INRIA

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 the CNRS, the École Normale Supérieure de Lyon, the Inria, and the Université Claude Bernard de Lyon. 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, CR CNRS]

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

Inria Scientists
Nicolas Brisebarre [Research Scientist CR, (on partial secondment), since 01/09/02]
Catherine Daramy-Loirat [Software Development Staff ODL, 01/09/02 to 31/08/04]
Claude-Pierre Jeannerod [Research Scientist, CR]
Nathalie Revol [Research Scientist, CR]
Arnaud Tisserand [Research Scientist, CR]

CNRS Scientists
Marc Daumas [Research Scientist, CR]
Jean-Michel Muller [Research Scientist, DR]

Faculty Member ÉNS Lyon
Florent Dupont de Dinechin [Associate Professor, *Maître de Conférences*]

Post-Doctoral Fellow
Jean-Luc Beuchat [Post-doctoral fellow of the *Fonds National Suisse de la Recherche Scientifique*, since 01/11/01]

Ph. D. Students
Sylvie Boldo [ÉNS student *Allocataire-monitrice* ÉNS, Ph.D. defense 11/22/04]
Nicolas Boullis [ÉNS student *Allocataire-moniteur* INSA, 4th year]
Francisco Cháves [*European Marie Curie grant* Mathlogaps, 1st year (since 10/01/04)]
Jérémie Detrey [ÉNS student *Allocataire-moniteur* INSA, 2nd year]
Pascal Giorgi [*Allocataire* MESR, Ph.D. defense 12/20/04]
Guillaume Melquiond [ÉNS student *Allocataire-moniteur* INSA, 2nd year]
Romain Michard [Inria grant, 1st year (since 09/01/04)]
Saurabh Kumar Raina [Grant from the *Région Rhône-Alpes*, 2nd year]
Nicolas Veyrat-Charvillon [*Allocataire-moniteur* MESR ÉNS, 1st year (since 10/01/04)]

Student interns
Nathalie Dessart [ÉNS-Lyon, graduate internship, March/June 2004]
Matthieu Gallet [ÉNS-Lyon, undergraduate internship, June/August 2004]
Nicolas Gast [ÉNS, Paris, undergraduate internship, June/August 2004]
Gaétan Leurent [ÉNS, Paris, undergraduate internship, June/August 2004]
David Pritchard [Inria-MIT undergraduate internship, June/August 2004]
Julien Robert [ÉNS-Lyon, undergraduate internship, June/August 2004]
Xavier Roche [ÉNS-Lyon, undergraduate internship, June/August 2004]

2. Overall Objectives

Keywords: *Computer arithmetic, FPGA circuit, VLSI circuit, approximated computation, computer algebra, elementary function, finite field, floating-point representation, integer computation, interval arithmetic, 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.

We contribute to the improvement of the available arithmetic, at the hardware level as well as at the software level, on computers, processors, dedicated or embedded chips, etc. Improving computing does not necessarily mean getting more accurate results or getting them more quickly: we also take into account other constraints such as power consumption, or the reliability of numerical software.

Whatever the target (hardware or software), the choice of the number system (and, more generally, of the data representation) is of uttermost importance. Typical examples are the *redundant number systems* (e.g., carry-save, borrow-save). Such systems are used inside multipliers, dividers, etc. The input and output operands of these operators are represented in a conventional number system: only their internal calculations are performed in redundant arithmetic. For a general purpose microprocessor, floating-point arithmetic seems an unavoidable choice (even if current implementations can certainly be improved), but for special-purpose systems, other ways of representing numbers might prove more useful (fixed-point format, some special redundant systems). The ways of representing the elements of a finite field are not standardized, and have strong impact on the performance of a special-purpose circuit. On a higher level, the performance of an interval arithmetic depends on the underlying real arithmetic.

Computer designers have always needed to implement the basic arithmetic functions (with software or hardware), for a medium-size precision (say, on words from 8 to 128 bits). Of course, addition and multiplication have been much studied, but their performance is still critical concerning silicon area (for multiplication) or speed (for both operations). Division and square-root are less critical, but with these operations there certainly remains more room for possible improvement. When elementary functions are at stake (cosine, sine, exponential, logarithm, etc.), algorithm designers have mainly focused on speed or savings of physical resources. Research on algorithms and architectures for multiplication, division and elementary or special functions is still very active. Implemented solutions are still evolving fast. The members of Arénaire have a strong reputation in these domains and they intend to continue to work on them.

Designing a hardwired operator is not only assembling small parts. The designer must also take into account numerous technological constraints and data. Due to the quick evolution of technologies, it is for instance necessary to master the placement and routing tools, if one wishes to design efficient chips. The power consumption of an integrated circuit depends, among other parameters, on its activity, which in turn depends on the value of the inputs: this makes the choice of the number system crucial. Some encodings are used specially in fast algorithms, some others allow to minimize energy consumption.

Validating numerical programs requires the ability to give formal proofs of algorithms, and control the propagation of rounding errors in floating-point computations. For verifying our formal proofs, we use the Coq proof assistant [67]. The formal specification of the operators in the assistant must be done with much care, so that it faithfully reflects the usual associated semantics. Now, the semantics of the floating-point operations is well defined. Indeed, the adoption of the IEEE-754 standard for floating-point arithmetic in 1985 was a major step forward in computer arithmetic. The standard specifies the various formats and the behavior of the floating-point operations. Thanks to the *Arithmétique des Ordinateurs Certifiée* (certified computer arithmetic) Inria New Investigation Grant, we have worked with members of the Lemme and Spaces projects on the proof of our arithmetic algorithms. This collaboration is still active.

Controlling round-off error propagation and, more generally, building systems that are numerically reliable is a more and more important topic. One performs computations that are much bigger than in the Seventies, whereas the accuracy of each individual operation only slightly improved. In many domains, the inaccuracy of a floating-point operation may lead to tragedies. A first solution (that will not solve all problems) consists in building our own floating-point libraries, so that they are better suited to the target applications. Such a library is being developed in cooperation with ST-Microeletronics, in the scope of a project funded by the *Région Rhône-Alpes*.

When conventional floating-point arithmetic does not suffice, we use other kinds of arithmetics. We work on an arbitrary precision interval arithmetic library, that allows to get certified and accurate bounds to solutions. Such intervals give an “exact” answer when the problem is to bound the result of a computation (global optimization). Algorithms dedicated to this arithmetic must be designed in order to get accurate solutions or sometimes simply to avoid divergence, i.e. infinite intervals. We also investigate exact arithmetics in computer

algebra, for computing in algebraic domains such as finite fields, unlimited precision integers, and polynomials (linear algebra in mathematical computing).

3. Scientific Foundations

3.1. Introduction

Our goal is to improve arithmetic operators. Under various hardware and software constraints we focus on reliability, accuracy, and speed. We identify three main directions: hardware arithmetic operators, floating-point operations, and impact of the arithmetic on the algorithms. These three interrelated topics are described below with the methodologies and techniques they implement.

3.2. Hardware arithmetic operators

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

- A software implementation, targeting off-the-shelf microprocessors, is easy to develop and reproduce, but will not always provide the best performance.
- For cost or performance reasons, some applications will be implemented as application specific integrated circuits (or ASIC). 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 (FPGA).

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. The design of hardware arithmetic operators is one of the goals of the Arénaire project.

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 may be used which would require costly emulation on a processor. 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.

A circuit may be optimized for speed or area (circuit cost). In addition, power consumption is becoming an increasingly important challenge in embedded applications. Here again, data and operator specialization has to be combined with generic power-aware techniques to achieve the lowest power consumption.

Those considerations motivate the study of arithmetic operators for ASIC and FPGA. More specifically we consider the following aspects.

3.2.1. Number representation

The choice of a number representation system may ease the implementation of a given operation. A typical example is the *logarithmic number system*, where a number is represented by its logarithm in radix 2. In this system, the multiplication and the division are exact (involving no rounding) and easy, but the addition becomes very expensive. A more standard example is that of *redundant* number systems, like carry-save and borrow-save, often used within multipliers and dividers to allow very fast addition of intermediate results. We also work on other number systems such as finite fields or residue number systems for cryptography. In the case of computations on real values, we consider two different solutions with fixed-point and floating-point number systems.

3.2.2. Algorithms

Many algorithms are available for the implementation of elementary operators. 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 (area and power budget, available low-level libraries).

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).

3.2.3. Architectures and tools

Implementing an algorithm (typically defined by equations) in hardware is a non-trivial task. For example, control signals are needed for correct initialization, most circuits involve memory elements and clock signals which have to be managed carefully, etc.

In this process, computer-aided design tools play a major role. Unfortunately, such tools currently have very poor arithmetic support (typically only radix-2 integer representations, with simple adders and sometimes multipliers). Improving this situation by developing specific design tools is an important research direction.

Finally, even though an algorithm has been formally proven, its hardware realization needs to be checked, as errors may be introduced by the synthesis process and in the physical realization. For this purpose, test vectors are used to validate the final circuit. For small circuits, such vectors may exhaustively test all the combinations of the inputs. When this exhaustive approach becomes impractical, it is the responsibility of the designer to provide test vectors ensuring sufficient coverage of all the possible faults. This again is a non-trivial task.

3.3. Floating-point arithmetic

Floating-point numbers are represented by triplets (s, n, e) associated with

$$(-1)^s \times n \times \beta^e,$$

where β is the radix of the system. In practice, $\beta = 2$ or $\beta = 10$, however, studying the system independently of the value of β allows a better understanding of its behaviour. An arithmetic operator handling floating-point numbers is more complex than the same operator restricted to integer numbers. It is necessary to correctly round the operation with one of the four rounding modes proposed by the IEEE-754 standard (this standard specifies the formats of the numbers and the arithmetic operations), to handle at the same time the mantissa and the exponent of the operands, and to deal with the various cases of exception (infinite, "denormal" numbers, etc).

3.3.1. Formal specifications and proofs

Very mediatized problems (Pentium's bug, $2001!/2000! = 1$ in Maple v7) show that arithmetic correctness is sometimes difficult to handle or to establish 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 peculiar cases generated by these calculations. The formalization of the standard, started with our collaboration with the Lemme project (ARC AOC) in year 2000, makes it possible to use a proof assistant such as Coq [67] to guarantee that each particular case is considered and handled correctly.

Systems such as Coq 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. For example, we used universal quantifiers to establish properties independently of the radix of the floating-point numbers or for an arbitrary rounding mode. 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 which contains all the details of derivations and guarantees that the theorem is valid.

3.3.2. Elementary functions and correct rounding

Many libraries for elementary functions are currently available. The functions in question are typically those defined by the C99 standard, and are offered by vendors of processors, compilers or operating systems. The majority of these libraries attempts to reproduce the mathematical properties of the given functions: monotony, symmetries and sometimes range.

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 accuracy necessary to ensure that rounding this evaluation is equivalent to rounding the exact result, for all possible inputs ? The answer to this question is generally unknown, which means that correctly rounding elementary functions requires arbitrary multiple-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 focussed in 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 large but fixed precision instead of arbitrary multiple-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 prove bounds on the worst-case performance of our functions. This proof concern is mostly absent from IBM's and Sun's libraries, and indeed we have found many misrounded values in each of them.

The design of a library with correct rouding 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.

3.4. Algorithms and arithmetics

Today, scientific computing needs not only floating-point arithmetic or multi-precision arithmetic. On the one hand, when validated results or certified enclosures of a solution are needed, *interval arithmetic* is the arithmetic of choice. It enables to handle uncertain data, such as physical measures, as well as to determine a global optimum of some criterion or to solve a set of constraints. On the other hand, there is an increasing

demand for exact solutions to problems in various areas such as cryptography, combinatorics or algorithmic geometry. Here, symbolic computation is used together with *exact arithmetic*.

General purpose computing environments such as Matlab or Maple now offer all these types of arithmetic and it is even possible to switch from one to another in the middle of a computation. Of course, such capabilities are quite useful and, in general, users already can enhance the quality of the answers to small problems.

However, most general purpose environments are still poorly suited for large computations and interfacing with other existing softwares remains an issue. Our goal is thus to provide high-performance easy-to-reuse software components for interval, mixed interval/multi-precision, finite field, and integer arithmetics. We further aim to study the impact of these arithmetics on algorithms for exact *linear algebra* and constrained as well as unconstrained *global optimization*.

3.4.1. 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 computes with 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.

3.4.2. Computational algorithms for exact linear algebra

The techniques for solving linear algebra problems exactly have been evolving rapidly since a few years, substantially improving the complexity of several algorithms. 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 for the most basic tasks (determinant, matrix inversion, matrix canonical forms, ...), to incorporate matrix multiplication in a systematic way, and to relate bit complexity to algebraic complexity. Another direction is to make these algorithms practically fast as well, especially since applications yield very large matrices that are either sparse or structured. The techniques used to achieve our goals are quite diverse: they range from probabilistic preconditioning via random perturbations to blocking, to the baby step /giant step strategy, to symbolic versions of the Krylov-Lanczos approach, and to approximate arithmetic.

Within the LinBox international project (see §5.5 and §8.3) we work on a software library that corresponds to our algorithmic research mentioned above. Our goal is to provide 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; it further offers very efficient implementations for dense linear algebra over finite fields. The library is being developed and improved, with a special emphasis on the sensitivity of computational costs to the underlying arithmetic implementations. The target matrix entry domains are finite fields and their algebraic extensions, integers and polynomials.

4. Application Domains

Keywords: *arithmetic operator, control, cypher, 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.

- Developments in **Coq** are used to formally **contain values and round-off errors** for **safety critical** applications such as flight control [33]. Our automatic tool (see §5.10) checks for overflows and performs forward error analysis with interval arithmetic. It generates all the necessary assessments and proofs related to each variable of a given program. Such technique has been coined as *invisible formal methods*. Our tool also refers to our growing library of validated properties to enhance the containment intervals.
- Developments of **correctly rounded elementary functions** is critical to the **reproducability** 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.
- 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 application domains of hardware arithmetic operators are **digital signal processing**, **image processing**, **embedded applications** and **cryptography**.

5. Software

5.1. Crlibm: A library of elementary functions with correct rounding

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

Participants: C. Daramy-Loirat, F. de Dinechin, J.-M. Muller.

This library is partially funded by an Inria ODL (C. Daramy-Loirat).

The Crlibm project [58] 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 2004, we added to the two functions already present (exp and log), the hyperbolic sine and cosine (M. Gallet ÉNS student), the arctangent (N. Gast ÉNS student), and the trigonometric functions (sine, cosine and tangent). Besides the `crlibm` project is beginning to serve as a testbench for experimenting with novel techniques [40][52][53]. Included in the distribution is an extensive documentation with the proof of each function (currently more than 90 pages), as well as all the Maple scripts used to develop the functions. This makes this library an excellent tutorial on software elementary function development.

The `crlibm` library also includes a lightweight library for multiple precision, `scslib`—Software Carry Save Library. This library has been developed specifically to answer the needs of the `crlibm` project: precision up to a few hundred bits, portability, compatibility with IEEE floating-point, performance comparable to or better than GMP, small footprint. It uses a data-structure which allows to avoid carry propagations during multiple-precision multiplications. Supported operations are essentially addition/subtraction and multiplication, and conversions. This library is independent from `crlibm`.

The library has been downloaded more than 200 times. It is used in the LHC@home project of CERN (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,Linux / **Programming Language:** C / **URL:** http://www.ens-lyon.fr/LIP/Arenaire

5.2. Divgen: a divider circuit generator

Keywords: *ASIC, FPGA, circuit, division.*

Participants: R. Michard, A. Tisserand, N. Veyrat-Charvillon.

Divgen is a divider generator. It generates synthesizable VHDL descriptions of division units. Various algorithms, representations, radices, and parameters are supported. Both ASIC and FPGA targets are supported. This generator is developped within a collaboration between Inria and CEA-Léti (see §6.1).

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

5.3. FPLibrary: A library of operators for “real” arithmetic on FPGAs

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

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. The ultimate purpose is to allow the comparison of these number systems on a per-application basis [15][48]. These operators are parameterized in terms of precision and dynamic range, and are available in combinatorial and pipelined versions. Operators for format conversion are also provided. In 2004, the LNS operators were greatly improved to track the state of the art of the litterature.

The FPLibrary package has been used by several research teams from, among others, NASA, Los Alamos National Observatory and Honk-Kong University, although most of them only used the floating point operators.

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

5.4. A VHDL library for integer and modular arithmetic

Keywords: *FPGA, arithmetic operators, modular arithmetic.*

Participant: J.-L. Beuchat.

This library provides a collection of arithmetic operators described in synthesizable VHDL for RNS or cryptographic applications:

- adders based on carry-ripple addition or prefix networks;
- multi-operand adders;
- modulo M addition and multiplication-addition;
- modulo $(2^n - 1)$ addition and multiplication;
- modulo $(2n + 1)$ addition, subtraction, and multiplication.

Status: Beta release / **Target:** FPGA / **License:** GPL / **OS:** any / **Programming Language:** VHDL / **URL:** http://www.ens-lyon.fr/LIP/Arenaire

5.5. LinBox: High performance software for matrix computation

Keywords: *black box, exact arithmetic, finite field, generic library, integer, matrix computation, polynomial, rational number, sparse or structured matrix.*

Participants: P. Giorgi, G. Villard.

This software library is developped within an international initiative between Canada, United States, and France (see §8.3).

LinBox is a C++ template library for exact and high-performance linear algebra computation with sparse and structured matrices. Base domains for the matrix coefficients are finite fields, the rational numbers, and univariate polynomials. Implementing standard interfaces the library uses a plug-and-play methodology [68], offers connections to external softwares like MAPLE, and provide online servers for homology computing. LinBox 0.2.0 (June 2004) is available. In addition to the general evolution and polishing of the software, in 2004 we have augmented the functionalities with BLAS-based fast linear algebra routines over finite fields, and with a toolbox for linear system and Diophantine linear system solving (see §6.8 and [35][6]). LinBox is part of the Roxane project, cf. §8.1.

5.6. MPFI: Multiple Precision Floating-point Interval arithmetic

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

Participant: N. Revol.

MPFI is a C library specifically developed for interval arithmetic using arbitrary precision [37][25]. For efficiency and portability reasons, it is based on GMP and MPFR and the implementation takes advantage of these specific libraries. MPFI implements the arithmetic and algebraic operations and elementary functions described for instance in the mathematical library of the C99 standard, along with classical set operations on intervals. It is part of the Roxane project, cf. §8.1.

So far, MPFI has been used by scientists both from France and abroad (Belgium, Germany, Great Britain, USA, India, Colombia, ...).

Ongoing developments concern the stabilization and completion of the C++ interface, and the development of automatic differentiation.

Status: stable (alpha for the C++ interface) / **Target:** x86, PPC / **License:** GPL / **OS:** Unix, Windows (Cygwin) / **Programming Language:** C, C++ / **Dependencies:** GMP v4.0 or higher, MPFR v2.0.1 or higher / **URL:** http://www.ens-lyon.fr/LIP/Arenaire

5.7. Boost interval arithmetic library

Keywords: *generic C++ library, interval arithmetic, policy-based design.*

Participant: G. Melquiond.

In collaboration with H. Brönnimann (Polytechnic U. Brooklin, NY USA) and S. Pion (Géométrica team, Sophia Antipolis).

This library of the Boost project (http://www.boost.org) is a C++ library designed to efficiently handle mathematical intervals in a generic way. Our design is unique in that it uses policies to specify three independent variable behaviors: rounding, checking, comparisons. As a result, with the proper policies, this interval library is able to emulate almost any of the specialized libraries available for interval arithmetic, without any loss of performance nor sacrificing the ease of use. The version 1.32 has been released and the library is now considered fully operational.

The interval arithmetic library is an integral part of the Boost project. This project aims at providing free peer-reviewed C++ libraries and the last release has been downloaded more than 90,000 times.

Status: stable / **Target:** x86, PPC, Sparc / **License:** Boost Software License 1.0 / **OS:** Unix, Linux, Windows / **Programming Language:** C++ / **URL:** http://www.boost.org

5.8. MEPLib : Machine-efficient polynomials library

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

Participants: N. Brisebarre, C.-P. Jeannerod, J.-M. Muller, A. Tisserand.

This software library is developped 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

5.9. PFF: Formal Proofs about Floats

Keywords: *Coq, floating-point arithmetic, formal proof.*

Participants: S. Boldo, M. Daumas, G. Melquiond, L. Théry.

Our library of theorems and proofs about floating-point arithmetic is based on the library originated in the ARC AOC and distributed by L. Théry (http://www-sop.inria.fr/lemme/AOC/coq). The theorems are in the most possible general form. Most of them do not depend on the radix or on the rounding mode. We based our results on many lemmas from the literature. This allows any reader to understand and use our results without having to learn our formalism. S. Boldo keeps the library of proof scripts up-to-date. The last properties added are some of the ones described in §6.5, §6.4 and [5].

Three research teams use their developer's privilege to add their proofs and theorems to PFF. Other teams download the library anonymously or use a more stable version available as a Coq contribution.

Status: stable / **License:** LGPL / **Programming Language:** Coq / **URL:** http://lipforge.ens-lyon.fr/www/pff

5.10. Gappa: A tool for certifying numerical programs

Keywords: *certification, floating-point arithmetic, formal proof, round-off error.*

Participant: G. Melquiond.

Given a low-level description of a floating-point application on which arithmetic properties have to be certified, the tool generates a formal proof stating and verifying these properties from some hypotheses on the input variables. These hypotheses and properties encompass ranges of computed values and bounds on absolute or relative round-off errors.

The tool is developed in C++. It is still at a very early stage: formal proofs are generated for the Coq proof checker only, and only a few basic floating-point operations are handled at present.

Status: prototype not yet appropriate for distribution.

6. New Results

6.1. Hardware Arithmetic Operators

Keywords: *ASIC, FPGA, LNS, addition, arithmetic operators, cryptography, digital signature, division, floating-point, low-power consumption, modular arithmetic, multiplication.*

Participants: J.-L. Beuchat, J. Detrey, F. de Dinechin, R. Michard, J.-M. Muller, A. Tisserand, N. Veyrat-Charvillon, G. Villard.

6.1.1. Modular Arithmetic for FPGAs

Modular arithmetic plays a crucial role in various fields such as cryptography or residue number system arithmetic. Several researchers described algorithms allowing to build an adder or a multiplier according to the required modulus. There are also dedicated architectures for specific moduli such as (2^n-1) and (2^n+1).

J.-L. Beuchat and J.-M. Muller presented two variants of a modular multiplication algorithm originally due to Koç and Hung [69], that are especially suited for FPGA implementation, and that allow to compute $(XY + W)$ modulo M, where there is no need to know M at design-time [8].

J.-L. Beuchat described an improved modulo $(2^n + 1)$ addition algorithm suited to FPGA and ASIC implementations. Then, he proposed three implementations of a modulo $(2^n + 1)$ multiplication algorithm based on a paper by A. Wrzyszcz and D. Milford [74]. The first operator is based on an $n \times n$ multiplication and a subsequent modulo $(2^n + 1)$ correction, and takes advantage of the arithmetic logic embedded in Xilinx Spartan or Virtex FPGAs. The second operator computes a sum of modulo-reduced partial products by means of a multioperand modulo $(2^n + 1)$ adder. Finally, radix-4 modified Booth recoding reduces the number of partial products, while making their generation more complex [41].

6.1.2. Code-based Digital Signature

An algorithm producing cryptographic digital signatures less than 100 bits long with a security level matching nowadays standards has been recently proposed by Courtois, Finiasz, and Sendrier [63]. This scheme is based on error correcting codes and consists in generating a large number of instances of a decoding problem until one of them is solved (about $9! = 362880$ attempts are needed). A careful software implementation requires more than one minute on a 2GHz Pentium 4 for signing.

J.-L. Beuchat, N. Sendrier, A. Tisserand, and G. Villard proposed a first hardware architecture which allows to sign a document in 0.86 second on an XCV300E-7 FPGA, hence making the algorithm practical [42].

6.1.3. Hardware Function Evaluation

A. Tisserand and F. de Dinechin have completed a study of multipartite table methods initiated in 2001. Improvements include comparisons with other table-and-addition methods, a study of table compression, and an argument of near-optimality of the generalized multipartite method among first-order methods [28].

J. Detrey and F. de Dinechin have then developed a new table-based method for the hardware evaluation of arbitrary continuous functions. This method is based on a piecewise second-order minimax approximation, which is implemented using only one small multiplier and a multioperand adder. The second-order term is stored in a table indexed by as few input bits as possible. For the first-order term, a tradeoff has to be reached between using a multiplier and tabulating products. Analytical formulas for all the error terms have been derived, so that the navigation in the parameter space is easy. The method is thus very flexible and general. When implemented on FPGAs, the resulting operators are both smaller and faster (despite the multiplier) than all previous table-and-addition methods [34].

Then the same authors have tried to generalize these ideas to higher-order approximations. This leads to more complex operators with further improvements in area and size, with a diminishing return. For the FPGA implementation of the studied functions, operator delay begins to increase between degree 3 and degree 4 polynomial approximation, while area still decreases. Using these higher-order methods allows the practical implementation of 24-bit operators on a fraction of the area of current FPGAs [49].

6.1.4. Complex Square Root

M. Ercegovac (*University of California at Los Angeles*) and J.-M. Muller have suggested a hardware-oriented algorithm for evaluating complex square roots [36]. Their algorithm is derived from the real digit-recurrence iteration, and uses a prescaling technique for making the root digit selection simple. Complex square root appears in numerical computations such as complex Givens rotation, complex singular value decomposition, and in applications such as principal component analysis, quantum defect theory and wave propagation. M. Ercegovac and J.-M. Muller have received the *ASAP'2004 best paper award* for their contribution.

6.1.5. Hardware-oriented Algorithms for the Evaluation of Functions

A. Pineiro (Intel Barcelona), S. Oberman (NVIDIA Santa Clara), J.-M. Muller and J. Bruguera (Univ. Santiago de Compostela) have studied an architecture for evaluating some functions in single precision using quadratic approximations [23].

6.1.6. Division Circuits (collaboration Inria/CEA-Léti)

R. Michard, A. Tisserand and N. Veyrat-Charvillon have developed a software for the generation of division circuits (see §5.2). This software allows the comparison of various parameters (radix, algorithm type, optimizations...) for architecture exploration. This work has been done within a collaboration between Inria and CEA-Léti.

6.1.7. Multiplication Algorithms and Implementations for Asynchronous Circuits

N. Veyrat-Charvillon and A. Tisserand have designed and compared various multiplication schemes for asynchronous circuits. The corresponding results are published in the intership report [59].

6.1.8. Modular Multiplication Algorithms

A. Tisserand and L. Imbert (ATIPS, University of Calgary) have worked on new algorithms for modular multiplication. They have started an FPGA implementation.

6.1.9. Opérateurs arithmétiques sur circuits FPGA

J.-L. Beuchat and A. Tisserand wrote a chapter entitled *Opérateurs arithmétiques sur circuits* FPGA [9] in the book *Calcul et arithmétique des ordinateurs* [4].

6.1.10. Low-power Arithmetic Operators

A. Tisserand wrote a chapter [27] on the design of low-power arithmetic operators in the book *Low Power Electronics Design*, CRC Press.

6.2. Correctly Rounded Elementary Functions

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

Participants: N. Brisebarre, C. Daramy-Loirat, F. de Dinechin, J.-M. Muller, N. Revol.

A former work on range reduction has been completed in [12]. To evaluate an elementary function $f(x)$ for any x, x is usually transformed into x^* such that it is known how to evaluate $g(x^*)$ and how to deduce $f(x)$ from $g(x^*)$. Range-reduction is the transformation that determines x^* from x. The proposed algorithm is fast for most cases and accurate over the full range. Furthermore, the statistical distribution of these cases has been determined.

F. de Dinechin, C. Daramy-Loirat and J.-M. Muller have continued the work on the `crlibm` library, with contributions by D. Defour, associate professor at Perpignan University, and three undergraduate students, M. Gallet (ÉNS-Lyon), N. Gast (ÉNS), and C. Lauter (TU München).

The specificity of this library is that it aims at providing a comprehensive proof of the correct rounding property for each function, along with the code. As of end 2004 the technical document which describes the methodology used, along with the proofs themselves, has more than 90 pages [58].

Each function evaluation consists of two distinct steps: a first step accurate to 60-64 bits provides correct rounding most of the time. This is decided by a rounding test, which depends on a bound on the overall relative error of the first step, computed statically. If this test fails, a second, more accurate step is launched and always returns the correctly rounded result.

An extensive experimentation of this methodology on the logarithm function has shown that unexpectedly, this quest for correctness also allows to improve performance, by helping the management of performance tradeoffs involved in this two-step approach [40].

In the portable version of `crlibm`, the average time will be that of a standard elementary function, but the worst-case time is typically two orders of magnitude slower, which may still be considered prohibitively expensive for real-time applications. F. de Dinechin, C. Lauter and D. Defour developed a technique to reduce this worst-case time in the case when (non-portable) double-extended precision is available: using double-double-extended arithmetic provides 128 bits of precision. Lefèvre and Muller found many cases where more

accuracy is needed for correct rounding (up to 157 bits) but these cases can be shown to be degenerate so that they can be handled using double-double-extended arithmetic only [52].

Of course, when double-extended precision is available it should be used to speed up the first step as well. N. Gast and F. de Dinechin studied practical issues such as the implementation of the rounding test in this case, the possibility of sharing computations between both steps, and the implication it has on the precision-performance tradeoffs. Two representative functions (exponential and arctangent) were studied on two processors (Pentium and Itanium). The result is consistently that the average case is that of the best available faithful implementation, while the worst case time is within a factor two to 7 [53].

This study lifts the last technical obstacle to a standardization of correct rounding for at least some elementary functions. D. Defour (Perpignan U.), G. Hanrot (Spaces team), V. Lefèvre (Spaces team), J.-M. Muller, N. Revol and P. Zimmermann (Spaces team) have published guidelines for the future possible specification of functions in floating-point arithmetic [14]. They are based on correct rounding for the implementation of elementary functions or at least on a guaranteed quality (i.e. bounded error and compliance with the mathematical properties of the computed function).

In a very different context, namely the context of arbitrary precision floating-point arithmetic, a study has been led on the error function erf(x). An efficient algorithm has been proposed to evaluate the error function in arbitrary precision and this algorithm returns the correctly rounded result [54].

6.3. Fast Floating-point Arithmetic for Integer Processors

Keywords: *DSP*, *VLIW*, *integer processor*, *single precision floating-point arithmetic*.

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

We have focused on the design and software implementation of very fast single precision floating-point operations for processors having integer units only. The processors studied so far are the ST200 VLIW processors from STMicroelectronics, whose main property is the ability to execute four instructions in parallel (two of which can be $16 \times 32 \rightarrow 32$ multiplications). We have proposed and implemented new fast algorithms for the five basic operations (addition, subtraction, multiplication, division and square root) on such processors; the resulting C library, which provides several levels of compliance to the IEEE 754 floating-point standard, is faster than the native ST200 library by 20% to 40% [29].

This work is joint work with STMicroelectronics' Compilation and Simulation Expertise Center (Grenoble, France) and is funded by the *Région Rhône-Alpes* within the "*Arithmétique flottante pour circuits DSP* " project.

6.4. Properties and Proofs on Floating-point Arithmetic

Keywords: *Coq*, *floating-point arithmetic*, *formal proof*, *fused multiply-and-add*.

Participants: S. Boldo, N. Brisebarre, M. Daumas, G. Melquiond, J.-M. Muller.

Some previous results that we had submitted to journals have finally been published [11][10][13]. All the formally proved results are available in the PFF library described in §5.9.

6.4.1. Formalization of Floating-point Numbers as Vectors of Bits

S. Boldo has enriched the Coq library with a formalization of floating-point numbers as vectors of bits. It means that we can now formally prove hardware-level algorithms [62] with the proof assistant and that hardware-level operations can interact with high-level properties. It also adds trust in our initial formalization as the translation from it to vector of bits is now possible [30].

6.4.2. Double-rounding

Double-rounding consists in a first rounding in an intermediate extended precision and then in a second rounding in the working precision [66]. The natural question is then of the accuracy and correctness of the final result. S. Boldo and G. Melquiond proved an efficient algorithm for the double rounding to give the

correct rounding to the nearest value assuming the first rounding is to odd. As this rounding is unusual and this property is surprising, we formally proved this property using the Coq automatic proof checker [43].

6.4.3. Functions Computable with a Fused Multiply-and-add Instruction

The fused multiply-and-add instruction (`fma`) that is available on some current processors such as the Power PC or the Itanium eases some calculations.

S. Boldo and J.-M. Muller have given examples of some floating-point functions (such as $\texttt{ulp}(x)$ or $\texttt{Nextafter}(x, y)$), or some useful tests, that are easily computable using a `fma`. Then, they have shown that, assuming the available arithmetic rounds to the nearest, the error of a `fma` instruction is exactly representable as the sum of two floating-point numbers. They have given an algorithm that computes that error [44].

N. Brisebarre and J.-M. Muller have shown that the `fma` instruction can sometimes be used for performing correctly-rounded multiplication by a constant C that is not exactly representable in floating-point arithmetic. They give methods for checking whether, for a given value of C and a given floating-point format, this algorithm returns a correctly rounded result for any x [45].

6.4.4. Taylor Models

Computing with a Taylor model amounts to determine a Taylor expansion of arbitrary order, often high, along with an interval which encloses both Lagrange remainder and roundoff errors due to previous computations. These models are implemented in the Cosy [70] software. Using the properties of the IEEE-754 floating-point arithmetic, proofs that the algorithms implemented in Cosy do indeed determine an enclosure of roundoff errors have been built and completed [24], the latter algorithms have been improved [57].

6.5. Intervals and Guaranteed Proofs to Bound Variables and Errors

Keywords: *Taylor model, floating-point arithmetic, formal proof, proof, property.*

Participants: S. Boldo, F. Cháves, M. Daumas, G. Melquiond, N. Revol.

6.5.1. Semi-Automatic Determination of Guaranteed Enclosures of a Result

Convergent linear recurrences can have divergent behaviour when interval arithmetic is used for simulation. Indeed, the condition for convergence are more stringent than when usual (exact or floating-point) arithmetic is employed. However, the convergence of such recurrences, using interval arithmetic, can be studied if only one step every k steps is simulated. The proof of this result and the determination of k have been presented in [38]. This result can be applied to linear infinite impulse response filters used in control theory, the value of k corresponds then to a better choice of the sampling time.

6.5.2. Generating Formally Certified Bounds on Values and Roundoff Errors

Some pen-and-paper proofs contain errors. In order to circumvent this situation, certifications by automatic proof checkers have already been used to detect or prevent errors in algorithms and implementations [60]. This work [33] aims at creating a tool (see §5.10) able to generate bounds on the values and the roundoff errors for programs relying on floating-point arithmetic. The tool is based on forward error analysis and interval arithmetic. The novelty of our tool is that it produces a formal proof of the bounds. Furthermore, this proof can be checked independently using an automatic proof checker such as Coq and using a complete model of floating point arithmetic. We can easily certify that simple numerical programs such as the ones usually found in real-time applications do not overflow and that roundoff errors are contained within limits we determine.

6.5.3. PVS-guaranteed Proofs using Interval Arithmetic

We have arranged with C. Muñoz (National Institute of Aerospace), a set of tools for mechanical reasoning using interval arithmetic in PVS proof assistant [72]. The tools implement two techniques for reducing variable dependency: interval subdivisions and Taylor expansions. Although the tools are designed for the proof assistant system PVS, expertise on PVS is not required. The ultimate goal of the tools is to provide guaranteed proofs of numerical properties with a minimal human-theorem prover interaction.

6.5.4. Formal Certification of Arithmetic Filters for Geometric Predicates

Floating-point arithmetic provides a fast but inexact way of computing geometric predicates. In order for these predicates to be exact, it is important to rule out all the numerical situations where floating-point computations could lead to wrong results [73]. Taking into account all the potential problems is a tedious and error-prone work to do by hand. In collaboration with S. Pion (Géométrica team), we have studied a floating-point implementation of the 2D orientation predicate, and we have put in evidence how a formal and partially automatized verification of this algorithm avoided many pitfalls [56]. The presented method is not limited to this particular filter though, it can easily be used to produce correct semi-static floating-point filters of other geometric predicates. These filters have been added to the latest release of the CGAL software http://www.cgal.org/.

6.6. Theory of Computer Arithmetic Algorithms

Keywords: *E-method*, *Newton-Raphson iteration*, *floating-point arithmetic*, *number systems*, *number theory*, *rational approximation*.

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

6.6.1. Analysis of Arithmetic Algorithms

P. Kornerup (Southern Danish University, Denmark) and J.-M. Muller have continued their 2003 work on the choice of seed values for some Newton-Raphson iterations [21]. They give formulas for finding the best possible seed values when computing $f(a) = a^{\frac{1}{p}}$ using the Newton-Raphson iteration in a given interval (the aim is to minimize the maximum possible distance between x_n and $f(a)$).

J.-L. Nicolas, X. Roblot (UCB Lyon) and J.-M. Muller have published their work of last year on the integer solutions to the equation $A^2 + B^2 = C^2 + C$, where A and B belong to the same binade [22]. This allows to bound the accuracy that is required in the intermediate calculations to round the function $f(x, y) = \sqrt{x^2 + y^2}$ (in the case $1 \leq x, y < 2$) in floating-point arithmetic.

N. Brisebarre and J.-M. Muller have shown transformations that allow to evaluate most rational functions using Ercegovac's E-method [64]. This might make the E-method an interesting solution for average-precision hardware implementation of regular enough functions [31].

6.6.2. Number Systems

P. Kornerup (Southern Danish University, Denmark) and J.-M. Muller have introduced the notion of "RN coding", studied the properties of these codings, and suggested applications and conversion algorithms [50]. An RN-coding (where "RN" stands for "Round to Nearest") is a radix-β signed-digit representation of numbers for which rounding to the nearest is always identical to truncation.

6.7. Efficient Polynomial Approximation

Keywords: *Chebyshev polynomials*, *automatic generation*, *floating-point arithmetic*, *linear programming*, *minimax approximation*, *polynomial approximation*, *polytopes*.

Participants: N. Brisebarre, J.-M. Muller, A. Tisserand.

Polynomial approximations are almost always used when implementing functions on a computing system. In most cases, the polynomial that best approximates (for a given distance and in a given interval) a function has coefficients that are not exactly representable with a finite number of bits. And yet, the polynomial approximations that are actually implemented do have coefficients that are represented with a finite - and sometimes small - number of bits: this is due to the finiteness of the floating-point representations (for software implementations), and to the need to have small, hence fast and/or inexpensive, multipliers (for hardware implementations). We then have to consider polynomial approximations that fit these constraints of form and size in bits. In [46], N. Brisebarre, J.-M. Muller and A. Tisserand provide a general and efficient method for finding the best polynomial approximation under such constraints. Moreover, this method also applies if some other constraints (such as requiring some coefficients to be equal to some predefined constants, or minimizing

relative error instead of absolute error) are required. The method described in [46] is currently implemented in the C library MEPLib (cf. §5.8).

In [32], N. Brisebarre, J.-M. Muller and A. Tisserand apply the results of [46] to get automatic generation of the best polynomial approximations dedicated to hardware implementation. The generated approximations yield high-speed and small hardware operators because of the presence of fixed strings of zeros in the binary representation of the coefficients. Our first results show that up to 47% smaller coefficients compared to standard minimax approximations for comparable accuracy.

6.8. Exact Linear Algebra, Algorithms and Software Components

Keywords: *Diophantine linear system, arithmetic and bit complexity, exact arithmetic, finite field, integer matrix, linear algebra, polynomial matrix, software library, sparse or structured matrix.*

Participants: P. Giorgi, C.-P. Jeannerod, N. Revol, G. Villard.

6.8.1. Efficient Software Components

This work is a twofold contribution to the LinBox software library (see §5.5). With FFPACK (Finite Field PACKage) we offer matrix routines that reflect theoretical reductions to matrix multiplication for small enough prime fields. Our highly optimized C++ routines allow to solve linear systems and to compute various matrix factorizations over finite fields with timings approaching BLAS (http://www.netlib.org/blas) performance. This is a joint work with J.-G. Dumas and C. Pernet (IMAG Grenoble) [35][6]. Our second contribution concerns linear system solution over rational numbers, including lifting, minimal denominator solutions and Diophantine solving [71]. We offer a toolbox providing various strategies and optimizations, and a high level interface to deal with different types of matrices. In particular, in the dense case we show how to efficiently use BLAS routines in an exact computation fashion [6]. D. Pritchard has been involved in this work during his MIT/Inria internship.

Interval arithmetic and algorithms specific to this arithmetic allow one to solve linear systems with guaranteed results. Preliminary to the study and development of linear system solving using arbitrary precision interval arithmetic, the required data structures have been added to MPFI (cf. §5.6), using LinBox (cf. §5.5), by N. Dessart during her graduate internship [55] and N. Revol. This is an example of interoperability of two components of Roxane, cf. §8.1.

6.8.2. Algorithmic Complexity

We study the interaction between matrix multiplication and other basic linear algebra problems over univariate polynomials. Few decades after the case of matrices over an abstract field [61], the relation to the polynomial matrix product has been established recently [65]. We solve the following problems over $\mathsf{K}[x]$, with about the same number of operations as required for the polynomial matrix problem plus the output size: *column (basis) reduction* (see also [7]); generic *matrix inversion* [17]; certified *rank* and small degree *nullspace basis* [51]. We have started to investigate how these techniques could be carried over to the integer case.

In [18] we establish the best known complexity estimate $n^{2.698}$ for computing the determinant, the adjoint, and the characteristic polynomial of an $n \times n$ matrix in terms of ring operations (without divisions).

We have worked on final versions of previously obtained results [16][19][20][26].

6.8.3. Lattice-based Memory Allocation

We pursue our collaboration with A. Darte (Compsys Inria team) and R. Schreiber (Hewlett Packard, Palo Alto, USA) on memory allocation. In [47][39] we have extended and further studied our mathematical framework based on integer critical lattices. The goal is new insights and strategies for solving the problem of memory reuse in the context of compilation of dedicated processors.

7. Contracts and Grants with Industry

7.1. Région Rhône-Alpes Grant

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

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

A joint project with ST-Microelectronics has started in September 2003 and is supported by the *Région Rhône-Alpes*. The goal is to design floating-point arithmetic algorithms (basic operations as well as elementary functions) suitable for an implementation on circuits that only have integer arithmetic units. The main issue here is to speed up computations by exploiting both the characteristics of the circuits (and especially, for a first design, those of the ST200 family processors) and possibilities of specialization due to applications.

8. Other Grants and Activities

8.1. National Initiatives

8.1.1. Ministry Grant ACI "Cryptology"

Keywords: *FPGA, encryption, hardware operator for cryptography*.

Participants: J.-L. Beuchat, A. Tisserand, G. Villard.

The OPAC—*OPérateurs Arithmétiques pour la Cryptographie*) project 2002-2005, is a collaboration with the team *Arithmétique Informatique* of the Lirmm laboratory and the GTA team of the University of Montpellier (see http://www.lirmm.fr/~bajard/ACI_CRYPTO). The goal is the development of hardware operators for cryptographic applications on FPGAs. The project focuses in particular on problems related to finite fields and elliptic curves.

8.1.2. Ministry Grant ACI "Security in computer science"

Keywords: *FPGA, arithmetic operator, digital signature*.

Participants: J.-L. Beuchat, A. Tisserand, G. Villard.

The Ministry Grant ACI "Security in computer science" funds the OCAM—*Opérateurs Cryptographiques et Arithmétique Matérielle*— project 2003-2006 in collaboration with the Codes team (Inria Rocquencourt) and the team *Arithmétique Informatique* of the Lirmm laboratory at Montpellier (see http://www-rocq.inria.fr/codes/OCAM). The goal of OCAM is the development of hardware operators for cryptographic applications based on the algebraic theory of codes. The FPGA implementation of a new digital signature algorithm is used as a first target application [42].

8.1.3. Ministry Grant ACI "New interfaces of mathematics"

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

Participants: N. Brisebarre, C.-P. Jeannerod, J.-M. Muller, A. Tisserand.

The GAAP—*étude et outils pour la Génération Automatique d'Approximants Polynomiaux efficaces en machine*—project, started in 2004, is a collaboration with the LArAl laboratory of the University of Saint-Étienne. 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 applications targeted are software and hardware implementations, such as embedded systems for instance.

8.1.4. CNRS Grant "Numerical validation for embedded computations"

Keywords: *embedded computation, restricted computing precision, safety*.

Participants: M. Daumas, G. Melquiond, J.-M. Muller, N. Revol.

This CNRS New Investigation Initiative (AS Stic), 2003-2004, aims at listing the numerical difficulties encountered with embedded computing. Existing methods to study and validate the numerical quality of such computations will also be listed and assessed. This action involves teams from Lip6 (Université Pierre et Marie Curie - Paris 6), Mano (Université de Perpignan), List (CEA Saclay), Lasti (Énssat Lannion), LE2I (Université de Bourgogne) and Lip (ÉNS Lyon).

8.1.5. Working group on "Set methods for control theory", CNRS GDR MACS

Keywords: *control theory*, *set computing*.

Participant: N. Revol.

This working group 2003-2005 focuses on the topic of set computing with applications to control theory. The goal of this group is to stimulate exchanges between researchers in computer science and researchers in control theory. It is part of the CNRS GDR MACS (Modélisation, Analyse et Conduite des Systèmes dynamiques). It is headed by S. Lesecq (Lag, INPG Grenoble) and N. Revol.

8.1.6. Roxane Initiative

Keywords: *algebraic computation*, *efficiency*, *numerical computation*, *open-software*, *reliability*.

Participants: P. Giorgi, N. Revol, G. Villard.

Roxane stands for *Reliable Open Software-Components for Algebraic and Numeric Efficiency*. The goal of this project is to mutualize the efforts of implementation that are done in different groups (in France for the time being). Roxane integrates, in a homogeneous environment, tools to build dedicated and efficient components for solving real problems, mainly in computer algebra. These tools can interoperate, or will be able to do so in a near future. The promotion of Roxane is done via the site http://www-sop.inria.fr/galaad/logiciels/roxane, and via schools, software distribution CDs etc.

8.2. European Initiatives

8.2.1. Mathlogaps Marie Curie Early Stage Training

Keywords: *Mathematical logic*, *PVS*, *applications*, *formal proof*, *interval arithmetic*.

Participants: M. Daumas, F. Cháves.

Mathlogaps is a multi-participant effort to offer Early Stage Research Training in Logic and Applications with three partners: (1) the Universities of Leeds and Manchester; (2) *Université Claude Bernard Lyon 1* and the *École Normale Supérieure at Lyon*; (3) Ludwig Maximilians Universität München. It is led by D. Macperson (Leeds) and our local leader is P. Koiran (Lip).

One PhD was started in Arenaire project in November 2004. F. Cháves will develop the use and certification of interval arithmetic with PVS automatic proof checker (see the related result in §6.5). M. Hofmann acts as a distant expert and a future host in Munich for this PhD.

8.3. International Initiatives

8.3.1. LinBox Initiative

Keywords: *exact arithmetic*, *finite field*, *generic software library*, *matrix computation*, *rational number*, *sparse or structured matrix*.

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

LinBox is an ongoing collaborative research project for efficient algorithms and a software library in exact linear algebra (see §5.5 and §6.8). About thirty researchers from nine institutions in Canada, the USA and France are participating—http://www.linalg.org.

8.3.2. Grant of the Japanese Society for the Promotion of Sciences

Keywords: *automatic differentiation*.

Participant: N. Revol.

N. Revol obtained a grant of the Japanese Society for the Promotion of Sciences for a short stay in Japan, to collaborate with Prof. K. Kubota, Chuo Univ., Tokyo, on automatic differentiation.

8.3.3. Certifications of properties of floating-point arithmetic (CNRS-NASA)

Keywords: *Coq, PVS, floating-point, formal method, interval arithmetic.*

Participants: S. Boldo, M. Daumas, G. Melquiond.

CNRS PICS 2533 on "certifications of properties and uses of floating-point arithmetic" supports our collaboration with the National Institute of Aeropsace in Hampton, Virginia. It also involves the *École Polytechnique* (G. Dowek) and the University of California at Berkeley (W. Kahan). French funding is matched on a mission basis by a Research Cooperative Agreement awarded by NASA Langley Research Center to NIA.

Funding started in Fall 2004 with the visit of Professor Kahan (1989 ACM Turing Award) in Arenaire project. He animated a series of workshops.

8.3.4. Contributions to standardization bodies (IEEE 754)

Keywords: *Coq, floating-point, formal method.*

Participants: S. Boldo, M. Daumas, G. Melquiond.

The Department of Development and Industrial Relations (DirDRI) of the INRIA has supported our participation to the ongoing revision of the IEEE Standard for Binary Floating-Point Arithmetic (ANSI-IEEE 754). We have managed many opportunities to raise the impact of results from our project and from the Spaces project.

9. Dissemination

9.1. Conferences, edition

- M. Daumas is co-program chair of the French symposium on computer architecture (SympA) to be held in 2005.
- M. Daumas and J.-M. Muller are members of the Steering Committee of RNC (Real Numbers and Computers).
- M. Daumas and N. Revol are guest editors of a special issue of *Theoretical Computer Science* on Real Numbers and Computers, that will appear in 2005.
- F. de Dinechin was a member of the Program Committee for the << 2004 IEEE International Conference on Field-Programmable Technology >> (FPT'04).
- C.-P. Jeannerod was a member of the Poster Committee for the << 2004 International Symposium on Symbolic and Algebraic Computation >> (ISSAC'04). He is in charge of the tutorials at ISSAC'05 and, with A. Enge (Inria, Lix) and A. Sedoglavic (UST Lille, LIFL), of the next *Journées Nationales de Calcul Formel* (Luminy, November 21-25, 2005).
- J.-M. Muller is member of the Steering Committee of the IEEE Symposium on Computer Arithmetic (ARITH). He is a member of the Program Committee of ARITH17, ASAP'2004 (15th IEEE International Conference on Application-specific Systems, Architectures and Processors).
- N. Revol co-organized the 2004 edition of the forum of young mathematician and computer scientist women on mathematics, computer science and life science in January 2004, Paris. She co-organizes a seminar in Dagstuhl "Reliable Implementation of Real Number Algorithms: Theory and Practice" (January 2006).

- G. Villard is member of the Steering Committee of the << International Symposium on Symbolic and Algebraic Computation (ISSAC) >> (2003-2005). He was member of the Program Committee of ISSAC'04, Santander, Spain, Jul. 2004, and of CASC'04 (Computer Algebra in Scientific Computing), Saint Petersburg, Russia, July 2004.

General public meetings:

- S. Boldo, M. Daumas, C.-P. Jeannerod, A. Tisserand, and N. Revol have been involved in the *2004 Science Festival* in Vaulx-en-Velin (October 13-16, 2004).
- S. Boldo and M. Daumas gave an interview for the "Télévision Lyon Métropole" local TV channel.
- N. Brisebarre, C.-P. Jeannerod, C. Loirat, N. Revol, and A. Tisserand participated to the animation of *8ième Mondial des Métiers* in Lyon (January 29-30, 2004).
- N. Revol gave tutorial presentations on computer arithmetic and computational complexity for high-school students and for a wider audience after the screening of a movie on neuro-sciences in Die (Drôme), she visited high-schools in Buis-les-Baronnies (Drôme), Charlieu (Loire), Gex (Ain) and forums in Lyon (Mondial des Métiers), Dijon (Academic Forum on Young Women Curricula for Technological and Scientific Professions) and Paris (Cité des Sciences) for making teenagers sensitive to scientific careers. Her professional portrait appears in a document dedicated to high school students. She discussed these initiatives in Grenoble (Le Goût des Sciences) and she presented faculty and research scientist careers to PhD students at Valorithèse. N. Revol and N. Portier (Lip, ÉNS Lyon) gave an interview for the "Télévision Lyon Métropole" local TV channel.

9.2. Doctoral School Teaching

- F. de Dinechin gives a 30h ÉNSL Master course "Hardware Arithmetic Operators" (2004 / 2005).
- N. Revol organizes a course of the Doctoral School MATHIF, "Applications of Computer Science to Research and Technological Development".
- A. Tisserand gives a 30h ÉNSL Master course "Digital Integrated Circuits" (2004 / 2005).
- G. Villard is the head of the ÉNSL Master2 *Informatique Fondamentale*.

9.3. Other teaching and Service

- N. Brisebarre, C.-P. Jeannerod and G. Villard give a 30h Master course "Algorithms for Computer Algebra and Applications" at *Université Claude Bernard - Lyon 1* (2004 / 2005).
- F. de Dinechin teaches *Computer Architecture* and *Computer Science for Non-Computer Scientists* in Licence, ÉNSL.
- M. Daumas, C.-P. Jeannerod and N. Revol have been examiner for the ÉNS admissions.
- C.-P. Jeannerod has been in charge of the Lip bimonthly seminar from October 2003 to June 2004.
- S. Boldo, N. Boullis, J. Detrey, G. Melquiond and N. Veyrat-Charvillon are teaching assistants—*moniteurs*—they give courses at the ÉNS and INSA.

9.4. Leadership within scientific community

- M. Daumas is a member of the board of the CNRS Nationwide Initiative GDR ARP.
- J.-M. Muller is head of the Lip laboratory (joint laboratory - UMR - of CNRS, *École Normale Supérieure de Lyon*, Inria and Lyon 1 University, about 90 persons).
- N. Revol and S. Lesecq (Lag, INPG Grenoble) are heads of a CNRS working group on "Set methods for control theory", which is part of the GDR MACS—*Modélisation, Analyse et Conduite des Systèmes dynamiques*.
- A. Tisserand installs and maintains the computers and softwares of CAD tools for the Lip laboratory.

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.
- J.-M. Muller was in the Ph. D. Advisory Committee of P. Guigue (U. Nice, December 2003) and in the *Habilitation* boards of examiners of L. Grandvilliers (U. Nantes, June 2004) and J.-L. Lamotte (U. Paris 6, november 2004).
- G. Villard was in the Ph. D. Advisory Committee of M. Finiasz (École Polytechnique, October 2004).

9.6. Seminars, conference and workshop committees, invited conferences

The team members regularly give talks at the Department Seminar and at other French Institutions Seminars (Amiens, Grenoble, Lyon, Nancy, Perpignan, Orsay, Paris 6, Saint-Étienne).

National meeting:

- S. Boldo spoke during the colloquium *Calcul formel, algorithmes certifiés, preuves constructives* of the MAP group, Luminy, January 2004.
- N. Revol organized a session on Computer Arithmetic at the School for Young Researchers on Algorithms and Computer Algebra in Grenoble, April 2004. During this school A. Tisserand and N. Revol gave a lecture and P. Giorgi gave a research talk.
- G. Villard spoke during the Days on *Sensibilisation et formation aux outils de calculs, aux ressources numériques en ligne et aux nouvelles technologies*, Université de Montpellier II, June 15-24 2004. He is invited to give a talk a the "33rd Theoretical Computer Science Spring School, Computational Complexity", Montagnac-les-truffes, France, June 2005.

International:

- J.-L. Beuchat was an invited speaker at the "CryptArchi 2004 Workshop", Labussière-Sur-Ouche, June 2004. He has been invited for a series of talks on cryptographic algorithms at the *Université du Québec à Chicoutimi*, November 2004. He was invited to give a talk at id Quantique (http://www.idquantique.com), November 2004.
- S. Boldo gave a talk at the "'Hewlett Packard math library status meeting" in Cupertino (California), in teleconference with Richardson (Texas).

- M. Daumas was invited for two weeks at the National Institute of Aerospace, Virginia, in Aug 2004. He also gave a talk during the seminar on “Air Trafic Control” held by Thalès Chair of Complex Industrial Systems in the École Polytechnique in November 2004.
- F. de Dinechin gave an invited talk at the Intel Nizhniy Novgorod Lab (Russia) in July 2004.
- C.-P. Jeannerod gave an invited talk at the Mathematics Department Seminar of Kingston University, Kingston, U.K., March 2004.
- N. Revol is an invited speaker at the 76th Annual Meeting of GAMM (Gesellschaft für Angewandte Mathematik und Mechanik) in Luxemburg, April 2005.
- G. Villard was an invited speaker at the conference “Mathematics of Computer Algebra and Analysis”, Waterloo, Ontario, Canada, May 2004. He has been asked to give a talk at the “Workshop on Real Number Complexity, Foundations of Computational Mathematics FoCM’05, Santander, Spain, July 2005; and at the workshop “Challenges in Linear and Polynomial Algebra in Symbolic Computation Software”, Banff, Canada, October 2005.

10. Bibliography

Major publications by the team in recent years

[1] M. DAUMAS, J.-M. MULLER (editors). *Qualité des calculs sur ordinateur : vers des arithmétiques plus fiables*, Masson, 1997, http://perso.ens-lyon.fr/jean-michel.muller/livre_masson.html.

[2] M. DAUMAS, L. RIDEAU, L. THÉRY. *A generic library of floating-point numbers and its application to exact computing*, in "14th International Conference on Theorem Proving in Higher Order Logics, Edinburgh, Scotland", 2001, p. 169-184, http://perso.ens-lyon.fr/marc.daumas/SoftArith/DauRidThe01.ps.

[3] J.-M. MULLER. *Elementary functions, algorithms and implementation*, Birkhauser, 1997, http://perso.ens-lyon.fr/jean-michel.muller/book_functions.html.

Books and Monographs

[4] J.-C. BAJARD, J.-M. MULLER (editors). *Calcul et Arithmétique des Ordinateurs*, IC 2, Hermès Science Publishing, 2004.

Doctoral dissertations and Habilitation theses

[5] S. BOLDO. *Preuves formelles en arithmétiques à virgule flottante*, Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, November 2004.

[6] P. GIORGI. *Algorithmique et arithmétique pour l’algèbre linéaire exacte à partir de la bibliothèque LinBox*, Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, December 2004.

Articles in referred journals and book chapters

[7] B. BECKERMANN, G. LABAHN, G. VILLARD. *Normal forms for general polynomial matrices*, in "Journal of Symbolic Computation", to appear.

[8] J.-L. BEUCHAT, J.-M. MULLER. *Modulo M Multiplication-Addition: Algorithms and FPGA Implementation*, in "Electronics Letters", vol. 40, n° 11, May 2004, p. 654–655.

[9] J.-L. BEUCHAT, A. TISSERAND. *Opérateurs arithmétiques sur circuits FPGA*, in "Calcul et arithmétique des ordinateurs", J.-C. BAJARD, J.-M. MULLER (editors)., Traité IC2, Lavoisier, 2004, p. 109–152.

[10] S. BOLDO, M. DAUMAS. *A simple test qualifying the accuracy of Horner's rule for polynomials*, in "Numerical Algorithms", vol. 37, n° 1, December 2004, p. 45-60.

[11] S. BOLDO, M. DAUMAS. *Properties of two's complement floating point notations*, in "International Journal on Software Tools for Technology Transfer", vol. 5, n° 2-3, 2004, p. 237-246, http://perso.ens-lyon.fr/marc.daumas/SoftArith/BolDau04a.pdf.

[12] N. BRISEBARRE, D. DEFOUR, P. KORNERUP, J.-M. MULLER, N. REVOL. *A new range reduction algorithm*, in "IEEE Transactions on Computers", to appear.

[13] N. BRISEBARRE, J.-M. MULLER, S. RAINA. *Accelerating Correctly Rounded Floating-Point Division when the Divisor is Known in Advance*, in "IEEE Transactions on Computers", vol. 53, n° 8, August 2004, p. 1069–1072.

[14] D. DEFOUR, G. HANROT, V. LEFÈVRE, J.-M. MULLER, N. REVOL, P. ZIMMERMANN. *Proposal for a Standardization of Mathematical Function Implementation in Floating-Point Arithmetic*, in "Numerical Algorithms", vol. 37, n° 1-4, 2004, p. 367–375.

[15] J. DETREY, F. DE DINECHIN. *Outils pour une comparaison sans a priori entre arithmétique logarithmique et arithmétique flottante*, in "Technique et science informatiques", to appear.

[16] C.-P. JEANNEROD. *On matrix perturbations with minimal leading Jordan structure*, in "Journal of Computational and Applied Mathematics", vol. 162(1), 2004, p. 113–132.

[17] C.-P. JEANNEROD, G. VILLARD. *Essentially optimal computation of the inverse of generic polynomial matrices*, in "Journal of Complexity", to appear, http://dx.doi.org/10.1016/j.jco.2004.03.005.

[18] E. KALTOFEN, G. VILLARD. *On the complexity of computing determinants*, in "Computational Complexity", to appear.

[19] E. KALTOFEN, G. VILLARD. *Computing the sign or the value of the determinant of an integer matrix, a complexity survey*, in "J. Comp. Applied Math", vol. 162, n° 1, 2004, p. 133–146.

[20] P. KOIRAN, N. PORTIER, G. VILLARD. *A rank theorem for Vandermonde matrices*, in "Linear Algebra and its Applications", vol. 378, 2004, p. 99–107.

[21] P. KORNERUP, J.-M. MULLER. *Choosing Starting Values for Certain Newton-Raphson Iterations*, in "Theoretical Computer Science", to appear.

[22] J.-M. MULLER, J.-L. NICOLAS, X. ROBLOT. *Nombre de Solutions dans une Binade de l'Equation $a^2+b^2 = c^2+c$ (in French)*, in "l'Enseignement Mathématique", vol. 50, 2004, p. 147–182.

[23] J. A. PINEIRO, S. F. OBERMAN, J.-M. MULLER, J. D. BRUGUERA. *High-Speed Function Approximation using a Minimax Quadratic Interpolator*, in "IEEE Transactions on Computers", to appear.

[24] 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", to appear.

[25] N. REVOL, F. ROUILLIER. *Motivations for an Arbitrary Precision Interval Arithmetic and the MPFI Library*, in "Reliable Computing", vol. 11, 2005, p. 1–16.

[26] B. D. SAUNDERS, A. STORJOHANN, G. VILLARD. *Matrix rank certification*, in "Elect. J. Linear Algebra", vol. 11, 2004, p. 16–23.

[27] A. TISSERAND. *Low Power Electronics Design*, chap. Low-Power Arithmetic Operators, CRC Press, 2004.

[28] F. DE DINECHIN, A. TISSERAND. *Multipartite table methods*, in "IEEE Transactions on Computers", to appear.

Publications in Conferences and Workshops

[29] C. BERTIN, N. BRISEBARRE, B. D. DE DINECHIN, C.-P. JEANNEROD, C. MONAT, J.-M. MULLER, S.-K. RAINA, A. TISSERAND. *A floating-point library for integer processors*, in "Proceedings of SPIE's 49th Annual Meeting, Denver, Colorado, U.S.A.", August 2004.

[30] S. BOLDO. *Bridging the gap between formal specification and bit-level floating-point arithmetic*, in "Proceedings of the 6th Conference on Real Numbers and Computers, Schloss Dagstuhl, Germany", November 2004, p. 22-36.

[31] N. BRISEBARRE, J.-M. MULLER. *Functions Approximable by E-Fractions*, in "Proc. 38th IEEE Conference on Signals, Systems and Computers", IEEE, November 2004.

[32] N. BRISEBARRE, J.-M. MULLER, A. TISSERAND. *Sparse-Coefficient Polynomial Approximations for Hardware Implementations*, in "Proc. 38th IEEE Conference on Signals, Systems and Computers", IEEE, November 2004.

[33] M. DAUMAS, G. MELQUIOND. *Generating formally certified bounds on values and round-off errors*, in "6th Conference on Real Numbers and Computers, Schloss Dagstuhl, Germany", November 2004, p. 55–70.

[34] J. DETREY, F. DE DINECHIN. *Second Order Function Approximation Using a Single Multiplication on FPGAs*, in "14th Intl Conference on Field-Programmable Logic and Applications, Antwerp, Belgium", Improved version of Inria Research report 5140 available, LNCS 3203, August 2004, p. 221-230, http://www.inria.fr/rrrt/rr-5140.html.

[35] J.-G. DUMAS, P. GIORGI, C. PERNET. *FFPACK: Finite Field Linear Algebra Package*, in "Proc. International Symposium on Symbolic and Algebraic Computation, Santander, Spain", ACM Press, July 2004, p. 119–126.

[36] M. D. ERCEGOVAC, J.-M. MULLER. *Complex Square Root with Operand Prescaling (best paper award)*, in "Proc. 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, Galveston, Texas", IEEE Computer Society Press, September 2004.

[37] M. GRIMMER, K. PETRAS, N. REVOL. *Multiple Precision Interval Packages: Comparing Different Approaches*, in "Lecture Notes in Computer Science", vol. 2991, 2004, p. 64–90, http://www.inria.fr/rrrt/rr-4841.html.

[38] N. REVOL. *Convergent linear recurrences (with scalar coefficients) with divergent interval simulations*, in "SCAN 2004 (11th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics), Fukuoka, Japan", 2004.

[39] G. VILLARD. *Lattice based memory allocation (invited talk)*, in "Mathematics of Computer Algebra and Analysis, Waterloo, Ontario, Canada", May 2004.

[40] F. DE DINECHIN, C. LOIRAT, J.-M. MULLER. *A proven correctly rounded logarithm in double-precision*, in "RNC6, Real Numbers and Computers, Schloss Dagstuhl, Germany", November 2004.

Internal Reports

[41] J.-L. BEUCHAT. *A Family of Modulo* $(2^n + 1)$ *Multipliers*, Research report, n° 5316, Institut National de Recherche en Informatique et en Automatique, September 2004.

[42] J.-L. BEUCHAT, N. SENDRIER, A. TISSERAND, G. VILLARD. *FPGA Implementation of a Recently Published Signature Scheme*, Research report, n° 5158, Institut National de Recherche en Informatique et en Automatique, March 2004, http://www.inria.fr/rrrt/rr-5158.html.

[43] S. BOLDO, G. MELQUIOND. *When double rounding is odd*, Research report, n° 2004–48, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, November 2004.

[44] S. BOLDO, J.-M. MULLER. *Some Functions Computable with a Fused Mac*, submitted to the 17th IEEE Symposium on Computer Arithmetic, Research Report, n° 5320, INRIA, October 2004, http://www.inria.fr/rrrt/rr-5320.html.

[45] N. BRISEBARRE, J.-M. MULLER. *Correctly rounded multiplication by arbitrary precision constants*, submitted to the 17th IEEE Symposium on Computer Arithmetic, Research Report, n° 5354, INRIA, November 2004, http://www.inria.fr/rrrt/rr-5354.html.

[46] N. BRISEBARRE, J.-M. MULLER, A. TISSERAND. *Computing machine-efficient polynomial approximations*, submitted, Technical report, 2004, http://perso.ens-lyon.fr/jean-michel.muller/publications.html.

[47] A. DARTE, R. SCHREIBER, G. VILLARD. *Lattice based memory allocation*, Research report, n° 2004-23, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364

Lyon Cedex 07, April 2004, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2004/RR2004-23.ps.gz.

[48] J. DETREY, F. DE DINECHIN. *A tool for unbiased comparison between logarithmic and floating-point arithmetic*, Research report, n° 2004–31, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, March 2004.

[49] J. DETREY, F. DE DINECHIN. *Table-based polynomials for fast hardware function evaluation*, Submitted to Arith'17., Technical report, n° RR2004-52, LIP, November 2004.

[50] P. KORNERUP, J.-M. MULLER. *RN-Codings of Numbers: definition and some properties*, Technical report, n° 2004-44, LIP, ENS-Lyon, 2004.

[51] A. STORJOHANN, G. VILLARD. *Computing the rank and a small nullspace basis of a polynomial matrix*, Research report, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, 2004.

[52] F. DE DINECHIN, D. DEFOUR, C. LAUTER. *Fast correct rounding of elementary functions in double precision using double-extended arithmetic*, Submitted to TOMS., Research report, n° 5137, INRIA, March 2004, http://www.inria.fr/rrrt/rr-5137.html.

[53] F. DE DINECHIN, N. GAST. *Towards the post-ultimate `libm`*, Submitted to Arith'17, Research report, n° 5367, INRIA, November 2004, http://www.inria.fr/rrrt/rr-5367.html.

Miscellaneous

[54] S. CHEVILLARD, N. REVOL. *Computation of the error functions erf and erfc in arbitrary precision with correct rounding*, 2004, Submitted to the 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Paris, July 2005.

[55] N. DESSART. *Arithmétique par intervalles, résolution de systèmes linéaires et précision*, Mémoire de DEA, École Normale Supérieure de Lyon, Lyon, France, 2004, http://www.ens-lyon.fr/LIP/Pub/Rapports/DEA/DEA2004/DEA2004-04.pdf.

[56] G. MELQUIOND, S. PION. *Formal certification of arithmetic filters for geometric predicates*, 2004, Submitted to the 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Paris, July 2005.

[57] N. REVOL. *Bounding roundoff errors in Taylor models arithmetic*, 2004, Submitted to the 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Paris, July 2005.

[58] THE ARÉNAIRE PROJECT. *CR-Libm, A library of correctly rounded elementary functions in double-precision*, 2004, http://lipforge.ens-lyon.fr/projects/crlibm/.

[59] N. VEYRAT-CHARVILLON. *Algorithmes de multiplication pour circuits asynchrones*, Mémoire de DEA, École Normale Supérieure de Lyon, Lyon, France, 2004, http://www.ens-lyon.fr/LIP/Pub/Rapports/DEA/DEA2004/DEA2004-02.pdf.

Bibliography in notes

[60] W. R. ADRION, M. A. BRANSTAD, J. C. CHERNIAVSKY. *Validation, verification and testing of computer software*, in "ACM Computing Surveys", vol. 14, n° 2, 1982, p. 159-192, http://www.acm.org/pubs/articles/journals/surveys/1982-14-2/p159-adrion/p159-adrion.pdf.

[61] P. BÜRGISSER, M. CLAUSEN, M. SHOKROLLAHI. *Algebraic Complexity Theory*, Volume 315, Grundlehren der mathematischen Wissenschaften, (Chapter 16), Springer-Verlag, 1997.

[62] J. T. COONEN. *Specification for a proposed standard for floating point arithmetic*, Memorandum, n° ERL M78/72, University of California, Berkeley, 1978.

[63] N. COURTOIS, M. FINIASZ, N. SENDRIER. *How to achieve a McEliece-based Digital Signature Scheme*, in "Advances in Cryptology – ASIACRYPT 2001", C. BOYD (editor)., Lecture Notes in Computer Science, n° 2248, Springer, 2001, p. 157–174.

[64] M. D. ERCEGOVAC. *A general hardware-oriented method for evaluation of functions and computations in a digital computer*, in "IEEE Transactions on Computers", vol. C-26, n° 7, July 1977, p. 667–680.

[65] P. GIORGI, C. JEANNEROD, G. VILLARD. *On the complexity of polynomial matrix computations*, in "Proc. Int. Symp. Symb. and Alg. Comput., Philadelphia, PE, USA", ACM Press, August 2003, p. 135–142.

[66] D. GOLDBERG. *What every computer scientist should know about floating point arithmetic*, in "ACM Computing Surveys", vol. 23, n° 1, 1991, p. 5-47.

[67] G. HUET, G. KAHN, C. PAULIN-MOHRING. *The Coq Proof Assistant: A Tutorial: Version 6.1*, Technical Report, n° 204, Inria, 1997, http://www.inria.fr/rrrt/rt-0204.html.

[68] E. KALTOFEN. *Challenges of symbolic computation: my favorite open problems*, in "J. Symbolic Computation", vol. 29, n° 6, 2000, p. 891–919.

[69] C. K. KOÇ, C. Y. HUNG. *Carry-save adders for computing the product AB modulo N*, in "Electronics Letters", vol. 26, n° 13, June 1990, p. 899–900.

[70] K. MAKINO, M. BERZ. *Higher order verified inclusions of multidimensional systems by Taylor models*, in "Nonlinear Analysis", vol. 47, 2001, p. 3503–3514.

[71] T. MULDERS, A. STORJOHANN. *Certified dense linear system solving*, in "J. Symb. Comput.", vol. 37, n° 4, 2004, p. 485–510.

[72] S. OWRE, J. M. RUSHBY, N. SHANKAR. *PVS: a prototype verification system*, in "11th International Conference on Automated Deduction, Saratoga, New-York", D. KAPUR (editor)., Springer-Verlag, 1992, p. 748-752, http://pvs.csl.sri.com/papers/cade92-pvs/cade92-pvs.ps.

[73] J. R. SHEWCHUK. *Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates*, in "Discrete and Computational Geometry", vol. 18, 1997, p. 305-363,

http://link.springer.de/link/service/journals/00454/papers97/18n3p305.pdf.

[74] A. WRZYSZCZ, D. MILFORD. *A New Modulo* $2^a + 1$ *Multiplier*, in "Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors", 1993, p. 614–617.