INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Arénaire

Computer Arithmetic

Rhône-Alpes

THEME 2B

Activity Report

2003

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 Project-Team
Jean-Michel Muller [CNRS Research Scientist DR]

Administrative Assistant
Sylvie Boyer [INRIA TR, 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, since 01/09/02]
Claude-Pierre Jeannerod [Research Scientist CR]
Nathalie Revol [Research Scientist CR]
Arnaud Tisserand [Research Scientist CR]

CNRS Scientists
Marc Daumas [Research Scientist CR]
Gilles Villard [Research Scientist CR]

Faculty Member ÉNS Lyon
Florent Dupont de Dinechin [*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 [*Allocataire-monitrice AC*, 3rd year]
Nicolas Boullis [*Allocataire-moniteur AC*, 3rd year]
David Defour [Defended 09/09/03, now ATER at the University of Perpignan]
Jérémie Detrey [ÉNS student, 1st year]
Pascal Giorgi [*Allocataire MENRT*, 3rd year]
Guillaume Melquiond [ÉNS student, 1st year]
Saurabh Kumar Raina [Grant from the *Région Rhône-Alpes*, 1st year]

2. Overall Objectives

Key words: *Computer arithmetic*, *integer computation*, *approximated computation*, *floating-point representation*, *elementary function*, *reliability of numerical software*, *multiple-precision arithmetic*, *interval arithmetic*, *computer algebra*, *finite field*, *linear algebra*, *VLSI circuit*, *FPGA circuit*, *low-power operator*.

The Arénaire project aims at elaborating and consolidating knowledge in the field of *Computer Arithmetic*. Reliability, accuracy, and speed 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 performances 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 performances are 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 [66]. 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. This is what we are starting to do, 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. In collaboration with the Spaces project, 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). 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 performance, unit 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 performances 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.

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 $\beta = 10$, but studying the system independently of the value of β makes it possible to better understand its operation. 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, during our collaboration with the Lemme and Spaces projects (ARC AOC), makes it possible to use a proof assistant such as Coq [66] 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. Manufacturers, such as Sun, Compaq, HP or INTEL, propose libraries with their compilers when their processors do not integrate these functions at the hardware level. Linux developers also propose several libraries. 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 the draft of the standard, it was considered that the correct rounding was impossible to obtain at a reasonable cost for elementary functions. It is one of our goals to provide environments where all the numerical primitives, including elementary functions, are completely specified.

Currently, two libraries offer elementary functions with correct rounding. A. Ziv (IBM) proposes only rounding to the nearest. MPFR offers the four IEEE-754 standard rounding modes (http://www.mpfr.org). These two libraries are comparatively much slower than the usual libraries, and that is the main obstacle to their generalized use.

During the evaluation of an elementary function, one must use a temporary precision higher than the precision used for the result in order to round correctly. The question is to determine which intermediate precision is sufficient so that the rounding of the approximation always coincides with the rounding of the exact result. If one does not have an answer to this question, the library must be able to use an arbitrary precision by increasing the precision until the answer is found. This is what the two aforementioned libraries do. On the contrary, since we know the worst-case needed precision, we can replace the arbitrary precision algorithms by an algorithm in fixed precision, certainly large but limited. That allows considerable optimizations and makes it possible to offer the correct rounding with an acceptable overcost. Indeed, some previous work in which we took part, has given us bounds on the intermediate precision for several algebraic functions such as the division, the square or higher order roots. We have also found the "worst cases" —*i.e.* the values asking for the greatest detail during intermediate calculations— for certain elementary double precision floating-point arithmetic functions such as the logarithm, the exponential or some trigonometric functions.

The design a library with correct rouding also requires the study of algorithms in large precision (but not arbitrary), 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 one 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 environements 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 §5.6 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-in external components in a plug-and-play fashion. The library is devoted to sparse or structured exact linear algebra and its applications. It 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

Key words: *arithmetic operator*, *hardware implementation*, *dedicated circuit*, *cypher*, *control*, *validation*, *proof*, *numerical software*.

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. On the other hand, it can also be applied to software programs, when numerical reliability issues arise: these issues can consist in improving the numerical stability of an algorithm, computing guaranteed results (either exact results or certified enclosures), or certifying numerical programs.

The two following directions are especially studied.

4.1.1. Dedicated arithmetic operators

An ongoing study in cryptography (cf. §8.1.2) focuses on the efficiency of public key protocols. It aims at designing very fast ciphering systems. Operations which can be efficiently performed will be identified and, as a follow-up, this should in turn guide the design choices for new ciphering algorithms (even if this last point lies outside our field of expertise).

4.1.2. Validation and proof of numerical pieces of software

Interval arithmetic is used in critical applications, to get guaranteed enclosures of the results. It can also be used a priori, for instance to bound the dynamics of a filter for digital signal processing (cf. §8.1.5), or a posteriori to check the quality of a computed result. However, the simple replacement of numerical data by interval ones in the code may not yield meaningful results: a thorough analysis of the program and an adequate use of interval arithmetic is often required.

Another approach consists in proving a piece of software, including its use of floating-point arithmetic. For the time being, only small programs can be proven, since a completely automatized approach is still out of reach. For instance, we are working with the National Institute of Aerospace and the NASA Langley Research Center (USA) on a 32 operation code producing 622 lines of proof script. We will also collaborate with the UCB Lyon on applications. The application must be critical in order to justify the need for validated proofs.

When validation with the techniques above is not possible, one may consider *exact arithmetic* instead. We offer various solution methods in this area, with an emphasis on symbolic linear algebra. The matrices we work on have entries which are either integers or polynomials: integer matrices are ubiquitous in algorithmic number theory and its applications to cryptography; polynomial matrices are the appropriate tool for the study of many problems in control theory.

5. Software

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

Participants: Catherine Daramy-Loirat, David Defour, Florent de Dinechin, Jean-Michel Muller.

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

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

The goal of Crlibm is to offer: all the elementary functions specified by the C99 standard (trigonometric and hyperbolic and their reciprocals, exponential and logarithms, etc.); correct rounding in double precision (see §6.5); the choice of the rounding mode as in the IEEE-754 standard; performance within a factor of two in average when compared to other libm which do not offer correct rounding; portability; a proof of the correct rounding property for every function. The language used is C. As of end of 2003, the exponential and logarithm functions are completed and proven.

The Crlibm library comes with an independent 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 (http://www.swox.com/gmp), small footprint. It uses a data-structure which allows to avoid carry propagations during multiple-precision multiplications. Supported operations are essentially addition/subtraction, multiplication, and conversions.

Crlibm and SCSLib are distributed under the GNU LGPL licence, and can be downloaded from the project homepage http://www.ens-lyon.fr/LIP/Arenaire.

5.2. Library of validated theorems on floating-point arithmetic

Participants: Sylvie Boldo, Marc Daumas.

Key words: *floating-point arithmetic*, *formal proof*, *Coq*.

Our library of floating-point properties 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 on the Internet at the address http://perso.ens-lyon.fr/sylvie.boldo/coq.

The last properties added are some of the ones described in §6.5, §6.6 and §6.7. Additional ones were key to the proof of the HP floating-point library for Intel's IA-64 by P. Markstein (HP).

5.3. Robust algorithm for air traffic conflict avoidance

Participants: Marc Daumas, Guillaume Melquiond.

Key words: *floating-point arithmetic*, *roundoff error*, *formal method*, *civil aviation*.

This program has been registered (IDDN.FR.001.430026.000.S.C.2003.000.30805) to the Agency for Program Protection. It implements the algorithm designed by Guillaume Melquiond during his master's thesis (see §6.8). Its distribution is not decided for the time being and will be done in collaboration with the National Institute of Aerospace and NASA Langley Research Center. In the meantime, interested people may contact the authors.

5.4. FPLibrary: operators for “real” arithmetic on FPGAs

Participants: Jérémie Detrey, Florent de Dinechin.

Key words: *arithmetic operator*, *floating-point arithmetic*, *LNS*, *FPGA*.

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 systems. 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.

This library is available under the GNU LGPL license and can be downloaded from the project homepage http://www.ens-lyon.fr/LIP/Arenaire.

5.5. Mpcheck: Testing the quality of elementary functions

This is a joint work with the project-team Spaces, INRIA Lorraine and Rocquencourt.
Participants: Nathalie Revol, Paul Zimmermann.

Key words: *elementary function*, *mathematical library*, *quality*, *rounding*.

In a first version of the software, we tested the direction of the roundings and the accuracy of the evaluation of elementary functions in double precision floating-point arithmetic. The following version also includes tests on other precisions, namely extended double and quadruple, and on other criteria on the quality of the evaluation: respect of the range of a function (such as $]-\pi/2, \pi/2[$ for atan), monotony and symmetries if any. This software is still under development and will be distributed as soon as possible.

5.6. LinBox: High performance software for matrix computation

This software library is developed within an international initiative between Canada, the USA and France (cf. §8.3).
Participants: Pascal Giorgi, Gilles Villard.

Key words: *generic library*, *matrix computation*, *black box*, *sparse or structured matrix*, *exact arithmetic*, *finite field*.

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 and the rational numbers. Implementing standard interfaces, the library uses a plug-and-play methodology [68] and offers connections to external softwares like Maple, GAP, and C (Saclib). In 2002/2003 the team worked on the first public distributions of the library that are available at http://www.linalg.org. The library imports or implements various arithmetics (for finite fields, unlimited precision integer computation or polynomials). With the central notion of black box [69], LinBox offers a wide spectrum of functionalities for sparse, structured, and dense matrices (direct and iterative methods). Online servers have been installed at the U. of Delaware and at Grenoble (LMC-IMAG), they provide linear algebra computations including matrix Smith form and its application to computing the full homology of simplicial complexes.

5.7. Multiple Precision Floating-point Interval (MPFI)

This software library has been improved and maintained in 2003 in collaboration with Rouillier (project-team Spaces, INRIA Lorraine and Rocquencourt).
Participant: Nathalie Revol.

Key words: *interval arithmetic*, *multiple precision interval*.

MPFI (*Multiple Precision Floating-point Interval arithmetic library*) is a C library for arbitrary precision interval arithmetic, based on MPFR [76]. MPFI provides arithmetic operations and elementary functions for interval arguments; it returns results as tight as possible. MPFI can be freely downloaded at http://perso.ens-lyon.fr/nathalie.revol/software.html. Recent improvements concern the number of available elementary functions and their quality, particularly the accuracy of the results (cf. §6.9).

6. New Results

6.1. Introduction

Following the presentation of our scientific foundations, we present our new results. In §6.2 we focus on hardware operators. Results about floating-point arithmetic are given in §6.3, §6.4, and §6.5. In §6.6 we study models and properties of floating-point computations. Sections 6.7, §6.8, and §6.9 are concerned with certified algorithms including formal proofs and interval arithmetic aspects. In §6.10 and §6.11, we present results in exact arithmetic.

6.2. Hardware arithmetic operators

Participants: Jean-Luc Beuchat, Nicolas Boullis, David Defour, Jérémie Detrey, Florent de Dinechin, Jean-Michel Muller, Arnaud Tisserand.

Key words: *arithmetic operators*, *addition*, *multiplication*, *division*, *modular arithmetic*, *cryptography*, *floating-point arithmetic*, *LNS*, *FPGA*, *VLSI*, *low-power consumption*.

6.2.1. Operators for "real" arithmetic on FPGAs

Many applications require "real" arithmetic, in the sense that the numbers manipulated have a dynamic range unfeasible using fixed-point formats. Standard digital signal processing, for example, relies on fixed-point computations, which is a problem for some algorithms like adaptive filtering. The floating-point format is a well-known solution, but alternatives exist, such as the logarithmic number system (LNS). Operators for these two formats have very different implementation costs. To allow a fair and accurate comparison of these two formats on a per-application basis, J. Detrey and F. de Dinechin have developed a library of parameterizable operators for floating-point and LNS [40] (cf. §5.4). The LNS operators are implemented using multipartite tables, and are smaller than previous published operators [39].

6.2.2. 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 suggested several improvements of an iterative algorithm for modular multiplication originally proposed by Jeong and Burleson [67]. They designed a new algorithm, making the implementation of modular exponentiation easier, and implemented it on Xilinx's Virtex-E FPGA [32][51].

J.-L. Beuchat, L. Imbert (LIRMM), and A. Tisserand wrote a VHDL code generator for several modulo m multiplication algorithms described in the literature. Then they used this tool to carry out a comparison of modular multipliers on FPGAs [31].

J.-L. Beuchat studied the implementation of modulo $2^n \pm 1$ adders and multipliers on FPGAs. In order to compare various architectures according to the width of the operands, he designed VHDL code generators [29][50].

6.2.3. Multiplication by constants

N. Boullis and A. Tisserand worked on the automatic generation of optimized operators for linear applications (multiplication of a vector by a constant matrix). Such operations are very useful in digital signal processing and multimedia applications, for instance with the Fourier transform, the DCT, or digital filters.

This work is based on extensions of the algorithm proposed by V. Lefèvre [71], which deals with the multiplication of one variable by several constants. They wrote a program that generates operators involving multiplication of a variable vector by a constant matrix [35].

This generator has been successfully used for the optimization of FIR filters for WCDMA communications in the case of a new collaboration with an IRISA team headed by Olivier Sentieys.

6.2.4. 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*, to appear in 2004, CRC Press.

6.2.5. Hardware implementation of block ciphers

J.-L. Beuchat proposed FPGA implementations of the IDEA [70] and RC6 [74] block ciphers [28][30] (some examples are available at http://perso.ens-lyon.fr/jean-luc.beuchat/). To our best knowledge, these dedicated processors offer the fastest encryption rates reported in the literature.

6.2.6. Hardware operators in Alpha/HandelC

Following a collaboration of F. de Dinechin with T. Risset (COMPSYS project, INRIA), J.M. Spivey and M. Manjunathaiah (Computing Laboratory, Oxford University) on the subject of high-level synthesis for FPGA using Alpha and HandelC, a paper published in the Forum on Design Languages 2002 has been selected as one of the best papers for inclusion in a book [18].

6.3. Division algorithms

Participants: Nicolas Brisebarre, Jean-Michel Muller, Saurabh Kumar Raina.

Key words: *floating-point arithmetic*, *division by software*, *division with fused-mac*, *complex division*, *compilation optimization*.

6.3.1. Floating-Point Division by a constant

N. Brisebarre, J.-M. Muller, and S. K. Raina developed in [15] several techniques to improve the computation of the division x/y when y is known before x. The proposed algorithms are oriented towards architectures with available fused-mac operators (for three operands x, y, and z the fused-mac operator implements $xy + z$ with a unique rounding error). The goal is to get exactly the same result with correct rounding as with the usual division. The algorithms from [15] reduce the latency to one multiplication and one fused-mac. This is achieved if an internal precision larger (one additional bit suffices) than the target precision is available, or if y satisfies some properties that can be easily checked at compile-time.

6.3.2. Complex Division

M. Ercegovac (UCLA) and J.-M. Muller have suggested a new hardware-oriented algorithm for complex division [41]. Complex division appears in numerous signal processing problems. Using the straightforward method leads to many potential troubles (intermediate overflows and poor accuracy). Some formulas are more stable than the straightforward one but are very costly in terms of computational delay. Our algorithm is derived from the real digit-recurrence iteration. We use a prescaling technique that makes the quotient digit selection fairly simple.

6.4. Function approximations

Participants: Nicolas Brisebarre, Jean-Michel Muller.

Key words: *polynomial approximation*, *floating-point*, *Chebyshev polynomial*, *Remes algorithm*, *polytope*, *norm*, *Newton method*.

6.4.1. Best polynomial approximation with bit size constrainted coefficients

When designing a dedicated circuit or a program in order to evaluate a function, we frequently use polynomial approximations. We first compute “ideal” coefficients and then we round these coefficients according to the size allowed by the operators used. This truncation is necessary because of the finiteness of the floating-point representations and the need of having small multipliers. We then have to consider polynomial approximations for which the degree-i coefficient has at most m_i fractional bits. N. Brisebarre and J.-M. Muller have proposed in [53] an efficient method that allows to obtain the best polynomial approximation under this constraint. The precision saved with respect to a simple truncation of the coefficients is often significant: J.-M. Muller showed in [46] that, for approximations by polynomials of degree 2, we can save approximately 3 bits.

6.4.2. Analysis of arithmetic algorithms

When using Newton-Raphson iterations for implementing division or square root, the first bits of the input operand are used as address bits for looking up in a table a seed value for the iterations. P. Kornerup (Odense University, Denmark) and J.-M. Muller have suggested a choice for the seed value that depends on the number of iterations [44].

The correct rounding of the function $f(a,b) = \sqrt{a^2+b^2}$ (in the case $1 \leq a,b < 2$) requires the knowledge of the smallest possible distance between $f(a,b)$ and the middle of two consecutive floating-point numbers when a and b are floating-point numbers. This reduces to looking for integer solutions to the equation $A^2+B^2 = C^2+C$ for which A and B belong to the same binade. J.-L. Nicolas, X. Roblot (UCB Lyon), and J.-M. Muller have solved this problem in [59].

6.5. Elementary functions

Participants: Sylvie Boldo, Catherine Daramy-Loirat, Marc Daumas, David Defour, Florent de Dinechin, Jean-Michel Muller, Nathalie Revol.

Key words: *elementary function*, *double precision*, *correct rounding*.

6.5.1. Correct rounding

Evaluating an elementary function with correct rounding requires in the worst case to compute an approximation with an accuracy much higher than that of the result. An efficient implementation tries to pay this price only when needed, and therefore proceeds in two steps:

- The first step is similar to an evaluation in a usual libm: it computes an approximation of the function with a precision only slightly higher than that of the result, and tries to round it. Its specificity is to be able to detect a misround, and to launch the second step in this case.
- The second step launches a computation with an accuracy that guaranties the correct rounding property. It is therefore much slower, but occurs sufficiently rarely to keep the average performance close to the performance of the first step. The accuracy required in the second step has been computed by the Arénaire project for several usual functions on selected intervals.

D. Defour and F. de Dinechin studied a format for representing multiple-precision numbers which is well suited to the second step (which typically consists of a polynomial evaluation in multiple precision). The format is a high-radix representation in which bits have been reserved to absorb the carries of the intermediate operations. This *Software Carry Save* structure wastes space but allows simpler code. More important, this code exposes more parallelism in the multiple-precision multiplication than a standard high-radix representation as used in GMP (http://www.swox.com/gmp). Modern superscalar processors are able to exploit this parallelism, leading to multiple-precision computations faster than GMP [37].

One of the goals of the Crlibm project is to provide a proof of the correct rounding property for each function under consideration. This means a proof of the algorithm used, but also a proof of its C implementation, provided the processor/compiler/OS combination supports the IEEE-754 and C99 standards. D. Defour wrote such a proof for the exponential [55].

Another issue is performance. A student of F. de Dinechin and D. Defour, C. Quirin Lauter, has worked at Intel in Nizhny Novgorod on an implementation of the correctly rounded exponential optimized for the Intel Itanium processor family. The main result is that the evaluation can be performed in one step only, thanks to the availability of double-extended multiply-accumulate units [58].

V. Lefèvre (INRIA research scientist at LORIA and former member of Arénaire) and J.-M. Muller have published [24] an algorithm for argument reduction which handles “online” data input in a most-significant-digit-first manner. This happens for example when a quotient or a square root is computed by the SRT algorithm.

S. Boldo and M. Daumas, in collaboration with R.-C. Li from University of Kentucky, also presented and proved in Coq a technique for argument reduction derived from the Cody and Waite algorithm, using a *fused-mac* similar to that of the IA-64 [45].

6.5.2. Standardization of mathematical function implementation in floating-point arithmetic

This is a joint work with the project-team Spaces, INRIA Lorraine and Rocquencourt.

As a consequence of our previous studies on correct rounding (Table Maker Dilemma), it is now realistic to propose a specification of the implementation of elementary functions in floating-point arithmetic. The main guidelines of such a specification are given in [17]. They are based on correct rounding for the evaluation of elementary functions or at least on a guaranteed quality (*i.e.* bounded error and compliance with the mathematical properties of the computed functions).

6.6. Models and properties of floating-point computations

Participants: Sylvie Boldo, Marc Daumas, Nathalie Revol.

Key words: *floating-point*, *roundoff error*, *formal proof*, *Coq*, *DSP*, *Taylor expansion*, *certified computation*.

6.6.1. Certified formal proof

> "I approve of what you're doing. I regard it as necessary. My hope is that continuing in that direction will ultimately lead to specifications and implementations which are not merely verifiable, but also maintainable."
>
> W. Kahan, ACM Turing Award.

Many properties are already available about the exact computation of the roundoff error for standard IEEE-754 operations (namely, addition, multiplication, division and square root). But all theses literature results assume that there is no *underflow*. S. Boldo and M. Daumas proved that this hypothesis is much too strong and gave a new necessary and sufficient condition for the roundoff error to be exactly representable. They also gave examples of surprising behaviors when those conditions were not fulfilled [33]. Some lacks of formal proofs have been recalled with future developments by S. Boldo, M. Daumas, and L. Théry [34].

Sterbenz's theorem is often used to prove algorithms with standard IEEE-754 arithmetic [16]. S. Boldo and M. Daumas proved that this theorem is still correct, when no underflow occurs, on the Texas Instrument's TMS 320 [14]. This is a DSP used in avionics and military applications that is not IEEE-754-compliant. They also checked many properties on the number representation used by the TMS 320.

6.6.2. Taylor models and floating-point arithmetic

N. Revol, in cooperation with M. Berz (U. Michigan, USA) and K. Makino (U. Illinois, USA), worked on Taylor models to certify computations. 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 [73][64] software. Properties of the IEEE-754 floating-point arithmetic have been extensively used to prove that the algorithms implemented in Cosy do indeed determine an enclosure of roundoff errors [60].

6.7. Faithful computation using Horner's rule

Participants: Sylvie Boldo, Marc Daumas.

Key words: *floating-point*, *round-off error*, *formal proof*, *Coq*.

The evaluation of a binomial $AX + Y$ seems to be the main operation of polynomial evaluation by Horner's rule. Moreover, some applications, such as approximations of elementary functions, compute this binomial knowing that AX is much smaller than Y.

S. Boldo and M. Daumas proved that, under given conditions, the floating-point evaluation of $AX + Y$ is faithful, without using the *fused-mac* operator available on some hardware. The evaluation may still be faithful if errors on approximations used for A and X are large [13]. A faithful result is either the rounded up or the rounded down value of the exact mathematical result; there is no way to force the rounding used *a priori* or even to know it *a posteriori*. This work leads to a simple condition on the polynomial that guarantees that the evaluation using Horner's rule is faithful.

6.8. Robust algorithm to avoid air traffic conflict

Participants: Marc Daumas, Guillaume Melquiond.

Key words: *floating-point*, *roundoff error*, *formal methods*, *civil aviation*, *ADS-B*, *NASA*, *NIA*.

We have characterized the behavior of algorithms based on the ADS-B protocol (conflict resolution protocol) and used by airplanes to maintain safety distances. Truncation errors caused by the Earth geometry, and roundoff errors caused by the implementation are both taken into account. This work is a collaboration with the National Institute of Aerospace, NIA (formerly ICASE), and the NASA Langley Research Center USA which have already verified some algorithms for the US Federal Aviation Administration. We have proposed a new algorithm [62], both simpler, and easier to verify.

6.9. Interval arithmetic with arbitrary precision

Participant: Nathalie Revol.

Key words: *interval arithmetic*, *arbitrary precision*, *validated computing*, *automatic adaptation of the computing precision*.

The work on the MPFI library for arbitrary precision interval arithmetic, in collaboration with F. Rouillier (Spaces, INRIA), went on with improvements of the library (cf. §5.7): every elementary function available in MPFR [76] has its counterpart in MPFI, and results are as tight as possible.

Linear systems have been studied [47], in particular the automatic adaptation of the computing precision in Hansen-Sengupta algorithm for interval linear system solving. Simulating and determining the dynamics of Infinite Impulse Response (IIR) linear filters used in digital signal processing, have been done in the framework of a CNRS *Action Spécifique* on the numerical validation of embedded computations (cf. §8.1.5).

6.10. Diophantine approximation

Participant: Nicolas Brisebarre.

Key words: *difference equation*, *exponential polynomial*, *exponential fraction*, *linear differential equation*, *modular function*, *modular invariant*, *Fourier coefficient*, *circle method*, *Hankel function*.

6.10.1. Difference equations

N. Brisebarre gave in [52] an algorithm that computes the entire solutions, i.e., holomorphic over all the complex plane, of systems of two difference equations and of systems made of one differential equation and one difference equation, all equations having complex polynomials coefficients. This algorithm, which uses some works by Abramov and Petkovšek, also allows to determine, for each of the considered systems, all the solutions of the form $R_1(z)e^{\delta_1 z} + \cdots + R_s(z)e^{\delta_s z}$, with $\delta_1, ..., \delta_s \in \mathbb{C}$ and $R_1(z), ..., R_s(z) \in \mathbb{C}(z)$.

6.10.2. Modular functions

The modular invariant j is an important function in number theory [75]. G. Philibert (LARAL, Saint-Étienne) and N. Brisebarre established in [54] precise upper and lower bounds for the Fourier coefficients of j^m for all $m \in \mathbb{N}^*$. These results improve on previously known results, especially those of Mahler [72] and Herrmann [65].

6.11. Exact linear algebra

Participants: Pascal Giorgi, Claude-Pierre Jeannerod, Gilles Villard.

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

In algebraic complexity over a field K, we may consider that basic linear problems can be solved in $n^{\omega+o(1)}$ operations where ω is the exponent of matrix multiplication. For polynomial and integer matrices it is not known in general whether the "equivalence" to matrix multiplication can be used. The difficulty and problem are to understand the impact of the data size (integer bit lengths or polynomial degrees) on the cost of solving the problems [11][49]. In collaboration with E. Kaltofen (North Carolina State University) we have studied the question in [21] for the particular case of the determinant. With the goal of a better understanding of algorithmic complexities in exact linear algebra, we have worked in the following directions.

6.11.1. Polynomial matrices

We have improved by a factor n the best known cost for inverting a generic $n \times n$ polynomial matrix of degree d. We propose an algorithm for computing the matrix inverse over $\mathsf{K}[x]$ with the essentially optimal complexity estimate $n^3 d \log^{O(1)}(nd)$ operations in K (this is roughly the order of the output size) [20]. We have established one of the first results on $\mathsf{K}[x]$, i.e., on a coefficient domain other than K, which identifies a correspondence between dense linear algebra problems (matrix greatest common divisor, determinant, and module basis reduction) and the associated matrix multiplication [43][1].

6.11.2. Computation without divisions and Krylov / Lanczos approach

Iterative methods of the Krylov / Lanczos type can be used in various exact computation situations. With E. Kaltofen we have extended our approach for computing the determinant without divisions. The solution we propose leads to the best known complexity estimate $O(n^{2.7})$ ring operations and is carried over to a new bit complexity estimate for the integer characteristic polynomial [57]. We also apply iterative methods to black box and especially sparse matrices, in [26] we propose a new rank certificate for randomized algorithms based on trace and orthogonalization.

6.11.3. Lattice-based memory allocation

We have started a collaboration with A. Darte (Compsys, LIP INRIA) and R. Schreiber (Hewlett Packard, Palo Alto, USA) on memory allocation. In [36] a general mathematical framework based on integer critical lattices is proposed. This leads to new insights and strategies for solving the problem of memory reuse in the context of compilation of dedicated processors.

6.11.4. LinBox software library

The role played by linear algebra relies both on theoretical progress and on the availability of high performance software libraries. LinBox is an ongoing project devoted to exact linear algebra and offers a generic software library that allows to plug-in external components in a plug-and-play fashion (cf. §5.6). New public distributions of the library have been realized in 2003. In addition to software development, the main objective in Lyon is the design and implementation of matrix interfaces. P. Giorgi has shown how to use external matrix multiplication routines. He has demonstrated that fast numerical libraries such as BLAS (http://www.netlib.org/blas) are very attractive and useful in an exact computation setting (matrix factorizations and block algorithms) [42].

7. Contracts and Grants with Industry

7.1. Région Rhône-Alpes Grant

Participants: Nicolas Brisebarre, Claude-Pierre Jeannerod, Jean-Michel Muller, Saurabh Kumar Raina, Arnaud Tisserand.

Key words: *emulation of floating-point*, *integer processor*, *DSP*.

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

[1]For his contribution to this paper, Pascal Giorgi has obtained the *Best Student Award* at the ISSAC'03 conference in Philadelphia, Aug. 2003.

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 "Computer arithmetic for FPGAs"

Participants: Jean-Luc Beuchat, Nicolas Boullis, Jérémie Detrey, Florent de Dinechin, Arnaud Tisserand.

Key words: *hardware arithmetic operator*, *CAD tool*, *FPGA*.

This *Action Concertée Incitative* Grant (*ACI Jeunes Chercheurs* 2000–2003) has funded in part our research on computer arithmetic for FPGAs. In 2003 the studied topics have been floating-point and LNS arithmetic, modular arithmetic, and multiple constant multiplication. In particular the Grant has funded licenses for FPGA CAD tools, as well as travels for J.-L. Beuchat, N. Boullis, J. Detrey, F. de Dinechin, and A. Tisserand.

8.1.2. Ministry Grant ACI "Cryptology"

Participants: Jean-Luc Beuchat, Arnaud Tisserand, Gilles Villard.

Key words: *hardware operator for cryptography*, *encryption*, *FPGA*.

The OPAC (*Opérateurs Arithmétiques pour la Cryptographie*) project, started in 2002, is a collaboration with the team *Arithmétique Informatique* of the LIRMM laboratory and the GTA team of the University of Montpellier. 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.

8.1.3. Ministry Grant ACI "Security in computer science"

Participants: Jean-Luc Beuchat, Arnaud Tisserand, Gilles Villard.

Key words: *digital signature*, *arithmetic operator*, *FPGA*.

The Ministry Grant ACI "Security in computer science" funds the OCAM (*Opérateurs Cryptographiques et Arithmétique Matérielle*) project (2003-2006) in collaboration with the Codes project (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 will be used as a first target application.

8.1.4. CNRS Grant "Computer arithmetic"

Participant: Arénaire project-team.

Key words: *computer arithmetic*, *floating-point arithmetic*, *fixed-point arithmetic*, *new floating-point operator*.

Headed by J.-M. Muller, this CNRS New Investigation Initiative "Computer arithmetic" (AS STIC), which started in October 2002, includes members of the following departments: LIP6 (Paris), LORIA (Nancy), LIAFA (Paris), LIRMM (Montpellier), MANO (Perpignan), and LASTI (ÉNSSAT Lannion). Its goal is to identify future research trends on the following topics:

- Hardware operators, algorithms, formal proofs, and computation accuracy for fixed-point arithmetic.
- New floating-point operators: hardware support for elementary function evaluation, "fused-mac" impact on algorithms and operators, etc.
- Industrial applications.

8.1.5. CNRS Grant "Numerical validation for embedded computations"

Participants: Marc Daumas, Guillaume Melquiond, Jean-Michel Muller, Nathalie Revol.

Key words: *embedded computation*, *safety*, *restricted computing precision*.

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).

8.2. European Initiatives

8.2.1. VASA: proposal for a Marie Curie Research Training Network

Participants: Marc Daumas, Jean-Michel Muller, Nathalie Revol.

Key words: *special function*, *arbitrary precision evaluation*, *quality*.

A proposal for a European project, named VASA for *Validated Algorithms for Special functions with Applications*, and linking 10 teams from 7 countries (Belgium, France, Germany, United Kingdom, Hungary, Poland and Spain), was submitted in November 2003. Its goal is to train young research scientists to the complete panel of knowledge and tools required to develop, implement, and apply algorithms for the evaluation of special functions in arbitrary precision with guaranteed error bounds.

8.3. International Initiatives

8.3.1. LinBox Initiative

Participants: Pascal Giorgi, Claude-Pierre Jeannerod, Gilles Villard.

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

LinBox is an ongoing collaborative research project for efficient algorithms and software library in exact linear algebra (cf. §5.6 and §6.11). About thirty researchers from nine institutions in Canada, the USA and France are participating. The project started in 1998 under a CNRS / NSF Grant (1998-2000). LinBox is funded (2002) by the NSF on the American side and by a CNRS JemStic Initiative on the French side.

9. Dissemination

9.1. Conferences, edition

- S. Boyer, N. Brisebarre, M. Daumas (Program Committee Chair), C.-P. Jeannerod, N. Revol, and G. Villard (Chair) have organized RNC'5, the "Fifth Conference on Real Numbers and Computers". Fifty participants from various countries came to Lyon in Sept. 2003.
- N. Brisebarre is involved in the organization of a day in honor of G. Philibert (Jan. 2004, U. St-Étienne). He participates to a working group of the ANRT National Initiative *Opération FutuRIS*.
- M. Daumas, D. Michelucci (U. Bourgogne), J-M. Moreau (UCB Lyon), and P. Langlois (U. Perpignan) organized the CNRS Thematic School "Algorithms and Computer Arithmetic" in March 2003 in Dijon.
- F. de Dinechin is member of the Editorial Board of *Technique et science informatiques*.

- Jean-Michel Muller was the ARITH (IEEE Symposium on Computer Arithmetic) Steering Committee President until June 2003 (two-year term). He has been Program Committee Member of ARITH-16, Santiago de Compostela, Spain, June 2003 and ASAP'2003, Den Hague, Netherlands, June 2003. He has been Guest Editor, with Peter Kornerup, Jean-Claude Bajard, and Christiane Frougny, of a special issue of Theoretical Computer Science [9] published in January 2003.
- N. Revol has been Guest Editor with S. El Hajji (U. Rabat) and P. Van Dooren (C.U. Louvain-La-Neuve) of a special issue of the Journal of Computational and Applied Mathematics (162(1), 2004) on linear algebra and computer arithmetic. With E. Hubert (INRIA, Sophia-Antipolis) and N. Portier (LIP, ÉNS Lyon), she organizes the 2004 edition of the forum of young women in mathematics (January 2004, Paris), the topic being *Mathematics, computer science, and life sciences*. She organizes a session on computer arithmetic at the school for young researchers on Algorithms and Symbolic Computations (Grenoble, 29 March-2 April 2004).
- A. Tisserand has co-organized the "Thematic School *Architectures des systèmes matériels enfouis et méthodes de conception associées*" in Roscoff, April 2003.
- G. Villard is member of the Steering Committee of the "International Symposium on Symbolic and Algebraic Computation (ISSAC)". He is member of the Program Committee of ISSAC'04, Santander, Spain, July 2004. He was Program Committee member for CASC'03 (Computer Algebra in Scientific Computing), Passau, Germany, September 2003. He has organized, with B.D. Saunders (U. Delaware), a session on symbolic linear algebra during the conference "Applications of Computer Algebra (ACA'03)", Raleigh, North Carolina, July 2003.

General public meetings:

- N. Revol gave a tutorial presentation on computational complexity during the Rhône-Alpes APMEP meeting in April 2003 (for high-school teachers); she visits high schools for making teenagers sensitive to scientific careers, she discussed of these initiatives during the "Meeting for More Women in Sciences", Paris, May 2003; she presented faculty and research scientist careers to Ph. D. students at "*Valorithèse*"; N. Revol has been invited to a discussion panel on women's parity in scientific professions during the "Meetings of the Jacques Cartier Center" (U. Lyon 2, December 2003), and to the "Academic Forum on Young Women Curricula for Technological and Scientific Professions" Dijon, February 2004.
- S. Boldo, F. de Dinechin, C.-P. Jeannerod, A. Tisserand, and N. Revol have been involved in the "2003 Science Festival" at the ÉNS Lyon.

9.2. Doctoral School Teaching

- N. Revol organizes a course of the Doctoral School MATHIF on "Applications of Computer Science to Research and Technological Development".
- Several members of the team teach 24h courses to graduate students (*DEA d'Informatique Fondamentale de l'ÉNSL*): F. de Dinechin, "Hardware arithmetic operators", 2003-2004; F. de Dinechin and A. Tisserand, "Hardware architectures design", 2002-2003; J.M. Muller, "Elementary functions" (jointly with the *DEA d'Informatique Fondamentale et d'Analyse Numérique* in 2003), 2002-2003; A. Tisserand, "Integrated Digital Circuits", 2003-2004; N. Revol et G. Villard, "Algorithms and arithmetics" (jointly with the *DEA d'Informatique Fondamentale et d'Analyse Numérique*), 2002–2003.

9.3. Other teaching and Service

- F. de Dinechin is a faculty member of the ÉNSL (*Maître de Conférences*) and teaches in the Computer Science and Modelling Master. He is the head of the European Program of the Master (MIM).

- S. Boldo and N. Boullis are teaching assistants (*moniteurs*) at the ÉNS and INSA Lyon.
- P. Giorgi teaches at the UCB Lyon.
- M. Daumas has been examiner for the ÉNS admissions.
- A. Tisserand manages machines and software tools for VLSI CAD and FPGA for the Arénaire and Compsys teams of the Department.
- The team members have been advisors for graduate student interns of the *Magistère* MMFAI (Paris), the *École Polytechnique* (Palaiseau), the DESS *Ingéniérie Mathématique* of Lyon, and the ÉNSL.

9.4. Leadership within scientific community

- M. Daumas is a member of the board of the CNRS National Initiative GDR ARP. With S. Boldo, N. Boullis, and P. Giorgi, he has organized the "Computer Arithmetic Days (AriNews)" in Dijon (March 2003) and Lyon (November 2003) within the CNRS National Initiatives GDR ARP and ALP.
- J.-M. Muller is the head of the LIP Department (UMR CNRS-ÉNS Lyon-INRIA-UCB Lyon 5668, about 90 persons). He is the head of the CNRS New Investigation Grant (AS STIC) on Computer Arithmetic.
- G. Villard, with J.-A. Weil (INRIA Sophia-Antipolis), has been the head of the CNRS New Investigation Grant (AS STIC) on Computer Algebra.

9.5. Committees

- *Hiring Committees.* N. Brisebarre, Math., U. J. Monnet Saint-Étienne. M. Daumas, Comp. Sc., ÉNS Lyon. F. de Dinechin, Comp. Sc., ÉNS Lyon. J.-M. Muller, Comp. Sc., ÉNS Lyon and U. Provence. N. Revol, App. Math., UJF Grenoble and U. Sc. Tech. Lille. A. Tisserand, Comp. Sc., UJF Grenoble. G. Villard, App. Math., U. Sc. Tech. Lille and Comp. Sc., U. Perpignan.
- J.M. Muller was in the Ph. D. Advisory Committee as *rapporteur* for the work of R. Chotin (Paris 6, June 2003), A. Pineiro (Univ. Santiago Compostella, June 2003), and P. Guigue (Univ. Nice, December 2003); he was member of the Ph. D. Advisory Committee of N. Magaud (Univ. Nice, October 2003) and of the *Habilitation* Committee of Gilles Villard (UCB Lyon, March 2003). In 2003 he was member of the committee for the *Région Rhône-Alpes* Ph. D. grants.
- G. Villard was in the Ph. D. Advisory Committee of E. Thomé (LIX, École Polytechnique, May 2003), and of A. Bostan as *rapporteur* (STIX, École Polytechnique, December 2003).

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 (Grenoble, Limoges, Montpellier, Orsay, Perpignan, Saint-Étienne, Toulouse, Rocquencourt, Sophia Antipolis). They gave talks at the "Computer Arithmetic Days (AriNews)" in Dijon, March 2003, and Lyon, Nov. 2003.

National meeting:

- P. Giorgi, C.-P. Jeannerod et G. Villard spoke at the "Computer Algebra Week" (Luminy, January 2003).

International:

- M. Daumas was invited by the NASA Langley Research Center for two weeks at the National Institute of Aerospace, Virginia, Nov. 2003. He presented a NIA Formal Methods Seminar.
- P. Giorgi and N. Revol have been speakers —proposed by the organizers— at the "Conference on Applications of Computer Algebra (ACA'03)" Raleigh, North Carolina, July 2003.
- N. Revol has been invited to stay one month (November 2003) at the *Institut für Algorithmen und Kognitive Systeme* (Karlsruhe, Germany).
- G. Villard was an invited speaker at the "SIAM Conference on Linear Algebra", Williamsburg, Virginia, July 2003. He gave a talk at the Mathematics Department Seminar, North Carolina State University, Raleigh, USA, April 2003.

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. ERCEGOVAC, T. LANG. *Digital arithmetic.* Morgan Kaufmann, 2004, http://www.cs.ucla.edu/digital_arithmetic/.

[3] J. GRABMEIER, E. KALTOFEN, V. WEISPFENNING, editors, *Computer Algebra Handbook.* Springer Verlag, Heidelberg, Germany, 2003.

[4] N. J. HIGHAM. *Accuracy and stability of numerical algorithms (2nd edition).* SIAM, 2002, http://www.ma.man.ac.uk/~higham/asna/.

[5] P. MARKSTEIN. *IA-64 and elementary functions: speed and precision.* Prentice Hall, 2000, http://www.markstein.org/.

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

[7] VON ZUR GATHEN, J. GERHARD. *Modern Computer Algebra.* Cambridge University Press, 1999, http://www-math.uni-paderborn.de/mca/.

Books and Monographs

[8] *Special issue "Linear Algebra and Arithmetic" of Journal of Computational and Applied Mathematics.* S. EL HAJJI, N. REVOL, P. VAN DOOREN, editors, number 1, volume 162, 2003, to appear.

[9] *Special issue "Real Numbers and Computers" of Theoretical Computer Science.* P. KORNERUP, J.-C. BAJARD, C. FROUGNY, J.-M. MULLER, editors, number 2, volume 291, January, 2003.

Doctoral dissertations and "Habilitation" theses

[10] D. DEFOUR. *Fonctions Élémentaires : algorithmes et implémentations efficaces pour l'arrondi correct en double précision.* Ph. D. Thesis, École Normale Supérieure de Lyon, Lyon, France, September, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/PhD/PhD2003/PhD2003-01.ps.gz.

[11] G. VILLARD. *Algorithmique en algèbre linéaire exacte.* Habilitation à Diriger des Recherches, Université Claude Bernard, Lyon, France, March, 2003.

Articles in referred journals and book chapters

[12] J.-L. BEUCHAT, A. TISSERAND. *Évaluation polynomiale en-ligne de fonctions élémentaires sur FPGA.* in « Technique et science informatiques », 2003, to appear.

[13] S. BOLDO, M. DAUMAS. *A simple test qualifying the accuracy of Horner's rule for polynomials.* in « Numerical Algorithms », 2003, to appear.

[14] S. BOLDO, M. DAUMAS. *Properties of two's complement floating point notations.* in « International Journal on Software Tools for Technology Transfer », 2003, http://perso.ens-lyon.fr/marc.daumas/SoftArith/BolDau03b.pdf, to appear.

[15] N. BRISEBARRE, J.-M. MULLER, S. K. RAINA. *Accelerating correctly rounded floating-point division when the divisor is known in advance.* in « IEEE Transactions on Computers », 2003, to appear.

[16] M. DAUMAS, P. LANGLOIS. *Additive symmetries: the non-negative case.* in « Theoretical Computer Science », number 2, volume 291, 2003, pages 143-157, http://dx.doi.org/10.1016/S0304-3975(02)00223-2.

[17] 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 », 2003, to appear.

[18] F. DE DINECHIN, T. RISSET, M. MANJUNATHAIAH, M. SPIVEY. *System Specification and Design Languages (best of FDL'02).* Kluwer, 2003, chapter Design of highly parallel architectures with Alpha and Handel.

[19] C.-P. JEANNEROD. *On matrix perturbations with minimal leading Jordan structure.* in « J. Comp. Applied Math », 2003, to appear.

[20] C.-P. JEANNEROD, G. VILLARD. *Essentially optimal computation of the inverse of generic polynomial matrices.* in « Journal of Complexity », 2003, to appear.

[21] 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 », 2003, to appear.

[22] P. KOIRAN, N. PORTIER, G. VILLARD. *A rank theorem for Vandermonde matrices.* in « Linear Algebra and its Applications », 2003, to appear.

[23] P. LANGLOIS, N. REVOL. *Validating polynomial numerical computations with complementary automatic methods.* in « Mathematics and Computers in Simulation », 2003, http://www.inria.fr/rrrt/rr-4205.html, to appear.

[24] V. LEFÈVRE, J.-M. MULLER. *On-the-Fly Range Reduction.* in « Journal of VLSI Signal Processing », number 1/2, volume 33, February, 2003, pages 31–35.

[25] N. REVOL. *Interval Newton iteration in multiple precision for the univariate case.* in « Numerical Algorithms », number 2, volume 34, 2003, pages 417–426.

[26] B. SAUNDERS, A. STORJOHANN, G. VILLARD. *Matrix rank certification.* in « Elect. J. Linear Algebra », 2003, to appear.

[27] A. TISSERAND. *Low Power Electronics Design.* CRC Press, 2003, chapter Low-Power Arithmetic Operators, to appear.

Publications in Conferences and Workshops

[28] J.-L. BEUCHAT. *Modular Multiplication for FPGA Implementation of the IDEA Block Cipher.* in « Proceedings of the 14th IEEE International Conference on Application-Specific Systems, Architectures, and Processors », IEEE Computer Society, E. DEPRETTERE, S. BHATTACHARYYA, J. CAVALLARO, A. DARTE, L. THIELE, editors, pages 412–422, 2003.

[29] J.-L. BEUCHAT. *Some Modular Adders and Multipliers for Field Programmable Gate Arrays.* in « Proceedings of the 17th International Parallel & Distributed Processing Symposium », IEEE Computer Society, 2003.

[30] J.-L. BEUCHAT. *FPGA Implementations of the RC6 Block Cipher.* in « Field-Programmable Logic and Applications », series Lecture Notes in Computer Science, number 2778, Springer, P. Y. K. CHEUNG, G. A. CONSTANTINIDES, J. T. DE SOUSA, editors, pages 101–110, 2003.

[31] J.-L. BEUCHAT, L. IMBERT, A. TISSERAND. *Comparison of Modular Multipliers on FPGAs.* in « Proceedings of SPIE 2003 », 2003.

[32] J.-L. BEUCHAT, J.-M. MULLER. *Multiplication-addition modulaire: algorithmes itératifs et implantations sur FPGA.* in « Actes de RenPar'15, CFSE'3 et SympAAA'2003 », M. AUGUIN, F. BAUDE, D. LAVENIER, M. RIVEILL, editors, pages 235–242, October, 2003.

[33] S. BOLDO, M. DAUMAS. *Representable correcting terms for possibly underflowing floating point operations.* in « Proceedings of the 16th Symposium on Computer Arithmetic », J.-C. BAJARD, M. SCHULTE, editors, pages 79-86, Santiago de Compostela, Spain, 2003, http://perso.ens-lyon.fr/marc.daumas/SoftArith/BolDau03a.pdf.

[34] S. BOLDO, M. DAUMAS, L. THÉRY. *Formal proofs and computations in finite precision arithmetic.* in « Proceedings of the 11th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning », Roma, Italy, 2003.

[35] N. BOULLIS, A. TISSERAND. *Some Optimizations of Hardware Multiplication by Constant Matrices.* in « ARITH 16 », pages 20–27, 2003.

[36] A. DARTE, R. SCHREIBER, G. VILLARD. *Lattice-based memory allocation.* in « International Conference on Compilers, Architecture and Synthesis for Embedded Systems, San Jose, California, USA », November, 2003.

[37] F. DE DINECHIN, D. DEFOUR. *Software Carry-Save: A case study for instruction-level parallelism.* in

« Seventh International Conference on Parallel Computing Technologies », Nizhny Novgorod, Russia, September, 2003.

[38] 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 Software with Result Verification », Dagstuhl, Germany, 2003.

[39] J. DETREY, F. DE DINECHIN. *A VHDL Library of LNS Operators.* in « 37th Asilomar Conference on Signals, Systems and Computers », Pacific Grove, USA, 2003.

[40] J. DETREY, F. DE DINECHIN. *Outils pour une comparaison sans a priori entre arithmétique logarithmique et arithmétique flottante.* in « Symposium en Architecture et Adéquation Architectures Algorithmes », La Colle sur Loup, France, October, 2003.

[41] M. ERCEGOVAC, J.-M. MULLER. *Complex Division with Prescaling of the Operands.* in « Proceedings of ASAP'03: 14th IEEE Conference on Application-Specific Systems, Architectures and Processors », IEEE Computer Society, E. DEPRETTERE, S. BHATTACHARYYA, J. CAVALLARO, A. DARTE, editors, pages 304-314, June, 2003.

[42] P. GIORGI. *From BLAS routine to finite field exact linear algebra solution.* series The 9th International IMACS Conference on Applications of Computer Algebra, Raleigh, North Carolina, USA, July, 2003.

[43] P. GIORGI, C.-P. JEANNEROD, G. VILLARD. *On the complexity of polynomial matrix computations.* in « Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation (ISSAC'03) », ACM Press, pages 135–142, August, 2003.

[44] P. KORNERUP, J.-M. MULLER. *Choosing Starting Values for Newton-Raphson Computation of Reciprocals, Square-Roots and Square-Root Reciprocals.* in « Proceedings of RNC5 », M. DAUMAS, N. REVOL, editors, Lyon, September, 2003.

[45] R.-C. LI, S. BOLDO, M. DAUMAS. *Theorems on efficient argument reductions.* in « Proceedings of the 16th Symposium on Computer Arithmetic », J.-C. BAJARD, M. SCHULTE, editors, pages 129-136, Santiago de Compostela, Spain, 2003, http://perso.ens-lyon.fr/marc.daumas/SoftArith/LiBolDau03.pdf.

[46] J.-M. MULLER. *"Partially rounded" Small-Order Approximations for Accurate, Hardware-Oriented, Table-Based Methods.* in « Proceedings of the 16th Symposium on Computer Arithmetic », J.-C. BAJARD, M. SCHULTE, editors, pages 114-121, Santiago de Compostela, Spain, 2003, http://computer.org/proceedings/.

[47] N. REVOL. *Multiple precision interval arithmetic and application to linear systems.* series The 9th International IMACS Conference on Applications of Computer Algebra, Raleigh, North Carolina, USA, July, 2003.

[48] N. REVOL, F. ROUILLIER. *The MPFI library: introduction and applications.* in « Numerical Software with Result Verification », Dagstuhl, Germany, 2003.

[49] G. VILLARD. *Exact computations on polynomial and integer matrices (invited talk).* series SIAM Conference on Applied Linear Algebra, Williamsburg, Virginia, USA, July, 2003.

Internal Reports

[50] J.-L. BEUCHAT. *More on Modulo $2^n - 1$ Addition.* Research report, number 2003-14, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, February, 2003.

[51] J.-L. BEUCHAT, J.-M. MULLER. *Opérateurs itératifs de multiplication-addition modulaire pour FPGA.* Research report, number 2003-40, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, August, 2003.

[52] N. BRISEBARRE. *An algorithm for finding entire solutions of systems of difference equations.* Research report, number 2003-53, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, November, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2003/RR2003-53.ps.gz.

[53] N. BRISEBARRE, J.-M. MULLER. *Finding the "truncated" polynomial that is closest to a function.* arXiv, number cs.MS/0307009, 2003, http://arxiv.org/abs/cs/0307009.

[54] N. BRISEBARRE, G. PHILIBERT. *Effective lower and upper bounds for the Fourier coefficients of powers of the modular invariant j.* Research report, number 2003-50, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, November, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2003/RR2003-50.ps.gz.

[55] D. DEFOUR, F. DE DINECHIN, J. MULLER. *Crlibm: The evaluation of the exponential.* Research report, number 2003-37, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, July, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2003/RR2003-37.ps.gz.

[56] M. GRIMMER, K. PETRAS, N. REVOL. *Multiple Precision Interval Packages: Comparing Different Approaches.* Research report, number 4841, INRIA, 2003, http://www.inria.fr/rrrt/rr-4841.html.

[57] E. KALTOFEN, G. VILLARD. *On the complexity of computing determinants.* Research report, number 2003-36, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2003/RR2003-36.ps.gz.

[58] C. Q. LAUTER. *A correctly rounded implementation of the exponential function on the Intel Itanium architecture.* Research report, number 2003-54, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, November, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2003/RR2003-54.ps.gz.

[59] J.-M. MULLER, J.-L. NICOLAS, X. ROBLOT. *Nombres de Solutions dans une Binade de l'Equation $a^2 + b^2 = c^2 + c$.* Research report, number 2003-45, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, 2003.

[60] N. REVOL, K. MAKINO, M. BERZ. *Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY.* Research report, number 4737, INRIA, 2003, http://www.inria.fr/rrrt/rr-4737.html.

Miscellaneous

[61] J. DETREY. *Bibliothèque d'opérateurs paramétrables pour l'arithmétique "réelle" sur FPGA.* Mémoire de DEA, École Normale Supérieure de Lyon, Lyon, France, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/DEA/DEA2003/DEA2003-07.ps.gz.

[62] G. MELQUIOND. *Robustesse d'algorithmes pour l'évitement des collisions aériennes.* Mémoire de DEA, École Normale Supérieure de Lyon, Lyon, France, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/DEA/DEA2003/DEA2003-03.pdf.

[63] S. K. RAINA. *Efficient division methods when the divisor is known beforehand.* Mémoire de DEA, École Normale Supérieure de Lyon, Lyon, France, 2003, http://www.ens-lyon.fr/LIP/Pub/Rapports/DEA/DEA2003/DEA2003-06.pdf.

Bibliography in notes

[64] M. BERZ, J. HOEFKENS. *Verified high-order inversion of functional dependencies and interval Newton methods.* in « Reliable Computing », number 5, volume 7, 2001, pages 379–398.

[65] O. HERRMANN. *Über die Berechnung der Fourierkoeffizienten der Funktion $j(\tau)$.* in « J. Reine Angew. Math. », volume 274/275, 1975, pages 187–195.

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

[67] Y.-J. JEONG, W. P. BURLESON. *VLSI Array Algorithms and Architectures for RSA Modular Multiplication.* in « IEEE Transactions on Very Large Scale Integration Systems », number 2, volume 5, June, 1997, pages 211–217.

[68] E. KALTOFEN. *Challenges of symbolic computation: my favorite open problems.* in « J. Symbolic Computation », number 6, volume 29, 2000, pages 891–919.

[69] E. KALTOFEN, B. TRAGER. *Greatest common divisors, factorization, separation of numerators and denominators.* in « J. Symbolic Computation », number 3, volume 9, 1990, pages 301–320.

[70] X. LAI. *On the Design and Security of Block Ciphers.* series ETH Series in Information Processing, Hartung–Gorre Verlag Konstanz, 1992.

[71] V. LEFÈVRE. *Multiplication par une constante.* in « Réseaux et Systèmes Répartis, Calculateurs Parallèles - numéro spécial sur l'Arithmétique des Ordinateurs », number 4-5, volume 13, 2001, pages 465–484.

[72] K. MAHLER. *On the coefficients of transformation polynomials for the modular function.* in « Bull. Austral. Math. Soc. », volume 10, 1974, pages 197–218.

[73] K. MAKINO, M. BERZ. *Higher order verified inclusions of multidimensional systems by Taylor models.* in « Nonlinear Analysis », volume 47, 2001, pages 3503-3514.

[74] R. L. RIVEST, M. J. B. ROBSHAW, R. SIDNEY, Y. L. YIN. *The* RC*6 Block Cipher.* 1998, http://www.rsasecurity.com/rsalabs/rc6.

[75] J.-P. SERRE. *Cours d'arithmétique.* P.U.F., 1970.

[76] SPACES PROJECT. *The MPFR library, version 2.0.1.* http://www.mpfr.org, 2002.