [c1] made available for the Cairn HLS toolbox Gecos, through the S2S4HLS project. APP registration in progress (May 2011).

Chuba Source-level optimizer that offloads a C kernel onto FPGA, with optimized communications to an external DDR memory [p6, c4]. Currently designed to be used as a front-end to the Altera HLS tool C2H. Roughly 1000 lines of C++. APP registration in progress (May 2011). Software at the heart of the Zettice start-up initiative.

C2fsm Extraction, from a C program, of an interpreted automaton. Not distributed yet. Used to interface C programs with the abstract interpretation tool Aspic [c29] (see details on Aspic at `http://laure.gonnord.org/pro/aspic`).

RanK Stand-alone tool to decide (when possible) the termination of an interpreted automaton. Connected to 2fsm and Aspic to handle C while loops and to give an upper bound on their number of iterations (kind of WCET) [c2]. Roughly 3000 lines of C++. See `http://www.ens-lyon.fr/LIP/COMPSYS/Tools/Ranking/`.

Simplifiers Stand-alone tool for the simplification of affine Boolean expressions [o18], in particular Quasts (quasi affine selection trees) extensively used in the polyhedral community. Not yet distributed. See also `http://www.ens-lyon.fr//LIP/COMPSYS/Tools/Simple/`.

Developments in LAO and Open64 All our aggressive and JIT code optimizations are implemented within the compiler toolchain Open64/LAO of STMicroelectronics (mainly in the research branch, some are then rewritten in the industrial branch). These algorithms concern SSA construction and destruction, liveness analysis, instruction cache optimizations, register allocation (coalescing, spilling, register constraints). This enables experimental studies, evaluation of algorithms, comparison of different approaches, and bug tracking as our techniques push the STMicroelectronics compiler beyond its limits.

MinIR MinIR (minimalist intermediate representation) is a new intermediate representation, designed to ease the interconnection of compilers, static analyzers, code generators, and other tools [c40]. In addition to its specification, generic core tools have been developed to offer a basic toolkit and to help the connection of client tools. See details at `http://www.assembla.com/spaces/minir-dev/wiki` and `https://compilation.ens-lyon.fr/`.

3.3 Industrial contracts and technology transfer

In 2004, we started a tight collaboration with the compilation team of STMicroelectronics (Christian Bertin, Benoît Dupont de Dinechin, Christophe Guillon, François de Ferrière). From 2006 to 2012, this joint research effort was funded through larger governmental contracts, Sceptre (2006-2009) and Mediacom (2009-2012), see also Section 4.

Sceptre project (2006-2009) Sceptre was funded by the "pôle de compétitivité" Minalogic (`http://www.minalogic.org/`). This project, led by STMicroelectronics, and with many partners mainly from Rhône-Alpes, aimed at the development of a toolkit to ease the implementation of multimedia algorithms and the generation of optimized codes for a multiprocessor reconfigurable platform. Our specific task was to work on combinatorial optimization problems coming from back-end optimization, in particular the removal of static single assignment (SSA), register allocation, and code placement for instruction cache optimization. This project was acknowledged in 2009 by the government as a great success and as the first Minalogic project that ended on time and smoothly.

Mediacom project (2009-2012) This contract started in September 2009 as part of the R&D funding mechanism Nano2012 and as the continuation of Sceptre. Mediacom focused on both aggressive optimizations and the application of the previously-developed techniques to just-in-time (JIT) compilation and implied four Inria teams: Alf, Alchemy, Arénaire, and Compsys. Unfortunately, due to a unilateral decision of the government, all fundings related to Nano2012 were cancelled, or at least frozen, in 2011 and 2012. Inria guaranteed the salary of PhD students and of some engineers/post-docs already in place, but all other salaries and the travelling budget were cut. Our activities continued but in a less ambitious format.

This long-term collaboration with the compilation team of STMicroelectronics was a real success, with a gain for both parties. This gives us access to real industrial compilation problems, to a set of representative benchmarks, to the ST assembly code optimizer in which we develop (LAO and Open64), and to an industrial expertise in compilers and processor architecture. Conversely, we help them develop new strategies, understand previously-published approaches that need accurate readings for a correct implementation, and our development activities contribute to debug their compiler.

In terms of scientific results, our joint efforts led to important contributions in instruction cache optimization, register allocation, and static single assignment (SSA). In particular, Compsys was the first group to push the use of SSA for register allocation and to completely deconstruct the classic view on register allocation. With our colleagues from STMicroelectronics, we are now well-identified internationally for this contribution. In addition to our results and publications (see also Section 2.4), this research created a lot of activity in seminars, tutorials [o13, o12, o7], organization of workshops (for example, we organized the first seminar on SSA in Autrans (`http://www.cdl.uni-saarland.de/ssasem/`) and we were involved in the organization of CGO'11), research proposals, hiring of young researchers (in both directions), PhDs [p2, p1], etc. This success also contributed to the signature of a R&D national agreement between Inria and STMicroelectronics (to which Alain Darte participated) and the activation of several other Nano2012 projects.

To support our second research axis (high-level loop transformations and high-level synthesis (HLS)), we established a second activity with STMicroelectronics, but with the HLS team (Pascal Urard, Roberto Guizzetti, Thierry Michel, Michel Favre). It was first supported by a CNRS/STMicroelectronics PhD funding (Clément Quinson), then as part of a second Nano2012 contract, S2S4HLS.

S2S4HLS project (2009-2011) S2S4HLS (source to source transformations for high-level synthesis) started in January 2009. The goal of this project, initiated by the Cairn Inria team, was the study and development of source-to-source program transformations, in particular loop transformations, that are worth applying on top of HLS tools. This includes restructuring transformations, program analysis, memory optimizations and array reshaping, etc. Our activities on the HLS tool UGH [c24], on the optimization of DDR communications with the HLS tool C2H [c3], and on the analysis of while loops [c2] arose in this context but we did not really succeed to find a good match between our activities and STMicroelectronics interests. Nevertheless, some of our tools (CL@K and BEE) were integrated to Cairn's toolbox. Finally, we were about to hire a post-doc on this topic when all Nano2012 projects were frozen. These successive difficulties pushed us to quit the project in Spring 2011.

Until 2008, Compsys was also involved in the Martes ITEA project (`http://www.martes-itea.org/public/news.php` focusing on a model-driven approach to real-time embedded systems development, using UML and SystemC.

Martes project (2006-2008) This project was completed in September 2008. The french partners of the project have focused their work on the interoperability of their respective tools using a common UML meta-model. A post-doc (Ouassila Labbani) was hired through Martes to focus on the interaction between Syntol (see Section 3.2) and the parallel design environment SPEAR of Thales Research [c39, o16]. A gateway, partially based on the Eclipse framework, was implemented and has been successfully demonstrated at several review meetings. The Martes project won a Silver Award at the 2008 ITEA Symposium.

Finally, to compensate the funding difficulties of Nano2012 projects and to prepare the future of the team, Compsys is now involved in a new industrial project led by Kalray (`http://www.kalray.eu/`).

ManyCoreLabs project (2012-2016) Kalray is a french start-up, partly arising from CEA and STMicroelectronics, whose activity is to develop new manycore processors for embedded computing. The ManyCoreLabs project, funded by the BGLE program ("briques génériques du logiciel embarqué", see `http://www.industrie.gouv.fr/fsn/logiciel-embarque`), is led by Kalray and involves many many partners, both academics and from industry (mainly potential customers for Kalray MPPA architecture). The role of Compsys in this project, in line with the objectives of Compsys III, is to explore compilation techniques for streaming-like languages for this platform. The kick-off meeting will be held in March 2012.

3.4 Teaching

No Compsys member has teaching duties (except Paul Feautrier until 2009). However, Compsys tries to be in charge of compilation courses at ENS-Lyon and/or UCBL.

"Compilation" Master 1, ENS-Lyon This 36 hours course, which presents all the basics of compilation (from parsing to code generation), was done by Paul Feautrier (2008, 2009), then by Christophe Alias (2010, 2011).

"Advanced code optimizations" Master 2, ENS-Lyon This 24 hours course covers advanced code optimizations in connection with Compsys activities: software pipelining, intermediate code representations, SSA, register allocation, polyhedral optimizations, high-level synthesis, etc. It was shared by Alain Darte (2007, 2009, 2010, 2011), Fabrice Rastello (2007, 2009), and Paul Feautrier (2008, 2011).

Here are some other activities directly linked to teaching and education:

- Paul Feautrier was in charge of the L3 "compilation project" in 2008-2009 and gave a M1 course on "operational research" in 2007-2008.
- Christophe Alias gave a L3-level "introduction to compilation" course at ENSI Bourges (2010, 2011) and a L2-level lab on "computer architecture" at UCBL in 2011. He belongs to the teaching council of the Computer Science Department of ENS-Lyon and organized two Winter Master School for students: "Beyond the PC. Application-specific systems: design and implementation" (in 2010) and "Verification and certification of software" (in 2012).
- Alain Darte was the vice-president of the admission exam to ENS-Lyon, responsible for computer science, from 2001 to 2010. He was also author of the 2008 exam [j3].

Other activities, such as participation to hiring committees, to PhD or habilitation (HDR) jurys, are to be found in Compsys annual reports.

3.5 General audience actions

None.

3.6 Visibility

Prizes and awards Compsys received 5 paper awards since 2007: at AICCSA'07 [c19], at CC'07 [c38] (work finalized by S. Hack while post-doc in Compsys), and three consecutive best paper awards at CGO (CGO'07, CGO'08, CGO'09), which is the best or one of the best conferences in compilation/code optimization. These three papers concerned the complexity of register coalescing [c11], a fast liveness check algorithm in SSA [c9], and an efficient technique to go out of SSA [c8].

Paul Feautrier received from the Euro-Par Steering Committee an award "in recognition of his outstanding contributions to parallel processing" (2009). He was also the special guest of a "Paul Feautrier evening" organized during CGO'11, with all his past and present international colleagues.

Editorial boards Compsys participates to the editorial boards of the following journals:

- *ACM Transactions on Embedded Computing Systems (ACM TECS)*: Alain Darte.
- *Parallel Computing*: Paul Feautrier.
- *International Journal of Parallel Programming*: Paul Feautrier.

Program committees Compsys participates to the following program committees:

ACCA *International workshop "analyze to compile, compile to analyze"*, Laure Gonnord (2011 – organizer).

CASES *ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, Alain Darte (2009, 2011), Fabrice Rastello (2008).

CC *International Conference on Compiler Construction*, Alain Darte (2008, 2011, 2012), Paul Feautrier (2009).

CGO *ACM/IEEE International Symposium on Code Generation and Optimization*, Fabrice Rastello (2009, 2011 – local chair).

CPC *International Workshop on Compilers for Parallel Computing*, Alain Darte (steering committee, 2013 – organizer).

DATE *International Conference on Design, Automation, and Test in Europe*, Alain Darte (2007, 2011, 2012).

EUROPAR *International Conference on Parallel Computing*, Alain Darte (2009).

IMPACT *International workshop on polyhedral compilation techniques*, Christophe Alias (steering committee, 2011 – organizer, 2012), Paul Feautrier (2012).

LCTES *ACM Conference on Languages, Compilers and Tools for Embedded Systems*, Alain Darte (2012).

NPC *IFIP International Conf. on Network and Parallel Computing*, Alain Darte (2007).

PARCO *International Conference on Parallel Computing*, Paul Feautrier (2011).

PLDI *ACM SIGPLAN Conference on Programming Language Design and Implementation*, Alain Darte (2008, 2009 – first "fun ideas and thoughts" (FIT) session).

SCOPES *International Workshop on Software and Compilers for Embedded Systems*, Alain Darte (2009, 2010).

Tutorials, keynote talks, contributions to books Following our research on SSA and register allocation, Fabrice Rastello, Florent Bouchez, and Alain Darte, in collaboration with Sebastian Hack, Fernando Pereira, Jens Palsberg, and Philip Brisk, organized three successive tutorials on "SSA-based register allocation" at CASES'08 [o13], CGO'09 [o12], and LCPC'09 [o7].

Alain Darte gave a keynote talk at MEMOCODE'10 on "understanding loops: the influence of the decomposition of Karp, Miller, and Winograd" [c23] and at IMPACT'11 on "approximations in the polyhedral model" [o14]. Paul Feautrier gave a keynote talk at LCPC'09 on "the polytope model, past, present, future" [o17].

Paul Feautrier has been an active participant in the elaboration of the **Encyclopedia of Parallel Programming**, (David Padua ed.), recently published by Springer [2]. He has been a member of the scientific committee and has contributed four entries, on array layout for parallel computing [j6], Bernstein's conditions [j7], dependences [j8], and the polyhedral model (with Christian Lengauer) [j9]. Alain Darte also contributed a chapter on the parallelism detection in loops [j4]. Following the workshop he organized in 2009 (see hereafter), Fabrice Rastello coordinated a book "SSA-based compiler design" [b1], to be published by Springer [3] and dedicated to static single assignment (SSA),

Organization of scientific events

SSA In 2009, Fabrice Rastello and Sebastian Hack, with the help of Compsys members, organized the very first international workshop entirely focused on static single assignment (SSA), although SSA was introduced in the late 80s. It regrouped 55 people during 4 days (see `http://www.prog.uni-saarland.de/ssasem/`), including personalities involved in the very first developments of SSA. This unique initiative gave rise to a book, coordinated by Fabrice Rastello, covering all aspects of SSA (semantics, analysis, optimizations, tools), that should be ready by the end of 2012.

[2] `http://www.springer.com/computer/swe/book/978-0-387-09765-7`

[3] `http://www.springer.com/engineering/circuits&systems/book/978-1-4419-6201-0`

French compilation days Until 2010, the french compiler community had no official national meetings. Fabrice Rastello, with the help of Laure Gonnord, decided to motivate the different french actors to meet regularly. All groups whose activities are related to compilation were contacted and the first "compilation day" was organized in September 2010 in Lyon. The next sessions took place in Aussois (2010), Dinard (2011), and Saint-Hippolyte (2011). This effort is a success: the community is now well identified and such an event occurs at least once a year (see `http://compilation.gforge.inria.fr/`).

CGO In 2011, for the very first time, CGO was organized outside the USA (see `http://www.cgo.org/cgo2011/`). Its organization involved members from the Alchemy and Compsys teams. Fabrice Rastello was responsible for the local organization in Chamonix, including the definition of new satellite workshops such as IMPACT'11 (co-organized by Christophe Alias), WIR'11 (co-organized by Florent Bouchez), ACCA'11 (co-organized by Laure Gonnord).

IMPACT Christophe Alias was the main organizer of IMPACT'11 (international workshop on polyhedral compilation techniques). This workshop was the very first international event on this topic, although it was introduced by Paul Feautrier in the late 80s. After its first very successful edition, IMPACT continued: IMPACT'12 was recently held as part of HIPEAC in Paris, IMPACT'13 will take place in Berlin.

Thematic quarter The Labex MILYON (`http://milyon.universite-lyon.fr/`) was created to fund the organization of 3-months period on specific topics in mathematics and computer science in Lyon. Compsys proposed to organize, in 2013, a thematic quarter centered on compilation, in connection with languages and architectures, with invited researchers and several events (CPC'13, HIPEAC meetings, summer schools, french compilation days, etc.). MILYON should fund around 75-100 Keuros.

4 External funding

For a description of the Martes European project, and the industrial contracts Sceptre, Mediacom, and S2S4HLS, see Section 3.3. The table provided Page 23 indicates the budget allocated to our project-team in Keuros, with two columns: the budget excluding salaries and the budget for salaries. Other salaries, not covered by these contracts are out of the budget of the team, and only the number of such scholarships is provided.

5 Objectives for the next four years

Compsys has always focused on the development of fundamental concepts or techniques whose applicability should go beyond a particular architectural or language trend. For instance, the core of our work on back-end optimizations was based on the mathematical properties of the SSA form. We also explored new techniques related to the polyhedral model (scheduling, memory mapping, simplification) as well as new applications (program termination, high-level synthesis). We will continue this type of research: we will try to push the theory beyond current knowledge, as independently as possible of technological trends (but aware of them of course) and we will develop small stand-alone tools – either as proofs of concepts or to be used as basic blocks in larger tools/compilers developed by others – and our own experimental prototypes. In particular, we will try to push polyhedral techniques beyond their present limits, using approximation techniques, and dynamic or runtime optimizations.

(Keuros)	2007		2008		2009		2010		2011		2012	
European projects												
ITEA MARTES	**4**	**39.6**[5]	**4**	**37.7**[5]								
Industrial contracts												
STMicro[1]	**9**				**4**							
Hewlett Packard[2]	**7**			**6**[8]								
SCEPTRE	**10**	**24.6**[6]	**10**	**39.5**[6]	**10**	**36.7**[6]						
S2S4HLS					**5**		**7**		7[3]	20.7[3]	7[3]	20.7[3]
MEDIACOM							**12**	**84.4**[6,7]	9[3]	**80.2**[6,7]	9[3]	**41.1**[6]
ManycoreLABS											**8**	**24.3**
Other funding												
Inria dotation	**15**		**15**		**15**		**10**		**12**		**12**	
ENS-Lyon dotation	**5**		**1**		**4**		**3**		**3**		**2**	
CNRS dotation	**3**		**1**		**1**		**5**		**2**		**1**	
Rhône-Alpes								**6**[4]				
Total	**53**	**64.2**	**31**	**83.2**	**39**	**36.7**	**37**	**90.4**	**17**	**80.2**	**23**	**65.4**
Scholarships (number of) between year-1 and year, excluding those supported by the above projects												
Post-doc		**2**[9,10]				**1**[11]						
PhD		**6**[4,8,12,13,14,15]		**4**[4,8,14,15]		**2**[4,15]		**2**[4,15]				**1**[16]
ITI Inria												**1**[4]

[1] *left-over of a direct contract with STMicroelectronics (Compsys I)*

[2] *left-over due to patents with HP in 2002* [3] *cancelled, due to Nano2012 problems*

[4] *Alexandru Plesco* [5] *Ouassila Labbani* [6] *Quentin Colombet* [7] *Florian Brandner*

[8] *Clément Quinson* [9] *Sebastian Hack* [10] *Christophe Alias* [11] *Laure Gonnord* [12] *Nicolas Fournel*

[13] *Philippe Grosse* [14] *Florent Bouchez* [15] *Benoit Boissinot* [16] *Guillaume Iooss*

The polyhedral model is neither a programming language nor an execution model; its status is rather that of a compiler intermediate representation (IR), albeit very different from usual IRs, like abstract syntax trees (AST) or control flow graphs (CFG). As such, it can be generated from several sequential high-level languages, like C and Fortran, or streaming languages like CRP (communicating regular processes, an extension of Kahn process networks), or equational languages like Alpha. While the structure of the model is the same in all three cases, it may enjoy different properties: for instance, the existence of a schedule is guaranteed for sequential programs, but it has to be checked in the other two cases. The import of the polyhedral model is that many questions relative to a program behavior and performances, and the applicability of many transformations, can be answered precisely and efficiently by applying well-known mathematical results to the model. The price to pay is that the expressive power of polyhedral programs is severely limited: they cannot handle either dynamic data structures or dynamic control.

Meanwhile, the evolution of the technology landscape has led to the introduction of massively parallel architectures at all levels of the performance spectrum, from embedded appliances to high-performance computers. Processor counts of a thousand up to a billion are now contemplated. Parallel applications, parallel compilers, and parallel languages must scale up to these figures. It is no longer possible to restrict oneself to the study of small regular kernels. Several research groups are attempting to trade predictability for expressiveness, in the form of parallel libraries whose behavior is completely data dependent and cannot be analyzed at compile time, see for instance the Galois system (Keshav Pingali) or the Concurrent Collections (Intel, Kathleen Knobe). One should note, however, that these groups are wondering how to detect and take advantage of regular program parts (SCoPs) in order to improve the performance of their systems.

The feeling at Compsys is that there is a continuum of approaches for parallel programming, with the polyhedral model at one of the extremities, and purely dynamical low-level

approaches at the other one. Most research teams start from one of these extremities and try to move, step by step, in the other direction. The work on parallel libraries is an attempt to alleviate the well-known difficulties of thread programming. The objective of Compsys will be to move in the opposite direction, i.e., to enlarge the applicability of the polyhedral model in a controlled and manageable way. But, instead of being driven by architecture issues or by languages features, we want to be driven by compiler issues, i.e., by what we know can be automated. We will also extend the polyhedral model itself, which has still many unsolved problems, mainly related to resources and memory constraints, especially in the context of hardware synthesis.

5.1 Inside the polyhedral model

This part of Compsys activity is mostly directed by the needs of hardware synthesis. The steps of the design of an accelerator circuit for streaming applications are (1) scheduling, (ii) sizing the local memories and specifying the communications inside the accelerator chip, (iii) specifying the communications with the external memory and the host processor, and (iv) generating the VHDL description of the chip. All these steps are interdependent, and cannot be solved in one pass due, in part, to the lack of a precise formal model, and also to the emergence of non linear problems. Compsys will attempt to improve this situation. Some possible optimization problems are sketched below. For that, polyhedral tools must be extended to handle problems beyond the reach of direct linear programming methods, such as scheduling under resource constraints or memory management. A step in this direction has been our work on array contraction with modulos or the introduction of sophisticated polyhedral optimizations to cope with tiling, pipelining, and data reuse simultaneously. All these extensions stress the polyhedral model and require new objective functions, new optimizations techniques, and a better control of complexity and scalability.

5.1.1 Communication with the external memory

Since the bandwidth to the external memory is limited, this is the most important performance optimization. Our solution consists in maximizing data reuse along the execution of the kernel, by identifying first reads and last writes for each array cell. The problem has been fully solved in the context of Alexandru Plesco PhD thesis, for the case of a perfect loop nest, optimized with loop tiling but run sequentially. It remains to extend it to more general kernels and to address parallel execution. Besides, this approach may need excessive amounts of local memory. This can be alleviated either by spilling to an external memory (live-range splitting), by not exploiting full data reuse, by multi-level tiling, or by slowing down the schedule.

5.1.2 Scheduling

When scheduling communicating processes, the obvious solution is firstly to compute local schedules for each process, then to compute inter-process schedules. As the CRP experience has shown, this is not the best way, as the processor schedules cannot be adjusted in order to simplify communications. Another drawback is that resource and memory constraints are not taken into account: for instance, it is not possible to construct pipelined schedules, except in an ad hoc and restricted way. Much progress is needed in this direction, in particular in the light of streaming specification or execution.

5.1.3 Access to local memories

Modern FPGAs have multiple local memory banks, which are easier to build than multi-port memories. The problem is how to partition the data set of the application in such a way that access to far-away memories is minimized. The schedule must also be adapted to the limited parallelism allowed by the number of available memory banks.

5.1.4 Control generation

The last step is the actual generation of the FPGA code. In hardware, control is the responsibility of a finite state machine (FSM), instead of a program counter as for software. The main question here is: how many FSMs for a given application? The construction of only one global FSM is possible and easy from the compiler point of view, but poses difficult problems for the clock signal distribution, and limits the amount of parallelism. One FSM per process or even more seems more attractive, as faster clocks can be used, but synchronization hardware is needed when two processes must exchange data.

5.2 Beyond the polyhedral model

As explained previously, we want to be able to go beyond the standard static control parts (SCoPs), to deal with a larger class of kernels. This can be done in several ways that need to be explored. Each extension requires to rethink the model that underlies the standard polyhedral techniques and poses complexity and scalability issues.

5.2.1 Incremental extensions

A proposal is to start from an existing purely polyhedral tool, e.g., Syntol, and to enlarge it by progressively introducing control in it. Syntol deals with regular process networks, which have a computational part and a communication part. Introducing conditionals in the computational part is easy: the method of "if conversion" offers a ready made solution. Introducing while loops is more difficult; using speculation may be a solution. The last step would be the introduction of conditional writes, which may necessitate major modifications in the communication protocol.

5.2.2 Dealing with approximations

The basic idea is to construct a polyhedral over-approximation of an irregular program, i.e., a program which has more operations, a larger memory footprint, and more dependences than the original. One can then parallelize the approximated program using polyhedral tools, and then return to the original, either by introducing guards, or by insuring that approximations are harmless. This technique is the standard way of dealing with approximated dependences. We also studied the impact of approximations in our work on Chuba, for optimizing remote communications. It is clear however that this method will apply only to mildly non-polyhedral programs. The restriction to arrays as the only data structure is still present. Its advantage is that it subsumes in a coherent framework many disparate tricks: the extraction of SCoPs, induction variable detection, the omission of non-affine subscripts, or the conversion of control dependences into data dependences. The link with the techniques developed in the PIPS compiler (based on array region analysis) is strong and will have to be explored.

5.2.3 Inductive compilation

There have been many attempts to trace the execution of a sequential program and to infer properties to be used in optimization and/or parallelization: the inspector/executor method, speculative parallelization for instance. Our proposal here is to apply sophisticated pattern matching techniques (one may say, polyhedral pattern matching techniques) to trace analysis. The inferred properties (regular access patterns, linear evolution of variables, absence of dependences) will be fed back to the compiler to be used to advantage in the generation of the target code. However, since no amount of experimentation can prove a theorem, the compiler will have to prepare two versions of the code, to be selected at run time depending on the truth or falsehood of the inference. This approach is one of the few methods that can be applied to machine or assembly code. Collaborations with the Camus Inria team already exist on this topic.

5.2.4 Regular programs in general

A program is called regular if its behavior can be predicted at compile time, and is relatively independent of its data. There probably exist many families of regular programs, but the only one that has been extensively studied is the family of static control programs, i.e., of programs that fit in the polyhedral model. To design other regular models, one needs to create new "parallel" data structures (i.e., data structures in which accessing a random element does not depend "too much" on the size of the structure), new control structures (e.g., natural enumerators) and new accessor functions (functions relating a position in the execution domain to an element of a data structure). Optimization and parallelization may rely on ad hoc transformations, specially adapted to the selected data and control structures, or use the universal concept of dependence, which must be adapted and whose decidability must be checked. This is a radical departure from the current concepts in parallel programming, and is a subject for long term research.

6 Bibliography of the project-team

The following references are those published during the evaluation period, i.e., 2007-2012. Most of them are directly related to the core topics of Compsys II. A few of them reflect research started earlier, in the context of Compsys I or research performed by non-permanent members (typically post-doc), but during their stay in Compsys II. Earlier references can be found on Compsys web site `http://www.ens-lyon.fr/LIP/COMPSYS` if needed.

Doctoral dissertations and "Habilitation" theses

[p1] Benoit Boissinot. *Towards an SSA-Based Compiler Back-End: Some Interesting Properties of SSA and Its Extensions.* PhD thesis, École normale supérieure de Lyon, September 2010.

[p2] Florent Bouchez. *A Study of Spilling and Coalescing in Register Allocation as Two Separate Phases.* PhD thesis, École normale supérieure de Lyon, April 2009.

[p3] Hadda Cherroun. *Scheduling for High-Level Synthesis.* PhD thesis, Université des Sciences et de la Technologie Houari Boumediene, Alger, December 2007.

[p4] Nicolas Fournel. *Estimation et optimisation de performances temporelles et énergétiques pour la conception de logiciels embarqués.* PhD thesis, École normale supérieure de Lyon, November 2007.

[p5] Philippe Grosse. *Gestion dynamique des tâches dans une architecture microélectronique intégrée à des fins de basse consommation.* PhD thesis, École normale supérieure de Lyon, December 2007.

[p6] Alexandru Plesco. *Program Transformations and Memory Architecture Optimizations for High-Level Synthesis of Hardware Accelerators.* PhD thesis, École normale supérieure de Lyon, September 2010.

Edition of books

[b1] Fabrice Rastello, editor. *SSA-Based Compiler Design.* Springer, 2012.

Articles in referred journals and book chapters

[j1] Benoit Boissinot, Philip Brisk, Alain Darte, and Fabrice Rastello. SSI properties revisited. *ACM Transactions on Embedded Computing Systems*, 2010. Special Issue on Software and Compilers for Embedded Systems, to appear.

[j2] Hadda Cherroun, Alain Darte, and Paul Feautrier. Reservation table scheduling: Branch-and-bound based optimization vs. integer linear programming techniques. *RAIRO-OR*, 41(4):427–454, December 2007.

[j3] Alain 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. *Revue de Mathématiques Spéciales*, 119(1), 2008.

[j4] Alain Darte. Optimal parallelism detection in nested loops. In David Padua, editor, *Encyclopedia of Parallel Programming.* Springer, 2011.

[j5] Paul Feautrier. Les compilateurs. In Jean-Eric Pin, editor, *Encyclopédie de l'Informatique.* Vuibert, 2007.

[j6] Paul Feautrier. Array layout for parallel processing. In David Padua, editor, *Encyclopedia of Parallel Programming.* Springer, 2011.

[j7] Paul Feautrier. Bernstein's conditions. In David Padua, editor, *Encyclopedia of Parallel Programming.* Springer, 2011.

[j8] Paul Feautrier. Dependences. In David Padua, editor, *Encyclopedia of Parallel Programming.* Springer, 2011.

[j9] Paul Feautrier and Christian Lengauer. The polyhedron model. In David Padua, editor, *Encyclopedia of Parallel Programming.* Springer, 2011.

[j10] Antoine Fraboulet and Tanguy Risset. Master interface for on-chip hardware accelerator burst communications. *Journal of VLSI Signal Processing*, 2(1):73–85, 2007.

[j11] Philippe Grosse, Yves Durand, and Paul Feautrier. Methods for power optimization in SOC-based data flow systems. *ACM Transactions on Design Automation of Electronic Systems*, 14(3):1–20, 2009.

[j12] Marie Rastello, Fabrice Rastello, Hervé Bellot, Frédéric Ousset, François Dufour, and Lorenz Meier. Size of snow particles in a powder-snow avalanche. *Journal of Glaciology*, 57(201):151–156, March 2011.

[j13] Antoine Scherrer, Nicolas Larrieu, Pierre Borgnat, Philippe Owezarski, and Patrice Abry. Non Gaussian and long memory statistical characterisations for internet traffic with anomalies. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 4(1):56–70, 2007.

Publications in conferences and workshops

[c1] Christophe Alias, Fabrice Baray, and Alain Darte. Bee+Cl@k: An implementation of lattice-based array contraction in the source-to-source translator ROSE. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'07)*, volume 42-7, pages 73–82, San Diego, USA, June 2007. ACM Press.

[c2] Christophe Alias, Alain Darte, Paul Feautrier, and Laure Gonnord. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In *17th International Static Analysis Symposium (SAS'10)*, pages 117–133, Perpignan, France, September 2010. ACM press.

[c3] Christophe Alias, Alain Darte, and Alexandru Plesco. Optimizing DDR-SDRAM communications at C-level for automatically-generated hardware accelerators. An experience with the Altera C2H HLS tool. In *21st IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'10)*, pages 329–332, Rennes, France, July 2010. IEEE Computer Society.

[c4] Christophe Alias, Alain Darte, and Alexandru Plesco. Optimizing remote accesses for offloaded kernels: Application to high-level synthesis for FPGA. In *17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'12)*, New Orleans, USA, February 2012. IEEE Computer Society. Short paper.

[c5] Christophe Alias, Alain Darte, and Alexandru Plesco. Optimizing remote accesses for offloaded kernels: Application to high-level synthesis for FPGA. In *2nd International Workshop on Polyhedral Compilation Techniques (IMPACT'12)*, Paris, January 2012.

[c6] Christophe Alias, Bogdan Pasca, and Alexandru Plesco. Automatic generation of FPGA-specific pipelined accelerators. In *7th International Symposium on Applied Reconfigurable Computing (ARC'11)*, pages 53–66, Belfast, UK, March 2011. Springer Verlag.

[c7] Benoit Boissinot, Florian Brandner, Alain Darte, Benoit Dupont de Dinechin, and Fabrice Rastello. A non-iterative data-flow algorithm for computing liveness sets in strict SSA programs. In *9th Asian Symposium on Programming Languages and Systems (APLAS'11)*. Springer Verlag, December 2011.

[c8] Benoit Boissinot, Alain Darte, Benoît Dupont de Dinechin, Christophe Guillon, and Fabrice Rastello. Revisiting out-of-SSA translation for correctness, code quality, and efficiency. In *International Symposium on Code Generation and Optimization (CGO'09)*, pages 114–125. IEEE Computer, March 2009. **Best paper award**.

[c9] Benoit Boissinot, Sebastian Hack, Daniel Grund, Benoît Dupont de Dinechin, and Fabrice Rastello. Fast liveness checking for SSA-form programs. In *Sixth Annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO'08)*, pages 35–44, Boston, USA, April 2008. ACM Press. **Best paper award**.

[c10] Florent Bouchez, Quentin Colombet, Alain Darte, Christophe Guillon, and Fabrice Rastello. Parallel copy motion. In *13th International Workshop on Software & Compilers for Embedded Systems (SCOPES'10)*, pages 1–10, St. Goar, Germany, June 2010. ACM Press.

[c11] Florent Bouchez, Alain Darte, and Fabrice Rastello. On the complexity of register coalescing. In *International Symposium on Code Generation and Optimization (CGO'07)*, pages 102–114. IEEE Computer, March 2007. **Best paper award**.

[c12] Florent Bouchez, Alain Darte, and Fabrice Rastello. On the complexity of spill everywhere under SSA form. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'07)*, volume 42-7, pages 103–112, San Diego, USA, June 2007. ACM Press.

[c13] Florent Bouchez, Alain Darte, and Fabrice Rastello. Advanced conservative and optimistic register coalescing. In *International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES'08)*, pages 147–156, Atlanta, GA, USA, October 2008. ACM Press.

[c14] Florian Brandner. Completeness of automatically generated instruction selectors. In *21st International Conference on Application-specific Systems Architectures and Processors (ASAP'10)*, pages 175–182, Rennes, France, July 2010. IEEE Computer Society.

[c15] Florian Brandner and Quentin Colombet. Copy elimination on data dependence graphs. In *Symposium on Applied Computing (SAC'12)*, Trento, Italy, March 2012. ACM Press.

[c16] Florian Brandner and Alain Darte. Compiler-driven optimization of the worst-case execution time. In Laure Gonnord and David Monniaux, editors, *Workshop "Analyse to Compile, Compile to Analyse" (ACCA'11), held with CGO'11*, Chamonix, April 2011.

[c17] Florian Brandner, Viktor Pavlu, and Andreas Krall. Execution models for processors and instructions. In *28th Norchip Conference (NORCHIP'10)*, November 2010.

[c18] G. Chelius, A. Fraboulet, and E. Fleury. Worldsens: A fast and accurate development framework for sensor network applications. In *22nd Annual ACM Symposium on Applied Computing (SAC'07)*, Seoul, Korea, March 2007. ACM.

[c19] Hadda Cherroun and Paul Feautrier. An exact resource constrained-scheduler using graph coloring technique. In *5th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'07)*, pages 554–561. IEEE Computer Society, May 2007. **Best paper award**.

[c20] Quentin Colombet, Benoit Boissinot, Philip Brisk, Sebastian Hack, and Fabrice Rastello. Graph coloring and treescan register allocation using repairing. In *International Conference on Compilers, Architectures, and Synthesis of Embedded Systems (CASES'11)*, Taipei, Taiwan, October 2011. IEEE Computer Society.

[c21] Quentin Colombet, Florian Brandner, and Alain Darte. Studying optimal spilling in the light of SSA. In *International Conference on Compilers, Architectures, and Synthesis of Embedded Systems (CASES'11)*, Taipei, Taiwan, October 2011. IEEE Computer Society.

[c22] Benoit Combemale, Laure Gonnord, and Vlad Rusu. A generic tool for tracing executions back to a DSML's operational semantics. In *7th European Conference on Modelling Foundations and Applications (ECMFA'11)*, volume 6698 of *Lecture Notes in Computer Science*, pages 35–51, Birmingham, United Kingdom, June 2011. Springer Verlag.

[c23] Alain Darte. Understanding loops: The influence of the decomposition of Karp, Miller, and Winograd. In *8th ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'10)*, pages 139–148, Grenoble, France, July 2010. IEEE Computer Society. Invited paper.

[c24] Alain Darte and Clément Quinson. Scheduling register-allocated codes in user-guided high-level synthesis. In *18th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'07)*, pages 554–561. IEEE Computer Society, July 2007.

[c25] Florent de Dinechin, Jean-Michel Muller, Bogdan Pasca, and Alexandru Plesco. An FPGA architecture for solving the table maker's dilemma. In *22nd IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'11)*, Santa Monica, CA, September 2011. IEEE Computer Society.

[c26] Boubacar Diouf, Albert Cohen, Fabrice Rastello, and John Cavazos. Split register allocation: Linear complexity without the performance penalty. In *International Conference on High-Performance Embedded Architectures and Compilers (HiPEAC'10)*, volume 5952 of *Lecture Notes in Computer Science*, pages 66–80. Springer Verlag, January 2010.

[c27] Nicolas Farrugia, Michel Paindavoine, and Clément Quinson. On the need for semi-automated source-to-source transformations in the user-guided high-level synthesis tool. In *High-Level Synthesis: Back to the Future (DAC'08 workshop)*, June 2008. Poster.

[c28] Paul Feautrier. Approximating the transitive closure of a boolean-affine relation. In *2nd International Workshop on Polyhedral Compilation Techniques (IMPACT'12)*, Paris, January 2012.

[c29] Paul Feautrier and Laure Gonnord. Accelerated invariant generation for C programs with Aspic and C2fsm. In *Workshop on Tools for Automatic Program Analysis (TAPAS'10)*, volume 267-2 of *Electronic Notes in Theoretical Computer Science*, pages 3–13, September 2010.

[c30] N. Fournel, M. Minier, and S. Ubéda. Survey and benchmark of stream ciphers for wireless sensor networks. In *Workshop in Information Security Theory and Practices (WISTP'07)*, Heraklion, Crete, Greece, May 2007.

[c31] Nicolas Fournel, Antoine Fraboulet, Guillaume Chelius, Eric Fleury, Bruno Allard, and Olivier Brevet. Worldsens: Embedded sensor network application development and deployment. In *26th Annual IEEE Conference on Computer Communications (INFOCOM'07)*, Anchorage, Alaska, USA, May 2007. IEEE.

[c32] Nicolas Fournel, Antoine Fraboulet, Guillaume Chelius, Eric Fleury, Bruno Allard, and Olivier Brevet. Worldsens: From lab to sensor network application development and deployment. In *International Conference on Information Processing in Sensor Networks (IPSN'07), demo session*, Cambridge, MA, USA., April 2007. ACM.

[c33] Nicolas Fournel, Antoine Fraboulet, and Paul Feautrier. eSimu: A fast and accurate energy consumption simulator for embedded systems. In *IEEE International Workshop: From Theory to Practice in Wireless Sensor Networks*, Helsinki, Finland, June 2007.

[c34] Nicolas Fournel, Antoine Fraboulet, and Paul Feautrier. Fast and instruction accurate embedded systems energy characterization using non-intrusive measurements. In *PATMOS Workshop - International Workshop on Power And Timing Modeling, Optimization and Simulation*, Göteborg, Sweden, September 2007.

[c35] Antoine Fraboulet, Guillaume Chelius, and Eric Fleury. Worldsens: Development and prototyping tools for application specific wireless sensors networks. In *IPSN Track on Sensor Platforms, Tools and Design Methods (SPOTS'07)*, Cambridge, MA, USA., April 2007. ACM.

[c36] Abdoulaye Gamatie and Laure Gonnord. Static analysis of synchronous programs in Signal for efficient design of multi-clocked embedded systems. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'11)*, pages 71–80, New York, NY, USA, 2011. ACM.

[c37] L. Gonnord and J.-P. Babau. Quantity of resource properties expression and runtime assurance for embedded systems. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'09)*, pages 428–435, Rabat, Morocco, May 2009.

[c38] Daniel Grund and Sebastian Hack. A fast cutting-plane algorithm for optimal coalescing. In Shriram Krishnamurthi and Martin Odersky, editors, *Compiler Construction (CC'07)*, volume 4420 of *Lecture Notes In Computer Science*, pages 111–125, Braga, Portugal, March 2007. Springer. **Best paper award**.

[c39] Ouassila Labbani, Paul Feautrier, Eric Lenormand, and Michel Barreteau. Elementary transformation analyses for Array-OL. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'09)*, pages 362–367, Rabat, Morocco, May 2009.

[c40] Julien Le Guen, Christophe Guillon, and Fabrice Rastello. MinIR, a minimalistic intermediate representation. In Florent Bouchez, Sebastian Hack, and Eelco Visser, editors, *Workshop on Intermediate Representations (WIR'11), held with CGO'11*, pages 5–12, Chamonix, April 2011.

[c41] Qingda Lu, Christophe Alias, Uday Bondhugula, Sriram Krishnamoorthy, J. Ramanujam, Atanas Rountev, P. Sadayappan, Yongjian Chen, Haibo Lin, and Tin fook Ngai. Data layout transformation for enhancing locality on NUCA chip multiprocessors. In *International ACM/IEEE Conference on Parallel Architectures and Compilation Techniques (PACT'09)*, pages 348–357. ACM Press, September 2009.

[c42] David Monniaux and Laure Gonnord. Using bounded model checking to focus fixpoint iterations. In Eran Yahav, editor, *Static analysis (SAS'11)*, volume 6887 of *Lecture Notes in Computer Science*, pages 369–385. Springer Verlag, 2011.

[c43] Alexandru Plesco and Tanguy Risset. Coupling loop transformations and high-level synthesis. In *SYMPosium en Architectures nouvelles de machines (SYMPA'08)*, February 2008.

[c44] Marie Rastello, Fabrice Rastello, Hervé Bellot, Frédéric Ousset, and François Dufour. Size of snow particles in a powder-snow avalanche. In *ASME Fluids Engineering Division Summer Meeting 2009 (FEDSM'09)*, August 2009.

[c45] Martin Schoeberl, Pascal Schleuniger, Wolfgang Puffitsch, Florian Brandner, Christian W. Probst, Sven Karlsson, and Tommy Thorn. Towards a time-predictable dual-issue microprocessor: The Patmos approach. In *Bringing Theory to Practice: Predictability and Performance in Embedded Systems, DATE Workshop PPES'11*, volume 18, pages 11–21, Grenoble, France, March 2011.

[c46] André Tavares, Quentin Colombet, Mariza Bigonha, Christophe Guillon, Fernando Pereira, and Fabrice Rastello. Decoupled graph-coloring register allocation with hierarchical aliasing. In *14th International Workshop on Software & Compilers for Embedded Systems (SCOPES'11)*, pages 1–10, St. Goar, Germany, June 2011. ACM Press.

Other: research reports, patents, tutorials, keynotes, etc.

[o1] Christophe Alias, Fabrice Baray, and Alain Darte. Lattice-based array contraction: From theory to practice. Research Report 2007-44, INRIA, November 2007.

[o2] Christophe Alias, Alain Darte, and Alexandru Plesco. Kernel offloading with optimized remote accesses. Research Report RR-7697, INRIA, July 2011.

[o3] Christophe Alias, Alain Darte, and Alexandru Plesco. Program analysis and source-level communication optimizations for high-level synthesis. Research Report RR-7648, INRIA, June 2011.

[o4] Christophe Alias, Bogdan Pasca, and Alexandru Plesco. FPGA-specific synthesis of loop-nests with pipelined computational cores. Research Report RR-7674, INRIA, July 2011.

[o5] Benoit Boissinot, Philip Brisk, Alain Darte, and Fabrice Rastello. SSI revisited. Research Report RR2009-24, LIP, July 2009.

[o6] Benoit Boissinot, Sebastian Hack, Daniel Grund, Benoît Dupont de Dinechin, and Fabrice Rastello. Fast liveness checking for SSA-form programs. Research Report RR2007-45, LIP, ENS-Lyon, France, September 2007.

[o7] Florent Bouchez, Philip Brisk, Sebastian Hack, Jens Palsberg, and Fabrice Rastello. SSA-based register allocation. In *22nd International Workshop on Languages and Compilers for Parallel Computers (LCPC'09)*, Newark, October 2009. **Tutorial**.

[o8] Florent Bouchez, Alain Darte, and Fabrice Rastello. Improvements to conservative and optimistic register coalescing. Research Report RR2007-41, LIP, ENS-Lyon, France, March 2007.

[o9] Florent Bouchez, Alain Darte, and Fabrice Rastello. On the complexity of spill everywhere under SSA form. Research Report RR2007-42, LIP, ENS-Lyon, France, March 2007.

[o10] Florian Brandner, Benoit Boissinot, Alain Darte, Benoît Dupont de Dinechin, and Fabrice Rastello. Computing liveness sets for SSA-form programs. Research Report RR-7503, INRIA, April 2011.

[o11] Florian Brandner and Quentin Colombet. Parallel copy elimination on data dependence graphs. Research Report RR-7735, INRIA, September 2011.

[o12] Philip Brisk, Alain Darte, Jens Palsberg, and Fabrice Rastello. SSA-based register allocation. In *International Symposium on Code Generation and Optimization (CGO'09)*, Seattle, March 2009. **Tutorial**.

[o13] Philip Brisk, Sebastian Hack, Jens Palsberg, Fernando Pereira, and Fabrice Rastello. SSA-based register allocation. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems(CASES'08, part of ESWEEK'08)*, Atlanta, October 2008. **Tutorial**.

[o14] Alain Darte. Approximations in the polyhedral model. In *1st International Workshop on Polyhedral Compilation Techniques (IMPACT'11), CGO'11 Workshop*, Chamonix, April 2011. **Keynote talk**.

[o15] Alain Darte and Rob Schreiber. System and method of optimizing memory usage with data lifetimes. **US patent** number 7363459, April 2008.

[o16] Paul Feautrier. Elementary transformation analysis for Array-OL. Research Report 6193, INRIA, May 2007.

[o17] Paul Feautrier. The polytope model, past, present, future. In *22nd International Workshop on Languages and Compilers for Parallel Computers (LCPC'09)*, Newark, October 2009. **Keynote talk**.

[o18] Paul Feautrier. Simplification of Boolean affine formulas. Research Report RR-7689, INRIA, July 2011.

[o19] Sebastian Hack. Register allocation for programs in SSA form. DATE'07 PhD Forum Poster, April 2007.