INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Arénaire

Computer Arithmetic

Grenoble - Rhône-Alpes

THEME SYM

Table of contents

1. Team

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.

Head of project-team
Gilles Villard [DR CNRS, HdR]

Administrative assistant
Sylvie Boyer [TR INRIA, 20% on the project]

Research scientists
Nicolas Brisebarre [CR CNRS]
Florent de Dinechin [Associate Professor, *Maître de Conférences*, HdR]
Claude-Pierre Jeannerod [CR INRIA]
Vincent Lefèvre [CR INRIA]
Jean-Michel Muller [DR CNRS, HdR]
Nathalie Revol [CR INRIA]
Damien Stehlé [CR CNRS]

Technical staff
Nicolas Jourdan [Technical Staff]
Serge Torres [Technical Staff, 40% on the project]

PhD students
Francisco Cháves [*European Marie Curie grant* Mathlogaps, thesis defended in September 2007]
Sylvain Chevillard [ÉNS student *Allocataire-moniteur*, 2nd year]
Jérémie Detrey [ÉNS student *Allocataire-moniteur* INSA, thesis defended in February 2007]
Christoph Lauter [*Allocataire-moniteur* MESR ÉNS, 3rd year]
Guillaume Melquiond [ÉNS student *Allocataire-moniteur* INSA, thesis defended in November 2006]
Romain Michard [INRIA grant, 4th year]
Ivan Morel [ÉNS student, 1st year]
Hong Diep Nguyen [INRIA grant, 1st year]
Guillaume Revy [*Allocataire-moniteur* MESR ÉNS, 2nd year]
Nicolas Veyrat-Charvillon [*Allocataire-moniteur* MESR ÉNS, thesis defended in June 2007]

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. Reliability, accuracy, and performance are the major goals that drive our studies. Our goals address various domains such as floating-point numbers, intervals, rational numbers, or finite fields. 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 (e.g., linear algebra in mathematical computing).

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

2007 has seen a decisive convergence of the revision of the IEEE-754 standard for floating-point arithmetic, a process begun in 2000. In its current draft, which many consider close to final, the standard advocates correct rounding of elementary functions, which will lead to more accurate results and better reproducibility. This is a result of many years of concerted efforts, covering most of the directions listed above, by members of Arénaire.

3. Scientific Foundations

3.1. Introduction

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

3.2. Hardware Arithmetic

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

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

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

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

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

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

Hardware-oriented algorithms. Many algorithms are available for the implementation of elementary operators (see for instance [4]). 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 [6] 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 [4] 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 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 [5]. 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 [2] for an essentially optimal result, or [3]). 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** 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. Our automatic tool (see §5.8) computes or verifies such bounds. For increased confidence in the numerical applications, it also generates formal proofs of the arithmetic properties. These proofs can then be used and 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 2007.

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

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

Participants: F. de Dinechin, C. Lauter, J.-M. Muller, G. 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 2007, we released the versions 0.17beta1, 0.18beta1 and 1.0beta1 with 7 more functions: $\sin(\pi x)$, $\cos(\pi x)$ $\tan(\pi x)$ and their inverse functions, and the power function x^y. We also have published our algorithm for correctly-rounded logarithms [28].

The library includes a documentation which makes an extensive tutorial on elementary function software development.

The library has been downloaded more than 3500 times. Its focus on performance and correctness has enabled acceptance of correct rounding as a goal of the revised floating-point standard IEEE-754R. It is also considered for inclusion in the GNU compiler collection.

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

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: C. Lauter, S. Chevillard, N. Jourdan.

Sollya aims at providing a safe, user-friendly, all-in-one environment for manipulating numerical functions. Sollya provides both safe and efficient algorithms: certified infinite norm, multiple-precision faithful-rounding evaluator of composed functions, efficient numerical approximate algorithms for searching approximations of the zeros of a function, finding a polynomial approximation, etc.

Sollya comes with an interpreter for the Sollya scripting language. Existing Sollya primitives can be extended with dynamically-linked user code.

Sollya used to be a small software project called *arenaireplot*. In 2007, it has been highly improved (quicker implementation of existing algorithms and implementation of new features such as the programming language).

Sollya has been used for implementing the polynomial evaluation inside the CRlibm functions added in 2007, along with their Gappa proofs.

Status: Release scheduled for December 2007 / **Target:** ia32, ia64, PPC, ia32-64, other / **License:** CeCILL-C / **OS:** Unix / **Programming Language:** C / **URL:** http://sollya.gforge.inria.fr/

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

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

Participants: J. Detrey, F. de Dinechin.

FPLibrary is a VHDL library that describes arithmetic operators (addition, subtraction, multiplication, division, and square root) 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 the first hardware library to offer parametrized hardware architectures for elementary functions in addition to the basic operations. In 2007, an efficient floating-point accumulator has been added.

FPLibrary is being used by tens of academic or industrial organizations in the world. In 2007, its trigonometric operators have been included in a standard set of benchmarks defined by the Altera corporation and the university of Berkeley.

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

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

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

Participant: F. de Dinechin.

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. The first release of FloPoCo, in 2007, include multipliers of an integer or a floating-point number by a constant, and a generalization of the long accumulator introduced by Kulisch.

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

5.6. MPFI: Multiple Precision Floating-Point Interval Arithmetic

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

Participants: N. Revol, S. Chevillard, C. Lauter, H.D. 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/

5.7. MEPLib: Machine-Efficient Polynomials Library

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

Participants: N. Brisebarre, J.-M. Muller, S. Torres.

This software library is developed within a national initiative Ministry Grant ACI "New interfaces of mathematics" (see §8.1).

MEPLib is a library for automatic generation of polynomial approximations of functions under various constraints, imposed by the user, on the coefficients of the polynomials. The constraints may be on the size in bits of the coefficients or the values of these coefficients or the form of these coefficients. It relies on the linear programming tool PIP and interacts with the Sollya library. It should be useful to engineers or scientists for software and hardware implementations.

Status: Beta release / **Target:** various processors, DSP, ASIC, FPGA / **License:** GPL / **OS:** Unix, Linux, Windows (Cygwin) / **Programming Language:** C / **URL:** http://lipforge.ens-lyon.fr/projects/meplib

5.8. Gappa: a Tool for Certifying Numerical Programs

Keywords: *certification*, *fixed-point arithmetic*, *floating-point arithmetic*, *formal proof*, *roundoff error*.

Participant: G. Melquiond.

Given a logical property involving interval enclosures of mathematical expressions, Gappa tries to verify this property and generates a formal proof of its validity. This formal proof can be machine-checked by an independent tool like the Coq proof-checker, so as to reach a high level of confidence in the certification.

Since these mathematical expressions can contain rounding operators in addition to usual arithmetic operators, Gappa is especially well suited to prove properties that arise when certifying a numerical application, be it floating-point or fixed-point. Gappa makes it easy to compute ranges of variables and bounds on absolute or relative roundoff errors.

Gappa is being used in several projects, including CRlibm, FLIP, Sollya, and CGal [23].

In 2007, the syntax of user rules was extended in order to formally prove a wider set of properties. The accessibility of Gappa was improved when starting new proofs. The support library was extended with new results on floating-point arithmetic and relative errors.

Status: Beta release / **Target:** any / **License:** GPL, CeCILL / **OS:** any / **Programming Language:** C++ / **URL:** http://lipforge.ens-lyon.fr/www/gappa/

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

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

Participants: C.-P. Jeannerod, N. Jourdan, G. Revy.

FLIP is a C library for efficient software support of single precision 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 2007 we have completely rewritten the code for add, sub, square, multiply, and for the most basic algebraic functions: square root, inverse square root, reciprocal, division. This resulted in substantial speed-ups compared to the previous codes. The tools used for the design include Gappa (§5.8) and Sollya (§5.3). Our approach to computing square roots is described in [46].

Status: Beta release / **Target:** VLIW processors (ST200 family from STMicroelectronics) / **License:** LGPL / **OS:** Linux, Windows / **Programming Language:** C / **URL:** http://lipforge.ens-lyon.fr/www/flip/

5.10. MPFR

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

Participant: V. Lefèvre.

MPFR is a multiple-precision floating-point library with correct rounding developed by the Arénaire and Cacao projects. The computation is both efficient and has a well-defined semantics. It copies the good ideas from the IEEE-754 standard. MPFR 2.3.0 was released on 29 August 2007.

The main changes done in 2007 are: new functions, improved tests (they allowed to find bugs in very particular cases), better documentation, bug fixes (which mainly consisted in completing the implementation of some functions on particular cases).

Several software systems use MPFR, for example: the KDE calculator Abakus by Michael Pyne; the ALGLIB.NET project; CGAL (Computational Geometry Algorithms Library) developed by the Geometrica team (INRIA Sophia-Antipolis); the Fluctuat tool developed and used internally at the CEA; Gappa (§5.8); Genius Math Tool and the GEL language, by Jiri Lebl; GCC; Giac/Xcas, a free computer algebra system, by Bernard Parisse; the iRRAM exact arithmetic implementation from Norbert Müller (University of Trier, Germany); the Magma computational algebra system; MPFI (§5.6); the mpfs library, an experiment in stochastic lazy floating-point arithmetic, from Keith Briggs; SAGE; the Wcalc calculator by Kyle Wheeler.

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

5.11. fpLLL

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

Participants: D. Stehlé, I. Morel, G. Villard.

Given a set of vectors with integer coordinates, the aim of fpLLL is to compute an LLL-reduced basis of the euclidean lattice that is spanned by the vectors. The reduction is performed efficiently thanks to the use of floating-point arithmetic for the computation of the underlying Gram-Schmidt orthogonalization (central in LLL).

The computation is both fully rigorous and efficient. A hierarchy of heuristic/fast and provable/slower variants is used in order to provide these two properties simultaneously.

fpLLL is used in the SAGE project, and an earlier version of fpLLL is in MAGMA (http://magma.maths.usyd.edu.au/magma/). fpLLL has been used for computations in fields as diverse as cryptography, computer arithmetic, algorithmic number theory and algorithmic group theory.

In 2007, fpLLL underwent a massive update, in major part by David Cadé, an ENS Master student who was hosted in the team during the summer. The new version, fpLLL-2.1, is faster, and more importantly considerably easier to use and extend.

Status: Version 2.1.6 / **Target:** any / **License:** GPL / **OS:** Linux / **Programming Language:** C++ / **URL:** http://perso.ens-lyon.fr/damien.stehle/english.html

5.12. Exhaustive Tests of the Mathematical Functions

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

Participant: V. Lefèvre.

The programs to search for the worst cases for the correct rounding of mathematical functions (exp, log, sin, cos, etc.) using Lefèvre's algorithm have been further improved, in particular:

- More robust control scripts, working in more environments, with more automation.
- More supported functions.
- One of the scripts has entirely been reimplemented, using MPFR instead of Maple, and with more features.
- The scripts have been modified to be able to support special functions whose worst-case information has been used for [52].
- Profiling information can now be automatically generated (with very little overhead) and used. This allows to gain up to 22% on Opteron processors.

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: N. Brisebarre, J. Detrey, F. de Dinechin, R. Michard, J.-M. Muller, N. Veyrat-Charvillon.

6.1.1. Digit-recurrence Algorithms

Milos D. Ercegovac (University of California at Los Angeles) and J.-M. Muller have improved their results of 2004 on digit recurrence algorithms for complex square-root (for which they obtained a best paper award at the ASAP-2004 conference). They have published the obtained results in a special issue of the Journal of VLSI Signal Processing [20].

The E-method, introduced by Ercegovac, allows efficient parallel solution of diagonally dominant systems of linear equations in the real domain using simple and highly regular hardware. Since the evaluation of polynomials and certain rational functions can be achieved by solving the corresponding linear systems, the E-method is an attractive general approach for function evaluation. Milos D. Ercegovac and J.-M. Muller have generalized the E-method to complex linear systems, and have shown some potential applications such as the evaluation of complex polynomials by a digit-recurrence method [42]. They have derived algorithms and schemes for computing common arithmetic expressions defined in the complex domain as hardware-implemented operators.The operators include Complex Multiply-Add, Complex Sum of Products, Complex Sum of Squares and complex Integer Powers. Again, the proposed approach is to map the expression to a system of linear equations, apply a complex-to-real transform, and compute the solutions to the linear system using a digit-by-digit, the most significant digit first, recurrence method. The components of the solution vector corresponds to the expressions being evaluated. The number of digit cycles is about m for m-digit precision. The basic modules are similar to left-to-right multipliers. The interconnections between the modules are digit-wide [43].

6.1.2. Multiplication in Finite Fields

With J.-L. Beuchat, T. Miyoshi and E. Okamoto (Tsukuba University, Japan), J.-M. Muller has written a survey [13] on the Horner's rule-based algorithms for Multiplication over $GF(p)$ and $GF(p^n)$. They present a generic architecture based on five processing elements and introduce a classification of several algorithms based on their model. They provide the readers with a detailed description of each scheme which should allow them to write a VHDL description or a VHDL code generator.

6.1.3. Improvements on a Method to Compute Hardware Function Evaluation Operators

R. Michard and N. Veyrat-Charvillon have worked with A. Tisserrand, former team member now at LIRMM. They have improved a new method to optimize hardware operators for the evaluation of fixed-point unary functions. Starting from an ideal minimax polynomial approximation of the function, it first looks for a polynomial whose coefficients are fixed-point numbers of small size. It then bounds accurately the evaluation error using the Gappa tool. This work was first published at the Sympa conference in 2006 and the improvements are to be published in 2008 [24].

6.1.4. Power Comparison of Evaluation Methods on FPGAs

R. Michard has started a work on power consumption. It is a study of several hardware operator families and aims at evaluating the consumption of each to be able to choose what kind of operator is required when designing a circuit. This work is based on Xilinx Power Estimator to characterize operators on Xilinx FPGAs. This work is in progress and will be the subject of a publication in 2008.

6.1.5. Hardware Floating-point Elementary Functions

Three hardware algorithms for elementary functions implemented in FPLibrary were published in 2007: a dual sine/cosine operator up to single precision, along with a study of the tradeoffs involved, by J. Detrey, F. de Dinechin [39], and an exponential and a logarithm up to double precision featuring quadratic area using an iterative range reduction, by J. Detrey, F. de Dinechin and X. Pujol (an ÉNS-Lyon student) [40].

6.1.6. 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 [38]. 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.

6.1.7. When FPGAs are Better at Floating-point than Processors

This collaboration with users of Arénaire's floating-point operators has pointed out that the flexibility of FPGA hardware is currently underused when floating-point applications are ported to this target. With his collaborators from UCTN, F. de Dinechin has written a prospective survey of the many ways to better exploit this flexibility [54]. Techniques surveyed include specific accumulator design, operator specialization, operator fusion, combining floating- and fixed-point, coarser operators such as elementary functions, 2Sum, 2Mult or interval operators, etc. This survey defines the philosophy of the FloPoCo project.

6.1.8. Hardware Implementation of the η_T Pairing in Characteristic 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 and E. Okamoto (Tsukuba Univ., Japan) and M. Shirase and T. Takagi (Future Univ. of Hakodate, Japan), N. Brisebarre and J. Detrey proposed in [31], [30], [56] several algorithms to compute the η_T pairing in characteristic three. These algorithms involve addition, multiplication, cubing, inversion, and sometimes cube root extraction over $\mathbb{F}_{3^m}$. The authors also propose 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. The article [30] received the CHES'2007 best paper award.

6.2. Efficient Polynomial Approximation

Keywords: *closest vector problem*, *floating-point arithmetic*, *lattice basis reduction*, *minimax*, *polynomial approximation*.

Participants: N. Brisebarre, S. Chevillard, J.-M. Muller, S. Torres.

We keep on developing methods for generating the best, or at least very good, polynomial approximations (with respect to the infinite norm or the L^2 norm) to functions, among polynomials whose coefficients follow size constraints (e.g., the degree-i coefficient has at most m_i fractional bits, or it has a given, fixed, value). This is a key step in the process of function evaluation.

A method [33], due to N. Brisebarre and S. Chevillard, uses the LLL lattice reduction algorithm to give very good polynomial approximation. This method also makes it possible to accelerate an older one by Brisebarre, Muller and A. Tisserand (CNRS, LIRMM, U. Montpellier) that computes a best polynomial approximation.

About the L^2 norm, N. Brisebarre and G. Hanrot (LORIA, INRIA Lorraine) published a complete study of the problem [35]. Using here again tools from euclidean lattice basis reduction, they gave theoretical and algorithmic results that make it possible to get optimal approximants.

We are developing libraries for generating such polynomial approximations (see Sections 5.3 and 5.7).

6.3. Efficient Floating-Point Arithmetic and Applications

Keywords: *Newton-Raphson iteration*, *emulation*, *floating-point arithmetic*, *integer processor*, *multiplication*, *rounding of algebraic functions*, *square root*.

Participants: N. Brisebarre, C.-P. Jeannerod, C. Lauter, V. Lefèvre, J.-M. Muller, G. Revy.

6.3.1. Multiplication by Constants

N. Brisebarre and J.-M. Muller [16] have improved their 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). Here is an example of a result they get with their techniques. Let C_h and C_ℓ be the constants

$$\begin{cases} C_h &= 884279719003555 \times 2^{-48}, \\ C_\ell &= 4967757600021511 \times 2^{-105}. \end{cases}$$

These two constant are exactly representable in double precision arithmetic. If executed with a fused multiply-add with round-to-nearest mode, the following algorithm

$$\begin{cases} u_1 &\longleftarrow (C_\ell x), \\ u_2 &\longleftarrow (C_h x + u_1). \end{cases}$$

will always return the floating-point number u_2 that is nearest πx.

6.3.2. Computation of Integer Powers in Floating-point Arithmetic

P. Kornerup (Odense University, Denmark), V. Lefèvre and J.-M. Muller [47] have introduced two algorithms for accurately evaluating powers to a positive integer in floating-point arithmetic, assuming a fused multiply-add (fma) instruction is available. They show that their log-time algorithm always produce faithfully-rounded results, discuss the possibility of getting correctly rounded results, and show that results correctly rounded in double precision can be obtained if extended-precision is available with the possibility to round into double precision (with a single rounding).

6.3.3. Emulation of Floating-point Square Roots on VLIW Integer Processors

Correctly-rounded floating-point square roots can be obtained in many ways: by iterative methods (restoring, non-restoring, SRT), by multiplicative methods (Newton, Goldschmidt), or by methods based on the evaluation of high-quality polynomial approximants. With H. Knochel and C. Monat (STMicroelectronics, Grenoble), C.-P. Jeannerod and G. Revy [46] compared those methods in the context of emulation in single precision on the ST231 VLIW integer processor cores of STMicroelectronics. They showed that polynomial-based methods can be made most efficient by exploiting some key architectural features. Then, in [59], they made this approach simpler (using a single polynomial instead of two) and faster (proposing an improved evaluation scheme). This resulted in a latency of 21 cycles for rounding to nearest. Other rounding modes have also been added, yielding similar latencies, and it was shown that supporting subnormals can be done with an overhead of at most 3 cycles. Finally, the approach turns out to extend immediately to other basic algebraic functions like reciprocal, inverse square root and division.

6.4. Correct Rounding of Elementary Functions

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

Participants: N. Brisebarre, S. Chevillard, F. de Dinechin, C. Lauter, V. Lefèvre, J.-M. Muller, D. Stehlé.

6.4.1. Worst-Cases of Periodic Functions for Large Arguments

G. Hanrot, V. Lefèvre, D. Stehlé and P. Zimmermann [44] studied the problem of finding hard to round cases of a periodic function for large floating-point inputs, more precisely when the function cannot be efficiently approximated by a polynomial. This was one of the last few issues that prevented from guaranteeing an efficient computation of correctly rounded transcendentals for the whole IEEE-754 double precision format. They introduced the first non-naive algorithm for that problem, with a heuristic complexity of $O(2^{0.676p})$ for a precision of p bits. The efficiency of the algorithm was shown on the largest IEEE-754 double precision binade for the sine function, and some corresponding bad cases were given.

6.4.2. Bounds for Correct Rounding of the Algebraic Functions

N. Brisebarre and J.-M. Muller have published a method that makes it possible to get bounds on the accuracy with which intermediate calculations must be carried-on to allow for correctly-rounded algebraic functions [16].

6.4.3. Exact Cases of the Power Function

C. Lauter and V. Lefèvre [52] designed and proved a new algorithm to detect the rounding boundaries for the power (x^y) function in double precision. The correct rounding of the power function is currently based on Ziv's iterative approximation process. In order to ensure its termination, cases when x^y falls on a rounding boundary (a floating-point number or a midpoint between two consecutive floating-point numbers) must be filtered out.

This new rounding boundary test consists of a few comparisons with pre-computed constants. These constants are deduced from worst cases for the Table Maker's Dilemma, searched over a small subset of the input domain. This is a novel use of such worst-case bounds.

6.4.4. Machine-assisted Proofs

F. de Dinechin, C. Q. Lauter and G. Melquiond have worked on the use of the Gappa tool for proving an elementary function [55].

6.4.5. Implementation of a Certified Infinite Norm

S. Chevillard and C. Lauter have worked on the problem of providing a certified bound on the infinite norm of a function [36]. This is useful when designing the implementation of a correctly rounded elementary function: the function is approximated by a polynomial, and one needs a validated, yet tight bound on this approximation error [28]. Commercial tools provide only an approximation of the infinite norm, and do not even guarantee that it is an over-approximation. The algorithm published in [36] has been implemented in the Sollya tool.

6.5. Interval Arithmetic, Taylor Models and Multiple or Arbitrary Precision

Keywords: *PVS*, *Taylor models*, *arbitrary precision*, *formal proof*, *implementation*, *interval arithmetic*.

Participants: F. Cháves, N. Revol.

6.5.1. Formal Proofs on Taylor Models Arithmetic

Our implementation of Taylor models arithmetic is developed in PVS. We provide new strategies to recursively apply operators on Taylor models, with as few intervention of the user as possible [37]. With these strategies, users can quickly evaluate a formula or prove an inequality and translate it into a fully qualified proof certified by PVS. We applied them to prove error bounds on polynomial approximations of some functions, one of the applications being a computation involved in air traffic conflict resolution [9].

6.5.2. Accurate Implementations of Taylor Models with Floating-point Arithmetic

The implementation of Taylor models arithmetic may use floating-point arithmetic to benefit from the speed of the floating-point implementation. In [49], we assume that the floating-point arithmetic is compliant with the IEEE-754 standard. We show how to bound roundoff errors and how to take these roundoff errors into account into the interval remainder part. We also propose an implementation of Taylor models with multiple precision coefficients.

6.5.3. Automatic Adaptation of the Computing Precision

When a given computation does not yield the required accuracy, the computation can be done (either restarted or continued) using an increasing computing precision. A natural question is how to increase the computing precision in order to minimize the time overhead, compared to the case where the optimal computing precision is known in advance and used for the computation. We generalize a result by Kreinovich and Rump: we study the case where the computation can benefit from results obtained with a lower precision and thus is not restarted but rather continued. We also propose an asymptotically optimal strategy for adapting the computing precision, which has an overhead tending to 1 when the optimal (unknown) precision tends to infinity.

6.6. Linear Algebra and Lattice Basis Reduction

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

Participants: C.-P. Jeannerod, I. Morel, H. D. Nguyen, N. Revol, G. Villard, D. Stehlé.

6.6.1. Verification Algorithms

Most of the algorithmic complexity questions we have been investigating in the past were in algebraic or bit-complexity models. Our typical target problems were system solution, matrix inversion or canonical forms (see [51]). Following the Arénaire objective of specification and certification (in an approximate arithmetic framework) we start studies around the complexity of computing certified error bounds. Based on componentwise perturbation analyses G. Villard proposes in [50] an approach for computing (certified and effective) error bounds for the R factor of the matrix QR factorization. Currently, the corresponding algorithm costs only six times as much as for computing the factorization itself numerically. The approach is applied for certifying the LLL reducedness of output bases of floating point and heuristic LLL variants.

H.D. Nguyen starts his PhD under the supervision of N. Revol and G. Villard. He studies and experiments the choice of the computing precision and the use of interval arithmetic to get certified enclosures of the error on the solution of linear systems, computed using floating-point arithmetic.

6.6.2. Algebraic Complexity for Linear Algebra: Structured and Black Box Matrices

Linear systems with structure such as Toeplitz, Vandermonde or Cauchy-likeness arise in many algebraic problems, like Hermite-Padé approximation or interpolation of bivariate polynomials. The cost of solving such systems was known to be in $\tilde{O}(\alpha^2 n)$, where n is the size of the matrix and α is its displacement rank. With A. Bostan (INRIA, project-team Algo) and É. Schost (University of Western Ontario), C.-P. Jeannerod showed in [32], [57] that this cost can be reduced to $\tilde{O}(\alpha^{\omega-1} n)$, where ω is a feasible exponent for matrix multiplication. (In both cases, such costs are obtained by means of Las Vegas probabilistic algorithms.) Since $\omega < 2.38$, this yields costs of order $\tilde{O}(\alpha^{1.38} n)$ and thus asymptotically faster solutions to the algebraic problems mentioned above. On the other hand, this bridges a gap between dense linear algebra (where $\alpha = n$ and for which system solving has cost $O(n^\omega)$) and structured linear algebra (where $\alpha = O(1)$ and for which system solving has cost $\tilde{O}(n)$).

Block projections have recently been used, in [61], to obtain an algorithm to find rational solutions for sparse linear systems (the input matrix can be multiplied by a vector using $\tilde{O}(n)$ operations). Unfortunately, the correctness of this algorithm depends on the existence of block projections of matrices and this has been conjectured but not proved. In [41], G. Villard and his co-authors establish the correctness of the algorithm by proving the existence of efficient block projections. The usefulness of these projections is demonstrated by improving bounds for the cost of several matrix problems: the dense inverse of a sparse matrix is computed using an expected number of $\tilde{O}(n^{2.27})$ operations; a basis for the null space of a sparse matrix — and a certification of its rank — are obtained at the same cost. An application to Kaltofen and Villard's Baby-Steps/Giant-Steps algorithms for the determinant and Smith Form of an integer matrix yields algorithms requiring $\tilde{O}(n^{2.66})$ machine operations. The algorithms are probabilistic of the Las Vegas type.

6.6.3. 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 [29] 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.

6.6.4. Improving LLL Reduction Algorithms

I. Morel starts his PhD under the supervision of D. Stehlé and G. Villard, he is studying LLL reduction algorithms.

D. Stehlé and G. Villard propose a componentwise perturbation analysis of the QR factorization for LLL reduced matrices [60]. This will be applied for designing new floating point variants of LLL reduction oblivious of the underlying orthogonalization procedure.

6.6.5. A Tighter Analysis of Kannan's Algorithm

The security of lattice-based cryptosystems such as NTRU, GGH and Ajtai-Dwork essentially relies upon the intractability of computing a shortest non-zero lattice vector and a closest lattice vector to a given target vector in high dimensions. The best algorithms for these tasks are due to Kannan, and, though remarkably simple, their complexity estimates had not been improved since over twenty years. Kannan's algorithm for solving the shortest vector problem (SVP) is in particular crucial in Schnorr's celebrated block reduction algorithm, on which rely the best known generic attacks against the lattice-based encryption schemes mentioned above. G. Hanrot and D. Stehlé [45] improved the complexity upper-bounds of Kannan's algorithms. The analysis provides new insight on the practical cost of solving SVP, and helps progressing towards providing meaningful key-sizes.

6.6.6. 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 [58] 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 [45]. 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.

6.7. 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: S. Chevillard, C.-P. Jeannerod, N. Jourdan, C. Lauter, V. Lefèvre, N. Revol, G. Revy, S. Torres, G. 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.

More precisely, we aim at doing accurate mathematics using a programming language, typically C. Our tools are the operators and functions specified by the IEEE-754 standard for floating-point arithmetic, the compilation of mathematical expressions and optimizations related to the context (ranges of the variables...). We simultaneously aim at providing certificates and proofs concerning the transformations performed during the code generation, using tools such as Gappa (cf. §5.8). The optimizations performed for the code generation imply to explore a solution space, more or less automatically. We aim at providing more automation.

The advantages of this approach are mainly the gain in productivity: a minor modification of the input would otherwise be time-consuming and error-prone if handled manually. A second main advantage is the gain in abstraction, through better understanding of models, specifications and constraints.

During this year, we worked on deciding upon the architecture of such a project (cf. §8.1 : EVA-Flo). What we already have at hand is a toolbox of software that intervene for the code generation. We are also defining a "conductor" algorithm, that will organize when and what each tool will do for the overall code generation. Our work focused mainly on what will be manipulated and transmitted between the tools: which notions and concepts must be represented. The representation is inspired from XML and more precisely from MathML Content, that we enrich with our own underlying mathematical structures and notions (for instance, floating-point arithmetic is not available). This representation is handled in the LEMA proposal (*LEMA: un langage pour les expressions mathématiques annotées*).

7. Contracts and Grants with Industry

7.1. Grant from Minalogic/EMSOC

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

Participants: C.-P. Jeannerod, N. Jourdan, J.-M. Muller, G. Revy, G. 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 HPC Compiler Expertise Center (Grenoble).

7.2. XtremeData University Program

Keywords: *FPGA coprocessor*, *high-performance computing*.

F. de Dinechin was included in the XtremeData University program co-sponsored by Altera, AMD, Sun Microsystems and XtremeData, inc. He received a donation of an XD1000 Development System costing $15,000, and the associated programming tools.

8. Other Grants and Activities

8.1. National Initiatives

8.1.1. ANR EVA-Flo Project

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

Participant: all Arénaire.

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: G. Villard.

The Gecko project (Geometrical Approach to Complexity and Applications, 2005-2008, http://gecko.inria.fr) is funded by ANR and headed by B. Salvy (Algo project-team, INRIA Paris-Rocquencourt). Other teams participating are at the École polytechnique, Université de Nice Sophia-Antipolis, and Université Paul Sabatier in Toulouse. The project is at the meeting point of numerical analysis, effective methods in algebra, symbolic computation and complexity theory. The aim is to improve significantly solution methods for algebraic or linear differential equations by taking geometry into account. In Lyon we concentrate on taking into account geometrical structure aspects for matrix problems.

8.1.3. ANR LaRedA Project

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

Participant: D. 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: F. de Dinechin, G. 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: N. Brisebarre, S. Chevillard, J.-M. Muller, S. Torres.

The GAAP project (*Étude et outils pour la Génération Automatique d'Approximants Polynomiaux efficaces en machine*, 2004-2008) is a collaboration with the LaMUSE laboratory (U. Saint-Étienne). A. Tisserand (CNRS, LIRMM, U. Montpellier) is also part of this project. The goal is 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 are 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: F. de Dinechin, J. Detrey.

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 T. Risset and A. Plesco from 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.

8.2. International Initiatives

8.2.1. Contributions to Standardization Bodies

F. de Dinechin, C. Lauter, V. Lefèvre, G. Melquiond, and J.M. Muller have participated in the balloting process for the revision of the IEEE-754 standard for floating-point arithmetic.

V. Lefèvre participates in the Austin Common Standards Revision Group (for the revision of POSIX IEEE Std 1003.1).

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: D. 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.

9. Dissemination

9.1. Conferences, Edition

N. Brisebarre is in the program committee of RNC 8 (8th Conference on Real Numbers and Computers).

F. de Dinechin has been in the Program Committee of FPL2007 (Field Programmable Logic and Applications). He is a member of the Steering Committee of the *Symposium en Architectures de Machines*.

J.-M. Muller was co-program chairman (with Peter Kornerup, Odense University) of the 18th IEEE Symposium on Computer Arithmetic (ARITH-18), that was held in Montpellier, France, in June 2007. He is co-associate editor of a special issue of the IEEE Transactions on Computers devoted to Computer Arithmetic (2008). He belongs to the editorial board of JUCS (Journal for Universal Computer Science).

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

D. Stehlé has been in the General Committee of the LLL+25 International meeting that was held to celebrate the 25th anniversary of the LLL algorithm.

G. Villard has been in the Program Committee of ISSAC'07 (International Symposium on Symbolic and Algebraic Computation) and PASCO'07 (Parallel Symbolic Computation). Starting 2008 he will be in the editorial board of Journal of Symbolic Computation (Chief ed. Hoon Hong NCSU).

General public meetings:

C.-P. Jeannerod organized with F. Rastello a one-day seminar on the theme of compiler optimization for embedded systems (ENS Lyon, September 2007). The goal was to present the collaborations that exist between the Arénaire and Compsys INRIA project-teams and STMicroelectronics.

N. Revol visited high-schools in the region of Lyon (Décines, Boen-sur-Lignon, Villefranche-sur-Saône). She debated with high-school females students at the 40th anniversary of INRIA (Lille, Dec. 2007).

9.2. Doctoral School Teaching

C.-P. Jeannerod gave a 30h ÉNSL Master course "Algorithms for Computer Arithmetic" (spring 2007).

C.-P. Jeannerod and G. Villard give a 30h ÉNSL Master course "Algebraic computation" (2007/2008).

J.-M. Muller gave a 30h ÉNSL Master course "Elementary Functions (spring 2007).

N. Revol organized a course of the Doctoral School MATHIF, "Applications of Computer Science to Research and Technological Development" and gave one of the lectures.

D. Stehlé gave a 30h ÉNSL Master course "Lattice reduction algorithms and applications" (2007).

9.3. Other Teaching and Service

N. Brisebarre has given undergraduate and graduate courses at Univ. de St-Étienne until July.

F. de Dinechin has taught at ENS-Lyon "Computer Science for Non-Computer Scientists", and "Architecture, Systems and Networks", for which he received the donation of an Altera DE2 development board. He is coordinator of international exchanges for the computer science department.

V. Lefèvre gave practical sessions (first steps with MPFR) at the CEA-EDF-INRIA School *Certified Numerical Computation* at LORIA (October 2007).

V. Lefèvre gives a 24h Master course "Computer Arithmetic" at Université Claude Bernard - Lyon 1 (2007/2008).

N. Revol gave a research talk for the students in computer science of the École Normale Supérieure of Lyon (September 2007).

D. Stehlé gave a general research talk at the welcoming day of the École Normale Supérieure of Lyon (September 2007).

9.4. Leadership within Scientific Community

J.-M. Muller is *Chargé de mission* in the ST2I (Engineering, Signal Processing and Computer Science) of CNRS (this involves participating to evaluation committees of laboratories, to hiring committees, etc.). He participated to the evaluation committee of ANR *Architectures du futur*, and to the steering committee of ANR *Calcul intensif et simulation*.

G. Villard is Vice Chair of LIP laboratory. He is a member of the board of the CNRS-GDR *Informatique Mathématique* (headed by B. Vallée).

9.5. Committees

Hiring Committees. N. Brisebarre, Math. Comm., U. J. Monnet Saint-Étienne. J.-M. Muller and N. Revol, Comp. Sc. Comm., ÉNS Lyon. N. Revol, Comp. Sc. Comm., U. Paris 6. G. Villard, App. Math. Comm., UJF Grenoble.

G. Villard was in the Habilitation Committee of P. Loidreau (ENSTA Paris, Jan. 07), the PhD Committee of F. Chávez (ÉNS Lyon, Sept. 07), and N. Méloni (U. Montpellier, Nov. 07).

C.-P. Jeannerod and N. Revol have been examiners for the ÉNS admissions (spring - summer 2007).

C.-P. Jeannerod has been in the INRIA Rhône-Alpes committee for recruiting junior software-development engineers (spring 2007).

J.-M. Muller was in Habilitation Committee of F. de Dinechin (Univ. de Lyon, June 2007), and in the PhD committees of J. Detrey (ENS Lyon, Jan. 2007), N. Veyrat-Charvillon (ENS Lyon, June 2007), P.-Y. Rivaille (U. Paris 6, September 2007), N. Fournel (ENS Lyon, Nov. 2007), N. Louvet (U. Perpignan, Nov. 2007) and P. Grosse (ENS Lyon, Dec. 2007).

N. Revol belongs to the INRIA Rhône-Alpes committee for recruiting post-doctoral researchers and to the 2007 committee for recruiting junior research scientists (CR2).

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

National meetings:

F. de Dinechin, G. Revy and G. Melquiond gave talks at the *Rencontres Arithmétique de l'Informatique Mathématique*, Montpellier, Jan. 2007.

J.-M. Muller was invited speaker at the Conference in Honor of Donald Knuth (Bordeaux, October 2007).

N. Revol organized a session on Computer Arithmetic at the *Rencontres Arithmétique de l'Informatique Mathématique*, (Montpellier, Jan. 2007). She gave an invited talk at the meeting on *Set methods for control theory*, co-organized by GDR MACS and Robotique (Lyon, Oct. 2007).

G. Revy gave a talk at the meeting on *Set methods for control theory* of the GDR MACS (Paris, November 2007).

D. Stehlé gave an invited talk at the national meeting of the *GDR Informatique Mathématique* (Paris, January 2007). He also gave a talk at the French Computer Algebra Meeting (Luminy, January 2007).

G. Villard gave a talk at the Gecko meeting, Sophia Antipolis, Nov. 2007.

International seminars and meetings:

N. Brisebarre gave a talk at the University of Tsukuba, Japan, April 2007.

S. Chevillard presented a poster at LLL+25 in Caen, France, June 2007.

J. Detrey gave a talk at the University of Tsukuba, Japan, April 2007.

C. Lauter gave a talk to the Intel Numerics Group in Nizhny Novgorod, Russia, in August, and another one at Intel Portland, Oregon, in October.

J.-M. Muller was invited speaker at the 41st Conference on Signals, Systems and Computers (Pacific Grove, California, Nov. 2007).

D. Stehlé gave invited seminar talks at the Universities of Frankfurt [Germany, April 2007] and Wollogong [Australia, December 2007]. He was also an invited speaker at the LLL+25 conference [Caen, June 2007].

G. Villard was invited tutorial speaker at the International Symposium on Symbolic and Algebraic Computation (Waterloo, Canada, Jul. 2007).

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] 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.

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

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

[5] 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.

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

Year Publications

Books and Monographs

[7] 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, to appear, 2008.

[8] P. KORNERUP, J.-M. MULLER (editors). *Proceedings of the 18th IEEE Symposium on Computer Arithmetic*, IEEE Conference Publishing Services, June 2007.

Doctoral dissertations and Habilitation theses

[9] F. J. CHÁVES ALONSO. *Utilisation et certification de l'arithmétique d'intervalles dans un assistant de preuves*, Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, September 2007, http://www.ens-lyon.fr/LIP/Pub/Rapports/PhD/PhD2007/PhD2007-05.pdf.

[10] J. DETREY. *Arithmétiques réelles sur FPGA : virgule fixe, virgule flottante et système logarithmique*, Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, January 2007, http://www.ens-lyon.fr/LIP/Pub/Rapports/PhD/PhD2007/PhD2007-01.pdf.

[11] N. VEYRAT-CHARVILLON. *Opérateurs arithmétiques matériels pour des applications spécifiques*, Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, June 2007, http://www.ens-lyon.fr/LIP/Pub/Rapports/PhD/PhD2007/PhD2007-04.pdf.

[12] F. DE DINECHIN. *Matériel et logiciel pour l'évaluation de fonctions numériques. Précision, performance et validation*, Habilitation à diriger des recherches, Université Claude Bernard - Lyon 1, June 2007, http://www.ens-lyon.fr/LIP/Pub/Rapports/HDR/HDR2007/HDR2007-01.pdf.

Articles in refereed journals and book chapters

[13] 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", to appear, 2008.

[14] S. BOLDO, G. MELQUIOND. *Emulation of a FMA and correctly-rounded sums: proved algorithms using rounding to odd*, in "IEEE Transactions on Computers", to appear, 2008.

[15] N. BRISEBARRE, J.-M. MULLER. *Correct Rounding of Algebraic Functions*, in "Theoretical Informatics and Applications", vol. 41, jan-mars 2007, p. 71–83.

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

[17] J. DETREY, F. DE DINECHIN. *A tool for unbiased comparison between logarithmic and floating-point arithmetic*, in "Journal of VLSI Signal Processing", vol. 49, n^{o} 1, 2007, p. 161–175.

[18] J. DETREY, F. DE DINECHIN. *Parameterized floating-point logarithm and exponential functions for FPGAs*, in "Microprocessors and Microsystems, Special Issue on FPGA-based Reconfigurable Computing", vol. 31, n^{o} 8, December 2007, p. 537–545.

[19] J. DETREY, F. DE DINECHIN. *Fonctions élémentaires en virgule flottante pour les accélérateurs reconfigurables*, in "Architecture des Ordinateurs", Technique et Science Informatiques, to appear, Lavoisier, 2008.

[20] M. D. ERCEGOVAC, J.-M. MULLER. *Complex Square Root with Operand Prescaling*, in "Journal of VLSI Signal Processing", vol. 49, n^{o} 1, October 2007, p. 19–30.

[21] L. FOUSSE, G. HANROT, V. LEFÈVRE, P. PÉLISSIER, P. ZIMMERMANN. *MPFR: A multiple-precision binary floating-point library with correct rounding*, in "ACM Trans. Math. Softw.", vol. 33, n^{o} 2, 2007, http://portal.acm.org/citation.cfm?doid=1236463.1236468.

[22] R. GLABB, L. IMBERT, G. JULLIEN, A. TISSERAND, N. VEYRAT-CHARVILLON. *Multi-mode Operator for SHA-2 Hash Functions*, in "Journal of Systems Architecture, Special issue on "Embedded Cryptographic Hardware"", vol. 53, n^{o} 2-3, 2007, p. 127–138.

[23] G. MELQUIOND, S. PION. *Formally certified floating-point filters for homogeneous geometric predicates*, in "Theoretical Informatics and Applications", vol. 41, n^{o} 1, 2007.

[24] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *Optimisation d'opérateurs arithmétiques matériels à base d'approximations polynomiales*, in "Technique et science informatiques (TSI)", to appear, 2008.

[25] P. NGUYEN, D. STEHLÉ. *Low-dimensional lattice basis reduction revisited*, in "ACM Transactions on Algorithms", to appear, 2008.

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

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

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

Publications in Conferences and Workshops

[29] A. AKHAVI, D. STEHLÉ. *Speeding-up Lattice Reduction With Random Projections*, in "Proc. 8th Latin American Theoretical Informatics, Rio de Janeiro, Brazil", to appear, 2008.

[30] J.-L. BEUCHAT, N. BRISEBARRE, J. DETREY, E. OKAMOTO. *Arithmetic Operators for Pairing-Based Cryptography*, in "Cryptographic Hardware and Embedded Systems – CHES 2007", P. PAILLIER, I. VERBAUWHEDE (editors), Lecture Notes in Computer Science, best paper award, n^{o} 4727, Springer, 2007, p. 239–255.

[31] J.-L. BEUCHAT, N. BRISEBARRE, M. SHIRASE, T. TAKAGI, E. OKAMOTO. *A Coprocessor for the Final Exponentiation of the η_T Pairing in Characteristic Three*, in "Proceedings of Waifi 2007", C. CARLET, B. SUNAR (editors), Lecture Notes in Computer Science, n^{o} 4547, Springer, 2007, p. 25–39.

[32] A. BOSTAN, C.-P. JEANNEROD, É. SCHOST. *Solving Toeplitz- and Vandermonde-like linear systems with large displacement rank*, in "Proc. International Symposium on Symbolic and Algebraic Computation, Waterloo, Canada", ACM Press, August 2007, p. 33–40.

[33] N. BRISEBARRE, S. CHEVILLARD. *Efficient polynomial L^{∞}-approximations*, in "Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH'18)", IEEE Computer Society, 2007, p. 169–176.

[34] 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", to appear, 2008.

[35] N. BRISEBARRE, G. HANROT. *Floating-point L^2 approximations to functions*, in "Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH'18)", IEEE Computer Society, 2007, p. 177–184.

[36] S. CHEVILLARD, C. LAUTER. *A certified infinite norm for the implementation of elementary functions*, in "Seventh International Conference on Quality Software (QSIC 2007)", A. MATHUR, W. E. WONG, M. F. LAU (editors), IEEE, 2007, p. 153–160.

[37] F. J. CHÁVES ALONSO, M. DAUMAS, C. MUÑOZ, N. REVOL. *Automatic strategies to evaluate formulas on Taylor models and generate proofs in PVS*, in "6th International Congress on Industrial and Applied Mathematics (ICIAM'07)", July 2007.

[38] O. CREŢ, I. TRESTIAN, F. DE DINECHIN, L. CREŢ, L. VĂCARIU, R. TUDORAN. *Computing the inductance of coils used for transcranial magnetic stimulation with FPGA devices*, in "Biomedical Engineering (BioMed 2008)", to appear, IASTED, 2008.

[39] J. DETREY, F. DE DINECHIN. *Floating-point trigonometric functions for FPGAs*, in "Intl Conference on Field-Programmable Logic and Applications", IEEE, August 2007, p. 29-34.

[40] J. DETREY, F. DE DINECHIN, X. PUJOL. *Return of the hardware floating-point elementary function*, in "18th Symposium on Computer Arithmetic", IEEE, June 2007, p. 161–168.

[41] W. EBERLY, M. GIESBRECHT, P. GIORGI, A. STORJOHANN, G. VILLARD. *Faster inversion and other black box matrix computation using efficient block projections*, in "Proc. International Symposium on Symbolic and Algebraic Computation, Waterloo, Canada", ACM Press, August 2007, p. 143–150.

[42] M. ERCEGOVAC, J.-M. MULLER. *A Hardware-Oriented Method for Evaluating ComplexPolynomials*, in "Proceedings of 18th IEEE Conference on Application-Specific Systems, Architectures and Processors (ASAP'2007)", IEEE Conference Publishing Services, July 2007.

[43] M. ERCEGOVAC, J.-M. MULLER. *Complex Multiply-Add and Other Related Operators*, in "Proceedings of SPIE Conf. Advanced Signal Processing Algorithms, Architectures and Implementation XVII", August 2007.

[44] G. HANROT, V. LEFÈVRE, D. STEHLÉ, P. ZIMMERMANN. *Worst Cases of a Periodic Function for Large Arguments*, in "Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH'18)", IEEE computer society, 2007, p. 133–140, http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=4272859.

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

[46] C.-P. JEANNEROD, H. KNOCHEL, C. MONAT, G. REVY. *Faster floating-point square root for integer processors*, in "IEEE Symposium on Industrial Embedded Systems (SIES'07)", 2007.

[47] P. KORNERUP, V. LEFÈVRE, J.-M. MULLER. *Computing Integer Powers in Floating-Point Arithmetic*, in "Proceedings of 41th Conference on signals, systems and computers", IEEE Conference Publishing Services, November 2007.

[48] V. LEFÈVRE, D. STEHLÉ, P. ZIMMERMANN. *Worst Cases for the Exponential Function in the IEEE 754r decimal64 Format*, in "Proc. Reliable Implementation of Real Number Algorithms: Theory and Practice", Lecture Notes in Computer Science, to appear, Springer-Verlag, 2008.

[49] N. REVOL. *Implementing Taylor models arithmetic with floating-point arithmetic*, in "6th International Congress on Industrial and Applied Mathematics (ICIAM'07)", July 2007.

[50] G. VILLARD. *Certification of the QR Factor R, and of Lattice Basis Reducedness*, in "Proc. International Symposium on Symbolic and Algebraic Computation, Waterloo, Canada", ACM Press, August 2007, p. 361–368.

[51] G. VILLARD. *Some recent progress in symbolic linear algebra and related questions (invited tutorial)*, in "Proc. International Symposium on Symbolic and Algebraic Computation, Waterloo, Canada", ACM Press, August 2007, p. 391–392.

Internal Reports

[52] C. LAUTER, V. LEFÈVRE. *An efficient rounding boundary test for pow(x,y) in double precision*, Technical report, n^{o} RR2007-36, Laboratoire de l'Informatique du Parallélisme, Lyon, France, August 2007, http://prunel.ccsd.cnrs.fr/ensl-00169409.

[53] G. VILLARD. *Certification of the QR Factor R, and of Lattice Basis Reducedness (extended version)*, Technical report, n^{o} hal-00127059, HAL, 2007, http://hal.archives-ouvertes.fr/hal-00127059/en.

[54] F. DE DINECHIN, J. DETREY, I. TRESTIAN, O. CREŢ, R. TUDORAN. *When FPGAs are better at floating-point than microprocessors*, Technical report, n^{o} ensl-00174627, École Normale Supérieure de Lyon, 2007, http://prunel.ccsd.cnrs.fr/ensl-00174627.

[55] F. DE DINECHIN, C. Q. LAUTER, G. MELQUIOND. *Certifying floating-point implementations using Gappa*, Technical report, n^{o} ensl-00200830, arXiv:0801.0523, École Normale Supérieure de Lyon, 2007, http://prunel.ccsd.cnrs.fr/ensl-00200830/.

Miscellaneous

[56] 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*, Cryptology ePrint Archive, Report 2007/417, 2007.

[57] A. BOSTAN, C.-P. JEANNEROD, É. SCHOST. *Solving structured linear systems with large displacement rank*, submitted to Theoretical Computer Science, 2007.

[58] G. HANROT, D. STEHLÉ. *Worst-Case Hermite-Korkine-Zolotarev Reduced Lattice Bases*, draft, 2007.

[59] C.-P. JEANNEROD, H. KNOCHEL, C. MONAT, G. REVY. *High-ILP emulation of IEEE floating-point square roots on integer VLIW processors*, draft, 2007.

[60] D. STEHLÉ, G. VILLARD. *Perturbation analysis for the QR factorization of LLL reduced matrices*, Gecko meeting, Sophia Antipolis, France, Nov. 2007, draft, 2007.

References in notes

[61] W. EBERLY, M. GIESBRECHT, P. GIORGI, A. STORJOHANN, G. VILLARD. *Solving sparse integer linear systems*, in "Proc. International Symposium on Symbolic and Algebraic Computation, Genova, Italy", ACM Press, July 2006, p. 63–70.