IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2012

Project-Team PARKAS

Parallélisme de Kahn Synchrone

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

Address : Département d'Informatique, École Normale Supérieure, 45 rue d'Ulm, 75005 Paris.

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

1. Members

Research Scientists

Albert Cohen [Senior Researcher Inria, HdR]
Francesco Zappa Nardelli [Junior Researcher Inria, since September 2012]
Timothy Bourke [Starting Researcher]

Faculty Members

Marc Pouzet [Team leader, ENS, Professor at UPMC, HdR]
Louis Mandel [Université Paris Sud]
Jean Vuillemin [ENS, Director of DI until August 2011, HdR]

PhD Students

Cedric Auger [Université Paris Sud, MESR scholarship]
Léonard Gérard [Université Paris Sud, AMN]
Robin Morisset [ENS contract, from November 2012]
Cédric Pasteur [Université Paris Diderot, AMX]
Ramakrishna Upadrasta [Université Paris Sud, MESR scholarship]
Tobias Grosser [Google Doctoral Fellowship]
Feng Li [UPMC, ENS contract, FP7 grant]
Riyadh Baghdadi [UPMC, ENS contract, FP7 grant]
Nhat Minh Le [ENS, FP7 grant]
Adrien Guatto [ENS contract]
Jean-Yves Vet [UPMC, External student, CEA DAM]
Camille Gallet [UPMC, External student, CEA DAM]
Ivan Llopard [UPMC, External student, CEA LETI]

Post-Doctoral Fellows

Antoniu Pop [FP7 grant]
Boubacar Diouf [BGLE grant (investissements d'avenir)]
Jun Inoue [FP7 grant]
Mehdi Dogguy [ANR grant, until nov. 2012]
Sven Verdoolaege [ENS, 80%, FP7 grant]
Serge Guelton [ENS, 50%, FP7 grant]

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).

3. Scientific Foundations

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 [19], 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 [19], [29], initialization analysis [30] and causality analysis [31]). 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 [28], [27], and modular code generation [20], [17].

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]

[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/

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 [42], [38] 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 [18] 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 [43].

REACTIVEML has been used for simulating routing protocols in ad-hoc networks [37] and large scale sensor networks [53]. 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 [52]. 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

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 [21], [22]. 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 [57]. It was algorithmically expensive and limited to periodic systems. In [25], 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 [39], as was a formalization of the theory in COQ [26]. 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 [44].

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 [47] (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).

[5] http://rml.lri.fr/
[6] More precisely a subset of OCAML without objects or functors.

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 [32]. 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 [36], [35], [54], [33] 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 [23], [34], [50], [49]. 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 [24], [23], we ported these techniques to the GCC compiler [48], [56], [55], 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 [50], [49], [51]. 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).

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 [45], [46]. 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. Application Domains

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

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). 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.

5.2. ReactiveML

Participants: Mehdi Dogguy, 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://rml.lri.fr. 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).

In 2012, a new automatic build system for ReactiveML program based on ocamlbuild has been implemented. A new static analysis which checks that programs cooperate has been developed. A full ReactiveML toplevel compiled into JavaScript has been made available at http://rml.lri.fr/tryrml. The ReactiveML distribution has also been cleaned up.

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. 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, java and VHDL code.

Heptagon is jointly developed by Gwenael Delaval and Alain Girault from the Inria POP ART team (Grenoble).

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

Participants: 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 2012, a first version of the code generator has been distributed. The typing algorithms has been improved.

5.5. ML-Sundials

Participants: Timothy Bourke, Marc Pouzet [contact].

ML-Sundials library provides an Ocaml interface to the Sundials numerical suite [7] (version 2.4.0). This library is used for solving and initial value problem and includes a zero-crossing detection mechanism. Only the CVODE solver with serial nvectors is currently supported. The structure and naming conventions largely follow the original libraries, both for ease of reading the existing documentation and for converting existing source code, but several changes have been made for programming convenience, namely:

- solver sessions are configured through algebraic data types rather than through multiple function calls,
- error conditions are signalled by exceptions rather than return codes (including in user-supplied callback routines),
- closures (partial applications of higher-order functions) are used to share user data between callback routines, and,
- explicit free commands are not necessary nor provided since Ocaml is a garbage-collected language.

The library is in use in a new synchronous hybrid language we are currently developping.

5.6. GCC

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

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, parallel data-flow programming, and automatic parallelization to a data-flow runtime or architecture. This experiment borrows key design elements to synchronous data-flow languages.

Tobias Grosser is the maintainer of the Graphite optimization pass of GCC.

5.7. 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 also includes an ILP solver based on generalized basis reduction. isl also supports affine transformations for polyhedral compilation.

5.8. ppcg

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

[7] https://computation.llnl.gov/casc/sundials/main.html

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

Licence: LGPLv2.1+

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 bet for many algorithms and heuristics published by our group, and is currently the best automatic parallelizer for CUDA (on the Polybench suite).

5.9. Ott: tool support for the working semanticist

Participant: Francesco Zappa Nardelli [contact].

Languages, semantics, tool support, theorem prouvers.

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.

In collaboration with Peter Sewell (Cambridge University).

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 2012 we implemented several bug-fixes and we kept the theorem prouver backends up-to date with the prover evolution. We have also been working toward a closer integration with the Lem tool.

The currently relased version is 0.21.2.

5.10. Lem: a tool for lightweight executable semantics

Participant: Francesco Zappa Nardelli [contact].

Languages, semantics, tool support, theorem prouvers.

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 and Isabelle code; the OCaml backend uses a finite set library (and does not yet support inductive relations). A Coq backend is in development.

Lem is already in use at Cambridge and Inria for research on relaxed-memory concurrency. We are currently preparing a feature-complete release with back-ends for HOL4, Isabelle/HOL, Coq, OCaml, and LaTeX. The project web-page is http://www.cl.cam.ac.uk/~so294/lem/.

In collaboration with Scott Owens (U. Kent, UK) and Peter Sewell (U. Cambridge, UK).

5.11. Cmmtest: a tool for hunting concurrency compiler bugs

Participants: Francesco Zappa Nardelli [contact], Robin Morisset, Pankaj Pawan.

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.

Although not yet publicly distributed, cmmtest already 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.

6. New Results

6.1. Reactive Programming

Participants: Mehdi Dogguy, 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 submitted. We have implemented a new runtime for ReactiveML, that uses the MPI (Message Passing Interface) library to run programs on multi-core and distributed architectures.

We have also investigated new static analyses for the language. Following the work of PhD thesis of Mehdi Dogguy, we have studied a new analysis which adds usages on signals to be able to ensure one to one communications. We have also studied a new reactivity analysis which ensures that a process can not prevent the other ones to from executing. This analysis will be published in [10].

6.2. n-Synchronous Languages

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

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

We have designed the language Lucy-n to program in this model of computation [40]. This language is similar to the first order synchronous data-flow language Lustre in which a buffer operator is added. A dedicated type system allows to check that programs can be executed in bounded memory and to compute sufficient buffer sizes. Technically it is done through the introduction of a subtyping constraint at each bufferization point.

- In collaboration with F. Plateau (Prove&Run), we developed a new resolution constraint algorithm for the clocking of Lucy-n programs [8]. Even if the new algorithm is less efficient that the one using abstraction, it has the advantage to be more precise and thus to accept more programs. It is useful for example for the static scheduling of Latency Insensitive Designs [41].
- We worked on an extension of the synchronous model with integer clocks. This extension allows to produce and consume several values at each activation. It has large implication on the semantics, clock typing, causality and code generation of the language.
- We have continue the work on the code generation. In particular, we have been designing a new intermediate representation that allows to deal with integer clocks.

6.3. Strong normal form for large integers, boolean functions and finite automata

Participant: Jean Vuillemin.

Jean Vuillemin's recent work focusses on finding Strong Normal Form for large Integers, Boolean functions and finite Automata, with applications to circuits and software.

- [16] is the latest version of JV's course notes at ENS "De l'algorithme au circuit".
- [9] shows that the ordered dimension of a Boolean function is a lower bound on the size of most known ordered Decision Diagrams, and that ordered decision diagrams can be efficiently constructed an operated upon.
- [6] shows an approach to circuit protection against side-channel attacks based on a statistical analysis of power traces derived from actual measures of the circuit in operation.

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

Participants: Francesco Zappa Nardelli [contact], Robin Morisset, Pankaj Pawan.

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 submitted to an international conference [15].

6.5. A verified compiler for relaxed-memory concurrency

Participant: Francesco Zappa Nardelli [contact].

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, and in 2012 we submitted a journal version, describing the correctness proof of all the phases of our CompCertTSO compiler (including experimental fence eliminations). This has been accepted for publication in Journal of the ACM [3].

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

6.6. Compiling C/C++ concurrency from C++11 to POWER

Participant: Francesco Zappa Nardelli [contact].

The upcoming C and C++ revised standards add concurrency to the languages, for the first time, in the form of a subtle relaxed memory model (the C++11 model). This aims to permit compiler optimisation and to accommodate the differing relaxed-memory behaviours of mainstream multiprocessors, combining simple semantics for most code with high-performance low-level atomics for concurrency libraries.

We studied the the correctness of two proposed compilation schemes for the C++11 load and store concurrency primitives to Power assembly, having noted that an earlier proposal was flawed. (The main ideas apply also to ARM, which has a similar relaxed memory architecture.)

This should inform the ongoing development of production compilers for C++11 and C1x, clarifies what properties of the machine architecture are required, and builds confidence in the C++11 and Power semantics.

A paper describing this work will appear in POPL 2012 [5].

In collaboration with Kayvan Memarian (previously student in the Moscova EPI, currently at U. Cambridge).

6.7. Compilation techniques for synchronous languages

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

- The generation of efficient sequential code for synchronous data-flow languages raises two intertwined issues: control and memory optimization. While the former has been extensively studied, for instance in the compilation of Lustre and SIGNAL, the latter has been only addressed in a restricted manner. Yet, memory optimization becomes a pressing issue when arrays are added to such languages, for example, SCADE 6 [8]. We have proposed a two-levels solution to the memory optimization problem. It combines a compile-time optimization algorithm, reminiscent of register allocation, paired with language annotations on the source given by the designer. Annotations express in-place modifications and control where allocation is performed. Moreover, they allow external functions performing in-place modifications to be imported safely. Soundness of annotations is guaranteed by a semilinear type system and additional scheduling constraints. A key feature is that annotations for well-typed programs do not change the semantics of the language: removing them may lead to a less efficient code but with the very same semantics.

  The method has been implemented in HEPTAGON, the compiler developed in the team of a Lustre-like synchronous language extended with hierarchical automata and arrays. Experiments show that the proposed approach removes most of the unnecessary array copies, resulting in faster code that uses less memory. This work has been presented at the *ACM Intern. Conf. on Languages, Compilers and Tools for Embedded Systems (LCTES'12)* in June 2012 and it has received the *Best paper award*.

[8] http://www.esterel-technologies.com/products/scade-suite/

6.8. Generation of Parallel Code from Synchronous Programs

Participants: Albert Cohen [contact], Léonard Gérard, Adrien Guatto, Nhat Minh Le, Marc Pouzet.

- Efficiently distributing synchronous programs is a challenging and long-standing subject. This paper introduces the use of futures in a Lustre-like language, giving the programmer control over the expression of parallelism. In the synchronous model where computations are considered instantaneous, futures increase expressiveness by decoupling the beginning from the end of a computation. Through a number of examples, we show how to desynchronize long computations and implement parallel patterns such as fork-join, pipelining and data parallelism. The proposed extension preserves the main static properties of the base language, including static resource bounds and the absence of deadlock, livelock and races. Moreover, we prove that adding or removing futures preserves the underlying synchronous semantics.

  This work has been presented at the *ACM Intern. Conf. on Embedded Software (EMSOFT 2012)*, in October 2012 and it received the *Best paper award*.

  Further work along these lines is taking place, to generate code for a variety of low-overhead execution models, to cope with real-time constraints, and to formalize and prove the correctness of the underlying concurrent data structures. On the latter point, a paper has been accepted at the ACM Conf. PPoPP 2013.

6.9. Semantics and Implementation of Hybrid System Modelers

Participants: Marc Pouzet [contact], Timothy Bourke.

Zélus is a new programming language for modeling systems that mix discrete logical time and continuous time behaviors. From a user's perspective, its main originality is to extend an existing -like synchronous language with Ordinary Differential Equations (ODEs). The extension is conservative: any synchronous program expressed as data-flow equations and hierarchical automata can be composed arbitrarily with ODEs in the same source code. A dedicated type system and causality analysis ensure that all discrete changes are aligned with zero-crossing events so that no side effects or discontinuities occur during integration. Programs are statically scheduled and translated into sequential code which, by construction, runs in bounded time and space. Compilation is effected by source-to-source translation into a small synchronous subset which is processed by a standard synchronous compiler architecture. The resulting code is paired with an off-the-shelf numeric solver.

This experiment show that it is possible to build a modeler for explicit hybrid systems à la Simulink/Stateflow on top of an existing synchronous language, using it both as a semantic basis and as a target for code generation. In parallel with the software development done during the year, we investigate, in collaboration with Albert Benveniste, Benoit Caillaud (Inria Rennes) and Dassault-Systèmes the treatment of Differential Algebraic Equations (DAEs), in explicit or semi-explicit form.

This work will be presented at the *ACM Intern. Conference on Hybrid Systems: Computation and Control (HSCC 2013)* in April 2013.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

+ Google European Doctoral Fellowship of Tobias Grosser. $62000 per year over 3 years. Studying the interaction of affine loop transformations and vectorization, for multicore processors and hardware accelerators.

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.

ANR CAFEIN, 2013-2015.

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.

8.2. International Research Visitors

8.2.1. Visits of International Scientists

September, 27 - October, 3, Peter Sewell (U. Cambridge) visited the Parkas team for collaboration with F. Zappa Nardelli and R. Morisset.

October, 6-13, Mike Hicks (U. Maryland) visited the Département d’informatique of the ENS.

January, 18-20, P. Sadayappan (Ohio State U.) visited the team to work with Tobias Grosser and Sven Verdoolaege. Similar visits took place in July and December.

June-July 2013. Stephen Edwards (Columbia U.) was invited by ENS to spend a month in the team.

8.2.1.1. Internships

January-July, Pankaj Pawan (IIT Kanpur) was intern student (M2) under the supervision of F. Zappa Nardelli.

May-September, Robin Morisset (ENS Ulm) was intern student (M2) under the supervision of F. Zappa Nardelli.

May-September, Fran cois Gindraud (ENS Ulm) was intern student (M2) under the supervision of A. Cohen.

December 2011-November 2012, Mehdi Dogguy was post-doc funded by the ANR Partout grant. Mehdi Dogguy worked on the static analysis of ReactiveML programs and was supervised by L. Mandel.

April-July 2012, Cyprien Lecourt (École Polytechnique) was intern student (M1) under the supervision of M. Pouzet.

April-September 2012, Guillaume Baudart (École normale supérieure de Cachan) was intern student (M2) under the supervision of M. Pouzet. Guillaume was a student from IRCAM and the supervision was joint with Florent Jacquemart (Inria Paris-Rocquencourt and IRCAM).

8.2.2. Visits to International Teams

Louis Mandel spent 7 weeks in the team of Vijay Saraswat at IBM T.J. Watson. He worked on the type system of the X10 language.

Albert Cohen and Tobias Grosser visited Prof. Uday Bondhugula at the Indian Institute of Science (IISc), CSA department, for 4 days and 2 weeks, respectively. Tobias Grosser gave a lecture/tutorial on optimizing compilation in LLVM to IISc students and AMD engineers.

9. Dissemination

9.1. Scientific Animation

9.1.1. Event organization

- J. Vuillemin is member of the International Scientific Advisory Board of the National ICT Australia.
- J. Vuillemin chairs the Scientific Advisory Board of the national Institute for InfoCom Research I2R in Singapore.
- F. Zappa Nardelli is member of the “comité executif” of the CEA-EDF-Inria summer schools.
- F. Zappa Nardelli served in the POPL 2012 ERC.
- F. Zappa Nardelli is member of the "comité de suivi doctoral" de l’Inria Saclay.
- A. Cohen is the General Chair of the 7th HiPEAC conference, January 2012. Louis Mandel is member of the local arrangements committee. The conference pioneered a new “journal first” publication model, and has been completely reorganized into a large networking event with 27 parallel events, an industry exhibit with 50 companies booths and posters, and a European project exhibit with 46 projects from the computing systems, embedded systems strategic objectives of the FP7 and from the FET programme. 516 registered participants, 34 ACM TACO journal papers presented (the record number for previous HiPEAC conferences was 210 participants).
- M. Pouzet is in charge, with Catherine Dubois (IIE CNAM, Evry) of the national research group LTP (Language, Types, Proofs) of the French GDR GLP (“Génie de la Programmation et du Logiciel”). Two meetings a year with three invited speakers.
- M. Pouzet organised 18th edition of SYNCHRON (http://synchron2011.di.ens.fr), the workshop on Synchronous Programming and Applications, in Dec. 2011 in Damaries-les-lys. We got the record track of 80 participants and a special half-day devoted to synchronous programming for music applications with collegues from IRCAM.

9.1.2. Editorial boards

- Jean Vuillemin is on the board of 5 international journals.
- Marc Pouzet is associate editor of the EURASIP Journal on Embedded systems (http://jes.eurasipjournals.com.
- Marc Pouzet is “directeur de collection” for Hermes publisher.
- Albert Cohen is on the editorial board of the International Journal on Parallel Programming (IJPP, http://www.springer.com/computer/theoretical+computer+science/journal/10766).
- Albert Cohen is on the distinguished reviewer board of the ACM Transactions on Architecture and Code Optimization (TACO).

9.1.3. Program committees

- M. Pouzet is a member of the program committee of the following conferences: Design, Automation & Test in Europe (DATE 2012); Embedded Software (EMSOFT 2012); Complex Systems Design & Management (CSD&M) 2012; Approches Formelles dans l’Assistance au Développement de Logiciels (AFADL 2012); ACM Conf. on the Principles and Applications of Declarative Programming (PADL 2012), located with POPL;
- M. Pouzet is an expert reviewer for the Design Automation Conference (DAC) in 2012 and 2013.
- A. Cohen is the chair of the DAC 2013 ESS1 TPC subcommittee, and was the co-chair of the DAC 2012 ESS1&2 TPC subcommittee.
- A. Cohen is the program chair of ETAPS CC 2014.

- A. Cohen is a member of the program committee of the following conferences: ICS 2012, ETAPS CC 2013, PPoPP 2013 external review committee.
- A. Cohen is the co-general chair of the ICPP-EMS 2013 workshop (with ICPP).
- A. Cohen is a member of the program committee of the following workshops: IMPACT 2012 and 2013 (with HiPEAC), HiRES 2013 (with HiPEAC), GPGPU 2012 and 2013 (with ASPLOS), COSMIC 2013 (with CGO), PLC 2012 and 2013 (with IPDPS), ICPP-EMS 2012 (with ICPP).
- L. Mandel, member of the program committee of the Journées Francophones des Langages Applicatifs (JFLA 2012).

9.1.4. Invited Presentations

- A. Cohen was a keynote speaker at the IMPACT 2013 workshop (with HiPEAC) "Adopt a Polyhedral Compiler!", July 2011.
- A. Cohen gave an invited presentation at AMD Bangalore titled "Languages and Compilers for Productivity and Efficiency", June 2012.
- A. Cohen gave a Priti Shankar seminar at the Indian Institute of Science, CSA department, titled "Polyhedral Compilation Off the Beaten Path", June 2012.
- M. Pouzet gave an invited presentation at the Int. Conference on Complex Systems, Design & Management (CSDM), Dec. 2011.
- M. Pouzet gave an invited presentation at the "Séminaire MaMux", IRCAM, Paris. Feb. 2012.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Licence: "Systèmes" (L3), L. Mandel (42h), Université Paris-Sud 11, France

Licence: "Systèmes et réseaux" (L3), M. Pouzet (24h), L. Mandel (24h), École Normale Supérieure, France

Licence: "Langages de programmation et compilation" (L3), L. Mandel (24h), École Normale Supérieure, France

Master Parisien de Recherche en Informatique (MPRI): "Synchronous systems" (M2), M. Pouzet (12h), J. Vuillemin (6h), L. Mandel (6h), École Normale Supérieure and Université Paris Diderot, France

Licence: "From Algorithm to Circuit" (L3), J. Vuillem (64h), École Normale Supérieure, France

Master Parisien de Recherche en Informatique (MPRI): "Semantics and tools for multicore programming" (M2), A. Cohen (9h), F. Zappa Nardelli (13.5h), École Normale Supérieure and Université Paris Diderot, France

Marc Pouzet is "responsable du concours d'entrée à l'ENS", for the Computer Science department (since Sept. 2010).

Marc Pouzet is director of studies (directeur des études) of the Computer Science department (since Sept. 2012).

9.2.2. Supervision

M2: R. Morisset (ENS), Correctness of optimisations in the C11/C++11 memory model, september 2012;

M2: P. Pawan (IIT Kanpur, India), Hunting concurrency compiler bugs, september 2012;

M2: F. Gindraud (ENS Lyon), definition, code generation, and formal verification of a software controlled cache coherence protocol;

M2: B. Arnoux (École Polytechnique, Telecom ParisTech), globally adressable memory model for data-flow execution.

M1: C. Lecourt (École Polytechnnique), numerical receipies and examples for Zélus.

M2: G. Baudart (École normale supérieure, Cachan), A synchronous semantics and implementation for Antescofo.

9.2.3. Juries

- A. Cohen was a reviewer of the following PhD theses: Nicolas Benoit (January 2012, Université de Versailles St-Quentin), Artur Pietrek (October 2012, Université Joseph Fourier), Mehdu Amini (November 2012, MINES ParisTech), Selma Saidi (November 2012, Université Joseph Fourier), Dmitry Nadezhkin (December 2012, Leiden University, The Netherlands).
- A. Cohen was a committee member of the following PhD theses: Delphine Demange (October 2012, École Normale Supérieure de Cachan – Université Européenne de Bretagne), Quentin Colombet (December 2012, École Normale Supérieure de Lyon).
- A. Cohen was a committee member of the following Habilitation theses: Gaël Thomas (December 2012, Université Pierre et Marie Curie), Cédric Bastoul (December 2012, Université Paris Sud).
- M. Pouzet was the President of the following PhD. thesis: Peter Schrammel (October 2012, Université de Grenoble).

9.3. Popularization

- L. Mandel and M. Pouzet, were invited to give a lecture at IRCAM in the Master of “Acoustique, Traitement du signal, Informatique, Appliqués à la Musique”. Nov. 2010, Nov. 2011, and Nov. 2012.

10. Bibliography

Publications of the year

Articles in International Peer-Reviewed Journals

[1] A. Benveniste, T. Bourke, B. Caillaud, M. Pouzet. *Non-Standard Semantics of Hybrid Systems Modelers*, in "Journal of Computer and System Sciences (JCSS)", 2012, vol. Special issue in honor of Amir Pnueli.

[2] D. Das, R. Upadrasta, B. Dupont De Dinechin. *Efficient Liveness Computation Using Merge Sets and DJ-Graphs*, in "ACM Transactions on Architecture and Code Optimization", January 2012, vol. 8, n^{o} 4 [*DOI :* 10.1145/2086696.2086706], http://hal.inria.fr/hal-00647369.

[3] J. Ševčík, V. Vafeiadis, F. Zappa Nardelli, P. Sewell, S. Jagannathan. *CompCertTSO: A Verified Compiler for Relaxed-Memory Concurrency*, in "JACM", 2012, to appear.

International Conferences with Proceedings

[4] M. Bachir, A. Cohen, S.-A.-A. TOUATI. *On the Effectiveness of Register Moves to Minimise Post-Pass Unrolling in Software Pipelined Loops*, in "HPCS 2012 : International Conference on High Performance Computing & Simulation", Madrid, Spain, IEEE, ACM (editors), Pr Waleed Smari, July 2012, http://hal.inria.fr/hal-00716183.

[5] M. Batty, K. Memarian, S. Owens, S. Sarkar, P. Sewell. *Clarifying and Compiling C/C++ Concurrency: from C++11 to POWER*, in "POPL 2012", 2012.

[6] E. Brier, Q. Fortier, R. Korkikian, D. Naccache, G. O. de Almeida, A. Pommellet, K. W. Magld, A. H. Ragab, J. Vuillemin. *Defensive Leakage Camouflage*, in "CARDIS 12", 2013, to appear.

[7] L. Gérard, A. Guatto, C. Pasteur, M. Pouzet. *A modular memory optimization for synchronous data-flow languages: application to arrays in a lustre compiler*, in "Proceedings of the 13th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems", Beijing, China, ACM, 2012, p. 51–60 [*DOI :* 10.1145/2248418.2248426], http://hal.inria.fr/hal-00728527.

[8] L. Mandel, F. Plateau. *Scheduling and Buffer Sizing of n-Synchronous Systems: Typing of Ultimately Periodic Clocks in Lucy-n*, in "Eleventh International Conference on Mathematics of Program Construction (MPC 2012)", Madrid, Spain, June 2012.

[9] J. Vuillemin. *The least diagram of a Boolean function*, in "Boole Conference, volume 3 of Luminy 12", 2012.

Conferences without Proceedings

[10] L. Mandel, C. Pasteur. *Réactivité des systèmes coopératifs : le cas de ReactiveML*, in "Vingt-quatrièmes Journées Francophones des Langages Applicatifs", Aussois, France, February 2013.

Research Reports

[11] B. Diouf, A. Cohen, F. Rastello. *A Polynomial Spilling Heuristic: Layered Allocation*, Inria, July 2012, n^{o} RR-8007, 23, http://hal.inria.fr/hal-00713693.

[12] A. Pop, A. Cohen. *Control-Driven Data Flow*, Inria, July 2012, n^{o} RR-8015, 36, http://hal.inria.fr/hal-00717906.

[13] A. Pop, A. Cohen. *Expressiveness and Data-Flow Compilation of OpenMP Streaming Programs*, Inria, June 2012, n^{o} RR-8001, 28, http://hal.inria.fr/hal-00710409.

Other Publications

[14] G. Baudart. *Antescofo : Vers une programmation synchrone*, Master ATIAM, Université Pierre et Marie Curie (UPMC) et IRCAM, Paris, September 2012, 46, http://hal.inria.fr/hal-00730443.

[15] R. Morisset, P. Pawan, F. Zappa Nardelli. *Compiler Testing via a Theory of Sound Optimisations in the C11/C++11 memory model*, 2012, submitted.

[16] J. Vuillemin. *De l'algorithme au circuit intégré*, 2012, Notes de cours de l'Ecole Normale Supérieure, pages 1-310, ENS.

References in notes

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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, p. 151–160.

[24] 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, p. 292–303.

[25] 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.

[26] 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://www.lri.fr/~plateau/hfl09/.

[27] 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.

[28] 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.

[29] 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.

[30] 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, p. 245–255.

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

[32] 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, p. 389-420, See also Part I, one dimensional time, 21(5):315–348.

[33] 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.

[34] 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, p. 261–317, Special issue on Microgrids.

[35] 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.

[36] 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, p. 173–182.

[37] 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.

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

[39] 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.

[40] L. Mandel, F. Plateau, M. Pouzet. *Lucy-n: a n-Synchronous Extension of Lustre*, in "Tenth International Conference on Mathematics of Program Construction (MPC 2010)", Québec, Canada, June 2010, http://www.lri.fr/~mandel/papiers/MandelPlateauPouzet-MPC-10.pdf.

[41] L. Mandel, F. Plateau, M. Pouzet. *Static Scheduling of Latency Insensitive Designs with Lucy-n*, in "International Conference on Formal Methods in Computer-Aided Design (FMCAD)", Austin, Texas, USA, October 30 – November 2 2011.

[42] 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.

[43] F. Maraninchi, N. Berthier, O. Bezet, G. Funchal. *Writing Simulators with Synchronous Languages*, 2008, Synchron 2008: International Open Workshop on Synchronous Programming.

[44] 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.

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

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

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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, p. 358–370.

[55] 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.

[56] 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.

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