INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Arénaire

Computer Arithmetic

Grenoble - Rhône-Alpes

THEME SYM

Table of contents

Arénaire is a joint project of CNRS, École Normale Supérieure de Lyon (U. Lyon), INRIA, and Université Claude Bernard de Lyon (U. Lyon). As part of the Laboratoire de l'Informatique du Parallélisme (LIP, UMR 5668), it is located at Lyon in the buildings of the ÉNS.

1. Team

Research Scientist

Nicolas Brisebarre [CR CNRS]
Claude-Pierre Jeannerod [CR INRIA]
Vincent Lefèvre [CR INRIA]
Jean-Michel Muller [DR CNRS, HdR]
Nathalie Revol [CR INRIA]
Damien Stehlé [CR CNRS, until July 2008]
Gilles Villard [DR CNRS, HdR]

Faculty Member

Florent de Dinechin [Associate Professor ÉNS Lyon, *Maître de Conférences*, HdR]
Nicolas Louvet [Associate Professor UCBL, *Maître de Conférences*]

Technical Staff

Nicolas Jourdan [Technical Staff, until February 2008]
Honoré Takeugming [Engineer on the ANR TCHATER project, from November 2008]
Serge Torres [Technical Staff, 40% on the project]

PhD Student

Sylvain Chevillard [*Allocataire-moniteur*, ÉNS grant, 3rd year]
Mioara Joldes [*Allocataire-moniteur*, Région Rhône-Alpes grant, 1st year]
Christoph Lauter [*Allocataire-moniteur*, MESR grant, thesis defended in October 2008]
Romain Michard [INRIA grant, thesis defended in June 2008]
Ivan Morel [*Allocataire-moniteur*, ÉNS grant, cotutelle with the University of Sydney, 2nd year]
Christophe Mouilleron [*Allocataire-moniteur*, ÉNS grant, 1st year]
Hong Diep Nguyen [INRIA grant, 2nd year]
Adrien Panhaleux [ÉNS student, 1st year]
Bogdan Pasca [*Allocataire-moniteur*, MESR grant, 1st year]
Guillaume Revy [*Allocataire-moniteur*, MESR grant, 3rd year]

Administrative Assistant

Isabelle Pera [TR CNRS, 30% on the project]

2. Overall Objectives

2.1. Introduction

Keywords: *FPGA circuit*, *VLSI circuit*, *approximate computation*, *computer algebra*, *computer arithmetic*, *elementary function*, *embedded chips*, *finite field*, *floating-point representation*, *integer computation*, *interval arithmetic*, *lattice basis reduction*, *linear algebra*, *low-power operator*, *multiple-precision arithmetic*, *reliability of numerical software*.

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 the 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 of floating-point), but internally, non-standard internal 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 mobilised manually for every problems.
- *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 of the Year

The IEEE-754-2008 standard for floating-point arithmetic has been adopted in August. Among the main novelties in this standard with respect to its predecessor, IEEE-754-1985, is the recommendation that elementary functions should be correctly rounded. This improvement, which means more portability and better accuracy for end-user programs, is a direct consequence of Arénaire's work over the past decade. This is illustrated by the bibliography of the IEEE-754-2008 document: Among the 22 recommended readings, 7 are publications from the Arénaire project-team.

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 tradeoffs 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 [5]). 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 [7] 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 [5] 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 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 is not required by the IEEE-754 standard: during the elaboration of this standard, it was considered that correctly rounded elementary functions was 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 cannot be answered in a simple manner, meaning that correctly rounding elementary functions requires arbitrary precision, which is very slow and resource-consuming.

Indeed, correctly rounded libraries already exist, such as 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.

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 [1]. 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 to 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.

Adequation 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 adequation" 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://lipforge.ens-lyon.fr/www/gappa/) 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 [6]. 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 [3] for an essentially optimal result, or [4]). 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

Keywords: *arithmetic operator*, *certified computing*, *control*, *dedicated circuit*, *hardware implementation*, *numerical software*, *proof*, *validation*.

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**.
- Developments 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 2008.

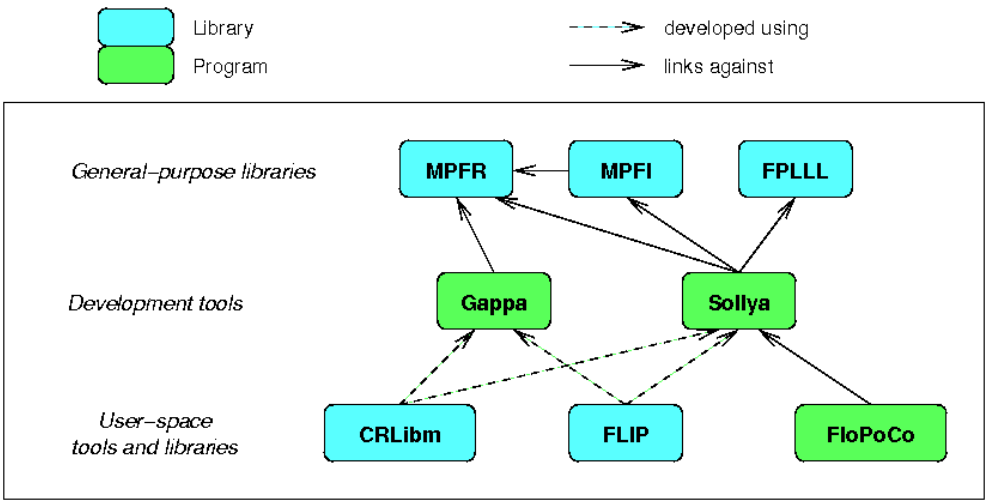

Figure 1. Relationships between some Arénaire developments.

5.2. FLIP: a floating-point library for integer processors

Keywords: *VLIW*, *correct rounding*, *integer processor*, *single-precision arithmetic*.

Participants: Claude-Pierre Jeannerod, Nicolas Jourdan, Guillaume Revy.

FLIP is a C library for efficient software support of *binary32* 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).

In 2008 C.-P. Jeannerod and G. Revy have written FLIP 1.0, which implements new algorithms for correctly-rounded addition, subtraction, multiplication, division, and square root. These codes support subnormal numbers as well as the four rounding direction attributes required by the IEEE 754-2008 standard. When compiled with the ST231 VLIW compiler, FLIP 1.0 leads to significant speed-ups compared to the previous version (FLIP 0.3).

Status: Beta release / **Target:** VLIW processors (ST200 family from STMicroelectronics) / **License:** LGPL / **OS:** Linux / **Programming Language:** C / **URL:** http://www.ens-lyon.fr/LIP/Arenaire/Ware/FLIP/flip.htm

5.3. Sollya: a toolbox of numerical algorithms for the development of safe numerical codes

Keywords: *Remez algorithm*, *automatic implementation of a function*, *infinite norm*, *numerical algorithms*, *plot*, *proof generation*.

Participants: Christoph Lauter, Sylvain Chevillard, Mioara Joldes, Nicolas Jourdan.

Sollya aims at providing a safe, user-friendly, all-in-one environment for manipulating numerical functions. Its distinguishing feature is the focus is on safety: numerical results are certified, or a warning is produced. Functionalities include plotting, infinite norm, polynomial approximation (including an original minimax approximation among polynomials with floating-point coefficients), zero finding, etc, and an interpreter for the Sollya scripting language.

Two stable versions of Sollya have been released in 2008, 1.0 and 1.1. Sollya is used by the CRlibm, FLIP and FloPoCo projects, and at Inria Saclay, Perpigan University and Ecole Polytechnique.

Status: Stable, version 1.1 / **Target:** ia32, ia64, PPC, ia32-64, other / **License:** CeCILL-C / **OS:** Unix / **Programming Language:** C / **URL:** http://sollya.gforge.inria.fr/

5.4. Metalibm, a code generator for elementary functions

Participants: Christoph Lauter, Sylvain Chevillard, Florent de Dinechin.

The Metalibm project provides a tool for the automatic implementation of mathematical (libm) functions. A function f is automatically transformed into Gappa-certified C code implementing an approximation polynomial in a given domain with given accuracy. Metalibm is based on the Sollya tool and has been used to produce large parts of CRlibm.

Status: alpha / **Target:** any / **License:** LGPL / **OS:** Unix / **Programming Language:** C / **URL:** http://lipforge.ens-lyon.fr/www/metalibm/

5.5. CRlibm: a Library of Elementary Functions with Correct Rounding

Keywords: *correct rounding*, *double precision arithmetic*, *elementary function*, *libm*.

Participants: Florent de Dinechin, Christoph Lauter, Jean-Michel Muller, Guillaume Revy.

The CRlibm project aims at developing a mathematical library (`libm`) which provides implementations of the double precision C99 standard elementary functions,

- correctly rounded in the four IEEE-754 rounding modes,
- with a comprehensive proof of both the algorithms used and their implementation,
- sufficiently efficient in average time, worst-case time, and memory consumption to replace existing `libms` transparently.

In 2008, the main objective of the CRlibm project was reached when the revised floating-point standard IEEE-754-2008 was published with a recommendation for correctly rounded functions.

Version 1.0beta2 was released with a much improved power function x^y. However the development focus has now turned to automated libm development with the Metalibm project.

Status: Beta release / **Target:** ia32, ia64, Sparc, PPC / **License:** LGPL / **OS:** Unix / **Programming Language:** C / **URL:** http://www.ens-lyon.fr/LIP/Arenaire

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

Keywords: *FPGA*, *arithmetic operators*, *fixed-point*, *floating-point*, *function evaluation*.

Participants: Florent de Dinechin, Bogdan Pasca.

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.

Non-standard operators developed in 2008 in FloPoCo include an application-specific floating-point accumulator, multipliers by constants, evaluators for fixed-point functions, and LNS operators. Versions 0.4, 0.5, 0.8 and 0.9 have been released in 2008.

Status: Beta release / **Target:** any / **License:** GPL/LGPL / **OS:** Unix, Linux, Windows / **Programming Language:** C++ / **URL:** http://www.ens-lyon.fr/LIP/Arenaire/Ware/FloPoCo/

5.7. FPLibrary: a Library of Operators for "Real" Arithmetic on FPGAs

Keywords: *FPGA*, *LNS*, *arithmetic operators*, *floating-point*, *function evaluation*.

Participant: Florent de Dinechin.

FPLibrary is a VHDL library that describes arithmetic operators (addition, subtraction, multiplication, division, and square root, exponential, logarithm, sine and cosine) for two formats of representation of real numbers: floating-point, and logarithmic number system (LNS). These formats are parametrized in precision and range.

FPLibrary is being used by tens of academic or industrial organizations in the world. In 2008 there have been some improvements to the LNS operators, but floating-point development has shifted completely to the FloPoCo project.

Status: Stable / **Target:** FPGA / **License:** GPL/LGPL / **OS:** Unix, Linux, Windows / **Programming Language:** VHDL / **URL:** http://www.ens-lyon.fr/LIP/Arenaire/Ware/FPLibrary/

5.8. MPFI: Multiple Precision Floating-Point Interval Arithmetic

Keywords: *arbitrary precision*, *interval arithmetic*.

Participants: Nathalie Revol, Sylvain Chevillard, Christoph Lauter, Hong Diep Nguyen.

Library in C for interval arithmetic using arbitrary precision (arithmetic and algebraic operations, elementary functions, operations on sets). Modifications made this year mainly concern the correction of bugs.

Status: Beta release / **Target:** any / **License:** LGPL / **OS:** Unix, Linux, Windows (Cygwin) / **Programming Language:** C / **URL:** http://mpfi.gforge.inria.fr/

5.9. MPFR: a multiple-precision floating-point library with correct rounding

Keywords: *arbitrary precision*, *correct rounding*.

Participant: Vincent Lefèvre.

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. Since the end of 2006, it has become a joint project between the Arénaire and CACAO project-teams.

MPFR 2.3.1 was released on 29 January 2008 and MPFR 2.3.2 on 12 September 2008. Two release candidates of MPFR 2.4.0 were distributed in December 2008, and we plan to release it in January 2009 as a GNU package.

The main changes done in 2008 on the Arénaire side are bug fixes (which mainly consisted in completing the implementation of some functions on particular cases).

Moreover an ODL (*Opération de Développement Logiciel*) called MPtools has been supported by the INRIA since 2007: A new engineer, Philippe Théveny, was hired in September 2007. He works in the CACAO project-team, but in addition to the collaboration by e-mail, Philippe came in Lyon for a few days in January 2008, and an MPtools meeting took place in Paris in September 2008.

Many software systems use MPFR. The most common one is GCC, which, as of its version 4.3.0 released in March 2008, now even *requires* MPFR (the use of MPFR was previously optional).

Status: stable / **Target:** any / **License:** LGPL / **OS:** Unix, Windows (Cygwin or MinGW) / **Programming Language:** C / **URL:** http://www.mpfr.org/

5.10. fpLLL: lattice reduction using floating-point arithmetic

Keywords: *floating-point arithmetic*, *lattice reduction*.

Participants: Damien Stehlé, Ivan Morel, 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 progressively becoming the reference for lattice reduction. This year, it has been included in SAGE and adapted to be included in PARI GP. A variant of it is also contained in the MAGMA computational algebra system.

Status: stable / **License:** LGPL / **Programming Language:** C++ / **URL:** http://perso.ens-lyon.fr/damien.stehle

5.11. Exhaustive Tests for the Correct Rounding of Mathematical Functions

Keywords: *correct rounding*, *elementary function*.

Participant: Vincent Lefèvre.

The programs to search for the worst cases for the correct rounding of mathematical functions (exp, log, sin, cos, etc.) in a fixed precision (mainly double precision) using Lefèvre's algorithm have been further improved, in particular concerning their robustness (e.g., to deal with network problems in a better way). More automation has also been added. Indeed these programs were originally designed to require a manual intervention every few weeks (in general), which was the time needed to obtain the complete results for a given function on a given "binade". But due to the improvement of the algorithms in the past and the new machines, this time is currently reduced to a few hours. Thus some manual work has been automatized.

6. New Results

6.1. Hardware Arithmetic Operators

Keywords: *ASIC*, *FPGA*, *LNS*, *arithmetic operators*, *circuit generator*, *cosine*, *digit-recurrence algorithms*, *division*, *exponential*, *fixed-point*, *floating-point*, *function evaluation*, *integrated circuit*, *logarithm*, *sine*, *square-root*, *tridiagonal systems*, *trigonometric*.

Participants: Nicolas Brisebarre, Florent de Dinechin, Jean-Michel Muller, Bogdan Pasca.

6.1.1. Multiplication by Constants

N. Brisebarre, F. de Dinechin and J.M. Muller have studied multiplication by a constant in the context of FPGAs. For integer constant multiplication, which resumes to additions and shifts, they have proposed a new algorithm that minimizes the size of the adders, and not only their count. They have also investigated multiplication and division by a floating-point constant, including correctly-rounded multiplication by an arbitrary precision constant [41], [29]. An implementation is available in FloPoCo.

6.1.2. Application-Specific Long Accumulator

F. de Dinechin and B. Pasca, in colalboration with O. Creţ, have proposed an improvement to the long accumulator concept introduced by Kulisch in the context of FPGA-accelerated floating-point computing. The proposed accumulator is a very wide fixed-point one. Its parameters (range and precision) are defined in an application-specific way, something that only makes sense in the context of reconfigurable computing. It holds numbers in partial carry-save representation and may therefore be operated at arbitrary frequency. For most application, this accumulator architecture may be tailored to be arbitrarily accurate, while remaining much more efficient than using a standard floating-point adder [40]. An implementation is available in FloPoCo.

6.1.3. Hardware complex polynomial evaluation

Milos Ercegovac (Univ. of California at Los Angeles) and Jean-Michel Muller have proposed an efficient hardware-oriented method for evaluating complex polynomials [17]. 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.

The main features of the method are: the latency of about m cycles for a 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 method can also be used to compute all successive integer powers of the complex argument with the same latency and a similar implementation cost. The design allows straightforward tradeoffs between latency and cost: a factor k decrease in cost leads to a factor k increase in latency. A similar tradeoff between precision, latency and cost exists. The proposed method is attractive for programmable platforms because of its regular and repetitive structure of simple hardware operators.

6.1.4. Generation of Modular Multipliers for FPGA applications

Since redundant number systems allow for constant time addition, they are often at the heart of modular multipliers designed for public-key cryptography (PKC) applications. Indeed, PKC involves large operands (160 to 1024 bits), and several researchers proposed carry-save or borrow-save algorithms. However, these number systems do not take advantage of the dedicated carry logic available in modern Field-Programmable Gate Arrays (FPGAs). To overcome this problem, Jean-Luc Beuchat (Tsukuba University, Japan) and Jean-Michel Muller [12] suggest to perform modular multiplication in a high-radix carry-save number system, where a sum bit of the carry-save representation is replaced by a sum word. Two digits are then added by means of a small Carry-Ripple Adder (CRA). Furthermore, They propose an algorithm that selects the best high-radix carry-save representation for a given modulus and generates a synthesizable VHDL description of the operator.

6.1.5. Horner's Rule-Based Multiplication in finite fields

J.-L. Beuchat, T. Miyoshi, and E. Okamoto (Tsukuba University, Japan), and Jean-Michel Muller have analyzed several possible tradeoffs that must be considered when implementing Horner's based multiolication in finite fields. They have published a survey [11].

6.1.6. Hardware Elementary Functions

Previous work on elementary function evaluation [22] has been ported to FloPoCo by Cristian Klein, an internship student.

6.1.7. Applications

With O. Creţ, I. Trestian, L. Creţ, L. Văcariu and R. Tudoran from the Technical University of Cluj-Napoca, F. de Dinechin has worked on the implementation of a hardware accelerator used for the simulation of coils to be used for transcranial magnetic stimulation [16]. The implementation of a full floating-point pipeline on an FPGA reduces simulation time from several hours to a few seconds, and allows practical trial-and-error design of complex coil conformations. This work relies on FPLibrary, and uses in particular its logarithm operator. This work has also lead to investigations on the accumulation of floating-point numbers [40].

6.1.8. Hardware Implementation of the η_T and Tate Pairings in Characteristics 2 and 3

Since their introduction in constructive cryptographic applications, pairings over (hyper)elliptic curves are at the heart of an ever increasing number of protocols. Software implementations being rather slow, the study of hardware architectures became an active research area.

In collaboration with J.-L. Beuchat (Tsukuba Univ., Japan), J. Detrey (CACAO, INRIA Nancy Grand-Est), E. Okamoto (Tsukuba Univ., Japan) and M. Shirase and T. Takagi (Future Univ. of Hakodate, Japan), N. Brisebarre proposed in [10] several algorithms to compute the η_T pairing (introduced by Barreto, Galbraith, Ó hÉigeartaigh and Scott) in characteristic 3. These algorithms involve addition, multiplication, cubing, inversion, and sometimes cube root extraction over $\mathbb{F}_{3^m}$. The authors also proposed a hardware accelerator based on a unified arithmetic operator able to perform the operations required by a given algorithm. and describe the implementation of a compact coprocessor for the field $\mathbb{F}_{3^{97}}$ given by $\mathbb{F}_3[x]/(x^{97}+x^{12}+2)$, which compares favorably with other solutions described in the open literature.

Then, in [27], a joint work with J.-L. Beuchat, J. Detrey, E. Okamoto and Francisco Rodríguez-Henríquez (CInvEstAv, México City, México), N. Brisebarre proposed a study of the modified Tate pairing in characteristics two and three. Starting from the η_T pairing, they detailed various algorithmic improvements in the case of characteristic two. Then they showed how to get back to the modified Tate pairing at almost no extra cost. Finally, they explored the trade-offs involved in the hardware implementation of this pairing for both characteristics two and three. From the experiments they've done, characteristic three appears to have a slight advantage over characteristic two.

6.2. Efficient Polynomial and Rational Approximations

Keywords: *E-method*, *closest vector problem*, *floating-point arithmetic*, *lattice basis reduction*, *linear programming*, *minimax*, *polynomial approximation*, *rational approximation*.

Participants: Nicolas Brisebarre, Sylvain Chevillard, Jean-Michel Muller, Serge Torres.

6.2.1. Towards An Automatic Approach for the Hardware Evaluation of Functions

In [28], a joint work with M. Ercegovac (University of California at Los Angeles), N. Brisebarre, S. Chevillard, J.-M. Muller and S. Torres extend the domain of applicability of the E-method (due to M. Ercegovac), as a hardware-oriented method for evaluating elementary functions using polynomial and rational function approximations. Until this paper, there was no systematic approach to obtain good approximations to f over an interval $[a, b]$ by rational functions satisfying the constraints required by the E-method. Such an approach which is based on linear programming and lattice basis reduction is presented. A design and performance characteristics of a corresponding implementation are also discussed.

6.3. Efficient Floating-Point Arithmetic and Applications

Keywords: *Newton-Raphson iteration*, *VLIW integer processor*, *code generation and validation*, *division*, *floating-point arithmetic*, *multiplication*, *polynomial evaluation*, *rounding of algebraic functions*, *software implementation*, *square root*.

Participants: Nicolas Brisebarre, Claude-Pierre Jeannerod, Christoph Lauter, Vincent Lefèvre, Nicolas Louvet, Jean-Michel Muller, Guillaume Revy, Gilles Villard.

6.3.1. Multiplication by Constants

N. Brisebarre and J.-M. Muller [15] have published their improved method (first presented at the ARITH'2005 conference) for checking whether multiplication by a given arbitrary precision constant can be done at very low cost (1 floating-point multiplication, and one floating-point fma – fused multiply-add).

6.3.2. Computation of Integer Powers in Floating-point Arithmetic

P. Kornerup, C. Lauter, N. Louvet, V. Lefèvre and J.-M. Muller have proposed several algorithms for accurately computing powers to a positive integer in IEEE-754 floating-point arithmetic, assuming in addition that a *fused multiply-add* (fma) instruction is available on the targeted architecture [19]. Two approaches have been studied : one uses the classic binary exponentation technique, and the other is based on the computation of logarithms and exponentials. For bounded, yet very large values of the exponent, the proposed algorithms return a correctly-rounded results in round-to-nearest mode,*i.e.*, the floating-point number that is nearest the exact value. The algorithms have been implemented on the Itanium architecture to compare their respective performances.

6.3.3. Correctly rounded sums

J.-M. Muller, P. Kornerup, V. Lefèvre and N. Louvet have worked on the problem of computing the correctly rounded sum of a set of floating-point numbers. They have shown that among the set of the algorithms with no comparison performing only rounded-to-nearest floating-point additions/subtractions: *(i)* the 2Sum algorithm introduced by Knuth is optimal, both in terms of number of operations and depth of the dependency graph; *(ii)* under reasonable assumptions, no algorithm exists to compute the rounded-to-nearest sum of $n \geq 3$ floating-point numbers. Starting from an algorithm due to Boldo and Melquiond, it has also been proved that the sum of three floating-point values rounded according to any of the standard directed rounding modes can be determined using only additions/subtractions, provided that the operands are of the same sign.

6.3.4. More accurate and fast polynomial evaluation

P. Langlois and N. Louvet have proposed a new polynomial evaluation algorithm in floating-point arithmetic [33]. The principle is to apply, once or recursively, an error-free transformation of the polynomial evaluation with the Horner algorithm and to accurately sum the final decomposition. This compensated algorithm is as accurate as the Horner algorithm performed in k times the working precision, for an arbitrary positive integer k. This is also an efficient alternative to other software solutions to improve the accuracy of polynomial evaluation, such as the double-double and quad-double libraries.

6.3.5. Binary Floating-point Square Root and Division on VLIW Integer Processors

In a joint work with H. Knochel and C. Monat (STMicroelectronics Compilation Expertise Center, Grenoble), C.-P. Jeannerod and G. Revy have shown in [54] how to reduce the computation of correctly-rounded square roots of binary floating-point data to the fixed-point evaluation of some bivariate integer polynomials. By designing parallel and accurate evaluation schemes for such polynomials, they have shown further that this approach allows for high ILP exposure, and thus potentially low latency implementations on VLIW integer processors. For the *binary32* format a C implementation has been written (and is part of FLIP 1.0 § 5.2). The experiments done with this code and the ST200 VLIW compiler show the practical interest of this approach: for all rounding modes, the generated assembly code is optimally scheduled and has low latency (23 cycles).

C.-P. Jeannerod, H. Knochel, C. Monat, G. Revy and G. Villard then extended this work to division in [55], again for the *binary32* format (but for round-to-nearest only and without subnormals). Compared to square root, the main difficulty was to automatically validate the numerical accuracy of the fast bivariate polynomial evaluation code that we were using to approximate the quotient. This required first to introduce a new set of approximation and evaluation error conditions that are sufficient to ensure correct rounding. Then, some efficient heuristics have been proposed to generate such evaluation codes and to validate their accuracy

according to the new error conditions. Finally, a complete C implementation has been written (which is also part of FLIP 1.0). With the ST200 VLIW compiler the speed-up factor is almost 1.8 (compared to FLIP 0.3).

6.4. Correct Rounding of Elementary Functions

Keywords: *correct rounding*, *double precision*, *elementary functions*, *interval arithmetic*, *libm*, *machine-assisted proofs*, *power function*.

Participants: Sylvain Chevillard, Florent de Dinechin, Mioara Joldes, Christoph Lauter, Vincent Lefèvre, Jean-Michel Muller, Nathalie Revol, Damien Stehlé.

6.4.1. Optimising polynomials for floating-point implementation

C. Lauter and F. de Dinechin have worked on automating the obtention of a machine-optimized polynomial approximation to of a function [34]. The resulting polynomial is implemented as a C program resorting to a minimum amount of double-double or triple-double arithmetic if needed, and provided with a Gappa proof. Tanks to a modified Remez implementation in Sollya, the algorithm extends textbook recipes, such as using an odd polynomial for an odd function, to a much wider class of functions. In addition, this method is purely numerical, and thus works for any function provided as a black box. This algorithm is the core of Metalibm.

6.4.2. Computing a certified bound of the supremum norm of an error

When implementing functions in libms, the function f to be implemented is often replaced by an approximation polynomial p on a closed bounded interval $[a, b]$. In order to guarantee the correctness of the implementation, one has to compute a tight upper bound of the supremum norm of the approximation error $p - f$ or $p/f - 1$ on the interval.

S. Chevillard, M. Joldes and C. Lauter proposed an algorithm for computing efficiently such a bound [52]. The algorithm uses automatic differentiation and interval arithmetic for overcoming the drawbacks of previous approaches. It has been implemented as a prototype using the scripting language of Sollya and is going to be integrated as a part of Sollya.

6.4.3. Computation of function erf with correct rounding in arbitrary precision

S. Chevillard and N. Revol designed an algorithm for computing the special function erf in arbitrary precision and with correct rounding in the sense of the IEEE-754 standard [30]. The algorithm uses the Taylor development of the function. A grouping technique makes it possible to reduce the number of high precision multiplications when evaluating the series. The approach proposed in the paper can be seen as a general scheme for the implementation of other special functions.

6.5. Interval Arithmetic, and Multiple or Arbitrary Precision

Keywords: *arbitrary precision*, *implementation*, *interval arithmetic*, *standardization*.

Participants: Hong Diep Nguyen, Nathalie Revol.

6.5.1. Standardization of Interval Arithmetic

Interval arithmetic now seems mature enough to be standardized. However, several points of view on the exact definition of interval arithmetic exist, in particular regarding the division when the denominator is an interval containing 0, but also on the underlying set (the real line or the real line completed with the infinities), the handling of exceptions (such as the square root of an interval containing negative values), whether wraparound intervals are allowed and what their meaning is, the definition of comparison operators and the implementation of the resulting standard with floating-point arithmetic (cf. §8.2).

These points of view must be surveyed and synthetized, and then discussed.

6.5.2. Automatic Adaptation of the Computing Precision

The goal of this work is to solve a linear system and at the same time certify the calculated solution. Our approach is based on the iterative refinement method for solving a linear system which allows us to approach the exact solution of the system. Meanwhile, instead of using the floating-point arithmetic, we use the interval arithmetic to solve the residual system which allows us to calculate a guaranteed bound of the exact result. In combining these two building blocks, we have managed to give out a precisely approximated solution and be able to state how many exact bits the calculated solution possesses by providing its error bound [45].

6.6. Linear Algebra and Lattice Basis Reduction

Keywords: *LLL lattice reduction*, *algebraic complexity*, *approximant basis*, *asymptotically fast algorithm*, *automatic differentiation*, *certified computation*, *integer matrix*, *lattice*, *matrix error bound*, *polynomial matrix*, *real matrix*, *reduction to matrix multiplication*, *strong lattice reduction*, *structured matrix*, *verification algorithm*.

Participants: Claude-Pierre Jeannerod, Nicolas Louvet, Ivan Morel, Hong Diep Nguyen, Nathalie Revol, Damien Stehlé, Gilles Villard.

6.6.1. Verification Algorithms

Condition numbers are mathematical quantities widely used either for measuring the sensitivity of the solution of a problem to perturbations of the input, or, after a backward error analysis, for deducing first-order error bounds on the computed solution. C.-P. Jeannerod, N. Louvet, N. Revol and G. Villard have worked on the computation of condition numbers using automatic differentiation. They address the question of effectively computing such numbers for basic algebraic problems like inverting and factoring a real matrix, or computing the solution to a linear system. Verification techniques such as multiple precision floating point arithmetic (with MPFI) are used to compute verified enclosures for the condition numbers. Preliminary results have been presented in [43], [42].

6.6.2. Automatic Differentiation for the Matrix Adjoint

Kaltofen has proposed a new approach in 1992 for computing matrix determinants without divisions over an abstract ring. By the results of Baur and Strassen, the approach also leads to algorithms for computing the adjoint matrix using the reverse mode of automatic differentiation, hence somehow that are not "explicit". G. Villard proposes an alternative (still closely related) algorithm for the adjoint that can be implemented directly, we mean without resorting to an automatic transformation [39], [57].

6.6.3. Algebraic Complexity for Structured Linear Algebra

The improved complexity results obtained by C.-P. Jeannerod —in collaboration with A. Bostan (INRIA, project-team Algo) and É. Schost (University of Western Ontario)— for structured linear system solving have been published in [14].

6.6.4. Probabilistic Reduction of Rectangular Lattices

Lattice reduction algorithms such as LLL and its floating-point variants have a very wide range of applications in computational mathematics and in computer science: polynomial factorisation, cryptology, integer linear programming, *etc*. It can occur that a lattice to be reduced has a dimension which is small with respect to the dimension of the space in which it lies. This happens within the LLL algorithm itself. A. Akhavi and D. Stehlé [25] described a randomised algorithm specifically designed for such rectangular matrices. It computes bases satisfying, with very high probability, properties similar to those returned by LLL. The algorithm significantly decreases the complexity dependence in the dimension of the embedding space. The technique mainly consists in randomly projecting the lattice on a lower dimensional space.

6.6.5. Improving the LLL-reducedness of an LLL-reduced Basis

The LLL algorithm allows one to reduce any given basis of a lattice to a "good basis" in polynomial time. The quality of the obtained reduction is directly related to a parameter δ: the higher δ is, the better the reduction. It is folklore that one could gradually reduce a basis by first using a small value of δ, and then increasing the value of δ to reach the maximum quality. I. Morel, D. Stehlé and G. Villard are studying the second phase, which consists in further reducing a basis already reduced with a small value of δ. They showed how to take advantage of the knowledge of the lattice obtained from the first reduction to perform the second one. Preliminary results related to this work have been presented by I. Morel on a poster at the ISSAC'08 conference [59].

6.6.6. LLL-reduction and numerical analysis

Schnorr proved that the LLL algorithm can be speeded up if one computes approximations to the underlying Gram-Schmidt orthogonalisations. Without approximations, these computations dominate the cost of the reduction. D. Stehlé surveyed the existing algorithms and implementations of floating-point LLL algorithms [49]. Also, recently, classical tools from the field of numerical analysis have been revisited and improved in order to strengthen Schnorr's approach, and further reducing the costs. I. Morel, D. Stehlé and G. Villard surveyed these developments, and showed especially how floating-point computations may be introduced at various levels in the reduction process, in [24].

6.6.7. Lower Bounds for Strong Lattice Reduction

The Hermite-Korkine-Zolotarev reduction plays a central role in strong lattice reduction algorithms. By building upon a technique introduced by Ajtai, G. Hanrot and D. Stehlé [53] showed the existence of Hermite-Korkine-Zolotarev reduced bases that are arguably least reduced. They proved that for such bases, Kannan's algorithm solving the shortest lattice vector problem requires $d^{\frac{d}{2e}(1+o(1))}$ bit operations in dimension d. This matches the best complexity upper bound known for this algorithm, obtained earlier by Stehlé and Hanrot. These bases also provide lower bounds on Schnorr's constants α_d and β_d that are essentially equal to the best upper bounds. They also showed the existence of particularly bad bases for Schnorr's hierarchy of reductions.

6.6.8. Floating-point Enumeration of Short Lattice Vectors

The Kannan-Fincke-Pohst enumeration algorithm for the shortest and closest lattice vector problems is the keystone of all strong lattice reduction algorithms and their implementations. In the context of the fast developing lattice-based cryptography, the practical security estimates derive from floating-point implementations of these algorithms. However, these implementations behave very unexpectedly and make these security estimates debatable. Among others, numerical stability issues seem to occur and raise doubts on what is actually computed. X. Pujol and D. Stehlé obtained in [36] the first results on the numerical behavior of the floating-point enumeration algorithm. They provide a theoretical and practical framework for the use of floating-point numbers within strong reduction algorithms, which could lead to more sensible hardness estimates.

6.7. Code and Proof Synthesis

Keywords: *automation*, *certified active code generation*, *code for the evaluation of mathematical expressions*, *representation of mathematical expressions with extra knowledge*.

Participants: Sylvain Chevillard, Claude-Pierre Jeannerod, Nicolas Jourdan, Christoph Lauter, Vincent Lefèvre, Nicolas Louvet, Nathalie Revol, Guillaume Revy, Serge Torres, Gilles Villard.

Our goal is to produce a certified active code generator. Starting from specifications given by the user, such a generator produces a target code (that may be modified, at a low level, by this user) along with a proof that the final code corresponds to the given specifications. The specifications we consider are, roughly speaking, mathematical expressions; the generated code evaluates these expressions using floating-point arithmetic and it satisfies prescribed quality criteria: for instance, it returns the correctly rounded result.

During this year, we worked on defining more precisely which mathematical information must be kept for a given expression and how it is represented, linked to this expression (cf. §8.1 : EVA-Flo).

7. Contracts and Grants with Industry

7.1. Grant from Minalogic/EMSOC

Keywords: *SOCs*, *compilation*, *embedded systems*, *synthesis of algorithms*.

Participants: Claude-Pierre Jeannerod, Nicolas Jourdan, Jean-Michel Muller, Guillaume Revy, Gilles Villard.

Since October 2006, we have been involved in Sceptre, a project of the EMSOC cluster of the Minalogic Competitivity Centre. This project, led by STMicroelectronics, aims at providing new techniques for implementing software on system-on-chips. Within Arénaire, we are focusing on the generation of optimized code for accurate evaluation of mathematical functions; our partner at STMicroelectronics is the Compiler Expertise Center (Grenoble).

7.2. Grant from Région Rhône-Alpes

Keywords: *approximation*, *compilation*, *floating-point arithmetic*, *function evaluation*.

Participants: Nicolas Brisebarre, Sylvain Chevillard, Claude-Pierre Jeannerod, Mioara Joldes, Jean-Michel Muller, Nathalie Revol, Guillaume Revy, Gilles Villard.

Since October 2008, we have obtained a 3-year grant from Région Rhône-Alpes. That grant funds a PhD student, Mioara Joldes. 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 people from the Compilation Expertise Center of STMicroelectronics in Grenoble (C. Bertin, H. Knochel, and C. Monat). STMicroelectronics should soon fund another PhD grant on these themes.

8. Other Grants and Activities

8.1. National Initiatives

8.1.1. ANR EVA-Flo Project

Keywords: *automation*, *floating-point computation*, *specification and validation*.

The EVA-Flo project (Évaluation et Validation Automatiques de calculs Flottants, 2006-2010) is headed by N. Revol (Arénaire). The other teams participating in this project are Dali (LP2A, U. Perpignan), Fluctuat (LIST, CEA Saclay) and Tropics (INRIA Sophia-Antipolis).

This project focuses on the way a mathematical formula (that includes transcendental functions and, possibly, iterations) is evaluated in floating-point arithmetic. Our approach is threefold: study of algorithms for approximating and evaluating mathematical formulae (with accuracy and performance being at stake), validation of such algorithms (especially their numerical accuracy) when developing them, in order to influence the algorithmic choices, and automation of the process.

8.1.2. ANR Gecko Project

Keywords: *algorithm analysis*, *geometry*, *integer matrix*, *polynomial matrix*.

Participant: Gilles Villard.

The Gecko project (Geometrical Approach to Complexity and Applications) funded for three years by ANR and headed by B. Salvy (Algorithms project-team, INRIA Paris-Rocquencourt), ended in october 2008. Other teams participating have been the École polytechnique, Université de Nice Sophia-Antipolis, and Université Paul Sabatier in Toulouse. The project at the meeting point of numerical analysis, effective methods in algebra, symbolic computation and complexity theory, had the goal to improve solution methods for algebraic or linear differential equations by taking geometry into account. Advances in Lyon have been related to taking into account geometrical structure aspects for matrix problems.

8.1.3. ANR LaRedA Project

Keywords: *average-case analysis*, *experiments*, *lattice reduction*.

Participants: Ivan Morel, Damien Stehlé.

The LaRedA project (Lattice Reduction Algorithms, 2008-2010) 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.

8.1.4. ANR TCHATER Project

Keywords: *40Gb/s*, *Digital Signal Processing*, *FPGA*, *optical networks*.

Participants: Florent de Dinechin, 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 will study the FPGA implementation of specific algorithms for polarisation demultiplexing and forward error correction with soft decoding.

8.1.5. Ministry Grant ACI "New Interfaces of Mathematics"

Keywords: *floating-point arithmetic*, *linear programming*, *minimax approximation*, *polynomial approximation*, *polytope*.

Participants: Nicolas Brisebarre, Sylvain Chevillard, Jean-Michel Muller, Serge Torres.

The GAAP project (Étude et outils pour la Génération Automatique d'Approximants Polynomiaux efficaces en machine, 2004-2008) was a collaboration with the LaMUSE laboratory (U. Saint-Étienne). A. Tisserand (CNRS, LIRMM, U. Montpellier) was also part of this project. The goal was the development of software tools aimed at obtaining very good polynomial approximants under various constraints on the size in bits and the values of the coefficients. The target applications were software and hardware implementations, such as embedded systems for instance. The software output of this project is made of the C libraries Sollya and MEPLib.

8.1.6. PAI Reconfigurable systems for biomedical applications

Keywords: *FPGA*, *biomedical engineering*, *floating-point*.

Participants: Florent de Dinechin, Bogdan Pasca.

This PAI *(programme d'actions intégrées)* is headed by F. de Dinechin and O. Creţ from Technical University of Cluj-Napoca in Romania. It also involves the INRIA Compsys team-project in Lyon. Its purpose is to use FPGAs to accelerate the computation of the magnetic field produced by a set of coils. The Cluj-Napoca team provides the biomedical expertise, the initial floating-point code, the computing platform and general FPGA expertise. Arénaire contributes the arithmetic aspects, in particular specializing and fusing arithmetic operators, and error analysis. Compsys brings in expertise in automatic parallelization.

8.1.7. CNRS Grant PEPS FiltrOptim

Keywords: *constrained coefficient*, *digital signal processing*, *fixed-point*, *floating-point*.

Participants: Nicolas Brisebarre, Sylvain Chevillard, Florent de Dinechin, Mioara Joldes, Jean-Michel Muller.

The project *FiltrOptim* (Calcul efficace et fiable en traitement du signal : optimisation de la synthèse de filtres numériques en virgules fixe et flottante) is within the 2008 program *Programmes Exploratoires Pluridisciplinaires (PEPS)* of the *ST2I* department of the *CNRS*. The participants belong to the CAIRN (IRISA, Lannion) and Arénaire (LIP, ENS Lyon) teams. The goal is to develop methods that allow for the efficient implementation of a numerical algorithm of a signal processing filter corresponding to some given constraints usually given by a frequential frame and the requested formats of the coefficients.

8.2. International Initiatives

8.2.1. Contributions to Standardization Bodies

F. de Dinechin, C. Lauter, V. Lefèvre, and J.M. Muller participated in the balloting process for the revision of the IEEE-754 standard for floating-point arithmetic, which was then approved by the IEEE-SA Standards Board in June 2008.

V. Lefèvre participated in the Austin Common Standards Revision Group for the revision of POSIX.1. Open Group Base Specifications Issue 7 / IEEE Std 1003.1-2008 was then approved by the Open Group and the IEEE-SA Standards Board in July and September 2008 respectively.

N. Revol was appointed to create and chair an IEEE working group on the standardization of interval arithmetic. The creation of the working group has been approved by IEEE in June 2008. She has been officially elected to chair the working group in November 2008.

F. de Dinechin gave a talk on arithmetic generators at a meeting of the CoreLib standardization group.

8.2.2. Australian Research Council Discovery Grant on Lattices and their Theta Series

Keywords: *lattice theta series*, *security estimate of NTRU*, *strong lattice reduction*.

Participant: Damien Stehlé.

Jointly with J. Cannon (University of Sydney) and R. Brent (Australian National University, Canberra), D. Stehlé recently obtained a three-year long funding to investigate on short lattice points enumeration. This is intricately related to strong lattice reduction and is thus important to assess the security of lattice-based cryptosystems such as NTRU (http://www.ntru.com). The main target of the project is to improve the theory and the algorithms related to lattice theta series.

8.2.3. Invitations

Ned Nedialkov (U. McMaster, Canada) has been invited, on an INRIA grant, to spend two months in Lyon. The work focused on the standardization of interval arithmetic and on applications to hybrid systems, originating from robotics.

9. Dissemination

9.1. Conferences, Edition

N. Brisebarre has been in the Program Committee of RNC8 (8th Conference on Real Numbers and Computers).

F. de Dinechin was in the program committee of FPL'08 (Field Programmable Logic and Applications). He is a member of the Steering Committee of the *Symposium en Architectures de Machines*.

J.-M. Muller is in the program committees of ARITH-19 (19th IEEE Symposium on Computer Arithmetic), and ASAP'2008 (19th IEEE International Conference on Application-specific Systems, Architectures and Processors). With P. Montuschi (Politecnico di Torino), E. Schwarz (IBM), P. Kornerup (Odense Univ.), he was guest editor of a special section on Computer Arithmetic, scheduled for the february issue of IEEE Transactions on Computers. He is a member of the steering committee of the RNC series of conferences. He is a member of the *board of foundation editors* of JUCS (*Journal for Universal Computer Science*).

N. Revol has been in the program committee of CCA'08 (Computability and Complexity in Analysis).

D. Stehlé has been in the program committee of SCC'2008 (First International Conference on Symbolic Computation and Cryptography).

G. Villard in the editorial board of Journal of Symbolic Computation.
General public meetings:

N. Revol visited high-schools in the region of Lyon (Lyon, Saint-Etienne, Charlieu).

9.2. Doctoral School Teaching

N. Brisebarre gave, jointly with J. Detrey and G. Hanrot (INRIA Nancy) a 30h ÉNSL Master course "Finite Field and Elliptic Curve Arithmetics; Applications to Cryptology" (winter 2008).

J.-M. Muller gave a 30h ÉNSL Master course "Floating-Point Arithmetic" (autumn 2008).

9.3. Other Teaching and Service

V. Lefèvre gives a 24h Master course "Computer Arithmetic" at Université Claude Bernard - Lyon 1 (2008/2009).

D. Stehlé gave a 32h Master course "Lattice reduction algorithms and their cryptographic applications", at Université Claude Bernard - Lyon 1 (spring 2008).

9.4. Leadership within Scientific Community

J.-M. Muller is "chargé de mission" at the ST2I department of CNRS.

N. Revol was appointed to create and chair an IEEE working group on the standardization of interval arithmetic. The creation of the working group has been approved by IEEE in June 2008. She has been officially elected to chair the working group in November 2008.

G. Villard has been vice chair of the LIP laboratory until December 2008, and is chair starting January 2009.

G. Villard is member of the board of the CNRS-GDR Informatique Mathématique (headed by B. Vallée).

9.5. Committees

C.-P. Jeannerod, N. Revol and D. Stehlé have been examiners for the ÉNS admissions (spring - summer 2008).

J.-M. Muller participated to the evaluation committees of the following laboratories: GSCOP (Grenoble, January 2008), LIP6 (Paris, January 2008), LIPN (Villetaneuse, January 2008), LIX (Palaiseau, February 2008), LORIA (Nancy, February 2008), LSV (Cachan, December 2008), LIFL (Lille, December 08).

J.-M. Muller is a member of the Scientific Committee of Grenoble INP (former INPG).

J.-M. Muller was a member of the board of examiners for the PhD defenses of N. Fournel (chair of the board), N. Louvet, P. Grosse, R. Michard and C. Lauter.

G. Villard was in the habilitation committee of L. Imbert (U. Montpellier), and member of the board of examiners for the PhD defense of O. Bouissou (École Polytechnique).

G. Villard was a member of the evaluation committee of the ANR program Défis.

9.6. Seminars, Conference and Workshop Committees, Invited Conference Talks

National meetings:

N. Brisebarre organized a session on Computer Arithmetic at the Rencontres Arithmétique de l'Informatique Mathématique, (Lille, June 2008).

S. Chevillard, C. Lauter, N. Louvet, H. D. Nguyen and B. Pasca gave talks at the *Rencontres Arithmétique de l'Informatique Mathématique* (RAIM).

J.-M. Muller gave a 3-hour invited lecture at the *Journées Nationales de Calcul Formel* (Marseille, October 2008)

N. Revol gave a talk at at the meeting on *Set methods for control theory* of the GDR MACS (Angers, March 2008).

D. Stehlé gave an invited talk at the *Journées Codage et Cryptographie* (C2, Carcans, March 2008).

International seminars and meetings:

N. Brisebarre gave an invited talk at the Australian National University, Macquarie University, Sydney University, Wollongong University (Australia), Tsukuba University (Japan).

S. Chevillard gave an invited talk at the Australian National University and at the Macquarie University (Sydney, Australia).

V. Lefèvre, N. Louvet, J.-M. Muller and N. Revol gave talks at the Dagstuhl Seminar 08021, Germany, January 2008.

D. Stehlé gave invited seminar talks at the Universities of Sydney, Macquarie, Wollongong (Australia) and McGill (Canada).

I. Morel gave an invited seminar talk at Macquarie University (Australia). He also presented a poster at the ISSAC'08 conference.

G. Villard gave an invited seminar talk ar the University of Sydney (Australia).

10. Bibliography

Major publications by the team in recent years

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

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

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

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

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

[6] 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", vol. 64, 2005, p. 135–154.

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

Year Publications

Doctoral Dissertations and Habilitation Theses

[8] C. LAUTER. *Arrondi correct de fonctions mathématiques - fonctions univariées et bivariées, certification et automatisation*, Thèse de doctorat, École Normale Supérieure de Lyon, October 2008.

[9] R. MICHARD. *Opérateurs arithmétiques matériels optimisés*, Ph. D. Thesis, Ecole normale supérieure de lyon - ENS LYON, 06 2008, http://tel.archives-ouvertes.fr/tel-00301285/en/.

Articles in International Peer-Reviewed Journal

[10] J.-L. BEUCHAT, N. BRISEBARRE, J. DETREY, E. OKAMOTO, M. SHIRASE, T. TAKAGI. *Algorithms and Arithmetic Operators for Computing the η_T Pairing in Characteristic Three*, in "IEEE Transactions on Computers", vol. 57, n^{o} 11, November 2008, p. 1454-1468.

[11] J.-L. BEUCHAT, T. MIYOSHI, J.-M. MULLER, E. OKAMOTO. *Horner's Rule-Based Multiplication over GF(p) and GF(p^n): A Survey*, in "International Journal of Electronics", vol. 95, n^{o} 7, 2008, p. 669–685.

[12] J.-L. BEUCHAT, J.-M. MULLER. *Automatic Generation of Modular Multipliers for FPGA Applications*, in "IEEE Transactions on Computers", vol. 57, n^{o} 12, December 2008.

[13] S. BOLDO, G. MELQUIOND. *Emulation of a FMA and Correctly Rounded Sums: Proved Algorithms Using Rounding to Odd*, in "IEEE Transactions on Computers", vol. 57, n^{o} 4, April 2008, p. 462–471, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4358278.

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

[15] N. BRISEBARRE, J.-M. MULLER. *Correctly rounded multiplication by arbitrary precision constants*, in "IEEE Transactions on Computers", vol. 57, n^{o} 2, February 2008, p. 165–174.

[16] O. CREŢ, I. TRESTIAN, R. TUDORAN, L. DARABANT, L. VĂCARIU, F. DE DINECHIN. *Accelerating The Computation of The Physical Parameters Involved in Transcranial Magnetic Stimulation Using FPGA Devices.*, in "Romanian Journal of Information, Science and Technology", vol. 10, n^{o} 4, 2008, p. 361-379.

[17] M. ERCEGOVAC, J.-M. MULLER. *An Efficient Method for Evaluating Complex Polynomials*, in "Journal of VLSI Signal Processing Systems", to appear, 2009.

[18] S. GRAILLAT, PH. LANGLOIS, N. LOUVET. *Algorithms for Accurate, Validated and Fast Polynomial Evaluation*, in "Japan Journal of Industrial and Applied Mathematics", to appear, 2009.

[19] 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", to appear, 2009.

[20] C. LAUTER, V. LEFÈVRE. *An efficient rounding boundary test for `pow(x,y)` in double precision*, in "IEEE Transactions on Computers", to appear, vol. 58, n° 2, February 2009, p. 197–207, http://doi.ieeecomputersociety.org/10.1109/TC.2008.202.

[21] P. Q. NGUYEN, D. STEHLÉ. *Low-dimensional lattice basis reduction revisited*, in "ACM Transactions on Algorithms", 2008, http://hal.inria.fr/inria-00328629/en/.

Articles in National Peer-Reviewed Journal

[22] J. DETREY, F. DE DINECHIN. *Fonctions élémentaires en virgule flottante pour les accélérateurs reconfigurables*, in "Technique et Science Informatiques", vol. 27, n° 6, 2008, p. 673–698.

[23] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *Optimisation d'opérateurs arithmétiques matériels à base d'approximations polynomiales*, in "Technique et science informatiques", vol. 27, n° 6, June 2008, p. 699–718, http://tsi.revuesonline.com/article.jsp?articleId=12222.

[24] I. MOREL, D. STEHLÉ, G. VILLARD. *Analyse numérique et réduction des réseaux*, in "Technique et Science Informatiques", to appear, 2009.

International Peer-Reviewed Conference/Proceedings

[25] A. AKHAVI, D. STEHLÉ. *Speeding-up Lattice Reduction With Random Projections*, in "Proc. 8th Latin American Theoretical Informatics (LATIN'08)", Lecture Notes in Computer Science, vol. 4957, Springer, 2008, p. 293–305.

[26] J.-C. BAJARD, P. LANGLOIS, D. MICHELUCCI, G. MORIN, N. REVOL. *Towards Guaranteed Geometric Computations with Approximate Arithmetics*, in "Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII, part of the SPIE Optics & Photonics 2008 Symposium", vol. 7074, August 2008, 12 pages.

[27] J.-L. BEUCHAT, N. BRISEBARRE, J. DETREY, E. OKAMOTO, F. RODRÍGUEZ-HENRÍQUEZ. *A Comparison between Hardware Accelerators for the Modified Tate Pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$*, in "Second International Conference on Pairing-based Cryptography (Pairing'08)", vol. 5209, Springer Verlag, 2008, p. 297–315.

[28] N. BRISEBARRE, S. CHEVILLARD, M. ERCEGOVAC, J.-M. MULLER, S. TORRES. *An efficient Method for Evaluating Polynomial and Rational Function Approximations*, in "Application-specific Systems, Architectures and Processors", IEEE, 2008, p. 245–250.

[29] N. BRISEBARRE, F. DE DINECHIN, J.-M. MULLER. *Integer and Floating-Point Constant Multipliers for FPGAs*, in "Application-specific Systems, Architectures and Processors", IEEE, 2008, p. 239–244.

[30] S. CHEVILLARD, N. REVOL. *Computation of the error function erf in arbitrary precision with correct rounding*, in "RNC 8 Proceedings, 8th Conference on Real Numbers and Computers", Javier D. Bruguera and Marc Daumas, July 2008, p. 27–36.

[31] O. CREŢ, F. DE DINECHIN, I. TRESTIAN, R. TUDORAN, L. CREŢ, L. VĂCARIU. *FPGA-based Acceleration of the Computations Involved in Transcranial Magnetic Stimulation*, in "Southern Programmable Logic Conference", IEEE, 2008, p. 43-48.

[32] R. BAKER. KEARFOTT, JOHN D. PRYCE, N. REVOL. *Discussions on an Interval Arithmetic Standard at Dagstuhl Seminar 08021*, in "Dagstuhl Seminar on Numerical Validation in Current Hardware Architectures", Lecture Notes in Computer Science, to appear, Annie Cuyt and Walter Krämer and Wolfram Luther and Peter Markstein, 2008.

[33] PH. LANGLOIS, N. LOUVET. *Compensated Horner algorithm in K times the working precision*, in "RNC 8 Proceedings, 8th Conference on Real Numbers and Computers", Javier D. Bruguera and Marc Daumas, July 2008, p. 157–166.

[34] C. LAUTER, F. DE DINECHIN. *Optimising polynomials for floating-point implementation*, in "RNC 8 Proceedings, 8th Conference on Real Numbers and Computers", J. D. BRUGUERA, M. DAUMAS (editors), Javier D. Bruguera and Marc Daumas, July 2008, p. 7–16.

[35] V. LEFÈVRE, D. STEHLÉ, P. ZIMMERMANN. *Worst Cases for the Exponential Function in the IEEE 754r decimal64 Format*, in "Reliable Implementation of Real Number Algorithms: Theory and Practice, Dagstuhl, Germany", Lecture Notes in Computer Science, vol. 5045, Springer, 2008, p. 114–126, http://hal.inria.fr/inria-00068731/en/.

[36] X. PUJOL, D. STEHLÉ. *Rigorous and Efficient Short Lattice Vectors Enumeration*, in "Proceedings of ASIACRYPT'08", Lecture Notes in Computer Science, vol. 5350, Springer, 2008, p. 390-405.

[37] I. TRESTIAN, O. CREŢ, L. CREŢ, L. VĂCARIU, R. TUDORAN, F. DE DINECHIN. *Computing the Inductance of Coils Used for Transcranial Magnetic Stimulation With FPGA Devices*, in "Biomedical Engineering (BioMED)", IASTED, 2008, p. 327-333.

[38] I. TRESTIAN, O. CREŢ, L. CREŢ, L. VĂCARIU, R. TUDORAN, F. DE DINECHIN. *FPGA-based Computation of the Inductance of Coils Used for the Magnetic Stimulation of the Nervous System*, in "Biomedical Electronics and Devices", vol. 1, 2008, p. 151-155.

[39] G. VILLARD. *Differentiation of Kaltofen's division-free determinant algorithm*, in "MICA'2008 : Milestones in Computer Algebra, Stonehaven Bay, Trinidad and Tobago", May 2008.

[40] F. DE DINECHIN, B. PASCA, O. CREŢ, R. TUDORAN. *An FPGA-specific Approach to Floating-Point Accumulation and Sum-of-Products*, in "Field-Programmable Technologies", IEEE, 2008, p. 33–40.

National Peer-Reviewed Conference/Proceedings

[41] N. BRISEBARRE, F. DE DINECHIN, J.-M. MULLER. *Multiplieurs et diviseurs constants en virgule flottante avec arrondi correct*, in "RenPar'18, SympA'2008, CFSE'6", 2008.

Workshops without Proceedings

[42] C.-P. JEANNEROD, N. LOUVET, N. REVOL, G. VILLARD. *Computing Condition Numbers with Automatic Differentiation*, in "SCAN 2008 - 13th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, El Paso, Texas", 2008.

[43] C.-P. JEANNEROD, N. LOUVET, N. REVOL, G. VILLARD. *On the Computation of some Componentwise Condition Numbers*, in "SNSC'08 - 4th International Conference on Symbolic and Numerical Scientific Computing, Hagenberg, Austria", 2008.

[44] P. LANGLOIS, N. LOUVET. *Accurate solution of triangular linear system*, in "SCAN 2008 - 13th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, El Paso, Texas", 2008.

[45] N. REVOL, H.-D. NGUYEN. *Solving and Certifying the Solution of a Linear System*, in "SCAN 2008 - 13th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, El Paso, Texas", 2008.

[46] N. REVOL. *Automatic Adaptation of the Computing Precision*, in "V Taylor Models Workshop", May 2008.

[47] N. REVOL. *Introduction to Interval Analysis and to some Interval-Based Software Systems and Libraries*, in "ECMI 2008, The European Consortium For Mathematics In Industry", July 2008.

[48] N. REVOL. *Survey of Proposals for the Standardization of Interval Arithmetic*, in "SWIM 08: Small Workshop on Interval Methods", June 2008.

Scientific Books (or Scientific Book chapters)

[49] D. STEHLÉ. *Floating-point LLL: theoretical and practical aspects*, in "LLL+25: 25th Anniversary of the LLL Algorithm Conference", to appear, Springer, 2008.

[50] F. DE DINECHIN, M. ERCEGOVAC, J.-M. MULLER, N. REVOL. *Digital Arithmetic*, in "Encyclopedia of Computer Science and Engineering", Wiley, 2008.

Books or Proceedings Editing

[51] P. HERTLING, C. M. HOFFMANN, W. LUTHER, N. REVOL (editors). *Special issue on Reliable Implementation of Real Number Algorithms: Theory and Practice*, Lecture Notes in Computer Science, vol. 5045, 2008.

Research Reports

[52] S. CHEVILLARD, M. JOLDES, C. LAUTER. *Certified and fast computation of supremum norms of approximation errors*, Technical report, n^{o} ensl-00334545, École Normale Supérieure de Lyon, 2008, http://prunel.ccsd.cnrs.fr/ensl-00334545.

[53] G. HANROT, D. STEHLÉ. *Worst-Case Hermite-Korkine-Zolotarev Reduced Lattice Bases*, Technical report, Maths Arxiv, 2008, http://arxiv.org/abs/0801.3331.

[54] C.-P. JEANNEROD, H. KNOCHEL, C. MONAT, G. REVY. *Computing floating-point square roots via bivariate polynomial evaluation*, Technical report, n^{o} ensl-00335792, École Normale Supérieure de Lyon, 2008, http://prunel.ccsd.cnrs.fr/ensl-00335792.

[55] C.-P. JEANNEROD, H. KNOCHEL, C. MONAT, G. REVY, G. VILLARD. *A new binary floating-point division algorithm and its software implementation on the ST231 processor*, Technical report, n^{o} ensl-00335892, École Normale Supérieure de Lyon, 2008, http://prunel.ccsd.cnrs.fr/ensl-00335892.

[56] J.-M. MULLER, P. KORNERUP, V. LEFÈVRE, N. LOUVET. *On the computation of correctly-rounded sums*, Technical report, n^{o} ensl-00331519, École Normale Supérieure de Lyon, 2008, http://prunel.ccsd.cnrs.fr/ensl-00331519/.

[57] G. VILLARD. *Kaltofen's division-free determinant algorithm differentiated for matrix adjoint computation*, Technical report, n^{o} ensl-00335918, École Normale Supérieure de Lyon, 2008, http://prunel.ccsd.cnrs.fr/ensl-00335918.

[58] F. DE DINECHIN, C. KLEIN, B. PASCA. *Generating high-performance arithmetic operators for FPGAs*, Technical report, n^{o} ensl-00321209, École Normale Supérieure de Lyon, 2008, http://prunel.ccsd.cnrs.fr/ensl-00321209/.

Other Publications

[59] I. MOREL. *From an LLL-reduced basis to another*, poster for the ISSAC'08 conference, 2008.