INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Arénaire

Computer Arithmetic

Rhône-Alpes

THEME SYM

Table of contents

1. Team

Arénaire is a joint project of CNRS, École Normale Supérieure de Lyon, INRIA, and Université Claude Bernard de Lyon. A 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 the team
Gilles Villard [Research Scientist, DR CNRS, HdR]

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

INRIA Scientists
Édouard Bechetoille [Technical Staff, from 2005-10-01 to 2006-11-30]
Claude-Pierre Jeannerod [Research Scientist, CR]
Vincent Lefèvre [Research Scientist, CR]
Nathalie Revol [Research Scientist, CR]

CNRS Scientists
Jean-Michel Muller [Research Scientist, DR, HdR]
Damien Stehlé [Research Scientist, CR]

ÉNS Lyon Scientists
Florent de Dinechin [Associate Professor, *Maître de Conférences*]
Serge Torres [Technical Staff]

U. St-Étienne Scientist
Nicolas Brisebarre [Associate Professor, *Maître de Conférences*]

PhD Students
Francisco Cháves [*European Marie Curie grant* Mathlogaps, 3rd year]
Sylvain Chevillard [ÉNS student *Allocataire-moniteur*, 1st year]
Jérémie Detrey [ÉNS student *Allocataire-moniteur* INSA, 4th year]
Christoph Lauter [*Allocataire-moniteur* MESR ÉNS, 2nd year]
Guillaume Melquiond [ÉNS student *Allocataire-moniteur* INSA, thesis defended in November 2006]
Romain Michard [INRIA grant, 3rd year]
Saurabh Kumar Raina [Grant from the *Région Rhône-Alpes*, thesis defended in September 2006]
Guillaume Revy [*Allocataire-moniteur* MESR ÉNS, 1st year]
Nicolas Veyrat-Charvillon [*Allocataire-moniteur* MESR ÉNS, 3rd year]

2. Overall Objectives

2.1. Overall Objectives

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 based on proof assistants; 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*, *formal proof and validation*, 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., the dedicated case where 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.
- *Formal proof and validation.* For certifying the properties we identify, and the compliance of the numerical programs we develop with their specifications, we rely on formal proving. Proofs are checked using proof assistants such as Coq and PVS. In particular, this is made possible by a careful specification of the arithmetic operators that are involved. Further, an increasingly growing demand exists for certified numerical results, that is, for computing with known or controlled errors. We answer with the conception of modern and efficient error-measurement tools (roundoff errors, polynomial and power series approximation errors, etc.).

- *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*.

3. Scientific Foundations

3.1. Introduction

As stated above, four major directions in Arénaire are *hardware arithmetic*, *algebraic and elementary functions*, *formal proof and validation*, 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. Many special-purpose processors, typically DSP cores, still do 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 software emulation of floating-point arithmetic should be compliant with the IEEE-754 standard. On the other hand, 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 the STMicroelectronics ST200 family of VLIW processor cores; such 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. Formal proof and validation

Specification of arithmetic operators. Very mediatized problems such as the Pentium bug show that arithmetic correctness is sometimes difficult to obtain on a computer. Few tools handle rigorous proofs on floating-point data. However, thanks to the IEEE-754 standard, the arithmetic operations are completely specified, which makes it possible to build proofs of algorithms and properties. But it is difficult to present a proof including the long list of special cases generated by these calculations. The formalization of the standard that we started in 2000 makes it possible to use a proof assistant such as Coq to guarantee that each particular case is considered and handled correctly. Thanks to funding from CNRS and NASA, the same specification is now also available in PVS.

However the IEEE-754 standard deals with neither elementary functions nor multiple precision. It does not provide language bindings either. For instance, a C implementation may use the extended precision (when available) for computations on double-precision types. Moreover, even when an x86 processor is configured to round on 53 bits (the mantissa size of the double precision), the exponent range is still larger than the one of the double-precision format. As a consequence, the proofs may need to be written for various specifications, depending on the considered targets.

Proof assistants. Systems such as Coq and PVS make it possible to define new objects and to derive formal consequences of these definitions. Thanks to higher order logic, we establish properties in a very general form. The proof is built in an interactive way by guiding the assistant with high level tactics. At the end of each proof, Coq builds an internal object, called a proof term, which contains all the details of derivations and guarantees that the theorem is valid. PVS is usually considered less reliable because it does not build any proof term.

Certification of numerical codes. Certifying a numerical code is error-prone. The use of a proof assistant will ensure the code correctly follows its specification. Part of the floating-point unit of AMD processors were formally proved, and so was an implementation of the exponential function. Another example is the certification of the arithmetic on Taylor models, which is an extension of interval arithmetic. 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.

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, we especially investigate computer arithmetic, cryptography, and algorithmic number theory. We work on improving the complexity estimates and providing better codes for LLL reduction, and for 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 §5.6) 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.

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 2006.

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 `libm`s transparently.

In 2006, we released the interim versions 0.11beta1 and 0.14beta1, with 4 more functions (`arcsine`, `arccosine`, `expm1` and `log1p`). In particular, some new polynomial evaluation algorithms have been introduced [57].

In these beta versions, the code is complete, working and validated by an extensive self-test procedure, but the proof of correct rounding is not complete yet for all the functions. Actually, most of the work in 2006 focused on improving and automating these proofs.

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

The library has been downloaded more than 2300 times. It is used in the LHC@home project of CERN[47] (http://lhcathome.cern.ch/), and is considered for inclusion as the default libm in several open-source compiler projects.

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. 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) [32]. 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.

It has been extended in 2006 by a dual sine/cosine operator [33], a new, more accurate LNS subtraction operator (contributed by S. Collange and M. Arnold from Lehigh University), and new versions of floating-point logarithm and exponential operators [53] which scale up to double precision.

Status: stable / **Target:** FPGA and ASIC / **License:** LGPL / **OS:** any / **Programming Language:** VHDL / **URL:** http://www.ens-lyon.fr/LIP/Arenaire

5.4. MPFI: Multiple Precision Floating-Point Interval Arithmetic

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

Participant: N. Revol.

Library in C for interval arithmetic using arbitrary precision (arithmetic and algebraic operations, elementary functions). Used by scientists both from France and abroad (Belgium, Germany, UK, USA, India, Colombia, etc.). Modifications made this year mainly concern the compatibility with new releases of MPFR.

Status: stable (alpha for the C++ interface) / **Target:** x86, PPC / **License:** GPL / **OS:** Unix, Windows (Cygwin) / **Programming Language:** C, C++ / **Dependencies:** GMP v4.1.0 or higher, MPFR v2.2.0 or higher / **URL:** http://gforge.inria.fr/projects/mpfi/

5.5. 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 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

5.6. 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.

Various people use this tool in order to certify their numerical code. For example, in CRlibm, floating-point elementary functions are proved with its help [45]. It has also been used to certify robust floating-point geometric filters for the CGAL library [24].

In 2006, the support for fixed-point arithmetic has been improved and the Coq library has been extended to fixed- and floating-point arithmetics. Moreover, support for the HOL Light proof assistant has been added.

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

5.7. FLIP: Floating-point Library for Integer Processors

Participants: C.-P. Jeannerod, S.-K. Raina.

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. This research project has been funded by Région Rhône-Alpes and STMicroelectronics from October 2004 to September 2006. The second release (FLIP 0.2) is now embedded into the standard tools STMicroelectronics delivers to its customers. FLIP 0.3 has been released in 2006.

Status: Beta release (Flip-0.3) / **Target:** VLIW processors (ST200 family from STMicroelectronics) / **License:** LGPL / **OS:** Linux, Windows / **Programming Language:** C / **URL:** http://lipforge.ens-lyon.fr/www/flip

5.8. MPFR

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

Participant: V. Lefèvre.

MPFR is a multiple-precision floating-point library developed by the SPACES team; in particular, it is used by Arénaire. Vincent Lefèvre, who moved from the SPACES team to the Arénaire team in September 2006, is still working on MPFR. MPFR 2.2.1 was released on November 29, 2006.

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

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

6.1.1. Digit-recurrence algorithms

In collaboration with Miloš Ercegovac (University of California at Los Angeles), we have improved our digit-recurrence algorithm for evaluating complex square-roots [19]. We have also adapted Ercegovac's E-method to the digit-recurrence solution of some tridiagonal linear systems [36].

6.1.2. Hardware operators for power computation

In collaboration with Arnaud Tisserand, former team member now at LIRMM, we have worked on new identities to compute power operations with fixed integer exponent ($x^3, x^4, ...$) in unsigned radix-2 fixed-point or integer representations. The proposed method reduces the number of partial products using simplifications based on new identities and transformations. These simplifications are performed both at the logical and the arithmetic levels. The proposed method has been implemented in a VHDL generator that produces synthesizable descriptions of optimized operators. The results of our method have been demonstrated on FPGA circuits [39].

6.1.3. A new method to compute hardware function evaluation operators

A new method to optimize hardware operators for the evaluation of fixed-point unary functions has been developed with Arnaud Tisserand [40]. 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.

6.1.4. Floating-point and LNS operators

We have demonstrated the use of FPLibrary as an unbiased comparison tool [17] to select the number representation (between floating-point and logarithmic number system) which leads to the best implementation for a given application [32]. Then, S. Collange (ENS-Lyon student), under the supervision of M. Arnold at Lehigh University, has designed more accurate LNS operators using a cotransformation-based subtraction.

6.1.5. Floating-point sine/cosine up to single precision

Parametrized operators for evaluating the sine and cosine functions in floating-point (up to single precision) were developed and integrated in the FPLibrary operator suite [33]. The difficulty of the range reduction for these operators brought us to also develop several alternative solutions, covering the accuracy-performance tradeoff.

6.1.6. Floating-point logarithm and exponential up to double precision

An iterative, hardware-oriented algorithms for the evaluation of exponential and logarithm has been designed and implemented by J. Detrey, F. de Dinechin and X. Pujol (ENS-Lyon student) [53]. Compared to the previous table-based approach, it has a longer latency but smaller area. More importantly, the area grows quadratically with the precision of the mantissa, where the previous approach had exponential growth. This allows us to offer double-precision logarithm and exponential operators using only a fraction of the resources of current large FPGAs.

6.2. Number Systems

Keywords: *guard digits*, *number systems*.

Participant: J.-M. Muller.

Redundant number representations are generally used to allow constant time additions, based on the fact that only bounded carry-ripples take place. But carries may ripple out into positions which may not be needed to represent the final value of the result, thus a certain amount of leading guard digits are needed to correctly determine the result. Also, when cancellation during subtractions occurs, there may be nonzero digits in positions not needed to represent the result of the calculation. With Peter Kornerup (Southern Danish University), we show that, for normal redundant digit sets with radix greater than two, a single guard digit is sufficient to determine the value of such an arbitrary length prefix of leading nonzero digits. This is also the case for the unsigned carry-save representation, whereas two guard digits are sufficient and may be necessary for additions in the binary signed-digit and 2's complement carry-save representations. Thus only the guard digits need to be retained during sequences of additions and subtractions. At suitable points, the guard digits may then be converted into a single digit, representing the complete prefix [23].

6.3. Efficient Polynomial Approximation

Keywords: *floating-point arithmetic*, *polynomial approximation*.

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

We have developed 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). Regarding to the infinite norm on a given compact interval, one of these methods, due to N. Brisebarre, J.-M. Muller and A. Tisserand (CNRS, LIRMM, U. Montpellier) reduces to scanning the integer points in a polytope [14], another one, due to N. Brisebarre and S. Chevillard, uses the LLL lattice reduction algorithm [49]. About the L^2 norm, N. Brisebarre and G. Hanrot (LORIA, INRIA Lorraine) started a complete study of the problem [50] and proposed theoretical and algorithmic results that make it possible to get optimal approximants.

Potential applications to hardwired operators have been studied [15] by N. Brisebarre, J.-M. Muller, A. Tisserand and S. Torres. We are developing a library for generating such polynomial approximations (see Section 5.5).

6.4. Tools for Efficient Floating-Point Arithmetic

Keywords: *Newton-Raphson iteration*, *floating-point arithmetic*, *multiplication*, *rounding of algebraic functions*.

Participants: N. Brisebarre, J.-M. Muller, C. Lauter.

6.4.1. Multiplication by constants

In 2005, we have presented at the ARITH symposium an algorithm for correctly rounded multiplication by "exact" constants (that is, by numbers that are not necessarily representable in finite binary precision). That extremely simple algorithm works for all possible floating-point inputs, provided that the constants satisfies conditions that we are able to check using methods we have introduced. We have extended our algorithm to the four rounding modes of the IEEE 754 standard, and to the frequent case where an internal format wider than the target precision is available. The paper presenting these new results is under minor revision for the journal IEEE Transactions on Computers.

6.4.2. Newton-Raphson iteration

P. Kornerup (Southern Danish University) and J.-M. Muller have improved and published [22] last year's results on Newton-Raphson iteration. They aim at finding the best possible seed values when computing $a^{1/p}$ using the Newton-Raphson iteration in a given interval. A natural choice of the seed value would be the one that best approximates the expected result. It turns out that in most cases, the best seed value can be quite far from this natural choice.

More generally, when one evaluates a monotone function $f(a)$ in an interval by building the sequence x_n defined by the Newton-Raphson iteration, the natural choice consists in choosing x_0 equal to the arithmetic mean of the endpoint values. This minimizes the maximum possible distance between x_0 and $f(a)$. And yet, if we perform n iterations, what matters is to minimize the maximum possible distance between x_n and $f(a)$. In several examples, the value of the best starting point varies rather significantly with the number of iterations.

6.4.3. Multiple precision using floating-point arithmetic

Some floating-point applications require an intermediate precision slightly higher than native precision. It is possible to emulate to some extend higher precision formats by representing numbers as an unevaluated sum of native precision numbers. We call these formats double-double or triple-double precision. We have shown that a usage of a combination of them speeds up the evaluation of correct rounding elementary functions for double precision by a factor 10 [25].

In [38] we extend this to other applications, such as Taylor models. Using native precision in Taylor models and accounting for round-off errors in the interval remainder may lead to too inaccurate results for some applications. We present techniques for actually computing the error of some types floating-point operations and for bounding precisely the round-off error induced by a double-double or triple-double base format. Here, we focus on adjusting the trade-off between the required accuracy and the cost of emulating higher precision.

6.5. 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, J.-M. Muller, N. Revol.

6.5.1. Bounds for correct rounding of the algebraic functions

In [13], N. Brisebarre and J.-M. Muller explicit the link between the computer arithmetic problem of providing correctly rounded algebraic functions and some Diophantine approximation issues. This allows them to get bounds on the accuracy with which intermediate calculations must be performed to correctly round these functions.

6.5.2. Exact cases of the power function

C. Lauter has described and implemented an efficient algorithm to filter floating-point numbers (x, y) such that x^y is a floating-point number or a mid-point between such numbers. It is important to detect these cases, which would otherwise lead to an infinite loop in a multilevel correctly rounded implementation of the power function. This new algorithm is much faster in average than the previous ones [56].

6.5.3. Machine-assisted proofs

The proof of the correct rounding property is the main difficulty of the implementation of a correctly rounded function. F. de Dinechin, C. Lauter and G. Melquiond have published the methodology currently used in the CRlibm library, which uses the Gappa tool [45]. This methodology was lacking a certified bound on the approximation error of a function by a polynomial. As no existing tool could provide such a validated infinite norm, S. Chevillard and C. Lauter have developed such a tool [29].

6.5.4. Interval elementary functions

Using CRlibm as a starting point, it is possible to derive elementary functions for double-precision interval arithmetic which 1/ are fully validated, 2/ return the smallest possible floating-point interval, and 3/ have a performance within a factor two of the scalar function, thanks to parallel evaluation of endpoints [44]. In collaboration with S. Maidanov from Intel, F. de Dinechin studied the implementation of such interval functions for the Itanium-2 processor [46].

6.5.5. Applications of CRlibm

F. de Dinechin collaborated with Eric McIntosh and Franck Schmidt at CERN, to ensure portability of a chaotic computation distributed over an untrusted network of heterogeneous machines. Among other techniques, this application uses CRlibm [25]. It has been published in a conference on computational physics [47]. As these issues interest a large community of physicists, F. de Dinechin and G. Villard were invited to publish a survey on the subject [26], following a presentation at the 2005 Workshop on Advanced Computing and Analysis Techniques in Physics Research.

6.5.6. Computation of the error functions in arbitrary precision with correct rounding

We continued our work on the evaluation of the error functions erf and erfc in arbitrary precision with correct rounding. We focused on the optimal strategy to adapt automatically the computing precision when the current precision does not suffice to round correctly the result [30].

6.6. Interval Arithmetic, Taylor Models and Certification

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

Participants: F. Cháves, G. Melquiond, N. Revol.

6.6.1. Interval arithmetic, arbitrary precision, validated infinite norm

Results computed using interval arithmetic and its variants (Taylor models, affine arithmetic...) are enclosures of the real, unknown sought value. This guarantee on the result is needed in the development of the CRlibm library [25], where an upper bound of the approximation error is needed, in order to guarantee the property of correct rounding. A. Touhami was hired as a post-doc and his task was to implement a validated infinite norm on one variable, using arbitrary precision interval arithmetic. Unfortunately (for us), he rapidly got a permanent position as assistant professor in Morocco and left us after only two months.

6.6.2. Formal proofs on Taylor models arithmetic

Computing with a Taylor model amounts to determine a Taylor expansion of arbitrary order, often high, along with an interval which encloses Lagrange remainder, truncation error etc. The advantage of Taylor models, compared to usual interval arithmetic, is to reduce the variable dependency.

To get a further level of certification, we have implemented Taylor models in a proof assistant that formally checks the enclosure property of the result along with computing it. We have enriched our library of arithmetic operations on Taylor models using the proof assistant system PVS, indeed we have proved several elementary functions [31]. This implied to prove several theorems of real analysis prior to the elementary functions.

In collaboration with Martin Hofmann (Maximilian-Ludwig Universität, Munich), we have implemented interval arithmetic in another proof assistant, Isabelle. This is a preliminary to the implementation of Taylor models.

6.6.3. Accurate implementations of Taylor models with floating-point arithmetic

When arithmetic on Taylor models is implemented using floating-point arithmetic for the coefficients of the Taylor models, roundoff errors due to the representation and to previous computations are also accounted for in the interval remainder.

The properties of IEEE-754 floating-point arithmetic enables us to compute with "double-double", roughly speaking as if we had twice the computing precision. We propose to take benefit of these properties to implement very accurately Taylor models using floating-point arithmetic [43].

6.7. Algorithms and Software for High Performance Linear Algebra

Keywords: *algebraic complexity*, *asymptotically fast algorithm*, *matrix factorization*, *polynomial matrix*, *reduction to matrix multiplication*, *structured matrix*.

Participants: C.-P. Jeannerod, G. Villard.

Our study of asymptotically fast algorithms for the most basic operations on *polynomial matrices* has appeared in [21]. There, the target matrices have entries in $K[x]$ for K an arbitrary commutative field and the emphasis is on reductions to the multiplication problem. In particular, we highlight the role played by two problems when designing asymptotically fast algorithms for any of the operations above: computing minimal bases of some matrix Padé approximants, and expanding/reconstructing polynomial matrix fractions.

For *linear algebra over a field*, we have studied fast algorithms both for dense matrices and some structured matrices. We propose in [54] some algorithms for LSP/LQUP matrix decomposition that rely on matrix multiplication and whose complexity is shown to be rank-sensitive. In [58] we present some asymptotically fast probabilistic algorithms for structured linear system solving (and their applications). There, the target matrices are Toeplitz-, Hankel- or Vandermonde-like matrices with large displacement rank.

We have found a new algorithm for exact sparse linear system solving [35]. It is the first algorithm whose worst-case complexity estimate may be less than the one of matrix multiplication (sparse enough matrices). The first published version is based on a conjecture for structured projections in the block-Krylov approach. Since then a full proof has been obtained.

7. Contracts and Grants with Industry

7.1. Région Rhône-Alpes Grant

Keywords: *emulation of floating-point arithmetic*, *integer processor*.

Participants: C.-P. Jeannerod, J.-M. Muller, S.-K. Raina, A. Tisserand.

A three-year joint project with STMicroelectronics has just ended (October 2003-September 2006). It was supported by both the Région Rhône-Alpes and STMicroelectronics. The goal was to design and implement an efficient floating-point library for some embedded processors (namely the ST200 family) that only have integer arithmetic units. This project has resulted in the FLIP software library (see §5.7) and the PhD of S.-K. Raina [11].

7.2. Grant from Minalogic/EMSOC

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

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 (and whose contact for Arénaire is C.-P. Jeannerod), aims at providing new techniques for implementing software on system-on-chips. Within Arénaire, we are focusing on the generation of function approximations at compile-time; our partner at STMicroelectronics is the HPC Compiler Expertise Center (Grenoble).

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 Arenaire.

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, INRIA 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 will concentrate on polynomial and matrix problems (with integer or polynomial entries) including some particular classes of structured matrices.

8.1.3. 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*, 2004-2007) is a collaboration with the LaMUSE laboratory (U. Saint-Étienne). A. Tisserand (CNRS, LIRMM, U. Montpellier) also participates to this project. The goal is the development of a C library MEPLib 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.

8.1.4. "Adaptive and Hybrid Algorithms", Imag-INRIA project

Keywords: *adaptive algorithm*, *optimization*, *reliable computation*.

Participants: N. Revol, G. Villard.

The AHA project (*Adaptive and Hybrid Algorithms*, March 2005-2007) is headed by J.-L. Roch (Laboratoire ID-Imag), and supported by Imag Grenoble and INRIA. Our motivation is the conception of algorithms that may adapt themselves automatically to the execution context. Arénaire is involved for building reliable algorithms (e.g. adaptive precision, general algorithms versus algorithms specific to interval arithmetic, ...). Other partners of the project will focus on parallel developments for problems in optimization and vision.

8.2. European Initiatives

8.2.1. 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. Cret 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.

8.3. International Initiatives

8.3.1. Contributions to Standardization Bodies

The IEEE-754 standard for floating-point arithmetic is under revision. Our work correctly-rounded elementary functions has had an influence on the revision committee: an appendix to the IEEE-754 R proposal now suggests that some elementary functions should be correctly rounded (see http://754r.ucbtest.org/). On November 2, 2006, Paul Zimmermann (Spaces), and Nicolas Brisebarre and Jean-Michel Muller (Arénaire) presented the latest results on correctly-rounded transcendentals to the revision committee.

The challenges encountered when developing the Boost library [16] made clear how an interval arithmetic library and the C++ language have to interact. As a consequence, G. Melquiond wrote, in collaboration with H. Brönnimann (Polytechnic U. Brooklyn, NY USA) and S. Pion (Géométrica team, Sophia-Antipolis), a proposal [51] for integrating interval arithmetic to the C++ standard library which was submitted to the C++ Standards committee (ISO/IEC JTC1/SC22/WG21). H. Brönnimann and S. Pion participated to the Fall meeting of the revision committee in 2006. The extension of this standard proposal to arbitrary precision interval arithmetic has been studied in [42].

9. Dissemination

9.1. Conferences, Edition

É. Bechetoille, F. Cháves, S.-K. Raina organized the *Journées Nationales d'Arithmétique des Ordinateurs* (JNAO) (Lyon, France, May-June 2006).

N. Brisebarre has been in the Program Committee of RNC7 (7th Conference on Real Numbers and Computers).

F. de Dinechin has been in the Program Committee of ICFPT 2006 (International Conference on Field-Programmable Technology).

C.-P. Jeannerod has been in the Program Committee of ISSAC 2006 (International Symposium on Symbolic and Algebraic Computing).

J.-M. Muller has been in the Program Committee of ASAP 2006 (IEEE Conference on Application-specific Systems, Architectures and Processors). He is a co-program chair of the 18th IEEE Symposium on Computer Arithmetic, that will be held in Montpellier in June 2007.

N. Revol co-organized with P. Hertling, C. Hoffmann and W. Luther the Dagstuhl seminar *Reliable Implementation of Real Number Algorithms: Theory and Practice*, 8-13 January 2006. She co-edits the proceedings of this seminar as a special issue of Lecture Notes in Computer Science, to appear in 2007. She was in the scientific committee for the SCAN'06 conference.

G. Villard has been a chair of the steering committee of the International Symposium on Symbolic and Algebraic Computation (2003-2006). He is a member of the program committee of ISSAC'07.

General public meetings:

C.-P. Jeannerod and N. Revol have been at *Mondial des Métiers* in February 2006 for presenting research careers to high-school students.

N. Revol visited high-schools in the region of Lyon (Rillieux-la-Pape, Oyonnax, Saint-Étienne).

9.2. Doctoral School Teaching

N. Brisebarre and C.-P. Jeannerod give a 46h Master course "Algorithms for Computer Algebra and Applications" at Université Claude Bernard - Lyon 1 (2005 / 2006 / 2007).

C.-P. Jeannerod gives a 30h ÉNSL Master course "Algorithms for Computer Arithmetic" (2006 / 2007).

V. Lefèvre gives a 24h Master course "Computer Arithmetic" at Université Claude Bernard - Lyon 1 (2006 / 2007).

J.-M. Muller gives a 30h ÉNSL Master course "Elementary Functions" (2006 / 2007).

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

N. Revol gives a 24h ÉNSL Master course "Validation in Scientific Computing" (2006 / 2007).

9.3. Other Teaching and Service

N. Brisebarre gives undergraduate and graduate courses at Univ. de St-Étienne.

F. de Dinechin gives graduate courses at ENS-Lyon, where he is also Departmental Erasmus Coordinator for computer science.

S. Chevillard, J. Detrey, C. Lauter, G. Melquiond, G. Revy and N. Veyrat-Charvillon are teaching assistants—*moniteurs*—they give courses at the ÉNS, INSA and École Centrale de Lyon.

J. Detrey developed *TutoRISC* [34], a framework for student assignments in *Computer Architecture*, in which the students take part in the design of a simple RISC micro-processor, up to its final implementation of an FPGA circuit (http://perso.ens-lyon.fr/jeremie.detrey/tutorisc/).

9.4. Leadership within Scientific Community

J.-M. Muller was the head of LIP laboratory (around 90 people) until July 2006. He is now "chargé de mission" at the ST2I department of CNRS;

G. Villard is the vice-head of LIP laboratory since July 2006. 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. F. de Dinechin, Comp. Sc. Comm., ÉNS Lyon. J.-M. Muller, Comp. Sc. Comm., ÉNS Lyon. N. Revol, App. Math. Comm., UJF Grenoble and Comp. Sc. Comm., ÉNS Lyon. G. Villard, App. Math. Comm., U. Sc. Tech. Lille and Comp. Sc. Comm., U. Perpignan.

C.-P. Jeannerod and J.-M. Muller were in the PhD Committee (as co-advisor and advisor, respectively) of Saurabh Raina (ENS Lyon, 2006).

J.-M. Muller was in the PhD Committee of Guillaume Melquiond (ENS Lyon, 2006).

N. Revol was in the PhD Committee of Loïc Lamarque (U. Bourgogne, Dec. 2006).

G. Villard has organized the evaluation seminar of INRIA SymB theme, 14-15 November, 2006, Paris.

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

The team members regularly give talks at the Department Seminar and at other French Institutions Seminars, in 2006: N. Brisebarre (LIP; GREYC, Caen), G. Melquiond (LIP6, Paris).

National meetings:

S. Chevillard, J. Detrey, F. de Dinechin, C. Lauter and J.-M. Muller gave talks at the *Journées Nationales d'Arithmétique des Ordinateurs* (JNAO), Lyon, France, May-June 2006.

International meetings:

G. Melquiond gave a talk at the Dagstuhl seminar, Germany, January 2006.

G. Melquiond gave a talk at the TYPES Workshop on Numbers and Proofs, Orsay, France, June 2006.

C.-P. Jeannerod gave an invited talk at the Symbolic Computation Group of the University of Waterloo, Canada, July 2006.

J.-M. Muller gave a 3-hour invited lecture on floating-point arithmetic at the workshop "Algorithmique et Programmation", CIRM, Luminy, May 2006.

N. Brisebarre and J.-M. Muller gave a seminar on the implementation of elementary functions at INTEL Portland (USA), November 2006.

N. Brisebarre and J.-M. Muller gave a presentation on the correct rounding of elementary functions and the design of polynomial approximation at the IEEE 754-R revision committee, Cupertino (USA), November 2006.

N. Brisebarre gave an invited talk at the University of Tsukuba (Japan), September 2006.

N. Brisebarre gave an invited talk at the University of Ostrava (Czech Republic), December 2006.

F. de Dinechin gave a seminar at the Technical University of Cluj-Napoca (Romania), June 2006.

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] M. DAUMAS, N. REVOL (editors). *Special issue on Real Numbers and Computers*, Theoretical Computer Science, vol. 351, n^{o} 1, 2006.

[8] 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, 2006.

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

Doctoral dissertations and Habilitation theses

[10] G. MELQUIOND. *De l'arithmétique d'intervalles à la certification de programmes*, Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, 2006.

[11] S. K. RAINA. *FLIP: a floating-point library for integer processors*, Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, September 2006.

Articles in refereed journals and book chapters

[12] B. BECKERMANN, G. LABAHN, G. VILLARD. *Normal forms for general polynomial matrices*, in "Journal of Symbolic Computation", vol. 41, n^{o} 6, 2006, p. 708–737.

[13] N. BRISEBARRE, J.-M. MULLER. *Correct Rounding of Algebraic Functions*, in "Theoretical Informatics and Applications", to appear, 2006.

[14] 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, http://doi.acm.org/10.1145/1141885.1141890.

[15] N. BRISEBARRE, J.-M. MULLER, A. TISSERAND, S. TORRES. *Hardware Operators for Function Evaluation Using Sparse-Coefficient Polynomials*, in "Electronic Letters", to appear, 2006.

[16] H. BRÖNNIMANN, G. MELQUIOND, S. PION. *The design of the Boost interval arithmetic library*, in "Theoretical Computer Science", vol. 351, 2006, p. 111–118, http://perso.ens-lyon.fr/guillaume.melquiond/doc/06-tcs-rnc5.pdf.

[17] J. DETREY, F. DE DINECHIN. *A tool for unbiased comparison between logarithmic and floating-point arithmetic*, in "Journal of VLSI Signal Processing", to appear, 2007.

[18] J. DETREY, F. DE DINECHIN. *Parameterized floating-point logarithm and exponential functions for FPGAs*, in "Microprocessors and Microsystems", to appear, Elsevier, 2007, http://perso.ens-lyon.fr/jeremie.detrey/publications/pub/DetDin2006_micpro.pdf.

[19] M. ERCEGOVAC, J.-M. MULLER. *Complex Square Root with Operand Prescaling*, in "Journal of VLSI Signal Processing", to appear, 2006.

[20] 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"", to appear, 2006.

[21] C.-P. JEANNEROD, G. VILLARD. *Asymptotically fast polynomial matrix algorithms for multivariable systems*, in "International Journal of Control", vol. 79, n° 11, 2006, p. 1359–1367.

[22] P. KORNERUP, J.-M. MULLER. *Choosing Starting Values for certain Newton-Raphson Iterations*, in "Theoretical Computer Science", vol. 351, n° 1, February 2006, p. 101-110.

[23] P. KORNERUP, J.-M. MULLER. *Leading Guard Digits in Finite-Precision Redundant Representations*, in "IEEE Transactions on Computers", vol. 55, n° 5, May 2006, p. 541–548.

[24] G. MELQUIOND, S. PION. *Formally certified floating-point filters for homogeneous geometric predicates*, in "Theoretical Informatics and Applications", to appear, 2006.

[25] F. DE DINECHIN, C. Q. LAUTER, J.-M. MULLER. *Fast and Correctly Rounded Logarithms in Double-Precision*, in "Theoretical Informatics and Applications", to appear, 2007.

[26] F. DE DINECHIN, G. VILLARD. *High precision numerical accuracy in physics research*, in "Nuclear Inst. and Methods in Physics Research, A", vol. 559, n° 1, 2006, p. 207–210.

Publications in Conferences and Workshops

[27] S. BOLDO, M. DAUMAS, W. KAHAN, G. MELQUIOND. *Proof and certification for an accurate discriminant*, in "SCAN 2006 - 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg, Germany", 2006.

[28] H. BRÖNNIMANN, G. MELQUIOND, S. PION. *Proposing Interval Arithmetic for the C++ Standard*, in "SCAN 2006 - 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg, Germany", 2006.

[29] S. CHEVILLARD, C. Q. LAUTER. *Certified infinite norm using interval arithmetic*, in "SCAN 2006 - 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg, Germany", 2006.

[30] S. CHEVILLARD, N. REVOL. *Computation of the error functions erf and erfc in arbitrary precision with correct rounding*, in "International Conference on Mathematical Software, Spain", 2006.

[31] F. CHÁVES, M. DAUMAS. *A Library to Taylor models for PVS automatic proof checker*, in "Proceedings of the NSF workshop on reliable engineering computing, Savannah, Georgia", 2006, p. 39-52, http://www.gtsav.gatech.edu/workshop/rec06/papers/Chaves_paper.pdf.

[32] S. COLLANGE, J. DETREY, F. DE DINECHIN. *Floating point or LNS: choosing the right arithmetic on an application basis*, in "9th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD'2006), Dubrovnik, Croatia", IEEE Computer Society, August 2006, p. 197-203, http://perso.ens-lyon.fr/jeremie.detrey/publications/pub/ColDetDin2006_dsd.pdf.

[33] J. DETREY, F. DE DINECHIN. *Opérateurs trigonométriques en virgule flottante sur FPGA*, in "RenPar'17, SympA'2006, CFSE'5 et JC'2006, Perpignan, France", October 2006, p. 96-105, http://perso.ens-lyon.fr/jeremie.detrey/publications/pub/DetDin2006_sympa.pdf.

[34] J. DETREY. *TutoRISC : un RISC dans mon FPGA*, in "9èmes Journées Pédagogiques du CNFM, Saint-Malo, France", November 2006, p. 153-158, http://perso.ens-lyon.fr/jeremie.detrey/publications/pub/Det2006_jpcnfm.pdf.

[35] 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.

[36] M. ERCEGOVAC, J.-M. MULLER. *Arithmetic Processor for Solving Tri-Diagonal Systems of Linear Equations*, in "proc. 40th Conference on Signals, Systems and Computers, Pacific Grove, CA", November 2006.

[37] R. GLABB, L. IMBERT, G. JULLIEN, A. TISSERAND, N. VEYRAT-CHARVILLON. *Multi-mode Operator for SHA-2 Hash Functions*, in "Proc. International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA), Las Vegas, Nevada, U.S.A.", June 2006, p. 207–210.

[38] C. Q. LAUTER. *A survey of multiple precision computation using floating-point arithmetic*, in "Taylor Models 2006, Florida", 2006.

[39] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *New Identities and Transformations for Hardware Power Operators*, in "Proc. Advanced Signal Processing Algorithms, Architectures and Implementations XVI, San Diego, California, U.S.A.", F. T. LUK (editor). , SPIE, August 2006.

[40] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *Optimisation d'opérateurs arithmétiques matériels à base d'approximations polynomiales*, in "11e SYMPosium en Architectures nouvelles de machines (SYMPA), Perpignan", October 2006.

[41] J.-M. MULLER. *Generating Functions at Compile-Time*, in "proc. 40th Conference on Signals, Systems and Computers, Pacific Grove, CA", November 2006.

[42] N. REVOL. *Design choices and their limits for an interval arithmetic library. The example of MPFI.*, in "SCAN 2006 - 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg, Germany", 2006.

[43] N. REVOL. *Implementing Taylor models arithmetic with floating-point arithmetic: bounding roundoff errors*, in "Taylor Models 2006, Florida", 2006.

[44] F. DE DINECHIN. *Elementary functions for double-precision interval arithmetic*, in "SCAN 2006 - 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg, Germany", 2006.

[45] F. DE DINECHIN, C. Q. LAUTER, G. MELQUIOND. *Assisted verification of elementary functions using Gappa*, in "Proceedings of the 2006 ACM Symposium on Applied Computing", 2006, p. 1318–1322.

[46] F. DE DINECHIN, S. MAIDANOV. *Software techniques for perfect elementary functions in floating-point interval arithmetic*, in "Real Numbers and Computers", July 2006.

[47] F. DE DINECHIN, E. MCINTOSH, F. SCHMIDT. *Massive Tracking on Heterogeneous Platforms*, in "9th International Computational Accelerator Physics Conference (ICAP)", October 2006.

Internal Reports

[48] S. BOLDO, G. MELQUIOND. *Emulation of a FMA and correctly-rounded sums: proved algorithms using rounding to odd*, Submitted to IEEE-TC, Technical report, n° inria-00080427, HAL, 2006, http://hal.inria.fr/inria-00080427.

[49] N. BRISEBARRE, S. CHEVILLARD. *Efficient polynomial L^∞-approximations*, Submitted to ARITH 18, Technical report, n° RR-6060, HAL, 2006, https://hal.inria.fr/inria-00119513.

[50] N. BRISEBARRE, G. HANROT. *Floating-point L^2 approximations*, Submitted to ARITH 18, Technical report, n° RR-6058, HAL, 2006, https://hal.inria.fr/inria-00119254.

[51] H. BRÖNNIMANN, G. MELQUIOND, S. PION. *A Proposal to add Interval Arithmetic to the C++ Standard Library*, Technical report, n° 2137, C++ standardization committee, 2006, http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2137.pdf.

[52] H. BRÖNNIMANN, G. MELQUIOND, S. PION. *Bool_set: multi-valued logic*, Technical report, n° 2136, C++ standardization committee, 2006, http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2136.pdf.

[53] J. DETREY, F. DE DINECHIN, X. PUJOL. *Return of the hardware floating-point elementary function*, Technical report, n° 2006-45, Laboratoire de l'Informatique du Parallélisme, 2006, http://prunel.ccsd.cnrs.fr/ensl-00117386.

[54] C.-P. JEANNEROD. *LSP Matrix Decomposition Revisited*, Research report, n° RR2006-28, Laboratoire de l'Informatique du Parallélisme, ENS Lyon, September 2006, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2006/RR2006-28.pdf.

[55] C. Q. LAUTER, S. CHEVILLARD. *A certified infinite norm for the validation of numerical algorithms*, Research report, n° RR2006-49, Laboratoire de l'Informatique du Parallélisme, Lyon, France, 2006, http://prunel.ccsd.cnrs.fr/ensl-00119810.

[56] C. Q. LAUTER. *Exact and mid-point rounding cases of power(x,y)*, Technical report, n° 2006-46, Laboratoire de l'Informatique du Parallélisme, 2006, http://prunel.ccsd.cnrs.fr/ensl-00117433.

[57] G. REVY. *Analyse et implantation d'algorithmes rapides pour l'évaluation polynomiale sur les nombres flottants*, Technical report, Laboratoire de l'Informatique du Parallélisme - ENS Lyon, 2006, http://prunel.ccsd.cnrs.fr/ensl-00119498.

Miscellaneous

[58] A. BOSTAN, C.-P. JEANNEROD, E. SCHOST. *Solving structured linear systems of large displacement rank*, July 2006, ISSAC 2006 poster.