INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Arénaire

Computer arithmetic

Grenoble - Rhône-Alpes

Theme : Algorithms, Certification, and Cryptography

Table of contents

1. Team

Research Scientists

Nicolas Brisebarre [Junior Researcher CNRS]
Claude-Pierre Jeannerod [Junior Researcher INRIA]
Vincent Lefèvre [Junior Researcher INRIA]
Micaela Mayero [Junior Researcher INRIA (on partial secondment), until August 2010]
Jean-Michel Muller [Senior Researcher CNRS, HdR]
Nathalie Revol [Junior Researcher INRIA]
Damien Stehlé [Junior Researcher CNRS, back in July 2010 after being seconded to Macquarie University and the University of Sydney]
Gilles Villard [Senior Researcher CNRS, HdR]

Faculty Members

Sylvain Collange [ATER ENS de Lyon, since October 2010]
Florent de Dinechin [Team leader, Associate Professor ENS de Lyon, HdR]
Guillaume Hanrot [Professor ENS de Lyon, HdR]
Nicolas Louvet [Associate Professor UCBL]

Technical Staff

Honoré Takeugming [Engineer on the ANR *TCHATER* project]
Philippe Théveny [Engineer on the ANR *EVA-Flo* project until August 2010]
Serge Torres [Technical Staff, 40% on the project]

PhD Students

Nicolas Brunie [CIFRE grant (Kalray), 1st year]
Mioara Joldeş [*Allocataire-moniteur*, Région Rhône-Alpes grant, 3rd year]
Jingyan Jourdan-Lu [CIFRE grant (STMicroelectronics), 2nd year]
Érik Martin-Dorel [MESR grant, 2nd year]
Ivan Morel [*Allocataire-moniteur*, ENS grant, cotutelle with the University of Sydney, 4th year]
Christophe Mouilleron [*Allocataire-moniteur*, ENS grant, 3rd year]
Hong Diep Nguyen [INRIA grant, thesis defended on January 18, 2011]
Adrien Panhaleux [*Allocataire-moniteur*, ENS grant, 3rd year]
Bogdan Pasca [*Allocataire-moniteur*, MESR grant, 3rd year]
David Pfannholzer [MEFI grant, 2nd year]
Xavier Pujol [*Allocataire-moniteur*, ENS grant, 2nd year]

Post-Doctoral Fellows

Andrew Novocin [ENS Lyon]
Ioana Pasca [INRIA, ANR *TaMaDi*, since December 2010]
Álvaro Vázquez Álvarez [INRIA grant, since October 2009]

Administrative Assistants

Sèverine Morin [CNRS, TR INRIA, until November 15, 50% on the project]
Damien Séon [ENS de Lyon, TR INRIA, since November 15, 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, lattice basis reduction, 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

- Three Arénaire members described the first LLL-reducing algorithm with a time complexity that is quasi-linear in the bit-length β of the entries and polynomial in the dimension d (see 6.5.1).
- Arénaire members organized the SCAN 2010 conference in Lyon (see 9.1).
- Arénaire members are designing the floating-point unit of a commercial processor (see 7.1.1).

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 [75]). 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 [76] 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 [75] 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 [74]). 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 2010.

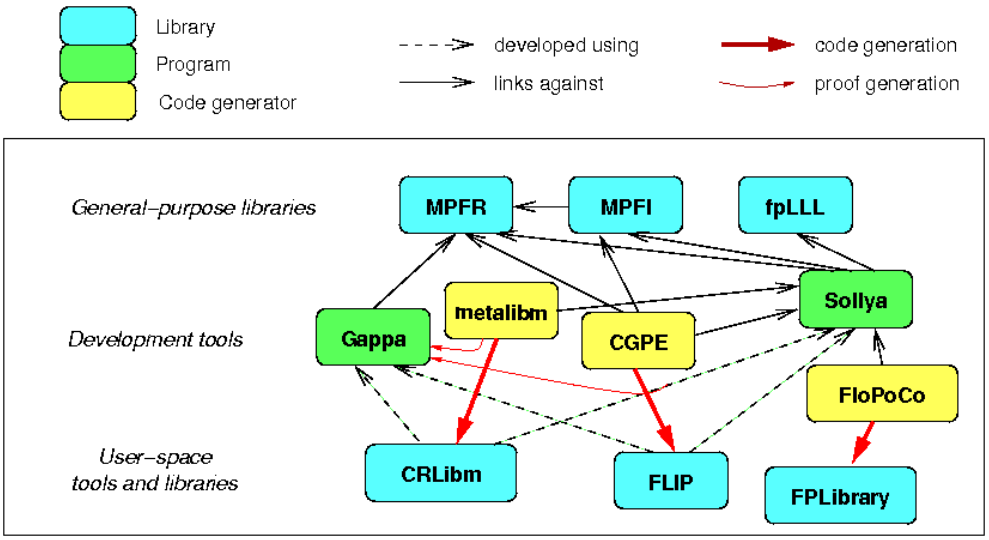

Figure 1. Relationships between some Arénaire developments.

5.2. 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. MPFR 3.0.0 was released on 10 June 2010.

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.

5.3. MPFI

Participants: Nathalie Revol, Hong Diep Nguyen, Philippe Théveny.

MPFI is a library in C for interval arithmetic using arbitrary precision (arithmetic and algebraic operations, elementary functions, operations on sets). It is based on MPFR (see §5.2). MPFI is maintained on par with MPFR: it offers the interval counterpart of most mathematical functions provided by MPFR. In August 2010, MPFI version 1.5 has been released: it offers more mathematical functions from MPFR, a modular structure and an extensive test suite, covering 99.7% of the lines of code. Minor corrections have been made.

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

- ACM: D.2.2 (Software libraries), G.4 (Mathematical software)
- AMS: 65G30 (Interval and finite arithmetic)
- Software benefit: interval arithmetic using arbitrary precision and outward rounding modes for a guarantee of inclusion
- License: LGPL
- Type of human computer interaction: C library, callable from any C program.
- OS/Middleware: any, as long as a C compiler is available (Cygwin on Windows)
- Required library or software: GMP 4.2.2 or above (gmplib.org) and MPFR 2.4 or above
- Programming language: C/C++
- Documentation: available in formats generated from texinfo.

5.4. Exhaustive Tests for the Correct Rounding of Mathematical Functions

Participant: Vincent Lefèvre.

The programs to 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 have slightly been improved. Amit Verma, an internship student, worked on these programs with the goal of improving the algorithms in a future full rewrite of the code.

5.5. fpLLL: lattice reduction using floating-point arithmetic

Participants: Damien Stehlé, Ivan Morel, Xavier Pujol, Gilles Villard.

FPLLL is a library which allows to perform several fundamental tasks on Euclidean lattices, most notably compute a LLL-reduced basis and compute a shortest non-zero vector in a lattice. The latter functionality has been added in July 2008 by X. Pujol at the end of his Master's degree internship with D. Stehlé. This led to the release of version 3.0 of fpLLL. The fpLLL library relies on floating-point arithmetic for all the computations involving the Gram-Schmidt orthogonalisation of the lattice bases under scope.

FPLLL is becoming the reference for lattice reduction. It is, or adaptations of it are, included in SAGE, PARI GP, and MAGMA.

URL: http://perso.ens-lyon.fr/damien.stehle

- License: LGPL
- 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: GMP 4.2.0 or above (gmplib.org) and MPFR 2.3.0 or above
- Programming language: C/C++
- Documentation: available in txt format.

5.6. FloPoCo: a Floating-Point Core generator for FPGAs

Participants: Florent de Dinechin, Bogdan Pasca, Mioara Joldeş.

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 2010, the pipeline framework of FloPoCo has been improved again (see 6.1.1). With respect to operators, most work this year was dedicated to the foundation operators: integer adder and multipliers, pipelined for arbitrary size to arbitrary frequency, and optimized for area and latency in a target-specific way. In addition, new non standard operators have also been implemented. The most important is a generic polynomial approximator using Sollya. This was developed with square root as a test case, then used for an overhaul of the exponential function. Other new operators include KCM-based multipliers by a constant and floating-point multiply-and-add.

Versions 2.0 and 2.1 were released in 2010.

Among the known users of FloPoCo are University of Cape Town, South Africa, U.T. Cluj-Napoca, Imperial College, University of 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.

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

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

5.7. CGPE: Code Generation for Polynomial Evaluation

Participants: Christophe Mouilleron, Claude-Pierre Jeannerod.

The CGPE project, created last year and 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. As the CGPE tool is used to generate significant parts of the codes in the FLIP library (see Section §5.8), speeding-up the tool has become an issue. Thus, several techniques based on or similar to abstract interpretation were introduced so as to select fast and accurate evaluation schemes more quickly.

- 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, GMP, MPFR, and MPFI
- Programming Language: C++
- Status: beta
- Documentation: available in html format on **URL:** http://cgpe.gforge.inria.fr/

5.8. 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 worked on improving the design and implementation of the following operators: square, scaleb, fused square-add, fused multiply-add, and butterfly. Speed-ups obtained via specialization range from 1.75 (square vs. multiply) to almost 2 (fused square-add vs. fused multiply-add). Furthermore, a framework has been set up which allows for each operator to simultaneously generate two codes, one for single precision (binary32) and one for double precision (binary64), both using 32-bit integer arithmetic.

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

6. New Results

6.1. Hardware Arithmetic Operators

Participants: Florent de Dinechin, Mioara Joldeş, Hong Diep Nguyen, Bogdan Pasca, Honoré Takeugming, Álvaro Vázquez Álvarez.

6.1.1. The FloPoCo Framework

From a user point of view, FloPoCo is an operator generator, but from a designer point of view it is also a framework that makes VHDL generation and testing much easier. F. de Dinechin and B. Pasca have improved this framework further [73]. It now exposes a very clear semantic separation between the functionality, described as a combinatorial VHDL component, and the pipeline construction, handled automatically by the tool. Pipeline construction is frequency-directed and target specific, so that a single C++ code generates pipelines optimized for a range of FPGA targets. Among other novelties is the possibility to share cycles across sub-components.

In collaboration with Ch. Alias and A. Plesco from the Compsys team, B. Pasca has also shown that FloPoCo can be used as a back-end to high-level synthesis tools [61]. The novelty here is to adapt the placement and scheduling of operations to exploit the internal pipeline of each operator, which is itself computed by FloPoCo.

6.1.2. Core Integer Operators for Reconfigurable Computing

Complex operators are built out of basic arithmetic blocks: large integer adders and multipliers. As FloPoCo now targets high precisions (binary64, and beyond) and high frequencies, these basic operators must themselves be pipelined. F. de Dinechin, H.-D. Nguyen and B. Pasca designed a range of novel architectures for pipelined addition. They were evaluated, and modelled so that the tool automatically chooses the best for a given context [49], [70]. For integer multipliers, B. Pasca and S. Banescu (an internship student), with F. de Dinechin, modelled the capabilities of the FPGA's DSP blocks as a set of tiles, so that the construction of a minimal-size multiplier can be expressed as a tiling problem [27]. This enables the construction of large-precision multiplier operating close to the DSP nominal frequency. These multipliers can also be truncated in contexts that allow it.

6.1.3. FPGA Operators based on Polynomial Evaluation

F. de Dinechin, M. Joldeş and B. Pasca worked on a generic polynomial evaluator for FloPoCo [47]. This generator inputs a function (provided as a mathematical expression), and a target precision, and ouptuts VHDL code of an operator that evaluates this function with last-bit accuracy to this precision. It relies on Sollya for polynomial approximation, and performs the evaluation with systematic truncation of all the values to the smallest precision required, in particular using truncated multipliers. This operator is used to build multiplicative square roots [48], and a floating-point exponential that is much better than the previous state of the art [50]. Meanwhile the logarithm still uses a recurrence method [72].

6.1.4. Decimal Arithmetic

Decimal arithmetic now belongs to the IEEE 754-2008 standard. A. Vázquez, with E. Antelo from U. Santiago de Compostela and P. Montuschi, from Politecnico di Torino, investigated hardware decimal multipliers units for microprocessors [21], and sharing hardware between decimal and binary operators [45].

Another option is to rely on FPGA acceleration for high-performance decimal computing. A. Vázquez studied with F. de Dinechin how to build large adder trees [60], then high-performance parallel multipliers [46] using efficiently the low-level architecture of modern FPGAs, in particular the 6-input look-up-table structure and fast-carry networks.

6.1.5. Hardware Complex Polynomial Evaluation

Milos Ercegovac (Univ. of California at Los Angeles) and Jean-Michel Muller proposed [14] an efficient hardware-oriented method for evaluating complex polynomials. The method is based on solving iteratively a system of linear equations. The solutions are obtained digit-by-digit on simple and highly regular hardware. The operations performed are defined over the reals. They described a complex-to-real transform, a complex polynomial evaluation algorithm, the convergence conditions, and a corresponding design and implementation. The main features of the method are: the latency of about m cycles for an m-bit precision; the cycle time independent of the precision; a design consisting of identical modules; and digit-serial connections between the modules. The number of modules, each roughly corresponding to serial-parallel multiplier without a carry-propagate adder, is $2(n+1)$ for evaluating an n-th degree complex polynomial. The design allows

straightforward tradeoffs between latency and cost: a factor k decrease in cost leads to a factor k increase in latency. The proposed method is attractive for programmable platforms because of its regular and repetitive structure of simple hardware operators.

6.1.6. High-performance Digital Signal Processing using FPGAs

For the TCHATER ANR project, F. de Dinechin and H. Takeugming designed with J.-M. Tanguy (Alcatel) a 128-tap FIR filter able to process 20 giga-samples per second in a single Stratix-IV FPGA [51]. Using an FFT approach, this amounts to more than 2 tera-operations per second. This was made possible by the design of a range of ad-hoc, small-precision constant multipliers.

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

Peter Kornerup, Vincent Lefèvre, Nicolas Louvet, and Jean-Michel Muller have presented a study of some basic blocks needed in the design of floating-point summation algorithms. In particular, they have shown that in a radix-2 arithmetic in any precision, 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 presented new results about the computation of the correctly-rounded sum of three floating-point numbers [58].

6.2.2. Implementing Decimal Floating-point Arithmetic through Binary

Nicolas Brisebarre, Milos Ercegovac (UC Los Angeles), Nicolas Louvet, Érik Martin-Dorel, Jean-Michel Muller and Adrien Panhaleux have proposed several algorithms and have provided some related results that make it possible to implement decimal floating-point arithmetic on a processor that does not have decimal operators, using the available binary floating- point functions. They have shown that several functions in decimal32 and decimal64 arithmetic can be implemented using binary64 and binary128 floating-point arithmetic, respectively. Specifically, they have discussed the decimal square root and some transcendental functions. They have also considered radix conversion algorithms [29].

6.2.3. Newton-Raphson Algorithms for Floating-Point Division Using an FMA

Since the introduction of the Fused Multiply and Add (FMA) in the IEEE-754-2008 standard for floating-point arithmetic, division based on Newton-Raphson's iterations becomes a viable alternative to SRT-based divisions. The Newton-Raphson iterations were already used in some architectures prior to the revision of the IEEE-754 standard. For example, the Itanium architecture already used this kind of iterations. Unfortunately, the proofs of the correctness of binary algorithms do not extend to the case of decimal floating-point arithmetic. Nicolas Louvet, Adrien Panhaleux and Jean-Michel Muller have presented general methods to prove the correct rounding of division algorithms using Newton-Raphson's iterations in software, for radix 2 and radix 10 floating-point arithmetic [37].

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 un-biased rounding-to-nearest to take place by a simple truncation, with the property that problems with double-roundings are avoided. Peter Kornerup (Odense University, Denmark), Jean-Michel Muller, and Adrien 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 [19].

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, Nicolas Brisebarre, Mioara Joldeş, Peter Kornerup (Odense University, Denmark), Érik Martin-Dorel, and Jean-Michel 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 [62].

6.2.6. Binary Floating-point Operators for VLIW Integer Processors

In a joint work with Christophe Monat (STMicroelectronics Compilation Expertise Center, Grenoble) and Guillaume Revy (DALI research team, Université de Perpignan and LIRMM laboratory), Claude-Pierre Jeannerod and Jingyan Jourdan-Lu focused on the problem of computing IEEE floating-point squares by means of integer arithmetic. They showed in [67] how the specific properties of squaring can be exploited in order to design and implement algorithms that have much lower latency than those for general multiplication, while still guaranteeing correct rounding. They also gave a C implementation for the binary32 format which, on targets like the ST231 VLIW integer processor from STMicroelectronics, is 1.75x faster than general multiplication.

In the invited paper [26], Claude-Pierre Jeannerod, Jingyan Jourdan-Lu, Christophe Mouilleron, and Jean-Michel Muller —jointly with Christian Bertin, Hervé Knochel, Christophe Monat (STMicroelectronics Compilation Expertise Center, Grenoble) and Guillaume Revy (DALI research team, Université de Perpignan and LIRMM laboratory)— reviewed the techniques and software tools used or developed so far for the FLIP library and, in particular, how they allowed very high instruction-level parallelism (ILP) exposure. The main points developed in this paper include a hierarchical description of function evaluation algorithms, the exploitation of the standard encoding of floating-point data, the automatic generation of fast and accurate polynomial evaluation schemes, and some compiler optimizations.

6.3. Correct Rounding of Elementary Functions

Participants: Florent de Dinechin, Vincent Lefèvre.

6.3.1. Search For Hardest-to-Round Cases (L-Algorithm)

A. Verma, an internship student, worked on Lefèvre's algorithm to search for hardest-to-round cases. The goal was to look for some way of improving the current algorithm, in particular, to extend the interval where the algorithm is applied, using the same degree-1 approximation of the function. The results look promising, but at this time, a proof could not be done (verification was done by comparing the results of the modified algorithm with the correct results).

6.3.2. Formal Proof Generation

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 [23]

6.4. Interval Arithmetic

Participants: Nicolas Brisebarre, Mioara Joldeş, Vincent Lefèvre, Hong Diep Nguyen, Nathalie Revol, Philippe Théveny.

6.4.1. Efficient Implementation of Algorithms for Interval Linear Algebra

H.-D. Nguyen has developed an algorithm for the product of interval matrices [38], [40], [53], which offers a tradeoff between efficiency and accuracy. It is more efficient than the straightforward product of interval matrices, thanks to the use of optimized floating-point routines, and up to $9/4$ slower than the fastest algorithm, due to S. M. Rump. It is more accurate than this latter algorithm: the factor of overestimation is 1.18, compared to 1.5 for Rump's algorithm and to 1 for the straightforward algorithm.

H.-D. Nguyen and N. Revol proposed an algorithm for interval linear system solving, based on iterative refinement techniques; a relaxation of the interval residual linear system [40] yields a gain in efficiency and carefully chosen computing precisions [39] yield accuracy to the last bit. This algorithm is competitive with the best competitors in terms of efficiency while obtaining more accurate solutions.

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 [41], for the year 2010, were the representation of intervals (endpoints, are midpoint-radius allowed?), comparisons, reverse operations (to get a reciprocal of some operations), exception handling (via decorations).

6.4.3. Validated Function Approximation

S. Chevillard, J. Harrison, M. Joldeş and Ch. Lauter proposed in [13] an algorithm for efficient and accurate computation of upper bounds of approximation errors. Key elements of this algorithm are the use of intermediate approximation polynomials with bounded truncation remainder and a non-negativity test based on a Sum-of-squares expression of polynomials.

This algorithm is fully automated and the accuracy obtained for the result is controlled "a priori" by the user. It can deal uniformly with both absolute and relative errors. The algorithm focuses also on formally proving the numerical result and paves the way for formally certified supremum norms.

In [30], N. Brisebarre and M. Joldeş proposed a new tool for performing validated computations (global optimization or quadrature for instance). Following the Taylor model approach, they associate to a smooth function f a pair made of a Chebyshev interpolant P and an interval remainder which contains all the values of the difference $f - P$. Chebyshev interpolants offer an almost optimal quality of approximation and polynomials expressed in the Chebyshev polynomial basis provide better numerical stability.

6.5. Linear Algebra, Polynomials, and Euclidean Lattices

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

6.5.1. Faster Lattice Reduction

With Xiao-Wen Chang (McGill University, Canada), D. Stehlé and G. Villard studied the effect of perturbations on the LLL-reducedness of lattice bases [63]. The algorithmic motivation was to speed up the LLL reduction algorithm by using (floating-point) approximations to the R-factor. Among others, they described a perturbation-friendly notion of LLL-reducedness. The obtained results have already proven to be very useful to devise LLL-type algorithms relying on floating-point approximations (see below). As an indirect by-product of this work, Xiao-Wen Chang and D. Stehlé [12] proved normwise perturbation bounds for the Cholesky, LU and QR factorizations with normwise or componentwise perturbations in the given matrix. Each of the new rigorous perturbation bounds is a small constant multiple of the corresponding first-order perturbation bound obtained by the refined matrix equation approach in the literature and can be estimated efficiently.

A. Novocin, D. Stehlé and G. Villard [71] have devised an algorithm, with the following specifications: It takes as input an arbitrary basis $B = (b_i)_i$ in $Z^{d\times d}$ of a Euclidean lattice L; It computes a basis of L which is reduced for the modified notion of LLL-reducedness (see above); It terminates in time $O(d^{5+\epsilon}\beta + d^{\omega+1+\epsilon}\beta^{1+\epsilon})$ where $\beta = \log \max \|b_i\|$ (for any $\epsilon > 0$ 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 β of the entries and polynomial in the dimension d.

6.5.2. Polynomial Arithmetic

With Bill Hart (Warwick Mathematics Institute, UK) and Mark van Hoeij (Florida State University, USA), A. Novocin proposed in [66] 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.

Mark van Hoeij (Florida State University, USA) and A. Novocin presented in [44] a lattice algorithm specifically designed for some classical applications of lattice reduction. The applications are for lattice bases with a generalized knapsack-type structure, where the target vectors are boundably short. If the bit-length of the target vectors is unrelated to the bit-length of the input, then the algorithm is only linear in the bit-length of the input entries, which is an improvement over the quadratic complexity floating-point LLL algorithms.

Bill Hart (Warwick Mathematics Institute, UK) and A. Novocin revisited in [65] a divide-and-conquer algorithm, originally described by Brent and Kung for composition of power series, showing that it can be applied practically to composition of polynomials in $Z[x]$ given in the standard monomial basis. They offer a complexity analysis, showing that it is asymptotically fast, avoiding coefficient explosion in $Z[x]$. The algorithm is straightforward to implement and practically fast, avoiding computationally expensive change of basis steps of other polynomial composition strategies.

6.5.3. Computing Short Lattice Vectors

D. Stehlé and Mark Watkins (University of Sydney, Australia) showed in [43] that a specific well-known 80-dimensional lattice is extremal (i.e., the minimal nonzero norm is 8). This is the third known extremal lattice in this dimension. The technique to show extremality involves using the positivity of the Theta-series, along with fast vector enumeration techniques including pruning, while also using the automorphisms of the lattice.

Jérémie Detrey (INRIA project-team Caramel), G. Hanrot, X. Pujol and D. Stehlé [33] devised an FPGA accelerator for the Kannan–Fincke–Pohst enumeration algorithm (KFP) solving the Shortest Lattice Vector Problem. This is the first FPGA implementation of KFP specifically targeting cryptographically relevant dimensions. In order to optimize this implementation, they theoretically and experimentally study several facets of KFP, including its efficient parallelization and its underlying arithmetic. For devices of comparable costs, their FPGA implementation is faster than a multi-core CPU implementation by a factor around 2.12.

6.5.4. Generalization of Lattice Reduction to Modules over Dedekind Domains

Lattices over number fields arise from a variety of sources in algorithmic algebra and more recently cryptography. Similar to the classical case of Z-lattices, the choice of a nice, "short" (pseudo)-basis is important in many applications. Claus Fieker (University of Sydney, Australia) and D. Stehlé [34] have proposed the first algorithm that computes such a "short" (pseudo)-basis. As it runs in polynomial time, this provides an effective variant of Minkowski's second theorem for lattices over number fields.

6.5.5. Fully Homomorphic Encryption

D. Stehlé and Ron Steinfeld (Macquarie University, Australia) studied and proposed proposed two improvements to Gentry's fully homomorphic scheme based on ideal lattices, which, combined together, lead to a faster fully homomorphic scheme [42]. A fully homomorphic scheme is an encryption scheme that allows to apply any circuit to plaintexts, while acting uniquely on the ciphertexts. The resulting scheme has a $\widetilde{O}(\lambda^{3.5})$ bit complexity per elementary binary add/mult gate, where λ is the security parameter.

6.5.6. Structured Matrix Inversion

The asymptotically fastest known divide-and-conquer methods for inverting dense structured matrices are essentially variations or extensions of the Morf/Bitmead-Anderson (MBA) algorithm. Most of them must deal with the growth in length of intermediate generators, and this is done by incorporating various generator compression techniques into the algorithms. One exception is an algorithm by Cardinal, which in the particular case of Cauchy-like matrices avoids such growth by focusing on well-specified, already compressed generators of the inverse. In [35], Claude-Pierre Jeannerod and Christophe Mouilleron have extended Cardinal's method to a broader class of structured matrices including those of Vandermonde, Hankel, and Toeplitz types. This paper also describes some experimental results which demonstrate the practical interest of the approach.

6.5.7. Automatic Differentiation for the Matrix Adjoint

Kaltofen (1992) has proposed a new approach for computing matrix determinants without divisions. For matrices over an abstract ring, by the results of Baur and Strassen (1983), the determinant program leads to an algorithm with the same complexity for computing the adjoint of a matrix. However, the latter adjoint algorithm is obtained by the reverse mode of automatic differentiation, and hence is in some way not explicit. Applying program differentiation techniques by hand, we propose in [22] an alternative (still closely related) algorithm for obtaining the adjoint that is completely described.

6.6. Code and Proof Synthesis

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

6.6.1. 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 [32]. 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.

6.6.2. LEMA: a Representation Language for Mathematical Expressions

A study has begun to establish and implement a language [36], called LEMA: *Langage pour les Expressions Mathématiques Annotées*, that enables the programmer to represent not only the numerical code, but also the intended semantics and the programmer's knowledge. This language is based on MathML, which enables to represent mathematical content, and it adds to MathML properties such as range and error bounds, taking the underlying arithmetic into account. This language should also facilitate the transformation of the original code into another code which can be more efficient and/or more accurate for instance. This work takes place in the framework of the ANR project EVA-Flo, described in §8.2.1.

6.6.3. Code Generation for Polynomial Evaluation

In [68] Christophe Mouilleron and Guillaume Revy (DALI research team, Université de Perpignan and LIRMM laboratory) have presented the design and implementation of CGPE, a software tool dealing with the generation of fast and certified codes for the evaluation of bivariate polynomials (see also §5.7). This paper offers insights into the combinatorics of polynomial evaluation schemes and the heuristics introduced to quickly find low-latency schemes for a given architecture. The use of CGPE is illustrated on some examples, showing how it makes it possible to generate fast and certified codes within a few seconds and thus to reduce the development time of libms like FLIP (see §5.8).

7. Contracts and Grants with Industry

7.1. Contracts with Industry

7.1.1. Kalray contract

Participants: Nicolas Brunie, Florent de Dinechin, Vincent Lefèvre, Jean-Michel Muller, Adrien Panhaleux.

This contract covered the development of synthesizable register-transfer level description of a floating-point unit for the Kalray processor, as well as compatible C models, for evaluation purposes. Details are confidential at this point. This 2.5 month contract brought 30,085 euros.

7.1.2. Adacsys contract

Participants: Florent de Dinechin, Bogdan Pasca.

Adacsys granted Arénaire free access to their RAVA remote hardware validation tool to help develop the FloPoCo project.

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 February 2012) from STMicroelectronics (Compilation Expertise Center, Grenoble) on the theme of 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.

8. Other Grants and Activities

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 funds a PhD student, Mioara Joldeş. 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 EVA-Flo Project

The EVA-Flo project (Évaluation et Validation Automatiques de calculs Flottants, 2006-2010) is headed by N. Revol (Arénaire). All documents are available at http://www.ens-lyon.fr/LIP/Arenaire/EVA-Flo/. The other teams participating in this project are DALI (Eliaus, U. Perpignan), Measi (LIST, CEA Saclay) and Tropics (INRIA Sophia-Antipolis).

This project focuses on the way a mathematical formula is evaluated in floating-point arithmetic. The approach is threefold: study of algorithms for approximating and evaluating mathematical formulae, validation of such algorithms, and automation of the process.

The EVA-Flo project appears on *Cahiers de l'ANR no 3* and has been the subject of a talk and a poster during the ANR STIC colloque "Quelle recherche pour les STIC de demain?" in January 2010 and of a talk during the ANR colloque "Systèmes embarqués, sécurité et sûreté de fonctionnement" in December 2010.

We had a meeting near Perpignan in May this year. The work done so far encompasses the definition of the LEMA language, cf §6.6. Philippe Théveny has been hired to implement LEMA and interfaces between LEMA and software related to EVA-Flo topics.

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.

We had a preliminary meeting in Paris on September 8, 2010, and the official kick-off meeting took place in Lyon on October 27-28, 2010. The various documents can be found at http://tamadiwiki.ens-lyon.fr/tamadiwiki/index.php/Main_Page.

The TaMaDi project is funding the postdoc of Ioana Pasca (started on December 1, 2010).

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 study the FPGA implementation of specific algorithms for polarization demultiplexing and forward error correction with soft decoding.

8.2.4. ANR LaRedA Project

Participants: Ivan Morel, 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.

8.2.5. CNRS Grant Échange de chercheurs

Participant: Damien Stehlé.

In 2009, D. Stehlé obtained a grant to visit Ron Steinfeld, from the Cryptography (ACAC) team at Macquarie University, Australia. The visit took place between September and November 2010. Its purpose was the study of the security of the NTRU cryptosystem. Ron Steinfeld and Damien Stehlé were able to modify it so that it can be proven secure assuming the worst-case hardness of standard lattice problems, over ideal lattices.

8.2.6. Rhône-Alpes Region Grant Exploradoc

Participant: Ivan Morel.

I. Morel obtained this funding (Jul. 2008 - Jul. 2010) to cover the costs of his cotutelle PhD thesis between ENS Lyon (local advisor: G. Villard) and the University of Sydney (local advisor: J. Cannon). His research focuses on the development, analysis and implementation of fast LLL-type lattice reduction algorithms.

8.2.7. ENS research fund "Collaboration with Spiral"

ENS granted us 8,000 euros for exchanges with the Spiral group from Carnegie Mellon University.

8.3. International Initiatives

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

9. Dissemination

9.1. Animation of the Scientific Community

- Florent de Dinechin, Claude-Pierre Jeannerod, Vincent Lefèvre, Nicolas Louvet, Sèverine Morin, Hong Diep Nguyen, and Nathalie Revol (chair) have organized the SCAN 2010 conference (Scientific Computing, Computer Arithmetic and Validated Numerics), held in Lyon, 27-30 September 2010, which gathered 123 participants from 20 countries. They were in charge of the scientific programme and of the local organization.
- Florent de Dinechin was in the program committees of FPL'09 (Field Programmable Logic and Applications) and FPT'09 (Field-Programmable Technologies). He was session chair of Track G, session 5: *Reconfigurable Architectures, Algorithms and Applications* of the Asilomar Conference, November 7-10, 2010. He is a member of the Steering Committee of the *Symposium en Architectures de Machines*.
- Guillaume Hanrot is vice-president of INRIA's Evaluation Committee, and deputy director of the Computer Science Department of ENS de Lyon. He was also program committee co-chair (with François Morain, INRIA project-team TANC) of the 9th Algorithmic Number Theory Symposium (ANTS IX, Nancy, July 2010).
- Jean-Michel Muller was in the program committee of ASAP 2010 (21st conference on Application-specific Systems, Architectures and Processors), and he is in the program committee of ARITH'20 (20th IEEE Symposium on Computer Arithmetic). He heads the TaMaDi project of the ANR (see Section 8.2.2).
- Nathalie Revol was in the program committee of NSV-3: Third International Workshop on Numerical Software Verification, of PASCO 2010: Parallel Symbolic Computation, of ICMS 2010: Third International Congress on Mathematical Software.
- Damien Stehlé was in the program committee of ACISP 2010 (15th Australasian Conference on Information Security and Privacy) and is in the program committee of ACISP 2011. He is also in the program committee of the C2 workshop (*Journées Codes et Cryptographie*), to be held in April 2011.
- Gilles Villard is chair of the LIP laboratory since January 2009.

9.2. Committees

- Nicolas Brisebarre has been examiner for the ENS admissions (spring 2010).
- Florent de Dinechin participated to the PhD committee of S. Collange (University of Perpignan, December 2010).
- Jean-Michel Muller is a member of the scientific council of Grenoble INP, and of the scientific council of École Normale Supérieure de Lyon; He was a member of the hiring committee for a full professor position in Henri Poincaré University (Nancy) in April-May 2010. He was a member of the evaluation committee for Laboratoire GREYC (UMR CNRS 6072, Caen) in November 2010. He participated to the PhD committees of J. Dusser (Rennes Univ., December 2010) and K. Gbolagade (TU Delft, the Netherlands, December 2010), and to the Habilitation committee of A. Tisserand (Rennes Univ., July 2010);
- Nathalie Revol was a member of the hiring committees in Lyon 1 and Paris 6 for positions of "maître de conférences", of the INRIA Grenoble - Rhône-Alpes hiring committee for post-doc positions.
- Nathalie Revol took part in the first INRIA seminar on the dissemination of "STIC culture".
- Gilles Villard was a member of the hiring committee for a professor position of the Université Montpellier 2 (March-April 2010). He was in the habilitation committee of J.-G. Dumas (Université de Grenoble, July 2010), and member of the board of examiners for the PhD defense of A. Marszalek Urbanska (Université de Grenoble, April 2010), and Y. Strozecki (U. Paris Diderot, December 2010).

9.3. Doctoral School Teaching

- Nicolas Brisebarre gave a 20h Master course “Cryptographic Algorithms” (November 2010) at the ENSIAS of Rabat (Morocco).
- Florent de Dinechin gave a 24h ENSL Master course “Reconfigurable computing” (Fall 2010). He advised the training periods of N. Brunie (Master 2) and M. Bonamy (Master 1).
- Guillaume Hanrot and Damien Stehlé gave a 24h ENSL Master course “Cryptology: new primitives and applications” (Fall 2010).
- Claude-Pierre Jeannerod and Nicolas Louvet gave a 36h Master Course “Numerical Algorithms” at Université Claude Bernard - Lyon 1 (Spring 2010).
- Vincent Lefèvre gave a 24h Master course “Computer Arithmetic” at Université Claude Bernard - Lyon 1 (Fall 2010).
- Jean-Michel Muller and Claude-Pierre Jeannerod gave a 24h ENSL Master course “Arithmetic algorithms” (Fall 2010).

9.4. Other Teaching and Service

- Florent de Dinechin gave License courses in computer architecture, networks and systems at ENS-Lyon. He manages the international exchange programme for the department of computing.
- Nicolas Louvet gave 182h of License courses during the academic year 2009/2010 at U. Claude Bernard - Lyon 1.
- Nathalie Revol gave conferences at “Collège Pablo Picasso”, Bron.

9.5. Seminars and Invited Conference Talks

- Nicolas Brisebarre, Sylvain Collange, and Nicolas Louvet gave invited talks during the workshop “Nouvelles tendances en calcul scientifique” organized by the GDR Calcul in Lyon in November 2010.
- Nicolas Brisebarre gave invited talks at Tokyo Technical University and Kyoto University (July 2010). He gave an invited talk within the colloquium of Institut Camille Jordan of Univ. Lyon 1 (December 2010).
- Sylvain Collange and Mioara Joldeş were invited to talk at the seminar of the *CARAMEL* INRIA project-team (Nancy, December and April 2010).
- Florent de Dinechin gave invited talks at University of Madrid, Technical University München, and CEA Grenoble.
- Guillaume Hanrot, Xavier Pujol, and Damien Stehlé were invited to talk at the CIRM workshop on the interactions between mathematics and computer science (Luminy, February 2010).
- Claude-Pierre Jeannerod was an invited speaker at PASCO’10 (International Workshop on Parallel and Symbolic Computation, Grenoble, July 2010).
- Claude-Pierre Jeannerod and Christophe Mouilleron gave invited talks at NASC’10 (international conference on Numerical Algebra and Scientific Computing), held in Beijing in October 2010.
- Mioara Joldeş, Christophe Mouilleron, Andrew Novocin, Bogdan Pasca, Philippe Théveny, and Nathalie Revol talked at the 5th meeting of the EVA-Flo ANR project (Perpignan, May 2010).
- Mioara Joldeş and Christophe Mouilleron gave invited talks at the seminar of the *Algorithms* INRIA project-team (Rocquencourt, October and November 2010).
- Mioara Joldeş gave invited talks at the SWIM’10 (3rd Small Workshop on Interval Methods, Nantes, June 2010) and at the Computer Algebra seminar of Limoges University (Limoges, November 2010).
- Vincent Lefèvre gave a talk at the seminar of the ENSL computer science department (March 2010).

- Érik Martin-Dorel was invited to talk at the seminar of the *Marelle* INRIA project-team (Sophia Antipolis, June 2010).
- Andrew Novocin and Damien Stehlé gave invited talks at the SIAM/MSRI workshop on Hybrid Methodologies for Symbolic-Numeric Computation, held in Berkeley (CA) in November 2010.
- Xavier Pujol was invited to talk at JNCF'10 (Journées Nationales de Calcul Formel, May 2010).
- Damien Stehlé was invited for two weeks in June/July 2010 at NTU (Nanyang Technological University) in Singapore to give a series of seminars on lattice-based cryptography.

9.6. Visiting Scientists

- Jean-Luc Beuchat, Associate Professor at University of Tsukuba, Japan (April 2010).
- Lihong Zhi, Professor at Academia Sinica, Beijing, China (from September 20 to October 15).
- Peter Kornerup, Professor at Odense University, Denmark (from September 30 to October 15).

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

Articles in International Peer-Reviewed Journal

[12] X.-W. CHANG, D. STEHLÉ. *Rigorous Perturbation Bounds of Some Matrix Factorizations*, in "SIAM Journal on Matrix Analysis and Applications (SIMAX)", November 2010, vol. 31, n^{o} 5, p. 2841–2859, http://hal.inria.fr/hal-00546885/en.

[13] S. CHEVILLARD, J. HARRISON, M. JOLDES, C. LAUTER. *Efficient and accurate computation of upper bounds of approximation errors*, in "Journal of Theoretical Computer Science (TCS)", 2010, vol. In Press, Corrected Proof [*DOI :* 10.1016/J.TCS.2010.11.052], http://hal.inria.fr/ensl-00445343/en.

[14] M. ERCEGOVAC, J.-M. MULLER. *An Efficient Method for Evaluating Complex Polynomials*, in "Journal of Signal Processing Systems", January 2010, vol. 58, n^{o} 1, p. 17-27 [*DOI :* 10.1007/s11265-008-0265-8], http://hal.inria.fr/ensl-00446889/en.

[15] K. GHAZI, V. LEFÈVRE, P. THÉVENY, P. ZIMMERMANN. *Why and How to Use Arbitrary Precision*, in "Computing in Science and Engineering", 2010, vol. 12, n^{o} 3, p. 62-65 [*DOI :* 10.1109/MCSE.2010.73], http://hal.inria.fr/inria-00543927/en.

[16] 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.

[17] 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 [*DOI :* 10.1109/TC.2010.144], http://hal.inria.fr/ensl-00409366/en.

[18] P. KORNERUP, C. LAUTER, V. LEFÈVRE, N. LOUVET, J.-M. MULLER. *Computing Correctly Rounded Integer Powers in Floating-Point Arithmetic*, in "Transactions on Mathematical Software", January 2010, vol. 37, n^{o} 1, p. 4:1-4:23 [*DOI :* 10.1145/1644001.1644005], http://hal.inria.fr/inria-00388501/en.

[19] 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.

[20] H. D. NGUYEN, N. REVOL. *Solving and Certifying the Solution of a Linear System*, in "Reliable Computing", 2011, http://hal.inria.fr/inria-00546856/en.

[21] A. VAZQUEZ, E. ANTELO, P. MONTUSCHI. *Improved Design of High-Performance Parallel Decimal Multipliers*, in "IEEE Transactions on Computers", May 2010, vol. 59, n^{o} 5, p. 679-693, http://hal.inria.fr/ensl-00560255/en.

[22] G. VILLARD. *Kaltofen's division-free determinant algorithm differentiated for matrix adjoint computation*, in "Journal of Symbolic Computation", 2010 [*DOI :* 10.1016/J.JSC.2010.08.012], http://hal.inria.fr/ensl-00335918/en.

[23] 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.

Articles in National Peer-Reviewed Journal

[24] I. MOREL, D. STEHLÉ, G. VILLARD. *Analyse numérique et réduction de réseaux*, in "Technique et Science Informatiques (TSI)", 2010, vol. 29, n^{o} 1, p. 115-144 [*DOI :* 10.3166/TSI.29.115-144], http://hal.inria.fr/ensl-00327678/en.

Articles in Non Peer-Reviewed Journal

[25] C. BERTIN, C.-P. JEANNEROD, C. MONAT. *Bringing fast floating-point arithmetic into embedded integer processors*, in "HiPEAC Newsletter", April 2010, n^{o} 22, p. 11-12, http://hal.inria.fr/ensl-00551245/en.

Invited Conferences

[26] C. BERTIN, C.-P. JEANNEROD, J. JOURDAN-LU, H. KNOCHEL, C. MONAT, C. MOUILLERON, J.-M. MULLER, G. REVY. *Techniques and tools for implementing IEEE 754 floating-point arithmetic on VLIW integer processors*, in "4th International Workshop on Parallel and Symbolic Computation (PASCO'10)", Grenoble, France, ACM, 2010 [*DOI :* 10.1145/1837210.1837212], http://hal.inria.fr/ensl-00549467/en.

International Peer-Reviewed Conference/Proceedings

[27] S. BANESCU, F. DE DINECHIN, B. PASCA, R. TUDORAN. *Multipliers for Floating-Point Double Precision and Beyond on FPGAs*, in "Highly Efficient Accelerators and Reconfigurable Technologies", Tsukuba, Japan, EIC, June 2010, http://hal.inria.fr/ensl-00475781/en.

[28] S. BOLDO, F. CLÉMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND, P. WEIS. *Formal Proof of a Wave Equation Resolution Scheme: the Method Error*, in "Interactive Theorem Proving", Edinburgh, United Kingdom, M. KAUFMANN, L. C. PAULSON (editors), Lecture Notes in Computer Science, Springer, July 2010, vol. 6172, p. 147-162 [*DOI :* 10.1007/978-3-642-14052-5_12], http://hal.inria.fr/inria-00450789/en.

[29] N. BRISEBARRE, M. ERCEGOVAC, N. LOUVET, É. MARTIN-DOREL, J.-M. MULLER, A. PANHALEUX. *Implementing decimal floating-point arithmetic through binary: some suggestions*, in "21st IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'2010)", Rennes, France, IEEE, 2010, p. 317-320, http://hal.inria.fr/ensl-00463353/en.

[30] N. BRISEBARRE, M. JOLDES. *Chebyshev Interpolation Polynomial-based Tools for Rigorous Computing*, in "ISSAC '10, 2010 International Symposium on Symbolic and Algebraic Computation", Münich, Germany, ACM New York, NY, USA, July 2010, p. 147-154, 17 pages [*DOI :* 10.1145/1837934.1837966], http://hal.inria.fr/ensl-00472509/en.

[31] F. BUTELLE, F. HIVERT, M. MAYERO, F. TOUMAZET. *Formal Proof of SCHUR Conjugate Function*, in "CALCULEMUS 2010", Paris, France, LNAI, Springer, 2010, vol. 6167, p. 158-171 [*DOI :* 10.1007/978-3-642-14128-7], http://hal.inria.fr/hal-00477574/en.

[32] P. COLLINS, M. NIQUI, N. REVOL. *A Taylor Function Calculus for Hybrid System Analysis: Validation in Coq*, in "NSV-3: Third International Workshop on Numerical Software Verification.", Edinburgh, United Kingdom, Fainekos, Georgios and Goubault, Eric and Putot, Sylvie, July 2010, http://hal.inria.fr/inria-00473270/en.

[33] J. DETREY, G. HANROT, X. PUJOL, D. STEHLÉ. *Accelerating lattice reduction with FPGAs*, in "First International Conference on Cryptology and Information Security in Latin America (LATINCRYPT'10)", Puebla, Mexico, M. ABDALLA, P. S. L. M. BARRETO (editors), Lecture Notes in Computer Science, August 2010, vol. 6212, p. 124-143 [*DOI :* 10.1007/978-3-642-14712-8_8], http://hal.inria.fr/inria-00539929/en.

[34] C. FIEKER, D. STEHLÉ. *Short bases of lattices over number fields*, in "ANTS-IX", France, 2010, vol. LNCS 6197, p. 157–173, http://hal.inria.fr/hal-00546895/en.

[35] C.-P. JEANNEROD, C. MOUILLERON. *Computing specified generators of structured matrix inverses*, in "35th International Symposium on Symbolic and Algebraic Computation (ISSAC 2010)", Münich, Germany, ACM, 2010 [*DOI :* 10.1145/1837934.1837988], http://hal.inria.fr/ensl-00450272/en.

[36] V. LEFÈVRE, P. THÉVENY, F. DE DINECHIN, C.-P. JEANNEROD, C. MOUILLERON, D. PFANNHOLZER, N. REVOL. *LEMA: Towards a Language for Reliable Arithmetic*, in "International Workshop on Programming Languages for Mechanized Mathematics Systems (PLMMS 2010)", Paris, France, ACM Communications in Computer Algebra, ACM, June 2010, vol. 44, p. 41-52 [*DOI :* 10.1145/1838599.1838622], http://hal.inria.fr/inria-00542143/en.

[37] N. LOUVET, J.-M. MULLER, A. PANHALEUX. *Newton-Raphson Algorithms for Floating-Point Division Using an FMA*, in "21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP), 2010", Rennes, France, IEEE, 2010, p. 200-207, http://hal.inria.fr/ensl-00549027/en.

[38] H. D. NGUYEN. *Efficient implementation of interval matrix multiplication*, in "Para 2010: State of the Art in Scientific and Parallel Computing", Reykjavik, Iceland, April 2010, http://hal.inria.fr/inria-00469472/en.

[39] H. D. NGUYEN, N. REVOL. *Certification of a Numerical Result: Use of Interval Arithmetic and Multiple Precision*, in "NSV-3: Third International Workshop on Numerical Software Verification.", Edinburgh, United Kingdom, Fainekos, Georgios and Goubault, Eric and Putot, Sylvie, July 2010, http://hal.inria.fr/inria-00544798/en.

[40] H. D. NGUYEN, N. REVOL. *High performance linear algebra using interval arithmetic*, in "PASCO'10 4th International Workshop in Parallel and Symbolic Computation", Grenoble, France, ACM Digital Library, 2010 [*DOI :* 10.1145/1837210.1837236], http://hal.inria.fr/inria-00544800/en.

[41] N. REVOL. *Standardized Interval Arithmetic and Interval Arithmetic Used in Libraries*, in "ICMS 2010 - Third International Congress on Mathematical Software", Kobe, Japan, Springer, 2010, vol. 6327, p. 337–341 [*DOI :* 10.1007/978-3-642-15582-6_54], http://hal.inria.fr/inria-00544803/en.

[42] D. STEHLÉ, R. STEINFELD. *Faster Fully Homomorphic Encryption*, in "ASIACRYPT 2010", Singapore, 2010, vol. LNCS 6477, p. 377-394, http://hal.inria.fr/hal-00546890/en.

[43] D. STEHLÉ, M. WATKINS. *On the Extremality of an 80-Dimensional Lattice*, in "ANTS-IX", France, 2010, vol. LNCS 6197, p. 340–356, http://hal.inria.fr/hal-00546901/en.

[44] M. VAN HOEIJ, A. NOVOCIN. *Gradual sub-lattice reduction and a new complexity for factoring polynomials*, in "LATIN 2010", Oaxaca, Mexico, LLNCS, April 2010, http://hal.inria.fr/ensl-00452881/en.

[45] A. VAZQUEZ, E. ANTELO. *Multi-Operand Decimal Addition by Efficient Reuse of a Binary Carry-Save Adder Tree*, in "44th ASILOMAR Conference on Signals, Systems and Computers", Pacific Grove, CA, United States, November 2010, http://hal.inria.fr/inria-00546040/en.

[46] A. VAZQUEZ, F. DE DINECHIN. *Efficient implementation of Parallel BCD Multiplication in LUT-6 FPGAs*, in "2010 International Conference on Field-Programmable Technology", Beijing, China, December 2010, http://hal.inria.fr/inria-00546028/en.

[47] F. DE DINECHIN, M. JOLDES, B. PASCA. *Automatic generation of polynomial-based hardware architectures for function evaluation*, in "Application-specific Systems, Architectures and Processors", Rennes, France, IEEE, July 2010, http://hal.inria.fr/ensl-00470506/en.

[48] F. DE DINECHIN, M. JOLDES, B. PASCA, G. REVY. *Multiplicative square root algorithms for FPGAs*, in "International Conference on Field Programmable Logic and Applications", Milano, Italy, IEEE, 2010 [*DOI :* 10.1109/FPL.2010.112], http://hal.inria.fr/ensl-00475779/en.

[49] F. DE DINECHIN, H. D. NGUYEN, B. PASCA. *Pipelined FPGA Adders*, in "International Conference on Field Programmable Logic and Applications", Milano, Italy, IEEE, 2010, p. 422-427 [*DOI :* 10.1109/FPL.2010.87], http://hal.inria.fr/ensl-00475780/en.

[50] F. DE DINECHIN, B. PASCA. *Floating-point exponential functions for DSP-enabled FPGAs*, in "International Conference on Field-Programmable Technology", Beijing, China, IEEE, December 2010, LIP research report RR2010-23, http://hal.inria.fr/ensl-00506125/en.

[51] F. DE DINECHIN, H. TAKEUGMING, J.-M. TANGUY. *A 128-Tap Complex FIR Filter Processing 20 Giga-Samples/s in a Single FPGA*, in "Proceedings of 44th Conference on signals, systems and computers", Asilomar, CA, IEEE, November 2010, http://prunel.ccsd.cnrs.fr/ensl-00542950/en/, http://prunel.ccsd.cnrs.fr/ensl-00542950/PDF/2010-36.pdf.

Workshops without Proceedings

[52] É. MARTIN-DOREL. *Formalization of Hensel's lemma in Coq*, in "TYPES 2010: The 17th Workshop Types for Proofs and Programs", Warsaw, Poland, October 2010, http://hal.inria.fr/ensl-00560449/en.

[53] H. D. NGUYEN, N. REVOL. *Accuracy issues in linear algebra using interval arithmetic*, in "SCAN 2010: 14th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics", Lyon, France, Revol, Nathalie and de Dinechin, Florent and Jeannerod, Claude-Pierre and Lefèvre, Vincent and Louvet, Nicolas and Morin, Sèverine and Nguyen, Hong Diep, 2010, http://hal.inria.fr/inria-00544805/en.

[54] F. DE DINECHIN, C.-P. JEANNEROD, D. PFANNHOLZER, N. REVOL. *Code generation for argument filtering and argument reduction in elementary functions*, in "SCAN 2010: 14th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics", Lyon, France, Revol,

Nathalie and de Dinechin, Florent and Jeannerod, Claude-Pierre and Lefèvre, Vincent and Louvet, Nicolas and Morin, Sèverine and Nguyen, Hong Diep, 2010, http://hal.inria.fr/inria-00544808/en.

Scientific Books (or Scientific Book chapters)

[55] 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*, Birkhauser Boston, 2010, http://hal.inria.fr/ensl-00379167/en.

[56] D. STEHLÉ. *Floating-point LLL: theoretical and practical aspects*, in "The LLL Algorithm: survey and applications", Springer, 2010, p. 179-213, http://hal.inria.fr/hal-00550987/en.

Books or Proceedings Editing

[57] G. HANROT, F. MORAIN, E. THOMÉ (editors). *Algorithmic Number Theory. 9th. International Symposium, ANTS-IX. Nancy, France, July 2010. Proceedings*, Lecture Notes in Computer Science, Springer-Verlag, July 2010, vol. 6197, 397 [*DOI :* 10.1007/978-3-642-14518-6], http://hal.inria.fr/inria-00544503/en.

Research Reports

[58] P. KORNERUP, V. LEFÈVRE, N. LOUVET, J.-M. MULLER. *On the Computation of Correctly-Rounded Sums*, INRIA, April 2010, n^{o} RR-7262, http://hal.inria.fr/inria-00475279/en.

[59] V. LEFÈVRE, P. THÉVENY, F. DE DINECHIN, C.-P. JEANNEROD, C. MOUILLERON, D. PFANNHOLZER, N. REVOL. *LEMA: Towards a Language for Reliable Arithmetic*, INRIA, April 2010, n^{o} RR-7258, http://hal.inria.fr/inria-00473767/en.

[60] A. VAZQUEZ, F. DE DINECHIN. *Multi-operand Decimal Adder Trees for FPGAs*, INRIA, October 2010, n^{o} RR-7420, http://hal.inria.fr/inria-00526327/en.

Other Publications

[61] C. ALIAS, B. PASCA, A. PLESCO. *Automatic Generation of FPGA-Specific Pipelined Accelerators*, http://hal.inria.fr/ensl-00549682/en.

[62] N. BRISEBARRE, M. JOLDES, P. KORNERUP, É. MARTIN-DOREL, J.-M. MULLER. *Augmented precision square roots, 2-D norms, and discussion on correctly rounding* $\sqrt{x^2+y^2}$, http://hal.inria.fr/ensl-00545591/en.

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

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

[65] W. HART, A. NOVOCIN. *A Practical Univariate Polynomial Composition Algorithm*, Submitted to Journal DMTCS, http://hal.inria.fr/ensl-00546102/en.

[66] W. HART, M. VAN HOEIJ, A. NOVOCIN. *Practical polynomial factoring in polynomial time*, Pre-Print, http://hal.inria.fr/ensl-00546114/en.

[67] C.-P. JEANNEROD, J. JOURDAN-LU, C. MONAT, G. REVY. *How to square floats accurately and efficiently on the ST231 integer processor*, http://hal.inria.fr/ensl-00532829/en.

[68] C. MOUILLERON, G. REVY. *Automatic Generation of Fast and Certified Code for Polynomial Evaluation*, http://hal.inria.fr/ensl-00531721/en.

[69] J.-M. MULLER. *Scaling Newton-Raphson division iterations to avoid double rounding*, http://hal.inria.fr/ensl-00496368/en.

[70] H. D. NGUYEN, B. PASCA, T. PREUSSER. *FPGA-Specific Arithmetic Optimizations of Short-Latency Adders*, http://hal.inria.fr/ensl-00542389/en.

[71] A. NOVOCIN, D. STEHLÉ, G. VILLARD. *An LLL-reduction algorithm with quasi-linear time complexity*, http://hal.inria.fr/ensl-00534899/en.

[72] F. DE DINECHIN. *A flexible floating-point logarithm for reconfigurable computers*, http://hal.inria.fr/ensl-00506122/en.

[73] F. DE DINECHIN, B. PASCA. *FPGA-Specific Custom Arithmetic Datapath Design*, http://hal.inria.fr/ensl-00542396/en.

References in notes

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

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

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