
IN PARTNERSHIP WITH:
**CNRS**

**Ecole normale supérieure de Paris**

Activity Report 2013

# Project-Team PARKAS

## Synchronous Kahn Parallelism

IN COLLABORATION WITH: Département d'Informatique de l'Ecole Normale Supérieure

RESEARCH CENTER
**Paris - Rocquencourt**

THEME
**Embedded and Real-time Systems**

# Table of contents

## Project-Team PARKAS

**Keywords:** Compiling, Embedded Systems, Parallelism, Programming Languages, Synchronous Languages

*Creation of the Team:* 2011 April 01*, updated into Project-Team:* 2012 January 01.

# 1. Members

**Research Scientists**

Timothy Bourke [Inria, Starting Research position]
Albert Cohen [Inria, Senior Researcher, HdR]
Francesco Zappa Nardelli [Inria, Researcher]

**Faculty Members**

Marc Pouzet [Team leader, UPMC and ENS, Professor, HdR]
Jean Vuillemin [ENS, Professor, HdR]
Louis Mandel [Univ. Paris XI, Associate Professor until Sep 2013]

**Engineers**

Jun Inoue [Inria, granted by Caisse des Dépôts et Consignations]
Martin Kong Moreno [Inria, granted by FP7 TERAFLUX project, from Jun 2013 until Sep 2013]
Feng Li [Inria, granted by FP7 TERAFLUX project]
Mircea Namolaru [Inria, granted by FP7 TERAFLUX project, from Oct 2013]
Antoniu Pop [Inria, granted by FP7 TERAFLUX project, until Oct 2013]
Ramakrishna Upadrasta [Inria, granted by FP7 TERAFLUX project, from Jan 2013 until Feb 2013]

**PhD Students**

Nhat Minh Lê [ENS Paris]
Riyadh Baghdadi [ENS Paris]
Guillaume Baudart [ENS Paris, from Sep 2013]
Camille Gallet [CEA]
Tobias Grosser [Inria, granted by Google UK Ltd]
Adrien Guatto [ENS Paris]
Léonard Gérard [Inria, granted by Caisse des Dépôts et Consignations, until Oct 2013]
Ivan Llopard [CEA]
Robin Morisset [Inria, granted by Google Inc]
Cédric Pasteur [Inria, granted by FP7 TERAFLUX project, until Nov 2013]
Jean-Yves Vet [CEA, until Aug 2013]

**Post-Doctoral Fellows**

Pejman Attar [Inria, granted by ANR WMC project, from Nov 2013 until Nov 2013]
Boubacar Diouf [Inria, granted by FP7 TERAFLUX project]
Antoine Madet [Inria, granted by Min. de l'Economie, from Jan 2013]
Sven Verdoolaege [ENS Paris]

**Visiting Scientist**

Govindarajan Ramaswamy [Inria, invited Professor, Indian Institute of Science, from May 2013 until Aug 2013]

**Administrative Assistant**

Assia Saadi [Inria]

**Others**

Guillaume Chelfi [Inria, student at Telecom ParisTech, from May 2013 until Sep 2013]
Pankaj Prateek Kewalramani [Inria, student at IIT Kanpur, India, from May 2013 until Jul 2013]

Anirudh Kumar [Inria, student at IIT Kanpur, India, from May 2013 until Jul 2013]
Pankaj More [Inria, student at IIT Kanpur, India, from May 2013 until Aug 2013]
Vincent Thiberville [Inria, student at École Polytechnique, from Apr 2013 until Jun 2013]

# 2. Overall Objectives

## 2.1. Overall Objectives

The goal of the project is the design, semantics and compilation of languages for the implementation of provably safe and efficient computing systems. We are driven by the ideal of a unique source code used both to *program* and *simulate* a wide variety of systems, including (1) embedded real-time controllers (e.g., fly-by-wire, engine control); (2) computationally intensive applications (e.g., video); (3) the simulation of (a possibly huge number of) embedded systems in close interaction (e.g., simulation of electrical or sensor networks, train tracking). All these applications share the need for formally defined languages used both for simulation and the generation of target code. For that purpose, we design languages and experiment with compilers that transform mathematical specifications of systems into target code, that may execute on parallel (multi-core) architectures.

Our research team draws inspiration and focus from the simplicity and complementarity of the data-flow model of Kahn process networks, synchronous concurrency, and the expression of the two in functional languages. To reach our goal, we plan to leverage a large body of formal principles: language design, semantics, type theory, concurrency models (including recent works on the formalisation of relaxed memory models), synchronous circuits and algorithms (code generation, optimization, polyhedral compilation).

## 2.2. Highlights of the Year

Robin Morisset was Awarded a Google Doctoral Fellowship.

Louis Mandel and Marc Pouzet received a reward for the paper introducing the ReactiveML language for the first time and presented at the French conference JFLA 2005 (“On the occasion of this quarter century, the program committees and steering selected four outstanding contributions from the articles published in JFLA past decade.")

Louis Mandel has been hired in Sept. 2014 at Collège de France, as an Assistant Professor.

# 3. Research Program

## 3.1. Presentation and originality of the PARKAS team

Our project is founded on our expertise in three complementary domains: (1) synchronous functional programming and its extensions to deal with features such as communication with bounded buffers and dynamic process creation; (2) mathematical models for synchronous circuits; (3) compilation techniques for synchronous languages and optimizing/parallelizing compilers.

A strong point of the team is its experience and investment in the development of languages and compilers. Members of the team also have direct collaborations for several years with major industrial companies in the field and several of our results are integrated in successful products. Our main results are briefly summarized below.

### *3.1.1. Synchronous functional programming*

In [35], Paul Caspi and Marc Pouzet introduced *synchronous Kahn networks* as those Kahn networks that can be statically scheduled and executed with bounded buffers. This was the origin of the language LUCID SYNCHRONE, [1] [2] an ML extension of the synchronous language LUSTRE with higher-order features, dedicated type systems (clock calculus as a type system [35], [45], initialization analysis [46] and causality analysis [47]). The language integrates original features that are not found in other synchronous languages: such as combinations of data flow, control flow, hierarchical automata and signals [44], [43], and modular code generation [36], [33].

In 2000, Marc Pouzet started to collaborate with the SCADE team of Esterel-Technologies on the design of a new version of SCADE. [3] Several features of LUCID SYNCHRONE are now integrated into SCADE 6, which has been distributed since 2008, including the programming constructs `merge`, `reset`, the clock calculus and the type system. Several results have been developed jointly with Jean-Louis Colaço and Bruno Pagano from Esterel-Technologies, such as ways of combining data-flow and hierarchical automata, and techniques for their compilation, initialization analysis, etc.

Dassault-Systèmes (Grenoble R&D center, part of Delmia-automation) developed the language LCM, a variant of LUCID SYNCHRONE that is used for the simulation of factories. LCM follows closely the principles and programming constructs of LUCID SYNCHRONE (higher-order, type inference, mix of data-flow and hierarchical automata). The team in Grenoble is integrating this development into a new compiler for the language Modelica. [4]

In parallel, the goal of REACTIVEML[5] was to integrate a synchronous concurrency model into an existing ML language, with no restrictions on expressiveness, so as to program a large class of reactive systems, including efficient simulations of millions of communicating processes (e.g., sensor networks), video games with many interactions, physical simulations, etc. For such applications, the synchronous model simplifies system design and implementation, but the expressiveness of the algorithmic part of the language is just as essential, as is the ability to create or stop a process dynamically.

The development of REACTIVEML was started by Louis Mandel during his PhD thesis [57], [55] and is ongoing. The language extends OCAML[6] with Esterel-like synchronous primitives — synchronous composition, broadcast communication, pre-emption/suspension — applying the solution of Boussinot [34] to solve causality issues.

Several open problems have been solved by Louis Mandel: the interaction between ML features (higher-order) and reactive constructs with a proper type system; efficient simulation that avoids busy waiting. The latter problem is particularly difficult in synchronous languages because of possible reactions to the absence of a signal. In the REACTIVEML implementation, there is no busy waiting: inactive processes have no impact on the overall performance. It turns out that this enables REACTIVEML to simulate millions of (logical) parallel processes and to compete with the best event-driven simulators [58].

REACTIVEML has been used for simulating routing protocols in ad-hoc networks [54] and large scale sensor networks [68]. The designer benefits from a real programming language that gives precise control of the level of simulation (e.g., each network layer up to the MAC layer) and programs can be connected to models of the physical environment programmed with LUTIN [67]. REACTIVEML is used since 2006 by the synchronous team at VERIMAG, Grenoble (in collaboration with France-Telecom) for the development of low-consumption routing protocols in sensor networks.

### *3.1.2. Relaxing synchrony with buffer communication*

[1] http://www.di.ens.fr/~pouzet/lucid-synchrone
[2] The name is a reference to Lustre which stands for "Lucid Synchrone et Temps réel".
[3] http://www.esterel-technologies.com/products/scade-suite/
[4] http://www.3ds.com/products/catia/portfolio/dymola/overview/
[5] http://rml.lri.fr/
[6] More precisely a subset of OCAML without objects or functors.

In the data-flow synchronous model, the clock calculus is a static analysis that ensures execution in bounded memory. It checks that the values produced by a node are instantaneously consumed by connected nodes (synchronous constraint). To program Kahn process networks with bounded buffers (as in video applications), it is thus necessary to explicitly place nodes that implement buffers. The buffers sizes and the clocks at which data must be read or written have to be computed manually. In practice, it is done with simulation or successive tries and errors. This task is difficult and error prone. The aim of the $n$-synchronous model is to automatically compute at compile time these values while insuring the absence of deadlock.

Technically, it allows processes to be composed whenever they can be synchronized through a bounded buffer [37], [38]. The new flexibility is obtained by relaxing the clock calculus by replacing the equality of clocks by a sub-typing rule. The result is a more expressive language which still offers the same guarantees as the original. The first version of the model was based on clocks represented as ultimately periodic binary words [72]. It was algorithmically expensive and limited to periodic systems. In [41], an abstraction mechanism is proposed which permits direct reasoning on sets of clocks that are defined as a rational slope and two shifts. An implementation of the $n$-synchronous model, named LUCY-N, was developed in 2009 [56], as was a formalization of the theory in COQ [42]. We also worked on low-level compiler and runtime support to parallelize the execution of relaxed synchronous systems, proposing a portable intermediate language and runtime library called ERBIUM [59].

This work started as a collaboration between Marc Pouzet (LIP6, Paris, then LRI and Inria Proval, Orsay), Marc Duranton (Philips Research then NXP, Eindhoven), Albert Cohen (Inria Alchemy, Orsay) and Christine Eisenbeis (Inria Alchemy, Orsay) on the real-time programming of video stream applications in set-top boxes. It was significantly extended by Louis Mandel and Florence Plateau during her PhD thesis [62] (supervised by Marc Pouzet and Louis Mandel). Low-level support has been investigated with Cupertino Miranda, Philippe Dumont (Inria Alchemy, Orsay) and Antoniu Pop (Mines ParisTech). Further directions of research and experimentation have been and are being followed through the theses of Léonard Gérard, Adrien Guatto and Nhat Minh Lê.

### 3.1.3. Polyhedral compilation and optimizing compilers

Despite decades of progress, the best parallelizing and optimizing compilers still fail to extract parallelism and to perform the necessary optimizations to harness multi-core processors and their complex memory hierarchies. *Polyhedral compilation* aims at facilitating the construction of more effective optimization and parallelization algorithms. It captures the flow of data between individual instances of statements in a loop nest, allowing to accurately model the behavior of the program and represent complex parallelizing and optimizing transformations. Affine multidimensional scheduling is one of the main tools in polyhedral compilation [48]. Albert Cohen, in collaboration with Cédric Bastoul, Sylvain Girbal, Nicolas Vasilache, Louis-Noël Pouchet and Konrad Trifunovic (LRI and Inria Alchemy, Orsay) has contributed to a large number of research, development and transfer activities in this area.

The relation between polyhedral compilation and data-flow synchrony has been identified through data-flow array languages [53], [52], [69], [49] and the study of the scheduling and mapping algorithms for these languages. We would like to deepen the exploration of this link, embedding polyhedral techniques into the compilation flow of data-flow, relaxed synchronous languages.

Our previous work led to the design of a theoretical and algorithmic framework rooted in the polyhedral model of compilation, and to the implementation of a set of tools based on production compilers (Open64, GCC) and source-to-source prototypes (PoCC, http://pocc.sourceforge.net). We have shown that not only does this framework simplify the problem of building complex loop nest optimizations, but also that it scales to real-world benchmarks [39], [50], [65], [64]. The polyhedral model has finally evolved into a mature, production-ready approach to solve the challenges of maximizing the scalability and efficiency of loop-based computations on a variety of high performance and embedded targets.

After an initial experiment with Open64 [40], [39], we ported these techniques to GCC [63], [71], [70] and LLVM [51], applying them to multi-level parallelization and optimization problems, including vectorization and exploitation of thread-level parallelism. Independently, we made significant progress in the design

of effective optimization heuristics, working on the interactions between the semantics of the compiler's intermediate representation and the structure of the optimization space [65], [64], [66] [2], [5]. These results open opportunities for complex optimizations that target larger problems, such as the scheduling and placement of process networks, or the offloading of computational kernels to hardware accelerators (such as GPUs). A new framework has been designed, centered on the Integer Set Library (isl, http://freecode.com/projects/isl) and implemented through multiple compiler interfaces (Graphite in GCC, Polly in LLVM) and a source-to-source research compiler (PPCG) [8], [13], [16], [25], [28]. This new framework underlies our collaborative research activities in the CARP and COPCAMS European projects, as well as emerging transfer projects through the TETRACOM European coordination action and bilateral industry contracts in preparation.

#### *3.1.4. Automatic compilation of high performance circuits*

For both cost and performance reasons, computing systems tightly couple parts realized in hardware with parts realized in software. The boundary between hardware and software keeps moving with the underlying technology and the external economic pressure. Moreover, thanks to FPGA technology, hardware itself has become programmable. There is now a pressing need from industry for hardware/software co-design, and for tools which automatically turn software code into hardware circuits, or more usually, into hybrid code that simultaneously targets GPUs, multiple cores, encryption ASICs, and other specialized chips.

Departing from customary C-to-VHDL compilation, we trust that sharper results can be achieved from source programs that specify bit-wise time/space behavior in a rigorous synchronous language, rather than just the I/O behavior in some (ill-specified) subset of C. This specification allows the designer to also program the (asynchronous) environment in which to operate the entire system, and to profile/measure/control each variable of the design.

At any time, the designer can edit a single specification of the system, from which both the software and the hardware are automatically compiled, and guaranteed to be compatible. Once correct (functionally and with respect to the behavioral specification), the application can be automatically deployed (and tested) on a hard/soft hybrid co-design support.

Key aspects of the advocated methodology were validated by Jean Vuillemin in the design of a PAL2HDTV video sampler [60], [61]. The circuit was automatically compiled from a synchronous source specification, decorated and guided by a few key hints to the hardware back-end, that targetted an FPGA running at real-time video specifications: a tightly-packed highly-efficient design at 240MHz, generated 100% automatically from the application specification source code, and including all run-time/debug/test/validate ancillary software. It was subsequently commercialized on FPGA by LetItWave, and then on ASIC by Zoran. This successful experience underlines our research perspectives on parallel synchronous programming.

# 4. Application Domains

## 4.1. Domain

The project addresses the design, semantics and implementation of programming languages together with compilation techniques to develop provably safe and efficient computing systems. Traditional applications can be found in safety critical embedded systems with hard real-time constraints such as avionics (e.g., fly-by-wire command), railways (e.g., on board control, engine control), nuclear plants (e.g., emergency control of the plant). While embedded applications have been centralized, they are now massively parallel and physically distributed (e.g., sensor networks, train tracking, distributed simulation of factories) and they integrate computationally intensive algorithms (e.g., video processing) with a mix of hard and soft real-time constraints. Finally, systems are heterogeneous with discrete devices communicating with physical ones (e.g., interface between analog and digital circuits). Programming and simulating a whole system from a unique source code, with static guarantees on the reproducibility of simulations together with a compiler to generate target embedded code is a scientific and industrial challenge of great importance.

# 5. Software and Platforms

## 5.1. Lucid Synchrone

**Participant:** Marc Pouzet [contact].

Synchronous languages, type and clock inference, causality analysis, compilation

Lucid Synchrone is a language for the implementation of reactive systems. It is based on the synchronous model of time as provided by Lustre combined with features from ML languages. It provides powerful extensions such as type and clock inference, type-based causality and initialization analysis and allows to arbitrarily mix data-flow systems and hierarchical automata or flows and valued signals.

It is distributed under binary form, at URL http://www.di.ens.fr/~pouzet/lucid-synchrone/.

The language was used, from 1996 to 2006 as a laboratory to experiment various extensions of the language Lustre. Several programming constructs (e.g. merge, last, mix of data-flow and control-structures like automata), type-based program analysis (e.g., typing, clock calculus) and compilation methods, originaly introduced in Lucid Synchrone are now integrated in the new SCADE 6 compiler developped at Esterel-Technologies and commercialized since 2008.

Three major release of the language has been done and the current version is V3 (dev. in 2006). As of 2013, the language is still used for teaching and in our research but we do not develop it anymore. Nonetheless, we have integrated several features from Lucid Synchrone in new research prototypes described below. The Heptagon language and compiler are a direct descendent of it. The new language Zélus for hybrid systems modeling borrows many features originaly introduced in Lucid Synchrone.

## 5.2. ReactiveML

**Participants:** Guillaume Baudart, Louis Mandel [contact], Cédric Pasteur.

Programming language, synchronous reactive programming, concurrent systems, dedicated type-systems.

ReactiveML is a programming language dedicated to the implementation of interactive systems as found in graphical user interfaces, video games or simulation problems. ReactiveML is based on the synchronous reactive model due to Boussinot, embedded in an ML language (OCaml).

The Synchronous reactive model provides synchronous parallel composition and dynamic features like the dynamic creation of processes. In ReactiveML, the reactive model is integrated at the language level (not as a library) which leads to a safer and a more natural programming paradigm.

ReactiveML is distributed at URL http://reactiveml.org. The compiler is distributed under the terms of the Q Public License and the library is distributed under the terms of the GNU Library General Public License. The development of ReactiveML started at the University Paris 6 (from 2002 to 2006).

The language was mainly used for the simulation of mobile ad hoc networks at the Pierre and Marie Curie University and for the simulation of sensor networks at France Telecom and Verimag (CNRS, Grenoble). A new application to mixed music programming has been developed.

In 2013, a new web site has been developed. New programming constructs have been added. The runtime system has been cleanup. Moreover, a new implementation based on the PhD of Cédric Pasteur has also been provided http://reactiveml.org/these_pasteur.

## 5.3. Heptagon

**Participants:** Cédric Pasteur [contact], Brice Gelineau, Léonard Gérard, Adrien Guatto, Marc Pouzet.

Synchronous languages, compilation, optimizing compilation, parallel code generation, behavioral synthesis.

Heptagon is an experimental language for the implementation of embedded real-time reactive systems. It is developed inside the Synchronics large-scale initiative, in collaboration with Inria Rhones-Alpes. It is essentially a subset of Lucid Synchrone, without type inference, type polymorphism and higher-order. It is thus a Lustre-like language extended with hierchical automata in a form very close to SCADE 6. The intention for making this new language and compiler is to develop new aggressive optimization techniques for sequential C code and compilation methods for generating parallel code for different platforms. This explains much of the simplifications we have made in order to ease the development of compilation techniques.

Some extensions have already been made, most notably automata, a parallel code generator with Futures, support for correct and efficient in-place array computations. It's currently used to experiment with linear typing for arrays and also to introduce a concept of asynchronous parallel computations. The compiler developed in our team generates C, C++, java and VHDL code.

Transfer activities based on our experience in Heptagon are taking place through the "Fiabilité and Sûreté de Fonctionnement" project at IRT SystemX, led by Alstom Transport, since 2013.

Heptagon is jointly developed with Gwenael Delaval and Alain Girault from the Inria POP ART team (Grenoble). Gwenael Delaval is developing the controller synthesis tool BZR (http://bzr.inria.fr/) above Heptagon. Both software are distributed under a GPL licence.

## 5.4. Lucy-n: an n-synchronous data-flow programming language

**Participants:** Albert Cohen, Louis Mandel [contact], Adrien Guatto, Marc Pouzet.

Lucy-n is a language to program in the n-synchronous model. The language is similar to Lustre with a buffer construct. The Lucy-n compiler ensures that programs can be executed in bounded memory and automatically computes buffer sizes. Hence this language allows to program Kahn networks, the compiler being able to statically compute bounds for all FIFOs in the program.

The language compiler and associated tools are available in a binary form at http://www.lri.fr/~mandel/lucy-n.

In 2013, a complete re-implemtantion has been started. This new version will take into account the new features developed during the PhD of Adrien Guatto. Parallel code generation for this new version also involves compilation and runtime system research in collaboration with Nhat Minh Lê and Robin Morisset.

## 5.5. ML-Sundials

**Participants:** Timothy Bourke, Jun Inoue, Marc Pouzet [contact].

The ML-Sundials bindings allow the use of the state-of-the-art Sundials numerical simulation library from OCaml programs (like, for instance, the Zélus runtime). The Sundials packages includes three main components: CVODE, IDA, and KINSOL.

This year we redesigned and reimplemented the interface to CVODE to fix a problem with memory leaks between OCaml and C heaps. We have submitted an APP request for this code. The CVODE component is an important part of our work on the Zélus programming language.

We also developed a new interface for the IDA component, which we have started to use in our experiments with DAEs (Modelica).

We plan to develop an interface for the remaining KINSOL component over the next three months and then to release the entire library under an open-source license.

## 5.6. Zélus

**Participants:** Timothy Bourke, Marc Pouzet [contact].

Zélus is a new programming language for hybrid system modeling. It is based on a synchronous language but extends it with Ordinary Differential Equations (ODEs) to model continuous-time behabiors. It allows for combining arbitrarily data-flow equations, hierarchical automata and ODEs. The language keeps all the fundamental features of synchronous languages: the compiler statically ensure the absence of deadlocks and critical races; it is able to generate statically scheduled code running in bounded time and space and a type-system is used to distinguish discrete and logical-time signals from continuous-time ones. The ability to combines those features with ODEs made the language usable both for programming discrete controllers and their physical environment.

The Zélus implementation has two main parts: a compiler that transforms Zélus programs into OCaml programs and a runtime library that orchestrates compiled programs and numeric solvers. The runtime can use the Sundials numeric solver, or custom implementations of well-known algorithms for numerically approximating continuous dynamics.

This year we reimplemented several basic numeric solver algorithms after a careful analysis of the Simulink versions together with the binding to SUNDIALS CVODE. This was necessary to enable detailed comparsions between our tool and Simulink (the de facto industrial standard in this domain). We also improved the algorithm for zero-crossing detection, simplified and streamlined the back-end interface.

We developed several new examples to aid in the development, debugging, and dissemination of our work together with various talks and demonstrations. These included a simple backhoe model (which served as a introducing example in the HSCC paper [12]), an adaptive control example from Astrom and Wittenmark's text, and a model of Zeno behaviour based on a zig-zagging object (presented at Synchron).

Zélus has been released officially in 2013 with several complete documented examples on http://zelus.di.ens.fr. An important software development has been done in the compiler internals during year 2013: a new *causality analysis* has been designed and implemented and a new back-end to generate efficient sequential code for both the discrete step and the continuous step.

## 5.7. GCC

**Participants:** Albert Cohen [contact], Tobias Grosser, Antoniu Pop, Feng Li, Riyadh Baghdadi, Nhat Minh Lê.

Compilation, optimizing compilation, parallel data-flow programming automatic parallelization, polyhedral compilation. http://gcc.gnu.org

Licence: GPLv3+ and LGPLv3+

The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these languages (libstdc++, libgcj,...). GCC was originally written as the compiler for the GNU operating system. The GNU system was developed to be 100% free software, free in the sense that it respects the user's freedom.

PARKAS contributes to the polyhedral compilation framework, also known as Graphite. We also distribute an experimental branch for a stream-programming extension of OpenMP called OpenStream (used in numerous research activites and grants). This effort borrows key design elements to synchronous data-flow languages.

Tobias Grosser is one of main contributors of the Graphite optimization pass of GCC.

## 5.8. isl

**Participants:** Sven Verdoolaege [contact], Tobias Grosser, Albert Cohen.

Presburger arithmetic, integer linear programming, polyhedral library, automatic parallelization, polyhedral compilation. http://freshmeat.net/projects/isl

Licence: MIT

isl is a library for manipulating sets and relations of integer points bounded by linear constraints. Supported operations on sets include intersection, union, set difference, emptiness check, convex hull, (integer) affine hull, integer projection, transitive closure (and over-approximation), computing the lexicographic minimum using parametric integer programming. It includes an ILP solver based on generalized basis reduction, and a new polyhedral code generator. isl also supports affine transformations for polyhedral compilation, and increasingly abstract representations to model source and intermediate code in a polyhedral framework.

isl has become the de-facto standard for every recent polyhedral compilation project. Thanks to a license change from LGPL to MIT, its adoption is also picking up in industry.

## 5.9. ppcg

**Participants:** Sven Verdoolaege [contact], Tobias Grosser, Riyadh Baghdadi, Albert Cohen.

Presburger arithmetic, integer linear programming, polyhedral library, automatic parallelization, polyhedral compilation. http://freshmeat.net/projects/ppcg

Licence: MIT

More tools are being developed, based on isl. PPCG is our source-to-source research tool for automatic parallelization in the polyhedral model. It serves as a test bed for many compilation algorithms and heuristics published by our group, and is currently the best automatic parallelizer for CUDA and OpenCL (on the Polybench suite).

## 5.10. Tool support for the working semanticist

**Participant:** Francesco Zappa Nardelli [contact].

Languages, semantics, tool support, theorem prouvers.

We are working on tools to support large scale semantic definitions, for programming languages and architecture specifications. For that we develop two complementary tools, Ott and Lem.

Ott is a tool for writing definitions of programming languages and calculi. It takes as input a definition of a language syntax and semantics, in a concise and readable ASCII notation that is close to what one would write in informal mathematics. It generates output:

1. a LaTeX source file that defines commands to build a typeset version of the definition;
2. a Coq version of the definition;
3. an Isabelle version of the definition; and
4. a HOL version of the definition.

Additionally, it can be run as a filter, taking a LaTeX/Coq/Isabelle/HOL source file with embedded (symbolic) terms of the defined language, parsing them and replacing them by typeset terms.

The main goal of the Ott tool is to support work on large programming language definitions, where the scale makes it hard to keep a definition internally consistent, and to keep a tight correspondence between a definition and implementations. We also wish to ease rapid prototyping work with smaller calculi, and to make it easier to exchange definitions and definition fragments between groups. The theorem-prover backends should enable a smooth transition between use of informal and formal mathematics.

Lem is a lightweight tool for writing, managing, and publishing large scale semantic definitions. It is also intended as an intermediate language for generating definitions from domain-specific tools, and for porting definitions between interactive theorem proving systems (such as Coq, HOL4, and Isabelle). As such it is a complementary tool to Ott. Lem resembles a pure subset of Objective Caml, supporting typical functional programming constructs, including top-level parametric polymorphism, datatypes, records, higher-order functions, and pattern matching. It also supports common logical mechanisms including list and set comprehensions, universal and existential quantifiers, and inductively defined relations. From this, Lem generates OCaml, HOL4, Coq, and Isabelle code.

In collaboration with Peter Sewell (Cambridge University) and Scott Owens (University of Kent).

The current version of Ott is about 30000 lines of OCaml. The tool is available from http://moscova.inria.fr/~zappa/software/ott (BSD licence). It is widely used in the scientific community. In 2013 we implemented several bug-fixes, we kept the theorem prouver backends up-to date with the prover evolution, and we have been working toward a closer integration with the Lem tool.

The development version of Lem is available from http://www.cs.kent.ac.uk/people/staff/sao/lem/.

## 5.11. Cmmtest: a tool for hunting concurrency compiler bugs

**Participants:** Francesco Zappa Nardelli [contact], Robin Morisset, Pankaj More, Anirudh Kumar, Pankaj Prateek Kewalramani, Pejman Attar.

Languages, concurrency, memory models, C11/C++11, compiler, bugs.

The cmmtest tool performs random testing of C and C++ compilers against the C11/C++11 memory model. A test case is any well-defined, sequential C program; for each test case, cmmtest:

1. compiles the program using the compiler and compiler optimisations that are being tested;
2. runs the compiled program in an instrumented execution environment that logs all memory accesses to global variables and synchronisations;
3. compares the recorded trace with a reference trace for the same program, checking if the recorded trace can be obtained from the reference trace by valid eliminations, reorderings and introductions.

Cmmtest identified several mistaken write introductions and other unexpected behaviours in the latest release of the gcc compiler. These have been promptly fixed by the gcc developers.

Cmmtest is available from http://www.di.ens.fr/~zappa/projects/cmmtest/ and a list of bugs reported thanks to cmmtest is available from http://www.di.ens.fr/~zappa/projects/cmmtest/gcc-bugs.html.

# 6. New Results

## 6.1. Reactive Programming

**Participants:** Guillaume Baudart, Louis Mandel, Cédric Pasteur, Marc Pouzet.

ReactiveML is an extension of OCaml with synchronous concurrency, based on synchronous parallel composition and broadcast of signals. The goal is to provide a general model of deterministic concurrency inside a general purpose functional language to program reactive systems. It is particularly suited to program discrete simulations, for instance of sensor networks.

One of the current focus of the research is being able to simulate huge systems, composed of millions of agents, by extending the current purely sequential implementation in order to be able to take advantage of multi-core and distributed architectures. This goal has led to the introduction of a new programming construct, *reactive domain*, which allows to define local time scales. These domains help for the distribution of the code but also increase the expressiveness of the language. In particular, it allows to do time refinement. A paper on this new construct and the related static analysis has been published [20]. An extended version is under submission.

We continued the work on a new reactivity analysis which ensures that a process can not prevent the other ones to from executing. This analysis has published in [19]. An English version is under submission.

The runtime of ReactiveML has been cleanup and a multi-threaded implementation has been developed. A paper describing this new implementation will be published in [27].

All these novelties has been described precisely in the PhD thesis of Cédric Pasteur [1].

During the year, ReactiveML has also bee applied to *mixed music*. Mixed music is about live musicians interacting with electronic parts which are controlled by a computer during the performance. It allows composers to use and combine traditional instruments with complex synthesized sounds and other electronic devices. There are several languages dedicated to the writing of mixed music scores. Among them, the Antescofo language coupled with an advanced score follower allows a composer to manage the reactive aspects of musical performances: how electronic parts interact with a musician. However these domain specific languages do not offer the expressiveness of functional programming.

We defined a synchronous semantics for the core language of Antescofo and an alternative implementation based on an embedding inside ReactiveML [9]. The semantics reduces to a few rules, is mathematically precise and leads to an interpretor of only a few hundred lines. The efficiency of this interpretor compares well with that of the actual implementation: on all musical pieces we have tested, response times have been less than the reaction time of the human ear. Moreover, this approach offers to the composer recursion, higher order, inductive types, as well as a simple way to program complex reactive behaviors thanks to the synchronous model of concurrency on which ReactiveML is built [10].

## 6.2. $n$-Synchronous Languages

**Participants:** Albert Cohen, Adrien Guatto, Louis Mandel, Marc Pouzet.

Synchronous programming languages in the vein of Lustre were designed for critical real-time systems. They are, however, not that well adapted to embedded applications with more pressing computational needs, since the generated code will usually not contain loops or arrays.

An essential task of a Lustre compiler is to determine whether a program can be executed within bounded memory. This process is called the "clock calculus", and consists in mapping every item of each program stream to a logical date in a global, discrete time scale. For a given stream, the mapping itself is called a "clock", and is a strictly increasing function from stream positions to natural numbers representing ticks: two items cannot be computed at the same time. In practice, this function is represented as an infinite binary stream where the boolean $b_i$ denotes presence (or absence) in the corresponding data stream at the i-th instant.

In recent work, Guatto, Cohen, Mandel and Pouzet considered the extension of the Lustre and Lucid Synchrone clock calculus to allow computing several values instantaneously. This simple idea has a deep impact on all aspects of the language: - its denotational semantics has to account for bursts of values; - the clock calculus now features integers rather than booleans: each integer denotes the size of the burst at the corresponding instant; - causality analysis has to take bursts into account when rejecting self-referential programs; - the code generation process translates bursts to arrays and clocks to counted loops.

A prototype implementation exploiting this idea and generating C code with loops is underway and a paper describing the base of the clock calculus will be published [26].

This work extends nicely the n-synchronous model that introduced a way to compose streams which have *almost the same clock* and can be synchronized through the use of a finite buffer.

## 6.3. Mechanization of AODV loop freedom proof

**Participant:** Timothy Bourke.

The Ad hoc On demand Distance Vector (AODV) routing protocol is described in RFC3561. It allows the nodes in a Mobile Ad hoc Network (MANET) to know where to forward messages so that they eventually reach their destinations. The nodes of such networks are *reactive systems* that cooperate to provide a global service (the sending of messages from node to node) satisfying certain correctness properties (namely ‘loop freedom’—that messages are never sent in circles).

We have mechanized an existing formal but pen-and-paper proof of loop freedom of AODV in the interactive theorem prover Isabelle/HOL. While the process algebra model and the fine details of the original proof are quite formal, the structure of the proof is much less so. This necessitated the development of new framework elements and techniques in Isabelle. In particular, we adapted standard theory on inductive assertions to show invariants over individual reactive nodes and introduced machinery for assume/guarantee reasoning to lift these invariants to networks of communicating processes. While the original proof reasoned informally over traces, the mechanized proof is purely based on invariant reasoning, i.e., on reasoning over pairs of reachable states. Our combination of techniques works very well and is likely useful for modelling and verifying similar protocols in an interactive theorem prover.

We are currently finalising a paper describing this work for submission in January.

In collaboration with Peter Hofner (NICTA) and Robert J. van Glabbeek (UNSW/NICTA).

## 6.4. Hybrid Synchronous Languages

**Participants:** Timothy Bourke, Jun Inoue, Antoine Madet, Marc Pouzet.

During year 2013, we mainly worked on three directions: (a) the treatment of DAEs; (b) the design and implementation of a causality analysis for hybrid systems modelers; (c) the study of numerical techniques for *non-smooth dynamical systems*.

DAEs As part of our participation in the European project MODRIO and SYS2SOFT projects, we have been developing a prototype for simulating DAE (Differential-Algebraic Equations) systems. DAEs are the basis of the language Modelica and their interaction with discrete features — in particular the novel ones introduced in 2012, like hierarchical automata and clocks — raise difficult semantical and compilation issues. The goal is to precisely define the interaction between synchronous programming constructs and DAEs, in term of semantics and compilation. One strong difficulty at the moment is that existing techniques (index reduction, dymmy derivative) are not modular and force, either to (a) write an interpretor where index reduction is done dynamically every time a mode change occurs or (b) statically enumerate all the modes, performing index reduction for every of those. While the first technique is too slow in practice (and it is not used in the most advanced Modelica compiler), the second one may explode in practice (putting $n$ two-state automata in parallel lead to $2^n$ states to be enumerated). During year 2013, we have investigated a new approach for index reduction.

Work to-date has focused on implementing standard algorithms from the literature (notably Pantelides, Dummy Derivatives, Dynamic State Selection). Despite the importance of these algorithms to tools like Modelica, we found that important implementation details and "tricks" are not always well documented.

This work is developed hand-in-hand with the interface to the Sundials IDA solver.

Causality Analysis We have designed a causality analysis for a language that mix stream equations, hierarchical automata and ODEs and implemented it in the Zélus compiler. Its purpose is to give a sufficient condition for a hybrid program can be turned into statically scheduled code. Moreover, the analysis ensures that absence of discontinuities outside of declared zero-crossing events. This result is novel and the proof deeply rely on the use of *non standard analysis* introduced in our previous works. This new result has been accepted for publication at HSCC 2014.

Non Smooth Dynamical Systems In parallel, we collaborate with Bernard Brogliato and Vincent Acary (Inria team BIBOP, Grenoble) on non smooth dynamical systems. Beside general-purpose techniques for solving DAEs and implemented in Modelica compilers, there exist dedicated methods for systems with a lot of discontinuities and contacts (in mechanical system, electrical analogous circuits, etc.). They are far more efficient and numerically accurate than general-purpose techniques when the number of contact is important (e.g., transient in electrical circuits, a bag of marbles). They are based on a time stepping execution and do not have to stop at every zero-crossing event. The combination of those techniques with event detection ones (as used in the Simulink tool) is largely unknown. We are currently inverstigating the extension of our previous work to take Brogliato and

Acary techniques into account. This is a novel but promising direction of research for the year to come.

In this research activity, we develop the new language Zélus used as a laboratory for experimenting novel programming constructs and compilation techniques. It serves to illustrate our research as Lucid Synchrone did in the past.

In collaboration with Benoit Caillaud and Albert Benveniste of the Inria HYCOMES team.

## 6.5. Fidelity in Real-Time Programming

**Participants:** Timothy Bourke, Guillaume Baudart.

We are close to completing a careful analysis of literature related to the quasi-synchronous model for real-time, distributed systems. We have extended existing results by increasing their precision, providing detailed proofs, and simplifying protocol descriptions. The work to-date is documented in a draft document which we expect will eventually become a technical report or journal article.

Quasi-synchronous architectures, sometimes termed Loosely Time-Triggered Architectures (LTTAs), are ubiquitious in the development of distributed, real-time systems. They represent a broad class of systems whose modelling and programming mixes elements of discrete time, physical time, and a notion of approximation. We expect that addressing these elements—in the Zélus programming language—will lead to insights and advances in a broader ambition to program in physical time.

## 6.6. A theory of safe optimisations in the C11/C++11 memory model and applications to compiler testing

**Participants:** Francesco Zappa Nardelli, Robin Morisset.

Compilers sometimes generate correct sequential code but break the concurrency memory model of the programming language: these subtle compiler bugs are observable only when the miscompiled functions interact with concurrent contexts, making them particularly hard to detect. In this work we design a strategy to reduce the hard problem of hunting concurrency compiler bugs to differential testing of sequential code and build a tool that puts this strategy to work. Our first contribution is a theory of sound optimisations in the C11/C++11 memory model, covering most of the optimisations we have observed in real compilers and validating the claim that common compiler optisations are sound in the C11/C++11 memory model. Our second contribution is to show how, building on this theory, concurrency compiler bugs can be identified by comparing the memory trace of compiled code against a reference memory trace for the source code. Our tool identified several mistaken write introductions and other unexpected behaviours in the latest release of the gcc compiler.

A paper on this work has been accepted in [22].

## 6.7. A verified compiler for relaxed-memory concurrency

**Participant:** Francesco Zappa Nardelli.

We studied the semantic design and verified compilation of a C-like programming language for concurrent shared-memory computation above x86 multiprocessors. The design of such a language is made surprisingly subtle by several factors: the relaxed-memory behaviour of the hardware, the effects of compiler optimisation on concurrent code, the need to support high-performance concurrent algorithms, and the desire for a reasonably simple programming model. In turn, this complexity makes verified (or verifying) compilation both essential and challenging. This project started in 2010. In 2013 an article, describing the correctness proof of all the phases of our CompCertTSO compiler (including experimental fence eliminations), appeared in the Journal of the ACM [7].

In collaboration with Jaroslav Sevcik (U. Cambridge), Viktor Vafeiadis (MPI-SWS), Suresh Jagannathan (Purdue U.), Peter Sewell (U. Cambridge).

## 6.8. Language design on top of JavaScript

**Participant:** Francesco Zappa Nardelli.

This research project aims at improving the design of the JavaScript language. In [23] we present a security infrastructure which allows users and content providers to specify access control policies over subsets of a JavaScript program by leveraging the con- cept of delimited histories with revocation. We implement our proposal in WebKit and evaluate it with three policies on 50 widely used websites with no changes to their JavaScript code and report performance overheads and violations. In [32] we propose a typed extension of JavaScript combining dynamic types, concrete types and like types to let developers pick the level of guarantee that is appropriate for their code. We have implemented our type system and we report on performance and software engineering benefits.

With Gregor Richards and Jan Vitek (Purdue University).

## 6.9. Tiling for iterated stencils

**Participants:** Tobias Grosser, Sven Verdoolaege, Albert Cohen.

Time-tiling is necessary for the efficient execution of iterative stencil computations. Classical hyper-rectangular tiles cannot be used due to the combination of backward and forward dependences along space dimensions. Existing techniques trade temporal data reuse for inefficiencies in other areas, such as load imbalance, redundant computations, or increased control flow overhead, therefore making it challenging for use with GPUs.

We proposed a time-tiling method for iterative stencil computations on GPUs. Our method is the first tiling algorithm solving the following constraints simultaneously: it does not involve redundant computations, it favors coalesced global-memory accesses, data reuse in local/shared-memory or cache, avoidance of thread divergence, and concurrency, combining hexagonal tile shapes along the time and one spatial dimension with classical tiling along the other spatial dimensions. Hexagonal tiles expose multi-level parallelism as well as data reuse. Experimental results demonstrate significant performance improvements over existing stencil compilers.

Part of this work also involved our colleagues from the POLYFLOW associate-team at the Indian Institute of Science, Bangalore, India.

## 6.10. Compilation for scalable on-chip parallelism

**Participants:** Antoniu Pop, Feng Li, Sven Verdoolaege, Govindarajan Ramaswamy, Albert Cohen.

Task-parallel programming models are getting increasingly popular. Many of them provide expressive mechanisms for inter-task synchronization. For example, OpenMP 4.0 will integrate data-driven execution semantics derived from the StarSs research language. Compared to data-parallel and fork-join models of parallelism, the advanced features being introduced into task-parallel models in turn enable improved scalability through load balancing, memory latency mitigation, mitigation of the pressure on memory bandwidth, and as a side effect, reduced power consumption.

We developed a systematic approach to compile a loop nest into concurrent, dependent tasks. We formulated a partitioning scheme based on the tile-to-tile dependences, represented as affine polyhedra. This scheme ensures at compilation time that tasks belonging to the same class have the same, fully explicit incoming and outgoing dependence patterns. This alleviates the burden of a full-blown dependence resolver to track the readiness of tasks at run time. We evaluated our approach and algorithms in the PPCG compiler, targeting OpenStream, our experimental data-flow task-parallel language with explicit inter-task dependences and a lightweight runtime. Experimental results demonstrate the effectiveness of the approach.

Part of this work also involved our colleagues from the POLYFLOW associate-team at the Indian Institute of Science, Bangalore, India.

## 6.11. Correct and efficient runtime systems

**Participants:** Nhat Minh Lê, Robin Morisset, Adrien Guatto, Antoniu Pop, Francesco Zappa Nardelli, Albert Cohen.

User-space scheduling and concurrent first-in first-out queues are two essential building blocks of parallel programming runtimes. They are, however, rarely used together since typical schedulers are oblivious to the ordering constraints introduced by buffered communication.

Chase and Lev's concurrent deque is a key data structure in shared-memory parallel programming and plays an essential role in work-stealing schedulers. We provided the first correctness proof of an optimized implementation of Chase and Lev's deque on top of the POWER and ARM architectures: these provide very relaxed memory models, which we exploit to improve performance but considerably complicate the reasoning. We also studied an optimized x86 and a portable C11 implementation, conducting systematic experiments to evaluate the impact of memory barrier optimizations. Our results demonstrate the benefits of hand tuning the deque code when running on top of relaxed memory models.

Based on this early success, we started working on a more global solution using a new lock-free algorithm for stalling and waking-up tasks in a user-space scheduler according to changes in the state of the corresponding queues. The algorithm is portable and correct, since it is written and proven against the C11 memory model. We showed through experiments that it can serve as a keystone to efficient parallel runtime systems.

These efforts underline the parallelizing compilation research for $n$-synchronous languages, and the scalable parallel execution of OpenStream.

## 6.12. Checking Synchronous Compiler Correctness

**Participants:** Francesco Zappa Nardelli, Guillaume Chelfi, Marc Pouzet.

During year 2013, we have worked on the use of formal verification of compilation steps in the compiler of a Lustre-like synchronous language. Two main directions has been taken:

- The use of SMT-based $k$-induction techniques to verify the correctness of the successive steps of a synchronous compiler. We used the tool KIND developed by Cesare Tinelli (Iowa state Univ.) and applied it to the Heptagon compiler. The compiler does several source-to-source transformations upto sequential code and KIND was used to verify the equivalence between those successive steps. We came to the conclusion that for most programs, equivalence checking fails unless extra traceability information is added by the compiler.
- The development of a dedicated verification technique to prove the equivalence between a Lustre program and its sequential implementation. We plan to pursue this work during year 2014. Cesare Tinelli will be visiting professor for a month during June 2014.

# 7. Bilateral Contracts and Grants with Industry

## 7.1. Bilateral Contracts with Industry

- Kalray 20K grant including the donation of an MPPA Developer workstation (with MPPA 256 accelerator) and support for a short-term research project (2 months of postdoc).
- Google Doctoral Fellowships of Tobias Grosser and Robin Morisset.

# 8. Partnerships and Cooperations

## 8.1. National Initiatives

### *8.1.1. ANR*

ANR WMC project (program "jeunes chercheuses, jeunes chercheurs"), 2012–2016, 200 Keuros. F. Zappa Nardelli is the main investigator.

ANR Boole project (program “action blanche”), 2009-2014.

ANR Partout (program “defis”), 2009-2012. Louis Mandel and Marc Pouzet.

ANR CAFEIN, 2013-2015. Marc Pouzet.

Action d’envergure Synchronics, 2008-2012. The action was driven by Alain Girault (Inria, PopArt, Grenoble) and Marc Pouzet (Inria, Parkas, Paris-Rocquencourt), to focus on “langages for embedded systems”. This has been instrumental in driving our new research on hybrid system modelers.

### *8.1.2. Competitivity Clusters*

FUI project OpenGPU, 2008–2012.

### *8.1.3. Investissements d’avenir*

Sys2Soft contract (Briques Génériques du Logiciel Embarqué). Partenaire principal: Dassault-Systèmes, etc. Inria contacts are Benoit Caillaud (HYCOMES, Rennes) and Marc Pouzet (PARKAS, Paris).

ManycoreLabs contract (Briques Génériques du Logiciel Embarqué). Partenaire principal: Kalray. Inria contacts are Albert Cohen (PARKAS, Paris) and Alain Darte (COMPSYS, Lyon).

### *8.1.4. Others*

Marc Pouzet is scientific advisor for the Esterel-Technologies/ANSYS company.

## 8.2. European Initiatives

### *8.2.1. FP7 Projects*

#### *8.2.1.1. TETRACOM*

Type: CAPACITIES

Defi: Alternative Paths to Components and Systems

Instrument: Coordination and Support Action

Objectif: Advanced Computing, embedded and Control systems

Duration: September 2013 – August 2016

Coordinator: Rainer Leupers

Partner: RWTH Aachen (Germany)

Inria contact: Albert Cohen

Abstract: coordination action to support bilateral technology transfer partnerships (TTPs); prototype of future H2020 transfer instruments.

#### *8.2.1.2. COPCAMS*

Type: ARTEMIS

Defi: Alternative Paths to Components and Systems

Instrument: ASP

Objectif: NC

Duration: April 2013 – March 2016

Coordinator: Christian Fabre

Partner: CEA Leti (Grenoble)

Inria contact: Albert Cohen

Abstract: cognitive/smart cameras enabled by hardware accelerators, including manycore processors (STHORM platform of ST) and GPUs.

### *8.2.2. Collaborations in European Programs, except FP7*

*8.2.2.1. MODRIO*

Duration: December 2012 - December 2014

Coordinator: EDF

Partner: Dassault-Systèmes, EDF, Institut Francais du Pétrole, DLR (Munich, Germany), LMS-Imagine, Inria.

Inria contact: Benoit Caillaud (HYCOMES, Rennes); Marc Pouzet (PARKAS, Paris)

## 8.3. International Initiatives

### *8.3.1. Inria Associate Teams*

*8.3.1.1. POLYFLOW*

Title: Polyhedral Compilation for Data-Flow Programming Languages

Inria principal investigator: Albert Cohen

International Partner (Institution - Laboratory - Researcher):

IISc Bangalore (India) - Department of Computer Science and Automation - Albert Cohen

Duration: 2013 - 2016

See also: http://polyflow.gforge.inria.fr

Polyhedral techniques for program transformation are now used in several proprietary and open source compilers. However, most of the research on polyhedral compilation has focused on imperative languages such as C, where computation is specified in terms of statements with zero or more nested loops and other control structures around them. Graphical data-flow languages, where there is no notion of statements or a schedule specifying their relative execution order, have so far not been studied using a powerful transformation or optimization approach. These languages are extremely popular in system analysis, modeling and design, in embedded reactive control. They also underline the construction of many domain-specific languages and compiler intermediate representations. The copy and execution semantics of data-flow languages impose a different set of challenges. We plan to bridge this gap by studying techniques that could enable extraction of a polyhedral representation from data-flow programs, transform them, and synthesize them from their equivalent polyhedral representation.

## 8.4. International Research Visitors

### *8.4.1. Visits of International Scientists*

We have regular invited professors in the PARKAS team:

- In 2012, one month (June/July), Prof. Stephen Edwards (Columbia Univ., New York, USA).
- In 2013, one month (June), Prof. Mary Sheeran from (Chalmers Univ., Sweden).

*8.4.1.1. Internships*

Pankaj Prateek, Anirudh Kumar, and Pankaj More, students at IIT Kanpur, India, worked in the Parkas team under the supervision of Francesco Zappa Nardelli from 4th May, 2013 to 23 July, 2013.

Guillaume Chelfi, student at Telecom Paris and the MPRI program, under the supervision of Francesco Zappa Nardelli and Marc Pouzet, from 1st of March, 2013, to 31st July, 2013. Guillaume Chelfi worked on the formal verification of the translation of synchronous programs to sequential code.

Louis Mandel supervised the 5-months MPRI Internship of Louis Jachiet from April to August. Louis Jachiet worked on the static scheduling of ReactiveML programs.

Albert Cohen supervised the 3-months Internship of Vincent Thiberville, 3rd year student at École Polytechnique, from April to June. Vincent conducted experimental studies and proposed enhanced methods to support array-based computations in the Heptagon synchronous language.

### *8.4.2. Visits to International Teams*

October, Louis Mandel spent 2 weeks in the team of Vijay Saraswat at IBM T.J. Watson. He worked on the type system of the X10 language.

# 9. Dissemination

## 9.1. Scientific Animation

- Albert Cohen was the program chair of CC 2014, the TPC chair of the DAC 2013 and 2014 ESS1 subcommittees, and the co-Program Chair of the APPT 2013 bi-annual Symposium on Advanced Parallel Processing Technology. Albert Cohen was also a member of the PC of PLDI 2014, and a member of the ERC of ASPLOS 2014, PPoPP 2014 and ICS 2014. Albert Cohen also participated to the PC of the IMPACT and HiRES workshops associated with HiPEAC 2014.
- Albert Cohen is an associate editor of ACM TACO and IJPP (Springer).
- Albert Cohen will be the general chair of PPoPP 2015.
- Albert Cohen was the sponsor chair for the HiPEAC 2013 and HiPEAC 2014 conference, and will serve as the exhibit and sponsor chair for HiPEAC 2015.
- Marc Pouzet was a member of the PC of DAC 2014, AFADL 2014, MSR 2013, RTNS 2013, DATE 2013.
- Marc Pouzet manages with Catherine Dubois (ENSIIE, Evry, France) the GDR (“Groupe de Recherche”) TLP (“Types, Langages et Preuves”) du CNRS (“Centre National de Recherche Scientifique”). Two one-day seminars are organised every year.

## 9.2. Teaching - Supervision - Juries

### *9.2.1. Teaching*

Licence: T. Bourke & J. Vuillemin, “Digital Systems”, 64h, L3, Ecole normale supérieure, France

Licence: L. Mandel, “Systèmes”, 42h, L3, Université Pars-Sud 11, France

Licence: L. Mandel & M. Pouzet, “Systèmes et réseaux”, 24h+24h, L3, Ecole normale supérieure, France

Licence: L. Mandel, “Langages de programmation et compilation”, 24h, L3, Ecole normale supérieure, France

Master: L. Mandel & M. Pouzet, “Synchronous Systems”, 8h+16h, M2, MPRI: Ecole normale supérieure and Université Paris Diderot, France

Master: A. Cohen & F. Zappa Nardelli, “Semantics, languages and algorithms for multicore programming”, 9h+14h, M2, MPRI: Ecole normale supérieure and Université Paris Diderot, France

Licence: “Components of a Computing System Introduction to Computer Architecture and Operating Systems” (L3), A. Cohen (44h), École Polytechnique, France

Master 1 École Polytechnique: “Operating Systems Principles and Programming” (M1), A. Cohen (38h), École Polytechnique, France

Marc Pouzet is supervising the national entry exam in computer science for École normale supérieure.

Marc Pouzet is director of studies (“Directeur des études”) for the CS department of École normale supérieure.

### *9.2.2. Supervision*

PhD : Léonard Gérard, Programmer le parallélisme avec des futures en Heptagon un langage synchrone flot de données et étude des réseaux de Kahn en vue d'une compilation synchrone, Université Paris-Sud 11, Orsay. Soutenue le 25 septembre 2013, au LRI, à Orsay.

PhD : Cédric Pasteur, Raffinement temporel et exécution parallèle dans un langage synchrone fonctionnel, Université Pierre et Marie Curie (UPMC), soutenue le 26 novembre 2013, au Collège de France. Encadrants: Louis Mandel et Marc Pouzet.

PhD in progress : Guillaume Baudart, Real-time fidelity in Quasi-synchronous Systems, Start: 1/10/2013, Timothy Bourke and Marc Pouzet

PhD in progress : Robin Morisset, Compiler Optimisations and Concurrency, 1/10/2013, F. Zappa Nardelli

### *9.2.3. Juries*

- Albert Cohen was the president of the Habilitation Thesis committee of Fabien Coelho, MINES ParisTech.
- Albert Cohen was the president of the PhD thesis committee of Bruno Bodin, UPMC (CIFRE Kalray).
- Albert Cohen was a reviewer for the PhD thesis of Martin Schindewolf at the Karlsruhe Institute of Technology.
- Albert Cohen was a reviewer for the PhD thesis of Daniel Cordes at TU Dortmund.
- Albert Cohen was a reviewer for the PhD thesis of Yuriy Kashnikov, UVSQ.
- Albert Cohen was an examiner in the PhD thesis committee of Thomas Preud'Homme, UPMC.
- Marc Pouzet was a reviewer for the PhD thesis of Boris Golden, École Polytechnique.
- Marc Pouzet was a reviewer for the PhD thesis of Gideon Smeding, Université de Grenoble.
- Albert Cohen was a member of a hiring committee for professors at the University of Strasbourg.
- Albert Cohen was a member of a hiring committee for assistant professors ("maître de conférences") at the University Claude Bernard de Lyon.

# 10. Bibliography

## Publications of the year

### Doctoral Dissertations and Habilitation Theses

[1] C. PASTEUR. , *Raffinement temporel et exécution parallèle dans un langage synchrone fonctionnel*, Université Pierre et Marie Curie - Paris VI, November 2013, http://hal.inria.fr/tel-00934919

### Articles in International Peer-Reviewed Journals

[2] R. BAGHDADI, A. COHEN, S. VERDOOLAEGE, K. TRIFUNOVIĆ. *Improved Loop Tiling based on the Removal of Spurious False Dependences*, in "ACM Transactions on Architecture and Code Optimization", 2013, vol. 9, n$^{o}$ 4, Selected for presentation at the HiPEAC 2013 Conf. [*DOI :* 10.1145/2400682.2400711], http://hal.inria.fr/hal-00786674

[3] T. BOURKE, A. SOWMYA. *Analyzing an Embedded Sensor with Timed Automata in Uppaal*, in "ACM Transactions in Embedded Computing Systems", December 2013, vol. 13, n$^{o}$ 3, pp. 44-1–44-26 [*DOI :* 10.1145/2539036.2539040], http://hal.inria.fr/hal-00909062

[4] B. DIOUF, C. HANTAŞ, A. COHEN, Ö. ÖZTURK, J. PALSBERG. *A Decoupled Local Memory Allocator*, in "ACM Transactions on Architecture and Code Optimization", 2013, vol. 9, n° 4, Selected for presentation at the HiPEAC 2013 Conf. [*DOI :* 10.1145/2400682.2400693], http://hal.inria.fr/hal-00786676

[5] E. PARK, J. CAVAZOS, L.-N. POUCHET, C. BASTOUL, A. COHEN, P. SADAYAPPAN. *Predictive Modeling in a Polyhedral Optimization Space*, in "International Journal of Parallel Programming", 2013, vol. 41, n° 5, pp. 704–750 [*DOI :* 10.1007/s10766-013-0241-1], http://hal.inria.fr/hal-00918653

[6] A. POP, A. COHEN. *OpenStream: Expressiveness and Data-Flow Compilation of OpenMP Streaming Programs*, in "ACM Transactions on Architecture and Code Optimization", 2013, vol. 9, n° 4, Selected for presentation at the HiPEAC 2013 Conf. [*DOI :* 10.1145/2400682.2400712], http://hal.inria.fr/hal-00786675

[7] J. SEVCIK, V. VAFEIADIS, F. ZAPPA NARDELLI, S. JAGANNATHAN, P. SEWELL. *CompCertTSO: A Verified Compiler for Relaxed-Memory Concurrency*, in "Journal of the ACM", 2013, vol. 60, n° 3, 22 p. [*DOI :* 10.1145/2487241.2487248], http://hal.inria.fr/hal-00909076

[8] S. VERDOOLAEGE, J. C. JUEGA, A. COHEN, J. I. GÓMEZ, C. TENLLADO, F. CATTHOOR. *Polyhedral Parallel Code Generation for CUDA*, in "ACM Transactions on Architecture and Code Optimization", 2013, vol. 9, n° 4, Selected for presentation at the HiPEAC 2013 Conf. [*DOI :* 10.1145/2400682.2400713], http://hal.inria.fr/hal-00786677

### International Conferences with Proceedings

[9] G. BAUDART, L. MANDEL, F. JACQUEMARD, M. POUZET. *A Synchronous Embedding of Antescofo, a Domain-Specific Language for Interactive Mixed Music*, in "EMSOFT 2013 - 13th International Conference on Embedded Software", Montreal, Canada, September 2013, http://hal.inria.fr/hal-00850299

[10] G. BAUDART, L. MANDEL, M. POUZET. *Programming Mixed Music in ReactiveML*, in "FARM '13 - ACM SIGPLAN Workshop on Functional Art, Music, Modeling and Design", Boston, United States, ACM, September 2013, pp. 11-22 [*DOI :* 10.1145/2505341.2505344], http://hal.inria.fr/hal-00850294

[11] A. BENVENISTE, T. BOURKE, B. CAILLAUD, B. PAGANO, M. POUZET. *A Type-based Analysis of Causality Loops in Hybrid Systems Modelers*, in "17th International Conference on Hybrid Systems: Computation and Control (HSCC 2014)", Berlin, Germany, April 2014, http://hal.inria.fr/hal-00939947

[12] T. BOURKE, M. POUZET. *Zélus: A Synchronous Language with ODEs*, in "HSCC - 16th International Conference on Hybrid systems: computation and control", Philadelphia, United States, C. BELTA, F. IVANČIĆ (editors), ACM, April 2013, pp. 113-118 [*DOI :* 10.1145/2461328.2461348], http://hal.inria.fr/hal-00909029

[13] A. COHEN, T. GROSSER, P. H. J. KELLY, J. RAMANUJAM, P. SADAYAPPAN, S. VERDOOLAEGE. *Split Tiling for GPUs: Automatic Parallelization Using Trapezoidal Tiles to Reconcile Parallelism and Locality, avoiding Divergence and Load Imbalance*, in "GPGPU 6 - Sixth Workshop on General Purpose Processing Using GPUs", Houston, United States, 2013, http://hal.inria.fr/hal-00786812

[14] C. COUVREUR, P. AVASARE, F. BROEKAERT, A. COHEN. *Two-layer Run-Time Power Management for embedded heterogeneous multi-core platforms*, in "DATE 13 - Workshop on Designing for Embedded Parallel Computing Platforms: Architectures, Design Tools, and Applications", Grenoble, France, 2013, 1 p. , http://hal.inria.fr/hal-00911895

[15] B. DIOUF, A. COHEN, F. RASTELLO. *A Polynomial Spilling Heuristic: Layered Allocation*, in "CGO 2013 - International Symposium on Code Generation and Optimization", Shenzhen, China, IEEE, 2013 [*DOI :* 10.1109/CGO.2013.6495005], http://hal.inria.fr/hal-00911887

[16] T. GROSSER, A. COHEN, J. HOLEWINSKI, P. SADAYAPPAN, S. VERDOOLAEGE. *Hybrid Hexagonal/Classical Tiling for GPUs*, in "Intl. Symp. on Code Generation and Optimization (CGO)", Orlando, FL, United States, 2014, http://hal.inria.fr/hal-00911177

[17] I. LLOPARD, A. COHEN, C. FABRE, J. MARTIN, H.-P. CHARLES, C. BERNARD. *Code Generation for an Application-Specific VLIW Processor With Clustered, Addressable Register Files*, in "ODES'13 - 10th Workshop on Optimizations for DSP and Embedded Systems, associated with CGO", Shenzhen, China, ACM, 2013, pp. 11-19 [*DOI :* 10.1145/2443608.2443612], http://hal.inria.fr/hal-00911896

[18] N. M. LÊ, A. POP, A. COHEN, F. ZAPPA NARDELLI. *Correct and Efficient Work-Stealing for Weak Memory Models*, in "PPoPP '13 - Proceedings of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming", Shenzhen, China, February 2013, pp. 69-80 [*DOI :* 10.1145/2442516.2442524], http://hal.inria.fr/hal-00802885

[19] L. MANDEL, C. PASTEUR. *Réactivité des systèmes coopératifs : le cas de ReactiveML*, in "JFLA - Journées francophones des langages applicatifs", Aussois, France, D. POUS, C. TASSON (editors), Damien Pous and Christine Tasson, February 2013, http://hal.inria.fr/hal-00779789

[20] L. MANDEL, C. PASTEUR, M. POUZET. *Time Refinement in a Functional Synchronous Language*, in "PPDP '13 - 15th ACM SIGPLAN International Symposium on Principles and Practice of Declarative Programming", Madrid, Spain, ACM, September 2013, pp. 169-180 [*DOI :* 10.1145/2505879.2505904], http://hal.inria.fr/hal-00850290

[21] N. MINH, A. GUATTO, A. COHEN, A. POP. *Correct and Efficient Bounded FIFO Queues*, in "SBAC-PAD 2013 : International Symposium on Computer Architecture and High Performance Computing", Porto de Galinhas, Brazil, IEEE, 2013, http://hal.inria.fr/hal-00911893

[22] R. MORISSET, P. PAWAN, F. ZAPPA NARDELLI. *Compiler testing via a theory of sound optimisations in the C11/C++11 memory model*, in "PLDI'13 - 34th ACM SIGPLAN conference on Programming language design and implementation", Seattle, WA, United States, ACM, 2013, pp. 187-196 [*DOI :* 10.1145/2491956.2491967], http://hal.inria.fr/hal-00909083

[23] G. RICHARDS, C. HAMMER, F. ZAPPA NARDELLI, S. JAGANNATHAN, J. VITEK. *Flexible access control for JavaScript*, in "OOPSLA'13 - CM SIGPLAN international conference on Object oriented programming systems languages & applications", Indianapolis, IN, United States, ACM, 2013, pp. 305-322 [*DOI :* 10.1145/2509136.2509542], http://hal.inria.fr/hal-00909080

[24] R. UPADRASTA, A. COHEN. *Sub-polyhedral scheduling using (unit-)two-variable-per-inequality polyhedra*, in "POPL'13 - 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages", Rome, Italy, ACM, 2013, pp. 483-496 [*DOI :* 10.1145/2429069.2429127], http://hal.inria.fr/hal-00911888

[25] S. VERDOOLAEGE, S. GUELTON, T. GROSSER, A. COHEN. *Schedule Trees*, in "IMPACT - 4th Workshop on Polyhedral Compilation Techniques, associated with HiPEAC", Vienna, Austria, ACM, 2014, http://hal.inria.fr/hal-00911894

### National Conferences with Proceedings

[26] A. GUATTO, L. MANDEL. *Réseaux de Kahn à rafales et horloges entières*, in "JFLA 2014 - Vingt-cinquièmes Journées Francophones des Langages Applicatifs", Fréjus, France, January 2014, http://hal.inria.fr/hal-00919281

[27] L. MANDEL, C. PASTEUR. *Exécution efficace de programmes ReactiveML*, in "JFLA 2014 - Vingt-cinquièmes Journées Francophones des Langages Applicatifs", Fréjus, France, January 2014, http://hal.inria.fr/hal-00919271

### Research Reports

[28] T. GROSSER, S. VERDOOLAEGE, A. COHEN, P. SADAYAPPAN. , *The Promises of Hybrid Hexagonal/Classical Tiling for GPU*, Inria, July 2013, n$^{O}$ RR-8339, http://hal.inria.fr/hal-00848691

[29] N. M. LÊ, A. GUATTO, A. COHEN, A. POP. , *Correct and Efficient Bounded FIFO Queues*, Inria, September 2013, n$^{O}$ RR-8365, http://hal.inria.fr/hal-00862450

### Other Publications

[30] A. BENVENISTE, T. BOURKE, B. CAILLAUD, B. PAGANO, M. POUZET. , *A Type-Based Analysis of Causality Loops In Hybrid Systems Modelers*, December 2013, Deliverable D3.1_1 v 1.0 of the Sys2soft collaborative project "Physics Aware Software", http://hal.inria.fr/hal-00938866

[31] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. , *Semantics of multi-mode DAE systems*, August 2013, Deliverable D.4.1.1 of the ITEA2 Modrio collaborative project, http://hal.inria.fr/hal-00938891

[32] G. RICHARDS, F. ZAPPA NARDELLI, C. ROULEAU, J. VITEK. , *Types You Can Count On: Like Types for JavaScript*, 2013, Submitted, http://hal.inria.fr/hal-00909092

## References in notes

[33] D. BIERNACKI, J.-L. COLAÇO, G. HAMON, M. POUZET. *Clock-directed Modular Code Generation of Synchronous Data-flow Languages*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)", Tucson, Arizona, June 2008

[34] F. BOUSSINOT, R. DE SIMONE. *The SL synchronous language*, in "IEEE Transaction on Software Engineering", 1996

[35] P. CASPI, M. POUZET. *Synchronous Kahn Networks*, in "ACM SIGPLAN International Conference on Functional Programming (ICFP)", Philadelphia, Pensylvania, May 1996

[36] P. CASPI, M. POUZET. *A Co-iterative Characterization of Synchronous Stream Functions*, in "Coalgebraic Methods in Computer Science (CMCS'98)", Electronic Notes in Theoretical Computer Science, March 1998, Extended version available as a VERIMAG tech. report no. 97–07 at www.lri.fr/∼pouzet

[37] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *Synchroning Periodic Clocks*, in "ACM International Conference on Embedded Software (EMSOFT'05)", Jersey city, New Jersey, USA, September 2005

[38] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Synchronous Kahn Networks: a Relaxed Model of Synchrony for Real-Time Systems*, in "ACM International Conference on Principles of Programming Languages (POPL'06)", Charleston, South Carolina, USA, January 2006

[39] A. COHEN, S. GIRBAL, D. PARELLO, M. SIGLER, O. TEMAM, N. VASILACHE. *Facilitating the Search for Compositions of Program Transformations*, in "Intl. Conf. on Supercomputing (ICS'05)", Boston, Massachusetts, June 2005, pp. 151–160

[40] A. COHEN, S. GIRBAL, O. TEMAM. *A Polyhedral Approach to Ease the Composition of Program Transformations*, in "Euro-Par'04", Pisa, Italy, LNCS, Springer-Verlag, August 2004, n$^{o}$ 3149, pp. 292–303

[41] A. COHEN, L. MANDEL, F. PLATEAU, M. POUZET. *Abstraction of Clocks in Synchronous Data-flow Systems*, in "The Sixth ASIAN Symposium on Programming Languages and Systems (APLAS)", Bangalore, India, December 2008

[42] A. COHEN, L. MANDEL, F. PLATEAU, M. POUZET. , *Relaxing Synchronous Composition with Clock Abstraction*, 2009, Workshop on Hardware Design using Functional languages (HFL 09) - ETAPS, http://hal.inria.fr/hal-00645333

[43] J.-L. COLAÇO, G. HAMON, M. POUZET. *Mixing Signals and Modes in Synchronous Data-flow Systems*, in "ACM International Conference on Embedded Software (EMSOFT'06)", Seoul, South Korea, October 2006

[44] J.-L. COLAÇO, B. PAGANO, M. POUZET. *A Conservative Extension of Synchronous Data-flow with State Machines*, in "ACM International Conference on Embedded Software (EMSOFT'05)", Jersey city, New Jersey, USA, September 2005

[45] J.-L. COLAÇO, M. POUZET. *Clocks as First Class Abstract Types*, in "Third International Conference on Embedded Software (EMSOFT'03)", Philadelphia, Pennsylvania, USA, october 2003

[46] J.-L. COLAÇO, M. POUZET. *Type-based Initialization Analysis of a Synchronous Data-flow Language*, in "International Journal on Software Tools for Technology Transfer (STTT)", August 2004, vol. 6, n$^{o}$ 3, pp. 245–255

[47] P. CUOQ, M. POUZET. *Modular Causality in a Synchronous Stream Language*, in "European Symposium on Programming (ESOP'01)", Genova, Italy, April 2001

[48] P. FEAUTRIER. *Some Efficient Solutions to the Affine Scheduling Problem, Part II, multidimensional time*, in "Intl. J. of Parallel Programming", December 1992, vol. 21, n$^{o}$ 6, pp. 389-420, See also Part I, one dimensional time, 21(5):315–348

[49] A. GAMATIÉ, E. RUTTEN, H. YU, P. BOULET, J.-L. DEKEYSER. *Synchronous Modeling and Analysis of Data Intensive Applications*, in "EURASIP Journal on Embedded Systems", 2008

[50] S. GIRBAL, N. VASILACHE, C. BASTOUL, A. COHEN, D. PARELLO, M. SIGLER, O. TEMAM. *Semi-Automatic Composition of Loop Transformations for Deep Parallelism and Memory Hierarchies*, in "Intl. J. of Parallel Programming", June 2006, vol. 34, n$^{o}$ 3, pp. 261–317, Special issue on Microgrids

[51] T. GROSSER, A. GRÖSSLINGER, C. LENGAUER. *Polly - Performing Polyhedral Optimizations on a Low-Level Intermediate Representation*, in "Parallel Processing Letters", 2012, vol. 22, n$^{o}$ 4

[52] A.-C. GUILLOU, F. QUILLERÉ, P. QUINTON, S. RAJOPADHYE, T. RISSET. *Hardware Design Methodology with the Alpha Language*, in "FDL'01", Lyon, France, September 2001

[53] H. LEVERGE, C. MAURAS, P. QUINTON. *The* ALPHA *language and its use for the design of systolic arrays*, in "J. of VLSI Signal Processing", 1991, vol. 3, pp. 173–182

[54] L. MANDEL, F. BENBADIS. *Simulation of Mobile Ad hoc Network Protocols in ReactiveML*, in "Proceedings of Synchronous Languages, Applications, and Programming (SLAP'05)", Edinburgh, Scotland, Electronic Notes in Theoretical Computer Science, April 2005, Workshop ETAPS 2005

[55] L. MANDEL. , *Conception, Sémantique et Implantation de ReactiveML : un langage à la ML pour la programmation réactive*, Université Paris 6, 2006

[56] L. MANDEL, F. PLATEAU, M. POUZET. *Lucy-n: a n-Synchronous Extension of Lustre*, in "10th International Conference on Mathematics of Program Construction (MPC'10)", Manoir St-Castin, Québec, Canada, Springer LNCS, June 2010

[57] L. MANDEL, M. POUZET. *ReactiveML, a Reactive Extension to ML*, in "ACM International Conference on Principles and Practice of Declarative Programming (PPDP)", Lisboa, July 2005

[58] F. MARANINCHI, N. BERTHIER, O. BEZET, G. FUNCHAL. , *Writing Simulators with Synchronous Languages*, 2008, Synchron 2008: International Open Workshop on Synchronous Programming

[59] C. MIRANDA, A. POP, P. DUMONT, A. COHEN, M. DURANTON. *Erbium: A Deterministic, Concurrent Intermediate Representation to Map Data-Flow Tasks to Scalable, Persistent Streaming Processes*, in "Intl. Conf. on Compilers Architectures and Synthesis for Embedded Systems (CASES'10)", October 2010

[60] J.-B. NOTE, M. SHAND, J. VUILLEMIN. *Realtime video pixel matching*, in "International Conference on Field Programmable Logic and Applications", 2006, pp. 507 – 512

[61] J.-B. NOTE, J. VUILLEMIN. *Towards automatically compiling efficient FPGA hardware*, in "International Workshop on Design and Functional Languages", IEEE, 2007, pp. 115 – 124

[62] F. PLATEAU. , *Modèle n-synchrone pour la programmation de réseaux de Kahn à mémoire bornée*, Université Paris-Sud 11Orsay, France, 6 janvier 2010, https://www.lri.fr/~mandel/lucy-n/~plateau/these/

[63] S. POP, A. COHEN, C. BASTOUL, S. GIRBAL, G.-A. SILBER, N. VASILACHE. *GRAPHITE: Loop Optimizations Based on the Polyhedral Model for GCC*, in "Proc. of the 4þ GCC Developper's Summit", Ottawa, Canada, June 2006

[64] L.-N. POUCHET, C. BASTOUL, A. COHEN, J. CAVAZOS. *Iterative Optimization in the Polyhedral Model: Part II, Multidimensional Time*, in "ACM Conf. on Programming Language Design and Implementation (PLDI'08)", Tucson, Arizona, June 2008

[65] L.-N. POUCHET, C. BASTOUL, A. COHEN, N. VASILACHE. *Iterative Optimization in the Polyhedral Model: Part I, One-Dimensional Time*, in "Intl. Symp. on Code Generation and Optimization (CGO'07)", San Jose, California, March 2007

[66] L.-N. POUCHET, U. BONDHUGULA, C. BASTOUL, A. COHEN, J. RAMANUJAM, P. SADAYAPPAN. *Combined Iterative and Model-driven Optimization in an Automatic Parallelization Framework*, in "ACM Supercomputing Conf. (SC'10)", New Orleans, Lousiana, November 2010, 11 p.

[67] P. RAYMOND, Y. ROUX, E. JAHIER. *Lutin: a language for specifying and executing reactive scenarios*, in "EURASIP Journal on Embedded Systems", 2008, vol. 2008, Article ID 753821

[68] L. SAMPER, F. MARANINCHI, L. MOUNIER, L. MANDEL. *GLONEMO: Global and Accurate Formal Models for the Analysis of Ad hoc Sensor Networks*, in "Proceedings of the First International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06)", Nice, France, May 2006

[69] J. SOULA, P. MARQUET, J.-L. DEKEYSER, A. DEMEURE. *Compilation principle of a specification language dedicated to signal processing*, in "Intl. Conf. on Parallel Computing Technologies", Novosibirsk, Russia, LNCS, Springer-Verlag, September 2001, vol. 2127, pp. 358–370

[70] K. TRIFUNOVIĆ, A. COHEN, D. EDELSOHN, F. LI, T. GROSSER, H. JAGASIA, R. LADELSKI, S. POP, J. SJÖDIN, R. UPADRASTA. *GRAPHITE Two Years After: First Lessons Learned From Real-World Polyhedral Compilation*, in "GCC Research Opportunities Workshop (GROW'10)", Pisa, Italy, January 2010

[71] K. TRIFUNOVIĆ, D. NUZMAN, A. COHEN, A. ZAKS, I. ROSEN. *Polyhedral-Model Guided Loop-Nest Auto-Vectorization*, in "Parallel Architectures and Compilation Techniques (PACT'09)", Raleigh, North Carolina, September 2009

[72] J. VUILLEMIN. , *On Circuits and Numbers*, Digital, Paris Research Laboratory, 1993