IN PARTNERSHIP WITH:
CNRS

Université Claude Bernard (Lyon 1)

Ecole normale supérieure de Lyon

Activity Report 2011

Project-Team ARENAIRE

Computer arithmetic

IN COLLABORATION WITH: Laboratoire de l'Informatique du Parallélisme (LIP)

RESEARCH CENTER
Grenoble - Rhône-Alpes

THEME
Algorithms, Certification, and Cryptography

Table of contents

Project-Team ARENAIRE

Keywords: Computer Arithmetic, Floating-Point Numbers, Elementary Function, Computer Algebra, Numerical Methods, Reliability, Interval Analysis, Algorithmic Numbers Theory, Cryptography, FPGA

1. Members

Research Scientists

Nicolas Brisebarre [Junior Researcher CNRS]
Laurent-Stéphane Didier [Associate Professor (Univ. Paris 6), sabbatical, since September 2011, HdR]
Claude-Pierre Jeannerod [Junior Researcher INRIA]
Fabien Laguillaumie [Junior Researcher CNRS (on partial secondment), since September 2011, HdR]
Vincent Lefèvre [Junior Researcher INRIA]
Micaela Mayero [Junior Researcher INRIA (on partial secondment), since September 2011]
Jean-Michel Muller [Senior Researcher CNRS, HdR]
Nathalie Revol [Junior Researcher INRIA]
Damien Stehlé [Junior Researcher CNRS, HdR]
Gilles Villard [Senior Researcher CNRS, HdR]

Faculty Members

Sylvain Collange [ATER ENS de Lyon, until September 2011]
Florent de Dinechin [Team leader, Associate Professor ENS de Lyon, HdR]
Eleonora Guerrini [ATER ENS de Lyon, since September 2011]
Guillaume Hanrot [Professor ENS de Lyon, HdR]
Nicolas Louvet [Associate Professor UCBL]
Ioana Pasca [ATER ENS de Lyon, since September 2011]

Technical Staff

Honoré Takeugming [Engineer on the ANR *TCHATER* project, until 16/01/2011]
Serge Torres [Technical Staff, 40% on the project]

PhD Students

Nicolas Brunie [CIFRE grant (Kalray), 2nd year]
Mioara Joldeş [*Allocataire-moniteur*, Région Rhône-Alpes grant, thesis defended on September 26, 2011]
Jingyan Jourdan-Lu [CIFRE grant (STMicroelectronics), 3rd year —maternity leave from November 2010 to May 2011]
Adeline Langlois [*Élève Normalienne*, ENS Cachan, cotutelle with Macquarie University, 1st year, since September 2011]
Érik Martin-Dorel [MESR grant, 3rd year]
Ivan Morel [*Allocataire-moniteur*, ENS grant, cotutelle with the University of Sydney, 4th year, until August 31, 2011]
Christophe Mouilleron [*Allocataire-moniteur*, ENS grant, thesis defended on November 4, 2011]
Hong Diep Nguyen [INRIA grant, thesis defended on January 18, 2011]
Adrien Panhaleux [*Allocataire-moniteur*, ENS grant, 4th year]
Bogdan Pasca [*Allocataire-moniteur*, MESR grant, thesis defended on September 21, 2011]
David Pfannholzer [MEFI grant, 2nd year]
Xavier Pujol [*Allocataire-moniteur*, ENS grant, 3rd year]
Philippe Théveny [Contrat doctoral, 1st year, since October 2011]

Post-Doctoral Fellows

Marc Mezzarobba [PostDoc INRIA, since December 2011]
Andrew Novocin [PostDoc ENS Lyon, until August 31, 2011]

Álvaro Vázquez Álvarez [PostDoc INRIA, until 31/03/2011]

Administrative Assistant

Damien Séon [ENS de Lyon, TR INRIA, 50% on the project]

2. Overall Objectives

2.1. Introduction

The Arénaire project aims at elaborating and consolidating knowledge in the field of Computer Arithmetic, which studies how a machine deals with numbers. Reliability, accuracy, and performance are the major goals that drive our research. We study basic arithmetic operators such as adders, dividers, etc. We work on new operators for the evaluation of elementary and special functions ($\log$, $\cos$, erf, etc.), and also consider the composition of previous operators. In addition to these studies on the arithmetic operators themselves, our research focuses on specific application domains (cryptography, signal processing, linear algebra, algorithms for Euclidean lattice, etc.) for a better understanding of the impact of the arithmetic choices on solving methods in scientific computing.

We contribute to the improvement of the available arithmetic on computers, processors, dedicated or embedded chips, etc., both at the hardware level and at the software level. Improving computing does not necessarily mean getting more accurate results or getting them faster: we also take into account other constraints such as power consumption, code size, or the reliability of numerical software. All branches of the project focus on algorithmic research and on the development and the diffusion of corresponding libraries, either in hardware or in software. Some distinctive features of our libraries are numerical quality, reliability, and performance.

The study of number systems and, more generally, of data representations is a first topic of uttermost importance in the project. Typical examples are: the redundant number systems used inside multipliers and dividers; alternatives to floating-point representation for special purpose systems; finite field representations with a strong impact on cryptographic hardware circuits; the performance of an interval arithmetic that heavily depends on the underlying real arithmetic.

Another general objective of the project is to improve the validation of computed data, we mean to provide more guarantees on the quality of the results. For a few years we have been handling those validation aspects in the following three complementary ways: through better qualitative properties and specifications (correct rounding, error bound representation, and portability in floating-point arithmetic); by proposing a development methodology focused on the proven quality of the code; by studying and allowing the cooperation of various kinds of arithmetics such as constant precision, intervals, arbitrary precision and exact numbers.

These goals may be organized in four directions: *hardware arithmetic*, *software arithmetic for algebraic and elementary functions*, *validation and automation*, and *arithmetics and algorithms for scientific computing*. These directions are not independent and have strong interactions. For example, elementary functions are also studied for hardware targets, and scientific computing aspects concern most of the components of Arénaire.

- *Hardware Arithmetic.* From the mobile phone to the supercomputer, every computing system relies on a small set of computing primitives implemented in hardware. Our goal is to study the design of such arithmetic primitives, from basic operations such as the addition and the multiplication to more complex ones such as the division, the square root, cryptographic primitives, and even elementary functions. Arithmetic operators are relatively small hardware blocks at the scale of an integrated circuit, and are best described in a structural manner: a large operator is assembled from smaller ones, down to the granularity of the bit. This study requires knowledge of the hardware targets (ASICs, FPGAs), their metrics (area, delay, power), their constraints, and their specific language and tools. The input and output number systems are typically given (integer, fixed-point, or floating-point), but internally, non-standard number systems may be successfully used.

- *Algebraic and Elementary Functions.* Computer designers still have to implement the basic arithmetic functions for a medium-size precision. Addition and multiplication have been much studied but their performance may remain critical (silicon area or speed). Division and square root are less critical, however there is still room for improvement (e.g. for division, when one of the inputs is constant). Research on new algorithms and architectures for elementary functions is also very active. Arénaire has a strong reputation in these domains and will keep contributing to their expansion. Thanks to past and recent efforts, the semantics of floating-point arithmetic has much improved. The adoption of the IEEE-754 standard for floating-point arithmetic has represented a key point for improving numerical reliability. Standardization is also related to properties of floating-point arithmetic (invariants that operators or sequences of operators may satisfy). Our goal is to establish and handle new properties in our developments (correct rounding, error bounds, etc.) and then to have those results integrated into the future computer arithmetic standards.
- *Validation and Automation.* Validation corresponds to some guarantee on the quality of the evaluation. Several directions are considered, for instance the full error (approximation plus rounding errors) between the exact mathematical value and the computed floating-point result, or some guarantee on the range of a function. Validation also comprises a proof of this guarantee that can be checked by a proof checker. Automation is crucial since most development steps require specific expertise in floating-point computing that can neither be required from code developers nor be mobilized manually for every problem.
- *Arithmetics and Algorithms.* When conventional floating-point arithmetic does not suffice, we use other kinds of arithmetics. Especially in the matter of error bounds, we work on interval arithmetic libraries, including arbitrary precision intervals. Here a main domain of application is global optimization. Original algorithms dedicated to this type of arithmetic must be designed in order to get accurate solutions, or sometimes simply to avoid divergence (e.g., infinite intervals). We also investigate exact arithmetics for computing in algebraic domains such as finite fields, unlimited precision integers, and polynomials. A main objective is a better understanding of the influence of the output specification (approximate within a fixed interval, correctly rounded, exact, etc.) on the complexity estimates for the problems considered. Those problems mainly come from two application domains: exact linear algebra and lattice basis reduction.

Our work in Arénaire since its creation in 1998, and especially since 2002, provides us a strong expertise in computer arithmetic. This knowledge, together with the technology progress both in software and hardware, draws the evolution of our objectives towards the *synthesis of validated algorithms*.

2.2. Highlights

BEST PAPER AWARD :
[45] **Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on**. F. DE DINECHIN, J.-M. MULLER, B. PASCA, A. PLESCO.

3. Scientific Foundations

3.1. Introduction

As stated above, four major directions in Arénaire are *hardware arithmetic*, *algebraic and elementary functions*, *validation and automation*, and *arithmetics and algorithms*. For each of those interrelated topics, we describe below the tools and methodologies on which it relies.

3.2. Hardware Arithmetic

A given computing application may be implemented using different technologies, with a large range of trade-offs between the various aspects of performance, unit cost, and non-recurring costs (including development effort):

- A software implementation, targeting off-the-shelf microprocessors, is easy to develop and reproduce, but will not always provide the best performance.
- For cost or performance reasons, some applications will be implemented as application specific integrated circuits (ASICs). An ASIC provides the best possible performance and may have a very low unit cost, at the expense of a very high development cost.
- An intermediate approach is the use of reconfigurable circuits, or field-programmable gate arrays (FPGAs).

In each case, the computation is broken down into elementary operations, executed by elementary hardware elements, or *arithmetic operators*. In the software approach, the operators used are those provided by the microprocessor. In the ASIC or FPGA approaches, these operators have to be built by the designer, or taken from libraries. Our goals include studying operators for inclusion in microprocessors and developing hardware libraries for ASICs or FPGAs.

Operators under study. Research is active on algorithms for the following operations:

- Basic operations (addition, subtraction, multiplication), and their variations (multiplication and accumulation, multiplication or division by constants, etc.);
- Algebraic functions (division, inverse, and square root, and in general, powering to an integer, and polynomials);
- Elementary functions (sine, cosine, exponential, etc.);
- Combinations of the previous operations (norm, for instance).

A hardware implementation may lead to better performance than a software implementation for two main reasons: parallelism and specialization. The second factor, from the arithmetic point of view, means that specific data types and specific operators, which would require costly emulation on a processor, may be used. For example, some cryptography applications are based on modular arithmetic and bit permutations, for which efficient specific operators can be designed. Other examples include standard representations with non-standard sizes, and specific operations such as multiplication by constants.

Hardware-oriented algorithms. Many algorithms are available for the implementation of elementary operators (see for instance [68]). For example, there are two classes of division algorithms: digit-recurrence and function iteration. The choice of an algorithm for the implementation of an operation depends on, and sometimes imposes, the choice of a number representation. Besides, there are usually technological constraints such as the area and power budget, and the available low-level libraries.

The choice of the number systems used for the intermediate results is crucial. For example, a redundant system, in which a number may have several encodings, will allow for more design freedom and more parallelism, hence faster designs. However, the hardware cost can be higher. As another example, the power consumption of a circuit depends, among other parameters, on its activity, which in turn depends on the distribution of the values of the inputs, hence again on the number system.

Alternatives exist at many levels in this algorithm exploration. For instance, an intermediate result may be either computed, or recovered from a precomputed table.

Parameter exploration. Once an algorithm is chosen, optimizing its implementation for area, delay, accuracy, or energy consumption is the next challenge. The best solution depends on the requirements of the application and on the target technology. Parameters which may vary include the radix of the number representations, the granularity of the iterations (between many simple iterations, or fewer coarser ones), the internal accuracies used, the size of the tables (see [69] for an illustration), etc.

The parameter space quickly becomes huge, and the expertise of the designer has to be automated. Indeed, we do not design operators, but *operator generators*, programs that take a specification and some constraints as input, and output a synthesizable description of an operator.

3.3. Algebraic and Elementary Functions

Elementary Functions and Correct Rounding. Many libraries for elementary functions are currently available. We refer to [68] for a general insight into the domain. The functions in question are typically those defined by the C99 and LIA-2 standards, and are offered by vendors of processors, compilers or operating systems.

Though the 1985 version of the IEEE-754 standard does not deal with these functions, there is some attempt to reproduce some of their mathematical properties, in particular symmetries. For instance, monotonicity can be obtained for some functions in some intervals as a direct consequence of accurate internal computations or numerical properties of the chosen algorithm to evaluate the function; otherwise it may be *very* difficult to guarantee, and the general solution is to provide it through correct rounding. Preserving the range (e.g., $\mathrm{atan}(x) \in [-\pi/2, \pi/2]$) may also be a goal though it may conflict with correct rounding (when supported).

Concerning the correct rounding of the result, it was not required by the IEEE-754-1985 standard: during the elaboration of this standard, it was considered that correctly rounded elementary functions were impossible to obtain at a reasonable cost, because of the so-called *Table Maker's Dilemma*: an elementary function is evaluated to some internal accuracy (usually higher than the target precision), and then rounded to the target precision. What is the minimum accuracy necessary to ensure that rounding this evaluation is equivalent to rounding the exact result, for all possible inputs? This question could not be answered in a simple manner, meaning that correctly rounding elementary functions may require arbitrary precision, which is very slow and resource-consuming.

Indeed, correctly rounded libraries already exist, such as GNU MPFR (http://www.mpfr.org/), the Accurate Portable Library released by IBM in 2002, or the `libmcr` library, released by Sun Microsystems in late 2004. However they have worst-case execution time and memory consumption up to 10,000 worse than usual libraries, which is the main obstacle to their generalized use.

We have focused in the previous years on computing bounds on the intermediate precision required for correctly rounding some elementary functions in IEEE-754 double precision. This allows us to design algorithms using a tight precision. That makes it possible to offer the correct rounding with an acceptable overhead: we have experimental code where the cost of correct rounding is negligible in average, and less than a factor 10 in the worst case. These performances led the IEEE-754 revision committee to recommend (yet not request) correct rounding for some mathematical functions. It also enables to prove the correct-rounding property, and to show bounds on the worst-case performance of our functions. Such worst-case bounds may be needed in safety critical applications as well as a strict proof of the correct rounding property. Concurrent libraries by IBM and Sun can neither offer a complete proof for correct rounding nor bound the timing because of the lack of worst-case accuracy information. Our work actually shows a posteriori that their overestimates for the needed accuracy before rounding are however sufficient. IBM and Sun for themselves could not provide this information. See also §3.4 concerning the proofs for our library.

Approximation and Evaluation. The design of a library with correct rounding also requires the study of algorithms in large (but not arbitrary) precision, as well as the study of more general methods for the three stages of the evaluation of elementary functions: argument reduction, approximation, and reconstruction of the result.

When evaluating an elementary function for instance, the first step consists in reducing this evaluation to the one of a possibly different function on a small real interval. Then, this last function is replaced by an approximant, which can be a polynomial or a rational fraction. Being able to perform those processes in a very cheap way while keeping the best possible accuracy is a key issue [2]. The kind of approximants we can work with is very specific: the coefficients must fulfill some constraints imposed by the targeted application, such as some limits on their size in bits. The usual methods (such as Remez algorithm) do not apply in that situation and we have to design new processes to obtain good approximants with the required form. Regarding the approximation step, there are currently two main challenges for us. The first one is the computation of excellent approximations that will be stored in hardware or in software and that should be called thousands or millions of times. The second one is the target of automation of computation of good approximants when the function is only known at compile time. A third question concerns the evaluation of such good approximants. To find a best compromise between speed and accuracy, we combine various approaches ranging from numerical analysis (tools like backward and forward error analysis, conditioning, stabilization of algorithms) to computer arithmetic (properties like error-free subtraction, exactly-computable error bounds, etc.). The structure of the approximants must further be taken into account, as well as the degree of parallelism offered by the processor targeted for the implementation.

Adequacy Algorithm/Architecture. Some special-purpose processors, like DSP cores, may not have floating-point units, mainly for cost reasons. For such integer or fixed-point processors, it is thus desirable to have software support for floating-point functions, starting with the basic operations. To facilitate the development or porting of numerical applications on such processors, the emulation in software of floating-point arithmetic should be compliant with the IEEE-754 standard; it should also be very fast. To achieve this twofold goal, a solution is to exploit as much as possible the characteristics of the target processor (instruction set, parallelism, etc.) when designing algorithms for floating-point operations.

So far, we have successfully applied this "algorithm/architecture adequacy" approach to some VLIW processor cores from STMicroelectronics, in particular the ST231; the ST231 cores have integer units only, but for their applications (namely, multimedia applications), being able to perform basic floating-point arithmetic very efficiently was necessary. When various architectures are targeted, this approach should further be (at least partly) automated. The problem now is not only to write some fast and accurate code for one given architecture, but to have this optimized code generated automatically according to various constraints (hardware resources, speed and accuracy requirements).

3.4. Validation and Automation

Validating a code, or generating a validated code, means being able to prove that the specifications are met. To increase the level of reliability, the proof should be checkable by a formal proof checker.

Specifications of qualitative aspects of floating-point codes. A first issue is to get a better formalism and specifications for floating-point computations, especially concerning the following qualitative aspects:

- *specification:* typically, this will mean a proven error bound between the value computed by the program and a mathematical value specified by the user in some high-level format;
- *tight error bound computation*;
- *floating-point issues:* regarding the use of floating-point arithmetic, a frequent concern is the portability of code, and thus the reproducibility of computations; problems can be due to successive roundings (with different intermediate precisions) or the occurrence of underflows or overflows;
- *precision:* the choice of the method (compensated algorithm versus double-double versus quadruple precision for instance) that will yield the required accuracy at given or limited cost must be studied;
- *input domains and output ranges:* the determination of input domain or output range also constitutes a specification/guarantee of a computation;

- *other arithmetics, dedicated techniques and algorithms for increased precision:* for studying the quality of the results, most of conception phases will require *multiple-precision* or *exact* solutions to various algebraic problems.

Certification of numerical codes using formal proof. Certifying a numerical code is error-prone. The use of a proof assistant will ensure the code correctly follows its specification. This certification work, however, is usually a long and tedious work, even for experts. Moreover, it is not adapted to an incremental development, as a small change to the algorithm may invalidate the whole formal proof. A promising approach is the use of automatic tools to generate the formal proofs of numerical codes with little help from the user.

Instead of writing code in some programming language and trying to prove it, we can design our own language, well-suited to proofs (e.g., close to a mathematical point of view, and allowing metadata related to the underlying arithmetics such as error bounds, ranges, and so on), and write tools to generate code. Targets can be a programming language without extensions, a programming language with some given library (e.g., MPFR if one needs a well-specified multiple-precision arithmetic), or a language internal to some compiler: the proof may be useful to give the compiler some knowledge, thus helping it to do particular optimizations. Of course, the same proof can hold for several targets.

We worked in particular also on the way of giving a formal proof for our correctly rounded elementary function library. We have always been concerned by a precise proof of our implementations that covers also details of the numerical techniques used. Such proof concern is mostly absent in IBM's and Sun's libraries. In fact, many misroundings were found in their implementations. They seem to be mainly due to coding mistakes that could have been avoided with a formal proof in mind. In CRlibm we have replaced more and more hand-written paper proofs by Gappa (http://gappa.gforge.inria.fr/) verified proof scripts that are partially generated automatically by other scripts. Human error is better prevented.

Integrated and interoperable automatic tools. Various automatic components have been independently introduced above, see §3.2 and §3.3. One of our main objectives is to provide an entire automatic approach taking in input an expression to evaluate (with possible annotations), and returning an executable validated code. The complete automation with optimal or at least good resulting performance seems to be far beyond the current knowledge. However, we see our objective as a major step for prototyping future compilers. We thus aim at developing a piece of software that automates the steps described in the previous pages. The result should be an easy-to-use integrated environment.

3.5. Arithmetics and Algorithms

When computing a solution to a numerical problem, an obvious question is that of the *quality* of the produced numbers. One may also require a certain level of quality, such as: approximate with a given error bound, correctly rounded, or –if possible– exact. The question thus becomes twofold: how to produce such a well-specified output and at what cost? To answer it, we focus on *polynomial and integer matrix operations*, *Euclidean lattices* and *global optimization*, and study the following directions:

- We investigate new ways of producing well-specified results by resorting to various arithmetics (intervals, Taylor models, multi-precision floating-point, exact). A first approach is to *combine* some of them: for example, guaranteed enclosures can be obtained by mixing Taylor model arithmetic with floating-point arithmetic [9]. Another approach is to *adapt* the precision or even *change* the arithmetic during the course of a computation. Typical examples are iterative refinement techniques or exact results obtained via floating-point basic operations. This often requires arithmetics with very-well *specified properties* (like the IEEE-754 standard for floating-point arithmetic).

- We also study the impact of certification on algorithmic complexity. A first approach there is to augment existing algorithms with validated error bounds (and not only error estimates). This leads us to study the (im)possibility of *computing such bounds* on the fly at a negligible cost. A second approach is to study the *algorithmic changes* needed to achieve a higher level of quality without, if possible, sacrificing for speed. In exact linear algebra, for example, the fast algorithms recently obtained in the bit complexity model are far from those obtained decades ago in the algebraic complexity model.

Numerical Algorithms using Arbitrary Precision Interval Arithmetic. When validated results are needed, interval arithmetic can be used. New problems can be solved with this arithmetic, which provides sets instead of numbers. In particular, we target the global optimization of continuous functions. A solution to obviate the frequent overestimation of results is to increase the precision of computations.

Our work is twofold. On the one hand, efficient software for arbitrary precision interval arithmetic is developed, along with a library of algorithms based on this arithmetic. On the other hand, new algorithms that really benefit from this arithmetic are designed, tested, and compared.

To reduce the overestimation of results, variants of interval arithmetic have been developed, such as Taylor models arithmetic or affine arithmetic. These arithmetics can also benefit from arbitrary precision computations.

Algorithms for Exact Linear Algebra and Lattice Basis Reduction. The techniques for exactly solving linear algebra problems have been evolving rapidly in the last few years, substantially reducing the complexity of several algorithms (see for instance [6] for an essentially optimal result, or [67]). Our main focus is on matrices whose entries are integers or univariate polynomials over a field. For such matrices, our main interest is how to relate the size of the data (integer bit lengths or polynomial degrees) to the cost of solving the problem exactly. A first goal is to design asymptotically faster algorithms, to reduce problems to matrix multiplication in a systematic way, and to relate bit complexity to algebraic complexity. Another direction is to make these algorithms fast in practice as well, especially since applications yield very large matrices that are either sparse or structured. Within the LinBox international project, we work on a software library that corresponds to our algorithmic research on matrices. LinBox is a generic library that allows to plug external components in a plug-and-play fashion. The library is devoted to sparse or structured exact linear algebra and its applications.

We recently started a direction around lattice basis reduction. Euclidean lattices provide powerful tools in various algorithmic domains. In particular, we investigate applications in computer arithmetic, cryptology, algorithmic number theory and communications theory. We work on improving the complexity estimates of lattice basis reduction algorithms and providing better implementations of them, and on obtaining more reduced bases. The above recent progress in linear algebra may provide new insights.

Certified Computing. Most of the algorithmic complexity questions that we investigate concern algebraic or bit-complexity models for exact computations. Much less seems to be known in approximate computing, especially for the complexity of computing (certified) error bounds, and for establishing bridges between exact, interval, and constant precision complexity estimates. We are developing this direction both for a theoretical impact, and for the design and implementation of algorithm synthesis tools for arithmetic operators, and mathematical expression evaluation.

4. Application Domains

4.1. Application Domains

Our expertise covers application domains for which the quality, such as the efficiency or safety, of the arithmetic operators is an issue. On the one hand, it can be applied to hardware oriented developments, for example to the design of arithmetic primitives which are specifically optimized for the target application and support. On the other hand, it can also be applied to software programs, when numerical reliability issues arise: these issues can consist in improving the numerical stability of an algorithm, computing guaranteed results (either exact results or certified enclosures) or certifying numerical programs.

- The application domains of hardware arithmetic operators are **digital signal processing**, **image processing**, **embedded applications**, **reconfigurable computing**, and **cryptography**.
- The development of **correctly rounded elementary functions** is critical to the **reproducibility** of floating-point computations. Exponentials and logarithms, for instance, are routinely used in accounting systems for interest calculation, where roundoff errors have a financial meaning. Our current focus is on bounding the worst-case time for such computations, which is required to allow their use in **safety critical** applications, and in proving the correct rounding property for a complete implementation.
- Certifying a numerical application usually requires bounds on rounding errors and ranges of variables. Some of the tools we develop compute or verify such bounds. For increased confidence in the numerical applications, they may also generate formal proofs of the arithmetic properties. These proofs can then be machine-checked by proof assistants like **Coq**.
- Arbitrary precision interval arithmetic can be used in two ways to **validate a numerical result**. To **quickly check the accuracy** of a result, one can replace the floating-point arithmetic of the numerical software that computed this result by high-precision interval arithmetic and measure the width of the interval result: a tight result corresponds to good accuracy. When **getting a guaranteed enclosure** of the solution is an issue, then more sophisticated procedures, such as those we develop, must be employed: this is the case of global optimization problems.
- The design of faster algorithms for matrix polynomials provides faster solutions to various problems in **control theory**, especially those involving multivariable linear systems.
- Lattice reduction algorithms have direct applications in public-key cryptography. They also naturally arise in computer algebra. A new and promising field of applications is communications theory.

5. Software

5.1. Introduction

Arénaire proposes various software and hardware realizations that are accessible from the web page http://www.ens-lyon.fr/LIP/Arenaire/Ware/. We describe below only those which progressed in 2011.

5.2. FloPoCo

Participants: Florent Dinechin [correspondant], Bogdan Pasca, Laurent-Stéphane Didier.

The purpose of the FloPoCo project is to explore the many ways in which the flexibility of the FPGA target can be exploited in the arithmetic realm. FloPoCo is a generator of operators written in C++ and outputting synthesizable VHDL automatically pipelined to an arbitrary frequency.

In 2011, FloPoCo was turned into a library which can be used as a back-end to high-level synthesis tools. An expression parser that generates a complete pipeline was also added for this context. The integer multiplier and floating-point adder were rewritten, and several new operators were added, including a floating-point power operator, and novel operators for integer and floating-point division by a constant.

Versions 2.2.0, 2.2.1, and 2.3.0 were released in 2011.

Among the known users of FloPoCo are U. Cape Town, U.T. Cluj-Napoca, Imperial College, U. Essex, U. Madrid, U. P. Milano, T.U. Muenchen, T. U. Kaiserslautern, U. Paderborn, CalTech, U. Pernambuco, U. Perpignan, U. Tokyo, Virginia Tech U. and several companies.

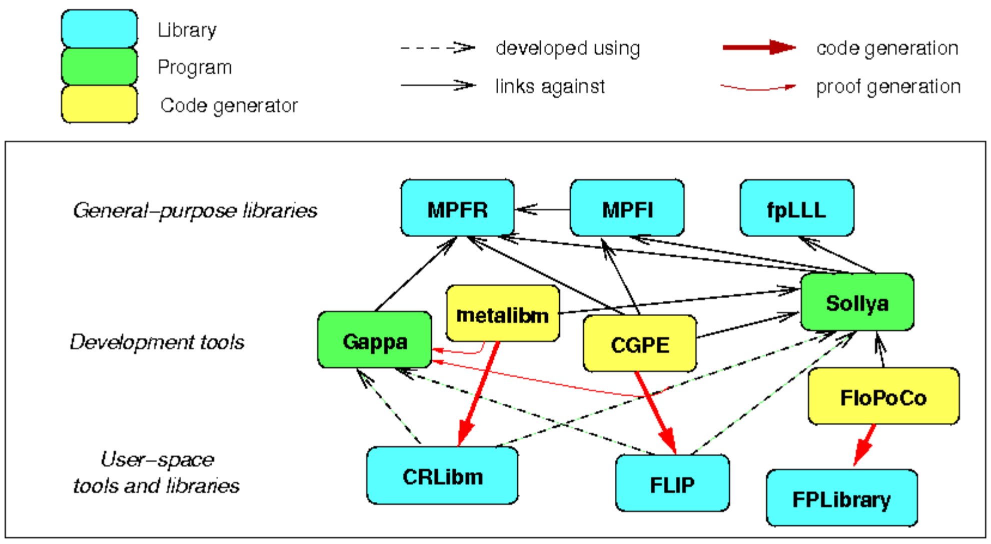

Figure 1. Relationships between some Arénaire developments.

URL: http://flopoco.gforge.inria.fr/

- Version: 2.3.0 (december 2011)
- APP: IDDN.FR.001.400014.000.S.C.2010.000.20600 (version 2.0.0)
- License: specific, GPL-like.
- Type of human computer interaction: command-line interface, synthesisable VHDL output.
- OS/Middelware: Linux, Windows/Cygwin.
- Required library or software: MPFR, flex, Sollya.
- Programming language: C++.
- Documentation: online and command-line help, API in doxygen format, articles.

5.3. GNU MPFR

Participants: Vincent Lefèvre [correspondant], Paul Zimmermann.

GNU MPFR is an efficient multiple-precision floating-point library with well-defined semantics (copying the good ideas from the IEEE-754 standard), in particular correct rounding in 5 rounding modes. GNU MPFR provides about 80 mathematical functions, in addition to utility functions (assignments, conversions...). Special data (*Not a Number*, infinities, signed zeros) are handled like in the IEEE-754 standard.

MPFR was one of the main pieces of software developed by the old SPACES team at Loria. Since late 2006, with the departure of Vincent Lefèvre to Lyon, it has become a joint project between the Caramel (formerly SPACES then CACAO) and the Arénaire project-teams. MPFR has been a GNU package since 26 January 2009. GNU MPFR 3.0.1 was released on 4 April 2011 and GNU MPFR 3.1.0 was released on 3 October 2011.

The main improvements are the generic tests in a reduced exponent range, the possibility to include the `mpfr.h` header file several times while still supporting optional functions, and, for the developers, the choice of the native type for the exponent (and various corrections related to these features).

URL: http://www.mpfr.org/

- ACM: D.2.2 (Software libraries), G.1.0 (Multiple precision arithmetic), G.4 (Mathematical software).
- AMS: 26-04 Real Numbers, Explicit machine computation and programs.
- APP: no longer applicable (copyright transferred to the Free Software Foundation).
- License: LGPL version 3 or later.
- Type of human computer interaction: C library, callable from C or other languages via third-party interfaces.
- OS/Middleware: any OS, as long as a C compiler is available.
- Required library or software: GMP.
- Programming language: C.
- Documentation: API in texinfo format (and other formats via conversion); algorithms are also described in a separate document.

5.4. Exhaustive Tests for the Correct Rounding of Mathematical Functions

Participant: Vincent Lefèvre.

The search for the worst cases for the correct rounding (hardest-to-round cases) of mathematical functions ($\exp$, $\log$, $\sin$, $\cos$, etc.) in a fixed precision (mainly double precision) using Lefèvre's algorithm is implemented by a set of utilities written in Perl, with calls to Maple/intpakX for computations on intervals and with C code generation for fast computations. It also includes a client-server system for the distribution of intervals to be tested and for tracking the status of intervals (fully tested, being tested, aborted).

These programs are run on the LIP network via Grid Engine (SGE). In June 2011, the SGE configuration was changed by the system administrator so that SIGSTOP/SIGCONT signals are sent to the jobs, allowing several users to use SGE at the same time. These signals make Maple crash (segmentation fault), and the Perl scripts needed to be improved to handle these crashes gracefully (by restarting the computations when need be, etc.). This SGE change made other problems appear, such as when the client is first stopped by SGE and is then killed by SGE (without being woke up by SGE), it cannot do its usual clean-up; workarounds were tried, but without success.

The above problems also made an inconsistency in the client-server protocol appear. The validity of the results was not affected, but the protocol had to be redesigned.

5.5. CGPE: Code Generation for Polynomial Evaluation

Participants: Christophe Mouilleron, Claude-Pierre Jeannerod.

The CGPE project, developed with Guillaume Revy (DALI research team, Université de Perpignan and LIRMM laboratory), aims at generating C codes for fast and certified polynomial evaluation, given various accuracy and architectural constraints. Several improvements for this tool, based on the addition of constraints in the first step of the generation process, were proposed in the PhD thesis of Ch. Mouilleron [12]. These improvements have been implemented, thus allowing us to reduce the whole generation time by about 50% on average.

- ACM: D.2.2 (Software libraries), G.4 (Mathematical software).
- Recommended library or software: MPFI or Gappa.
- License: CeCiLL
- Type of human computer interaction: command-line interface
- OS/Middelware: Unix
- Required library or software: Xerces-C++ XML Parser library and MPFR
- Programming Language: C++
- Status: beta
- Documentation: available in html format on **URL:** http://cgpe.gforge.inria.fr/

5.6. FLIP: Floating-point Library for Integer Processors

Participants: Claude-Pierre Jeannerod, Jingyan Jourdan-Lu.

FLIP is a C library for the efficient software support of *binary32* IEEE 754-2008 floating-point arithmetic on processors without floating-point hardware units, such as VLIW or DSP processors for embedded applications. The current target architecture is the VLIW ST200 family from STMicroelectronics (especially the ST231 cores). This year, we have mostly worked on improving the design and implementation of the following operators with correct rounding “to nearest even”: DP2 (fused dot product in dimension two) and sum of two squares. The impact of the DP2 operator has been evaluated on the UTDSP benchmark, and on some kernels speed-ups of 1.46 have been observed. On the other hand, specializing DP2 to a sum of squares brings a speed-up of 2.

URL: http://flip.gforge.inria.fr/

- ACM: D.2.2 (Software libraries), G.4 (Mathematical software)
- AMS: 26-04 Real Numbers, Explicit machine computation and programs.
- APP: IDDN.FR.001.230018.S.A.2010.000.10000
- License: CeCILL v2
- Type of human computer interaction: C library callable, from any C program.
- OS/Middleware: any, as long as a C compiler is available.
- Required library or software: none.
- Programming language: C

5.7. SIPE: Small Integer Plus Exponent

Participant: Vincent Lefèvre.

SIPE (Small Integer Plus Exponent) is a C header file providing a fast floating-point arithmetic with correct rounding to the nearest in very small precision. Implemented operations are the addition, subtraction, multiplication, FMA, and minimum/maximum/comparison functions (of the signed numbers or in magnitude). SIPE has been written for exhaustive tests of simple algorithms in small precision in order to prove results or find conjectures (which could then be proved). In 2011, a research report was written about SIPE [62], including documentation and proof of the implementation; some bugs were fixed at the same time.

- ACM: D.2.2 (Software libraries), G.4 (Mathematical software).
- AMS: 26-04 Real Numbers, Explicit machine computation and programs.
- License: LGPL version 2.1 or later.
- Type of human computer interaction: C header file.
- OS/Middleware: any OS.
- Required library or software: GCC compiler.
- Programming language: C.
- Documentation: Research report RR-7832. [62]
- URL: http://www.vinc17.net/software/sipe.h

6. New Results

6.1. Hardware Arithmetic and Architecture

Participants: Florent de Dinechin, Hong Diep Nguyen, Bogdan Pasca, Honoré Takeugming, Álvaro Vázquez Álvarez, Nicolas Brunie, Sylvain Collange.

6.1.1. FPGA-specific arithmetic

Reconfigurable computing has the opportunity of using exotic operators that would not make sense in a general-purpose microprocessor [43], for instance the constant dividers studied in 6.1.2. Such operators must be also be matched to the precision and performance needed by applications. F. de Dinechin and B. Pasca described the FloPoCo framework that assists the construction of correct pipelines and the automatic testing of such operators [28]. For this context, B. Pasca, with H. D. Nguyen, now at U.C. Berkeley, and T. Preusser, from T. U. Darmstadt, described improved architectures for short-latency adders on modern FPGAs [39]. With Ch. Alias and A. Plesco (Compsys project-team), he studied the integration in of deeply pipelined arithmetic datapath in high-level synthesis tools [51].

6.1.2. Multiplication by Rational Constants versus Division by a Constant

Motivated by the division by 3 or by 9 appearing in some stencil kernels, F. de Dinechin investigated how the periodicity of the binary representation of a rational constant could be exploited to design an architecture multiplying by this constant [26]. With L. S. Didier, this approach was then compared to a specialisation of divider architectures to the division by small integer constants, which is shown to match well the fine structure of FPGAs [44].

6.1.3. Elementary Functions

A. Vázquez worked with J. Bruguera, from U. Santiago de Compostella, on hardware architectures for evaluating q-th roots [66]. Their solution composes digit-recurrence operators for reciprocal, logarithm, multiplication and exponential.

6.1.4. Extensions of the fused-multiply-and-add operator

With B. de Dinechin, from Kalray, N. Brunie and F. de Dinechin proposed to extend the classical fused-multiply-and-add operator with a larger addend and result. This enables higher-precision computation of sums of products at a cost that remains close to that of the classical FMA [56].

6.1.5. Emerging throughput-oriented architecture

On massively multi-threaded processors like GPUs, neighbor threads are likely to operate on similar data. S. Collange showed with A. Kouyoumdjian how it is possible to take advantage of this inter-thread value correlation at the hardware level with a hardware cache-compression technique on GPUs [59]. With D. Sampaio, R. Martins, and F. Magno Quintão Pereira (U. Minas Gerais), he then addressed this question also at the compiler level using a compiler stage to identify statically data patterns in GPGPU programs [65].

Current GPU architectures require specific instruction sets with control-flow reconvergence annotations, and only support a limited number of control-flow constructs. S. Collange and N. Brunie, with G. Diamos (NVIDIA) generalized dynamic vectorization to arbitrary control flow on standard instruction sets with no compiler involvment [46], [57], [54]. In addition, this technique allows divergent branches to be executed in parallel, as a way to increase the throughput of parallel architectures [55].

6.2. Efficient Floating-Point Arithmetic and Applications

Participants: Nicolas Brisebarre, Claude-Pierre Jeannerod, Mioara Joldeş, Jingyan Jourdan-Lu, Vincent Lefèvre, Nicolas Louvet, Érik Martin-Dorel, Christophe Mouilleron, Jean-Michel Muller, Adrien Panhaleux.

6.2.1. Correctly Rounded Sums

P. Kornerup (Odense Univ., Denmark), V. Lefèvre, N. Louvet and J.-M. Muller have given a study of some basic blocks needed in the design of floating-point summation algorithms. In particular, in radix-2 floating-point arithmetic, they have shown that among the set of the algorithms with no comparisons performing only floating-point additions/subtractions, the 2Sum algorithm introduced by Knuth is minimal, both in terms of number of operations and depth of the dependency graph. They have investigated the possible use of another algorithm, Dekker's Fast2Sum algorithm, in radix-10 arithmetic. Under reasonable conditions, they have also proven that no algorithms performing only round-to-nearest additions/subtractions exist to compute the round-to-nearest sum of at least three floating-point numbers. Starting from an algorithm due to Boldo and Melquiond, they have also presented new results about the computation of the correctly-rounded sum of three floating-point numbers [21].

6.2.2. Error of an FMA

The fused multiply-add (FMA) instruction, specified by the IEEE 754-2008 Standard for Floating-Point Arithmetic, eases some calculations, and is already available on some current processors such as the Power PC or the Itanium. S. Boldo (EPI Proval) and J.-M. Muller first extended an earlier work on the computation of the exact error of an FMA (by giving more general conditions and providing a formal proof). Then, they presented a new algorithm that computes an approximation to the error of an FMA, and provide error bounds and a formal proof for that algorithm [16].

6.2.3. Accurate computation of $ad - bc$ with an FMA

C.-P. Jeannerod, N. Louvet, and J.-M. Muller have provided in [60] a detailed rounding error analysis of Kahan's FMA-based algorithm for the computation of expressions of the form $ad - bc$. They showed that Kahan's algorithm is always highly accurate, and under mild assumptions on the radix and the precision gave an optimal bound on the absolute error and an asymptotically optimal bound on the relative error. They also studied how the relative error varies as a function of the relative order of magnitude of the two products ad and bc. Finally, they investigated whether the error bounds can be improved in special cases like sums of squares and discriminants.

6.2.4. Performing Arithmetic Operations on Round-to-Nearest Operations

During any composite computation, there is a constant need for rounding intermediate results before they can participate in further processing. Recently, a class of number representations denoted RN-Codings were introduced, allowing an unbiased rounding-to-nearest to take place by a simple truncation, with the property that problems with double-roundings are avoided. P. Kornerup (Odense Univ., Denmark), J.-M. Muller and A. Panhaleux first investigate a particular encoding of the binary representation. This encoding is generalized to any radix and digit set; however, radix complement representations for even values of the radix turn out to be particularly feasible. The encoding is essentially an ordinary radix complement representation with an appended round-bit, but still allowing rounding-to-nearest by truncation, and thus avoiding problems with double-roundings. Conversions from radix complement to these round-to-nearest representations can be performed in constant time, whereas conversion the other way, in general, takes at least logarithmic time. Not only is rounding-to-nearest a constant time operation, but so is also sign inversion, both of which are at best log-time operations on ordinary two's complement representations. Addition and multiplication on such fixed-point representations are first analyzed and defined in such a way that rounding information can be carried along in a meaningful way, at minimal cost. The analysis is carried through for a compact (canonical) encoding using two's complement representation, supplied with a round-bit. Based on the fixed-point encoding, it is shown possible to define floating-point representations, and a sketch of the implementation of an FPU is presented [22].

6.2.5. Augmented Precision Square Roots, 2-D Norms, and Discussion on Correctly Rounding $\sqrt{x^2 + y^2}$

Define an "augmented precision" algorithm as an algorithm that returns, in precision-p floating-point arithmetic, its result as the unevaluated sum of two floating-point numbers, with a relative error of the order of 2^{-2p}. Assuming an FMA instruction is available, N. Brisebarre, M. Joldeş, P. Kornerup (Odense University, Denmark), E. Martin-Dorel and J.-M. Muller perform a tight error analysis of an augmented precision algorithm for the square root, and introduce two slightly different augmented precision algorithms for the 2D-norm $\sqrt{x^2 + y^2}$. Then they give tight lower bounds on the minimum distance (in ulps) between $\sqrt{x^2 + y^2}$ and a midpoint when $\sqrt{x^2 + y^2}$ is not itself a midpoint. This allows them to determine cases when their algorithms make it possible to return correctly-rounded 2D-norms [30].

6.2.6. *Midpoints and exact points of some algebraic functions in floating-point arithmetic*

When implementing a function f in floating-point arithmetic, if we wish correct rounding and good performance, it is important to know if there are input floating-point values x such that $f(x)$ is either the middle of two consecutive floating-point numbers (assuming rounded-to-nearest arithmetic), or a floating-point number (assuming rounded toward $\pm\infty$ or toward 0 arithmetic). In the first case $f(x)$ is a *midpoint*, and in the second case it is an *exact point*. In [20] C.-P. Jeannerod, N. Louvet, J.-M. Muller, and A. Panhaleux have studied whether such midpoints and exact points exist for some usual algebraic functions and various floating-point formats. When midpoints or exact points exist, they have been characterized or, when possible, listed exhaustively. The results and the techniques presented in this paper can be used in particular to deal with both the binary and the decimal formats defined in the IEEE 754-2008 standard for floating-point arithmetic.

6.3. Correct Rounding of Elementary Functions

Participants: Florent de Dinechin, Vincent Lefèvre, Jean-Michel Muller, Bogdan Pasca, Serge Torres.

6.3.1. *FPGA Acceleration of the Search For Hardest-to-Round Cases*

The IEEE 754-2008 standard for floating-point arithmetic recommends (yet does not dictate) that some elementary functions should be correctly rounded. That is, given a rounding function $\circ$, (e.g., round to nearest even, or round to $\pm\infty$), when evaluating function f at the floating-point number x, the system should always return $\circ(f(x))$.

Building a fast correctly rounded library for some target floating-point (FP) format requires preliminarily solving a problem called the *table maker's dilemma*. This requires very large computations which may use environments and formats totally different from the target environment and format. F. de Dinechin, V. Lefèvre, J.-M. Muller, B. Pasca and A. Plesco suggest performing these computations on an FPGA. Their paper [45] won the best paper award at the ASAP2011 conference.

6.3.2. *Hierarchical Polynomial Approximation of a Function by Polynomials*

Algorithms used to search for the hardest-to-round cases of a function requires the approximation of the function by small-degree polynomials on small intervals. This can be done efficiently by a hierarchical polynomial approximation. Work is being done to improve this method by replacing interval arithmetic (as partly used in the current tools) by static error bounds. This will allow us to better control the precision needed to compute the coefficient of the polynomials. The implementation will also be simpler.

6.4. Validation and Automation

Participants: Nicolas Brisebarre, Florent de Dinechin, Claude-Pierre Jeannerod, Jingyan Jourdan-Lu, Mioara Joldeş, Vincent Lefèvre, Nicolas Louvet, Christophe Mouilleron, Hong Diep Nguyen, David Pfannholzer, Nathalie Revol, Philippe Théveny, Gilles Villard.

6.4.1. *Efficient Implementation of Algorithms for Interval Linear Algebra*

H.-D. Nguyen and N. Revol proposed an algorithm to solve linear systems with interval coefficients. The same approach can be used to verify the solution of a linear system with floating-point coefficients, i.e. to compute an interval enclosing the error between the exact solution and an approximate solution. The goal is twofold: on the one hand the accuracy of the solution is desired up to the last bit of the floating-point solution, on the other hand the efficiency of the implementation is obtained through the use of optimized BLAS3 routines [48]. The PhD thesis of H.-D. Nguyen [13] contains in particular the algorithm [24] and its properties. Its complexity has been established [47] and its potential use for symbolic-numeric computations has been discussed [50].

6.4.2. *Standardization of Interval Arithmetic*

We contributed to the creation, and now chair, the IEEE 1788 working group on the standardization of interval arithmetic http://grouper.ieee.org/groups/1788/. The main discussion topics of this working group [49], for the year 2011, were exception handling (via decorations). An emerging topic is the repeatibility and reproducibility of interval computations. on the same platform or across different platforms.

6.4.3. Formal Proofs of the Arithmetic on Polynomial Models

Using as starting point [9], the calculus with polynomial models, such as Taylor models, based on floating-point coefficients and floating-point operations, has been formalized and checked in Coq [18]. This calculus is at the core of Ariadne, an environment for the study of hybrid systems: the idea is to prove the environment itself, instead of using model-checking on the systems.

6.4.4. Formal Proof Generation for Elementary functions

The proof of the correct rounding property for an elementary function requires tight bounds on the error involved in the function code. F. de Dinechin, with Ch. Lauter (LIP6) and G. Melquiond (INRIA Proval) have described the use of the Gappa proof assistant to compute such tight bounds rigorously [27].

6.4.5. Code Generation for Polynomial Evaluation

A given arithmetic expression may be evaluated on a computer in several ways, depending on the parenthesization and the ordering of terms in use. Among all the possible evaluations, one may want to choose one that is as fast and accurate as possible. In [12] Ch. Mouilleron introduced a set of algorithms in order to generate all these possible evaluations, to count them, and to find an optimal or nearly optimal one according to a given criteria. Thanks to this work, several sequences related to numbers of evaluations have been discovered and added to Sloane's on-line encyclopedia of integer sequences (OEIS). Moreover, this allowed to show experimentally that an algorithm by Paterson and Stockmeyer for the evaluation of a polynomial p at a matrix point is optimal for small degrees of p. Finally, this work has led to the revamping of the software tool CGPE presented in [38] (see also §5.5).

6.5. Arithmetic and Algorithms

Participants: Guillaume Hanrot, Claude-Pierre Jeannerod, Adeline Langlois, Ivan Morel, Christophe Mouilleron, Andrew Novocin, Xavier Pujol, Damien Stehlé, Gilles Villard.

6.5.1. Faster Lattice Reduction

Andrew Novocin, Damien Stehlé and Gilles Villard [40] designed an algorithm, $\widetilde{L}^1$, with the following specifications: It takes as input an arbitrary basis B in $Z^{d\times d}$ of a lattice L; It computes a basis of L which is reduced for a mild modification of the Lenstra-Lenstra-Lovász reduction; It terminates in time $\widetilde{O}(d^5\beta + d^{\omega+1}\beta)$ where $\beta = \log\|B\|$ (and ω is a valid exponent for matrix multiplication). This is the first LLL-reducing algorithm with a time complexity that is quasi-linear in the bit-length beta of the entries and polynomial in the dimension d. A critical ingredient for achieving this result was the study of the effect of small perturbations on the LLL-reducedness of a lattice basis [17].

6.5.2. Computing Short Lattice Vectors

Among all known lattice reduction algorithms, BKZ provides the best trade-off between run-time and smallness of the computed lattice basis. Guillaume Hanrot, Xavier Pujol and Damien Stehlé [32] showed that BKZ can be terminated long before its completion, while still providing bases of excellent quality. More precisely, if it is terminated within a polynomial number of calls to a lower-dimensial Shortest Vector Problem solver, then the bounds on the output quality are as close as desired to the bounds that can be obtained by letting BKZ run until completion.

Guillaume Hanrot, Xavier Pujol and Damien Stehlé also surveyed the known algorithms for solving the Shortest Vector Problem [31].

6.5.3. Lattice-Based Cryptography

NTRUEncrypt is the fastest known lattice-based encryption scheme. Its moderate key-sizes, excellent asymptotic performance and conjectured resistance to quantum computers could make it a desirable alternative to factorisation and discrete-log based encryption schemes. Damien Stehlé and Ron Steinfeld [41] showed how to modify NTRUEncrypt to make it provably resistance to Chosen Plaintext Attacks, under the assumed quantum hardness of standard worst-case lattice problems restricted to a family of lattices related to some cyclotomic fields.

6.5.4. Lattices and Communication Theory

Cong Ling, Shuiyin Liu, Laura Luzzi and Damien Stehlé studied and optimized lattice algorithms that are relevant for MIMO communications [23], [37]. These algorithms tackle the Bounded Distance Decoding Problem: Given a point within a small prescribed distance to a given lattice, find the lattice vector closest to it.

6.5.5. Other Applications of Lattice Reduction Algorithms

In [35] Jürgen Klüners, Mark van Hoeij, and Andrew Novocin showed how to use the LLL lattice reduction algorithm for computing a compact representation of the set of all subfields of any given number field. William Hart (Warwick Mathematics Institute, UK), Mark van Hoeij (Florida State University, USA) and Andrew Novocin exploited the very latest progress in lattice reduction to propose a fine-tuned cutting-edge implementation of a polynomial factorization algorithm.

6.5.6. Polynomial Arithmetic

With William Hart and Mark van Hoeij, A. Novocin proposed in [33] a state of the art algorithm for factoring polynomials in Z[x] . The algorithm is fast in practice, saving in a large class of common examples, without sacrificing performance on worst-case polynomials. The presented algorithm is structured along the lines of algorithms with the best theoretical complexity. In [34] William Hart and A. Novocin proposed an efficient algorithm for computing the composition of two univariate polynomials. Their work builds upon the Brent-Kung algorithm.

6.5.7. Exact Linear Algebra

Transforming a matrix over a field to echelon form, or decomposing the matrix as a product of structured matrices that reveal the rank profile, is a fundamental building block of computational exact linear algebra. For such tasks the best algorithms available so far were either rank sensitive (i.e., of complexity expressed in terms of the exponent of matrix multiplication and the rank of the input matrix) or in place (i.e., using essentially no more memory that what is needed for matrix multiplication). In [61] C.-P. Jeannerod, Clément Pernet (U. Joseph Fourier, Grenoble), and Arne Storjohann (U. Waterloo, Canada) have proposed algorithms that are both rank sensitive and in place. These algorithms are based on a new matrix factorization, namely $A = CUP$ with C a column echelon form revealing the row rank profile of A, U a unit upper triangular matrix, and P a permutation matrix.

7. Contracts and Grants with Industry

7.1. Contracts with Industry

One contract with STMicroelectronics and one contract with Kalray, in the context of two PhD CIFRE grants; see §7.2.

7.2. Grants with Industry

7.2.1. STMicroelectronics CIFRE PhD Grant

Participants: Claude-Pierre Jeannerod, Jingyan Jourdan-Lu, Jean-Michel Muller.

Jingyan Jourdan-Lu is supported by a CIFRE PhD grant (from March 2009 to September 2012) from STMicroelectronics (Compilation Expertise Center, Grenoble) on the theme of floating-point arithmetic code generation and specialization for embedded processors. Advisors: Claude-Pierre Jeannerod and Jean-Michel Muller (Arénaire), Christophe Monat (STMicroelectronics). A contract between STMicroelectronics and INRIA (duration: 36 months; amount: 36,000 euros; signature: fall 2010) aims at supporting the developments done in the context of this PhD.

7.2.2. Mediacom Project with STMicroelectronics

Participants: Florent de Dinechin, Claude-Pierre Jeannerod, Jingyan Jourdan-Lu, Jean-Michel Muller, David Pfannholzer, Nathalie Revol.

We have been involved in Mediacom since September 1, 2009. Mediacom is a 40-month joint project with the Compiler Expertise Center (STMicroelectronics Grenoble) and INRIA project-teams Alchemy, Alf, and Compsys, and a Nano 2012 partner project. For Arénaire, it funds in particular the 3-year MEFI PhD grant of David Pfannholzer. The development this year is the generation of some elementary functions, focusing on the pre-processing (argument reduction, exception handling) and post-processing (argument reconstruction). Our long-term goal with this project is the design and implementation of a dynamic code generation tool, for numerical kernels typical of intensive mediaprocessing, and that could be integrated into production compilers.

7.2.3. STMicroelectronics CIFRE PhD Grant

Nicolas Brunie is supported by a CIFRE PhD grant (from 15/04/2011 to 14/04/2014) from Kalray. Its purpose is the study of a tightly-coupled reconfigurable accelerator to be embedded in the Kalray multicore processor. Advisors: Florent de Dinechin (Arénaire) and B. de Dinechin (Kalray). The support contract between Kalray and Inria amounts to 76,000 euros on three years.

7.2.4. Altera hardware donation

Altera donated to the team an FPGA-based acceleration card (Altera DK-DEV-4SGX530N) worth 8000 euros for the Table-Maker's Dilemma acceleration project.

8. Partnerships and Cooperations

8.1. Regional Initiatives

8.1.1. Cible Grant from Région Rhône-Alpes

Participants: Nicolas Brisebarre, Claude-Pierre Jeannerod, Mioara Joldeş, Jingyan Jourdan-Lu, Jean-Michel Muller, Nathalie Revol, Gilles Villard.

Since October 2008, we have obtained a 3-year grant from Région Rhône-Alpes. That grant has funded a PhD student, Mioara Joldeş, who defended her PhD thesis on September 26, 2011. The project consists in automating as much as possible the generation of code for approximating functions. Instead of calling functions from libraries, we wish to elaborate approximations at compile-time, in order to be able to directly approximate compound functions, or to take into account some information (typically, input range information) that might be available at that time. In this project, we collaborate with the STMicroelectronics' Compilation Expertise Center in Grenoble (C. Bertin, H. Knochel, and C. Monat). STMicroelectronics is funding another PhD grant on these themes.

8.2. National Initiatives

8.2.1. ANR HPAC Project

Participants: Claude-Pierre Jeannerod, Nicolas Louvet, Nathalie Revol, Damien Stehlé, Philippe Théveny, Gilles Villard.

"High-performance Algebraic Computing" (HPAC) is a four year ANR project that will start in January 2012. HPAC is headed by Jean-Guillaume Dumas (CASYS team, LJK laboratory, Grenoble); it involves Arénaire as well as the INRIA project-team MOAIS (LIG, Grenoble), the INRIA project-team SALSA (LIP6 lab., Paris), the ARITH group (LIRMM laboratory, Montpellier), and the HPC Project company.

The overall ambition of HPAC is to provide international reference high-performance libraries for exact linear algebra and algebraic systems on multi-processor architecture and to influence parallel programming approaches for algebraic computing. The central goal is to extend the efficiency of the LinBox and FGb libraries to new trend parallel architectures such as clusters of multi-processor systems and graphics processing units in order to tackle a broader class of problems in lattice cryptography and algebraic cryptanalysis. HPAC will conduct researches along three axes:
- A domain specific parallel language (DSL) adapted to high-performance algebraic computations;
- Parallel linear algebra kernels and higher-level mathematical algorithms and library modules;
- Library composition and innovative high performance solutions for cryptology challenges.

8.2.2. ANR TaMaDi Project

Participants: Nicolas Brisebarre, Florent de Dinechin, Guillaume Hanrot, Vincent Lefèvre, Érik Martin-Dorel, Micaela Mayero, Jean-Michel Muller, Andrew Novocin, Ioana Pasca, Damien Stehlé, Serge Torres.

The TaMaDi project (Table Maker's Dilemma, 2010-2013) is funded by the ANR and headed by Jean-Michel Muller. It was submitted in January 2010, accepted in June, and started in October 2010. The other French teams involved in the project are the MARELLE team-project of INRIA Sophia Antipolis-Méditerranée, and the PEQUAN team of LIP6 lab., Paris.

The aim of the project is to find "hardest to round" (HR) cases for the most common functions and floating-point formats. In floating-point (FP) arithmetic having fully-specified "atomic" operations is a key-requirement for portable, predictable and provable numerical software. Since 1985, the four arithmetic operations and the square root are IEEE specified (it is required that they should be correctly rounded: the system must always return the floating-point number nearest the exact result of the operation). This is not fully the case for the basic mathematical functions (sine, cosine, exponential, etc.). Indeed, the same function, on the same argument value, with the same format, may return significantly different results depending on the environment. As a consequence, numerical programs using these functions suffer from various problems. The lack of specification is due to a problem called the Table Maker's Dilemma (TMD). To compute $f(x)$ in a given format, where x is a FP number, we must first compute an approximation to $f(x)$ with a given precision, which we round to the nearest FP number in the considered format. The problem is the following: finding what the accuracy of the approximation must be to ensure that the obtained result is always equal to the "exact" $f(x)$ rounded to the nearest FP number. In the last years, our team-project and the CACAO team-project of INRIA Nancy-Grand Est designed algorithms for finding hardest-to-round cases. These algorithms do not allow to tackle with large formats. The TaMaDi project mainly focuses on three aspects:

- big precisions: we must get new algorithms for dealing with precisions larger than double precision. Such precisions will become more and more important (even if double precision may be thought as more than enough for a final result, it may not be sufficient for the intermediate results of long or critical calculations);
- formal proof: we must provide formal proofs of the critical parts of our methods. Another possibility is to have our programs generating certificates that show the validity of their results. We should then focus on proving the certificates;
- aggressive computing: the methods we have designed for generating HR points in double precision require weeks of computation on hundreds of PCs. Even if we design faster algorithms, we must massively parallelize our methods, and study various ways of doing that.

There was a meeting in Sophia-Antipolis in February 2011, and two other ones in Lyon in June and December 2011. The various documents can be found at http://tamadiwiki.ens-lyon.fr/tamadiwiki/index.php/Main_Page.

8.2.3. ANR TCHATER Project

Participants: Florent de Dinechin, Honoré Takeugming, Gilles Villard.

The TCHATER project (Terminal Cohérent Hétérodyne Adaptatif TEmps Réel, 2008-2010) is a collaboration between Alcatel-Lucent France, E2V Semiconductors, GET-ENST and the INRIA Arénaire and ASPI project/teams. Its purpose is to demonstrate a coherent terminal operating at 40Gb/s using real-time digital signal processing and efficient polarization division multiplexing. In Lyon, we studied the FPGA implementation of specific algorithms for polarization demultiplexing and forward error correction with soft decoding.

TCHATER was extended by the ANR until 9/06/2011, which allowed us to finalize the demonstrator.

8.2.4. ANR LaRedA Project

Participants: Fabien Laguillaumie, Adeline Langlois, Ivan Morel, Xavier Pujol, Damien Stehlé.

The LaRedA project (Lattice Reduction Algorithms, 2008-2011) is funded by the ANR and headed by Brigitte Vallée (CNRS/GREYC) and Valérie Berthé (CNRS/LIRMM). The aim of the project is to finely analyze lattice reduction algorithms such as LLL, by using experiments, probabilistic tools and dynamic analysis. Among the major goals are the average-case analysis of LLL and its output distribution. In Lyon, we concentrate on the experimental side of the project (by using fpLLL and MAGMA) and the applications of lattice reduction algorithms to cryptography.

8.3. European Initiatives

8.3.1. Other European Initiatives

- Guillaume Hanrot and Damien Stehlé collaborate with Cong Ling (Imperial College London, UK) on lattices and communication theory. The collaboration is jointly funded by the CNRS and the Royal Society, from January 2011 to December 2012.

8.4. International Initiatives

8.4.1. INRIA International Partners

- Nathalie Revol chairs the IEEE 1788 working group on the standardization of interval arithmetic, cf. http://grouper.ieee.org/groups/1788/.

8.4.2. Visits of International Scientists

- San Ling (Nanyang Technological University, Singapore) visited for two months (March and April), for collaborating on lattice-based cryptography. Visit partly funded by NTU and Inria Rhône-Alpes (invited researcher).
- Xiao-Wen Chang (McGill University, Canada) visited for one month (July), for collaborating on the numerical aspects of lattice reduction algorithms. Visit funded by ENS de Lyon (invited professor).
- Ron Steinfeld (Macquarie University, Australia) visited for one month (August), for collaborating on lattice-based cryptography. Visit funded by the French Embassy in Australia.

8.4.3. Participation In International Programs

- Guillaume Hanrot and Damien Stehlé participate in the LaBaCry project (Lattice-Based Cryptography), with San Ling and Huaxiong Wang (Cryptography and Coding group of Nanyang Technological University, Singapore). Project jointly funded by NTU and the MERLION program from the French Embassy in Singapore.
- Damien Stehlé is a Partner Investigator in the Australian Research Council Discovery Grant *Lattices as a Constructive and Destructive Tool in Cryptography*, with Christophe Doche, Igor Shparlinski and Ron Steinfeld (Macquarie University).
- Florent de Dinechin was invited 4 months by Nizhniy Novgorod State University (Russia).

9. Dissemination

9.1. Animation of the scientific community

- Florent de Dinechin was in the program committee of HEART 2011 (Highly Efficient Accelerators for Reconfigurable Computing), FPL 2011 (Field-Programmable Logic), FPT 2011 (Field-Programmable Technologies) and ARC 2011 (Applied Reconfigurable Computing). He was a reviewer for the thesis of Pedro Echeverria (U. Madrid) and the habilitation thesis of Steven Derrien (U. Rennes 1). He gave talks in the Radiophysics department of Nizhniy Novgorod State University and in the Intel Russian Summer School.
- Claude-Pierre Jeannerod was in the program committee of SNC 2011 (Fourth International Workshop on Symbolic-Numeric Computation).
- Jean-Michel Muller was in the program committee of ARITH'20 (20th IEEE Symposium on Computer Arithmetic) and ASAP'2011 (Application-Specific Systems, Architectures and Processors). He was a reviewer of the Habilitation thesis of D. Ménard (U. Rennes 1) and chaired the board of examiners for the PhD defense of M. Mezzarobba (École Polytechnique). He was a member (until December 2011) of the Scientific Council of Grenoble INP, and is a member of the Scientific Councils of Ecole Normale Supérieure de Lyon and CERFACS.
- Nathalie Revol was in the program committee of NSV 11 (Fourth International Workshop on Numerical Software Verification) and of MACIS 2011 (Fourth International Conference on Mathematical Aspects of Computer and Information Sciences).
- Damien Stehlé was in the program committees of ACISP 2011 (16th Australasian Conference on Information Security and Privacy), PQCRYPTO 2011 (4th International Conference on Post-Quantum Cryptography), CANS 2011 (10th International Conference on Cryptography and Network Security) and INDOCRYPT 2011 (12th International Conference on Cryptology in India). He was in the program committee of C2 (National Workshop on Coding and Cryptography, Oléron). He gave invited talks at the RAIM workshop (Rencontres Arithmétiques de l'Informatique Mathématique, Perpignan), at the NUS/NTU Workshop on Coding, Cryptology and Combinatorial Design (Singapore), at the International Workshop on Coding and Cryptology (IWCC 2011, Qingdao, China) and at the Magic@LIX workshop (Palaiseau).
- Vincent Lefèvre gave a talk at the Dagstuhl Seminar 11371 on Uncertainty modeling and analysis with intervals: Foundations, tools, applications. [29]
- Gilles Villard is chair of the LIP laboratory since January 2009.

9.2. Teaching

Licence : *Computer Architecture*, 71h, L2, U. Claude Bernard, taught by Nicolas Louvet.

Licence : *Operating Systems*, 35h, L3, U. Claude Bernard, taught by Nicolas Louvet.

Licence : *Algorithms and Data Structures*, 32h, L3, U. Claude Bernard, taught by Nicolas Louvet.

Master: *Arithmetic Algorithms*, 24h, M1, École normale supérieure de Lyon, taught by Eleonora Guerrini, Guillaume Hanrot, and Claude-Pierre Jeannerod (autumn 2011).

Master: *Reconfigurable Computing*, 24h, M2, École normale supérieure de Lyon, taught by Florent de Dinechin (autumn 2011).

Master: *Cryptology: new primitives and applications*, 24h, M2, École normale supérieure de Lyon, taught by Guillaume Hanrot and Damien Stehlé (autumn 2011).

Master: *Algorithms for Verified Linear Algebra*, 24h, M2, École normale supérieure de Lyon, taught by Claude-Pierre Jeannerod, Nicolas Louvet, and Nathalie Revol (autumn 2011).

Master: *Numerical Algorithms*, 36h, M2, U. Claude Bernard, taught by Claude-Pierre Jeannerod and Nicolas Louvet.

Master: *Computer Arithmetic*, 28h, M2, U. Claude Bernard, taught by Vincent Lefèvre.

Doctorate: *Floating-Point Arithmetic*, 12h, Nizhniy Novgorod State University, taught by Florent de Dinechin (spring 2011).

PhD & HdR:

HdR: Damien Stehlé, *Euclidean lattices: algorithms and cryptography* [15], ENS de Lyon - Université de Lyon, defended on October 14, 2011.

HdR: Fabien Laguillaumie, *Public-Key Cryptography: Design and Algorithmic*, Université de Caen, defended on December 12, 2011.

PhD: Mioara Joldes, *Rigorous Polynomial Approximations and Applications* [11], ENS de Lyon - Université de Lyon, defended on September 26, 2011. Supervisors: Nicolas Brisebarre and Jean-Michel Muller.

PhD: Christophe Mouilleron, *Efficient computation with structured matrices and arithmetic expressions* [12], ENS de Lyon - Université de Lyon, defended on November 4, 2011. Supervisors: Claude-Pierre Jeannerod and Gilles Villard.

PhD: Hong Diep Nguyen, *Efficient algorithms for verified scientific computing: Numerical linear algebra using interval arithmetic* [24], ENS de Lyon - Université de Lyon, defended on January 18, 2011. Supervisors: Nathalie Revol and Gilles Villard.

PhD: Bogdan Pasca, *High-performance floating-point computing on reconfigurable circuits* [14], ENS de Lyon - Université de Lyon, defended on September 21, 2011. Supervisor: Florent de Dinechin.

PhD in progress: Nicolas Brunie, *Embedding a tightly-coupled reconfigurable accelerator in multi-core processor*, started in december 2010. Supervisor: Florent de Dinechin.

PhD in progress: Jingyan Jourdan-Lu, *Floating-point arithmetic and compilation for embedded processors*, started in March 2009. Supervisors: Claude-Pierre Jeannerod, Christophe Monat (STMicroelectronics Compilation Expertise Center), and Jean-Michel Muller. Jingyan has been on maternity leave from November 2010 to May 2011.

PhD in progress: Adeline Langlois, *Cryptography based on ideal lattices*, started in September 2011. Supervisors: Guillaume Hanrot and Damien Stehlé.

PhD in progress: Érik Martin-Dorel, *Contributions to the elaboration of arithmetic algorithms with their formal proof*, started in October 2009. Supervisors: Micaela Mayero and Jean-Michel Muller.

PhD in progress: Adrien Panhaleux, *Floating-point arithmetic algorithms*, started in September 2008. Supervisors: Nicolas Louvet and Jean-Michel Muller.

PhD in progress: David Pfannholzer, *Generation of specialized numerical codes*, started in November 2009, until October 2011. Supervisors: Florent de Dinechin and Nathalie Revol.

PhD in progress: Xavier Pujol, *Efficient lattice reduction algorithms*, started in September 2009. Supervisors: Guillaume Hanrot and Damien Stehlé.

PhD in progress: Philippe Théveny, *Numerical quality and high performance in scientific computing on emerging architectures*, started in October 2011. Supervisor: Nathalie Revol.

PhD in progress: Serge Torres, *Some tools for the design of efficient and reliable function evaluation libraries*, started in September 2010. Supervisors: Nicolas Brisebarre and Jean-Michel Muller.

Other teaching:

Nathalie Revol gave conferences at “Lycée Lalande” (Bourg-en-Bresse), “Collège Jean Zay” (Brignais), “Collège Eugénie de Pomey” (Amplepuis) and “Collège Pablo Picasso” (Bron).

Jean-Michel Muller gave an introductory talk on Floating-Point Arithmetic at the “Journées nationals de l’association des professeurs de Mathématiques de l’enseignement Public”, Grenoble, Oct. 2011.

Damien Stehlé gave introductory talks on lattice-based cryptography at ENS de Casablanca and ENSA de SAFI (Morocco) .

Vincent Lefèvre gave a presentation of GNU MPFR at the GNU Hackers Meeting in Paris, Aug. 2011.

10. Bibliography

Major publications by the team in recent years

[1] A. BOSTAN, C.-P. JEANNEROD, É. SCHOST. *Solving structured linear systems with large displacement rank*, in "Theoretical Computer Science", November 2008, vol. 407, n^{o} 1:3, p. 155–181.

[2] N. BRISEBARRE, J.-M. MULLER, A. TISSERAND. *Computing machine-efficient polynomial approximations*, in "ACM Transactions on Mathematical Software", June 2006, vol. 32, n^{o} 2, p. 236–256.

[3] J. DETREY, F. DE DINECHIN. *Parameterized floating-point logarithm and exponential functions for FPGAs*, in "Microprocessors and Microsystems, Special Issue on FPGA-based Reconfigurable Computing", December 2007, vol. 31, n^{o} 8, p. 537–545, http://dx.doi.org/10.1016/j.micpro.2006.02.008.

[4] G. HANROT, V. LEFÈVRE, D. STEHLÉ, P. ZIMMERMANN. *Worst Cases of a Periodic Function for Large Arguments*, in "Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH-18)", IEEE computer society, 2007, p. 133–140, http://doi.ieeecomputersociety.org/10.1109/ARITH.2007.37.

[5] G. HANROT, D. STEHLÉ. *Improved Analysis of Kannan’s Shortest Lattice Vector Algorithm (Extended Abstract)*, in "Proceedings of Crypto 2007", LNCS, Springer, 2007, vol. 4622, p. 170–186.

[6] C.-P. JEANNEROD, G. VILLARD. *Essentially optimal computation of the inverse of generic polynomial matrices*, in "Journal of Complexity", 2005, vol. 21, n^{o} 1, p. 72–86.

[7] P. KORNERUP, C. LAUTER, V. LEFÈVRE, N. LOUVET, J.-M. MULLER. *Computing Correctly Rounded Integer Powers in Floating-Point Arithmetic*, in "ACM Transactions on Mathematical Software", 2010, vol. 37, n^{o} 1, p. 4:1-4:23.

[8] J.-M. MULLER, N. BRISEBARRE, F. DE DINECHIN, C.-P. JEANNEROD, V. LEFÈVRE, G. MELQUIOND, N. REVOL, D. STEHLÉ, S. TORRES. *Handbook of Floating-Point Arithmetic*, Birkhäuser Boston, December 2010, ISBN: 978-0-8176-4704-9, http://hal.inria.fr/ensl-00379167/en.

[9] N. REVOL, K. MAKINO, M. BERZ. *Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY*, in "Journal of Logic and Algebraic Programming", 2005, vol. 64, p. 135–154.

[10] F. DE DINECHIN, C. LAUTER, J.-M. MULLER. *Fast and correctly rounded logarithms in double-precision*, in "Theoretical Informatics and Applications", 2007, vol. 41, p. 85-102.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] M. JOLDES. *Rigorous Polynomial Approximations and Applications*, École normale supérieure de Lyon - ENS LYON, September 2011, http://tel.archives-ouvertes.fr/tel-00657843/en.

[12] C. MOUILLERON. *Efficient computation with structured matrices and arithmetic expressions*, École normale supérieure de Lyon - ENS LYON, November 2011, http://hal-ens-lyon.archives-ouvertes.fr/ensl-00659014/en.

[13] H. D. NGUYEN. *Efficient algorithms for verified scientific computing: Numerical linear algebra using interval arithmetic*, École normale supérieure de Lyon - ENS LYON, January 2011, http://hal.inria.fr/ensl-00560188/en.

[14] B. PASCA. *High-performance floating-point computing on reconfigurable circuits*, École normale supérieure de Lyon - ENS LYON, September 2011, http://hal.inria.fr/tel-00654121/en.

[15] D. STEHLÉ. *Euclidean lattices: algorithms and cryptography*, École normale supérieure de Lyon - ENS LYON, October 2011, Habilitation à Diriger des Recherches, http://hal.inria.fr/tel-00645387/en.

Articles in International Peer-Reviewed Journal

[16] S. BOLDO, J.-M. MULLER. *Exact and Approximated error of the FMA*, in "IEEE Transactions on Computers", February 2011, vol. 60, n^{o} 2, p. 157-164 [*DOI :* 10.1109/TC.2010.139], http://hal.inria.fr/inria-00429617/en.

[17] X.-W. CHANG, D. STEHLÉ, G. VILLARD. *Perturbation Analysis of the QR Factor R in the Context of LLL Lattice Basis Reduction*, in "Mathematics of Computation", 2012, http://hal.inria.fr/ensl-00529425/en.

[18] P. COLLINS, M. NIQUI, N. REVOL. *A validated real function calculus*, in "Mathematics in Computer Science", December 2011, http://hal.inria.fr/hal-00641648/en.

[19] C.-P. JEANNEROD, H. KNOCHEL, C. MONAT, G. REVY. *Computing floating-point square roots via bivariate polynomial evaluation*, in "IEEE Transactions on Computers", February 2011, vol. 60, n^{o} 2, p. 214-227 [*DOI :* 10.1109/TC.2010.152], http://hal.inria.fr/ensl-00559236/en.

[20] C.-P. JEANNEROD, N. LOUVET, J.-M. MULLER, A. PANHALEUX. *Midpoints and exact points of some algebraic functions in floating-point arithmetic*, in "IEEE Transactions on Computers", February 2011, vol. 60, n^{o} 2, p. 228-241, Recherche en partie supportée par le "Pole de competitivite mondial" Minalogic et le projet ANR EVA-Flo. [*DOI :* 10.1109/TC.2010.144], http://hal.inria.fr/ensl-00409366/en.

[21] P. KORNERUP, V. LEFÈVRE, N. LOUVET, J.-M. MULLER. *On the Computation of Correctly-Rounded Sums*, in "IEEE Transactions on Computers", 2011 [*DOI :* 10.1109/TC.2011.27], http://hal.inria.fr/hal-00646179/en.

[22] P. KORNERUP, J.-M. MULLER, A. PANHALEUX. *Performing Arithmetic Operations on Round-to-Nearest Representations*, in "IEEE Transactions on Computers", February 2011, vol. 60, n^{o} 2, p. 282-291 [*DOI :* 10.1109/TC.2010.134], http://hal.inria.fr/ensl-00548988/en.

[23] C. LING, S. LIU, D. STEHLÉ. *Decoding by Sampling: A Randomized Lattice Algorithm for Bounded Distance Decoding*, in "IEEE Transactions on Information Theory", 2011, p. 5933-5945, http://hal.inria.fr/hal-00640634/en.

[24] H. D. NGUYEN, N. REVOL. *Solving and Certifying the Solution of a Linear System*, in "Reliable Computing", 2011, vol. 15, n^{o} 2, p. 120-131, The Reliable Computing journal has no more paper publication, only free, electronic publication., http://hal.inria.fr/inria-00546856/en.

[25] G. VILLARD. *Kaltofen's division-free determinant algorithm differentiated for matrix adjoint computation*, in "Journal of Symbolic Computation", 2011, vol. 46, n^{o} 7, p. 773-790 [*DOI :* 10.1016/J.JSC.2010.08.012], http://hal.inria.fr/ensl-00335918/en.

[26] F. DE DINECHIN. *Multiplication by rational constants*, in "IEEE Transactions on Circuits and Systems. Part II, Express Briefs", 2012 [*DOI :* 10.1109/TCSII.2011.2177706], http://hal.inria.fr/ensl-00610328/en.

[27] F. DE DINECHIN, C. LAUTER, G. MELQUIOND. *Certifying the floating-point implementation of an elementary function using Gappa*, in "IEEE Transactions on Computers", February 2011, vol. 60, n^{o} 2, p. 242-253 [*DOI :* 10.1109/TC.2010.128], http://hal.inria.fr/inria-00533968/en.

[28] F. DE DINECHIN, B. PASCA. *Designing Custom Arithmetic Data Paths with FloPoCo*, in "IEEE Design and Test of Computers", July 2011, vol. 28, p. 18-27, http://hal.inria.fr/ensl-00646282/en.

Invited Conferences

[29] V. LEFÈVRE. *Generating a Minimal Interval Arithmetic Based on GNU MPFR*, in "Uncertainty modeling and analysis with intervals: Foundations, tools, applications (Dagstuhl Seminar 11371)", Dagstuhl, Germany, I. E. ELISHAKOFF, V. KREINOVICH, W. LUTHER, E. D. POPOVA (editors), Dagstuhl Reports, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, December 2011, vol. 1, 43, Abstract only [*DOI :* 10.4230/DAGREP.1.9.26], http://hal.inria.fr/hal-00651939/en.

International Conferences with Proceedings

[30] N. BRISEBARRE, M. JOLDES, P. KORNERUP, É. MARTIN-DOREL, J.-M. MULLER. *Augmented precision square roots, 2-D norms, and discussion on correctly rounding* $\sqrt{x^2 + y^2}$, in "20th IEEE Symposium on Computer Arithmetic (ARITH-20)", Tübingen, Germany, IEEE Computer Society, July 2011, p. 23-30 [*DOI :* 10.1109/ARITH.2011.13], http://hal.inria.fr/ensl-00545591/en.

[31] G. HANROT, X. PUJOL, D. STEHLÉ. *Algorithms for the Shortest and Closest Lattice Vector Problems*, in "IWCC 2011", China, 2011, p. 159-190, http://hal.inria.fr/hal-00640637/en.

[32] G. HANROT, X. PUJOL, D. STEHLÉ. *Analyzing Blockwise Lattice Algorithms using Dynamical Systems*, in "CRYPTO 2011", United States, 2011, p. 447-464, http://hal.inria.fr/hal-00640638/en.

[33] W. HART, M. V. HOEIJ, A. NOVOCIN. *Practical polynomial factoring in polynomial time.*, in "Proceedings of ISSAC 2011", United States, 2011, p. 163-170, http://hal.inria.fr/hal-00650391/en.

[34] W. HART, A. NOVOCIN. *Practical Divide-and-Conquer Algorithms for Polynomial Arithmetic*, in "Proceedings of CASC 2011", Germany, 2011, p. 200-214, http://hal.inria.fr/hal-00650389/en.

[35] M. V. HOEIJ, J. KLÜNERS, A. NOVOCIN. *Generating subfields*, in "Proceedings of ISSAC 2011", United States, 2011, p. 345-352, http://hal.inria.fr/hal-00650392/en.

[36] C.-P. JEANNEROD, J. JOURDAN-LU, C. MONAT, G. REVY. *How to Square Floats Accurately and Efficiently on the ST231 Integer Processor*, in "20th IEEE Symposium on Computer Arithmetic (ARITH)", Tübingen, Germany, August 2011, p. 77-81 [*DOI :* 10.1109/ARITH.2011.19], http://hal.inria.fr/ensl-00644147/en.

[37] C. LING, S. LIU, L. LUZZI, D. STEHLÉ. *Decoding by Embedding: Correct Decoding Radius and DMT Optimality*, in "Proceedings of ISIT 2011", Russian Federation, 2011, p. 1106 - 1110, http://hal.inria.fr/hal-00640636/en.

[38] C. MOUILLERON, G. REVY. *Automatic Generation of Fast and Certified Code for Polynomial Evaluation*, in "20th IEEE Symposium on Computer Arithmetic (ARITH)", Tübingen, Germany, August 2011, p. 233-242 [*DOI :* 10.1109/ARITH.2011.39], http://hal.inria.fr/ensl-00531721/en.

[39] H. D. NGUYEN, B. PASCA, T. PREUSSER. *FPGA-Specific Arithmetic Optimizations of Short-Latency Adders*, in "2011 International Conference on Field Programmable Logic and Applications (FPL)", Chania, Greece, 2011, September 2011, p. 232 - 237 [*DOI :* 10.1109/FPL.2011.49], http://hal.inria.fr/ensl-00542389/en.

[40] A. NOVOCIN, D. STEHLÉ, G. VILLARD. *An LLL-reduction algorithm with quasi-linear time complexity*, in "STOC'11 - 43rd annual ACM symposium on Theory of computing", San Jose, United States, ACM New York, NY, USA, 2011, p. 403-412 [*DOI :* 10.1145/1993636.1993691], http://hal.inria.fr/ensl-00534899/en.

[41] D. STEHLÉ, R. STEINFELD. *Making NTRU as secure as worst-case problems over ideal lattices*, in "Proceedings of EUROCRYPT 2011", Estonia, 2011, p. 27-47, http://hal.inria.fr/hal-00640635/en.

[42] G. VILLARD. *Recent progress in linear algebra and lattice basis reduction (invited)*, in "ISSAC'11 - International symposium on Symbolic and algebraic computation", San Jose, United States, ACM proceedings, 2011, p. 3-4 [*DOI :* 10.1145/1993886.1993889], http://hal.inria.fr/hal-00644796/en.

[43] F. DE DINECHIN. *The arithmetic operators you will never see in a microprocessor*, in "20th IEEE Symposium of Computer Arithmetic", Germany, IEEE, July 2011, p. 189-190, http://hal.inria.fr/ensl-00642164/en.

[44] F. DE DINECHIN, L.-S. DIDIER. *Table-based division by small integer constants*, in "Applied Reconfigurable Computing", Hong Kong, Hong Kong, March 2012, http://hal.inria.fr/ensl-00642145/en.

[45] *Best Paper*
F. DE DINECHIN, J.-M. MULLER, B. PASCA, A. PLESCO. *An FPGA architecture for solving the Table Maker's Dilemma*, in "Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on", Santa Monica, United States, IEEE Computer Society, 2011, p. 187-194, Cet article a obtenu le "best paper award" de la conférence [*DOI :* 10.1109/ASAP.2011.6043267], http://hal.inria.fr/ensl-00640063/en.

National Conferences with Proceeding

[46] S. COLLANGE. *Une architecture unifiée pour traiter la divergence de contrôle et la divergence mémoire en SIMT*, in "SYMPosium en Architectures", Saint-Malo, France, May 2011, http://hal.inria.fr/hal-00576049/en.

Conferences without Proceedings

[47] H. D. NGUYEN, N. REVOL. *Refining and verifying the solution of a linear system*, in "SNC 2011 - Symbolic Numeric Computation", San Jose, United States, ACM Digital Library, June 2011, http://hal.inria.fr/hal-00641659/en.

[48] N. REVOL, H. D. NGUYEN. *Refining and verifying efficiently the solution of a linear system*, in "Dagstuhl Seminar 11371: Uncertainty modeling and analysis with intervals: Foundations, tools, applications", Dagstuhl, Germany, September 2011, http://hal.inria.fr/hal-00641669/en.

[49] N. REVOL. *IEEE 1788 Working Group for the Standardization of Interval Arithmetic: a brief overview*, in "Dagstuhl Seminar 11371: Uncertainty modeling and analysis with intervals: Foundations, tools, applications", Dagstuhl, Germany, September 2011, http://hal.inria.fr/hal-00641674/en.

[50] N. REVOL. *Verified Numerical Linear Algebra: Linear System Solving*, in "2011 SIAM Conference on Applied Algebraic Geometry", Raleigh, United States, October 2011, http://hal.inria.fr/hal-00641663/en.

Research Reports

[51] C. ALIAS, B. PASCA, A. PLESCO. *FPGA-Specific Synthesis of Loop-Nests with Pipelined Computational Cores*, INRIA, July 2011, n^{o} RR-7674, http://hal.inria.fr/inria-00606977/en.

[52] S. BOLDO, F. CLEMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND, P. WEIS. *Wave Equation Numerical Resolution: Mathematics and Program*, INRIA, December 2011, n^{o} RR-7826, http://hal.inria.fr/hal-00649240/en.

[53] N. BRISEBARRE, M. JOLDES, É. MARTIN-DOREL, M. MAYERO, J.-M. MULLER, I. PASCA, L. RIDEAU, L. THÉRY. *Rigorous Polynomial Approximation using Taylor Models in Coq*, December 2011, http://hal.inria.fr/ensl-00653460/en.

[54] N. BRUNIE, S. COLLANGE. *Assouplir les contraintes des architectures SIMT à faible coût*, December 2011, http://hal.inria.fr/ensl-00649186/en.

[55] N. BRUNIE, S. COLLANGE, G. DIAMOS. *Simultaneous Branch and Warp Interweaving for Sustained GPU Performance*, December 2011, http://hal.inria.fr/ensl-00649650/en.

[56] N. BRUNIE, F. DE DINECHIN, B. DE DINECHIN. *Mixed-precision Fused Multiply and Add*, November 2011, http://hal.inria.fr/ensl-00642157/en.

[57] S. COLLANGE. *Identifying scalar behavior in CUDA kernels*, January 2011, http://hal.inria.fr/hal-00555134/en.

[58] S. COLLANGE. *Stack-less SIMT reconvergence at low cost*, September 2011, http://hal.inria.fr/hal-00622654/en.

[59] S. COLLANGE, A. KOUYOUMDJIAN. *Affine Vector Cache for memory bandwidth savings*, December 2011, http://hal.inria.fr/ensl-00649200/en.

[60] C.-P. JEANNEROD, N. LOUVET, J.-M. MULLER. *Further analysis of Kahan's algorithm for the accurate computation of 2 x 2 determinants*, December 2011, http://hal.inria.fr/ensl-00649347/en.

[61] C.-P. JEANNEROD, C. PERNET, A. STORJOHANN. *Rank-profile revealing Gaussian elimination and the CUP matrix decomposition*, December 2011, http://hal.inria.fr/hal-00655543/en.

[62] V. LEFÈVRE. *SIPE: Small Integer Plus Exponent*, INRIA, December 2011, n^{o} RR-7832, http://hal.inria.fr/hal-00650659/en.

[63] É. MARTIN-DOREL. *Univariate and bivariate integral roots certificates based on Hensel's lifting*, March 2011, n^{o} RRLIP2011-1, http://hal.inria.fr/ensl-00575673/en.

[64] É. MARTIN-DOREL, G. MELQUIOND, J.-M. MULLER. *Some issues related to double roundings*, November 2011, http://hal.inria.fr/ensl-00644408/en.

[65] D. SAMPAIO, R. MARTINS, S. COLLANGE, F. MAGNO QUINTÃO PEREIRA. *Divergence Analysis with Affine Constraints*, November 2011, http://hal.inria.fr/hal-00650235/en.

[66] A. VAZQUEZ, J. BRUGUERA. *Composite Iterative Algorithm and Architecture for q-th Root Calculation*, INRIA, March 2011, n^{o} RR-7564, http://hal.inria.fr/inria-00575573/en.

References in notes

[67] E. KALTOFEN, G. VILLARD. *On the complexity of computing determinants*, in "Computational Complexity", 2004, vol. 13, p. 91–130.

[68] J.-M. MULLER. *Elementary Functions, Algorithms and Implementation*, Birkhäuser Boston, 2nd Edition, 2006.

[69] F. DE DINECHIN, A. TISSERAND. *Multipartite table methods*, in "IEEE Transactions on Computers", 2005, vol. 54, n^{o} 3, p. 319-330.