INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Arénaire

Computer Arithmetic

Rhône-Alpes

THEME SYM

Activity Report

2005

Table of contents

1. Team

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

The year 2005 is marked by the departure of three researchers whose investment for Arénaire has been considerable. Jean-Luc Beuchat who has finished a four years postdoctoral period, Marc Daumas who moves as CR1 CNRS to Montpellier-Perpignan, and Arnaud Tisserand who has been hired CR1 CNRS at Montpellier.

Head of the team
Gilles Villard [Research Scientist, CR CNRS]

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

Inria Scientists
Édouard Bechetoille [Technical Staff, since October 1, 2005]
Nicolas Brisebarre [Research Scientist, CR (on partial secondment) until August 31, 2005, then Associate Professor, *Maître de Conférences* U. J. Monnet St-Étienne]
Claude-Pierre Jeannerod [Research Scientist, CR]
Nathalie Revol [Research Scientist, CR]
Arnaud Tisserand [Research Scientist, CR until September 30, 2005, then visiting scientist]

CNRS Scientists
Marc Daumas [Research Scientist, CR, until November 30, 2005]
Jean-Michel Muller [Research Scientist, DR]

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

Post-Doctoral Fellows
Jean-Luc Beuchat [Post-doctoral fellow of the *Fonds National Suisse de la Recherche Scientifique*, from November 1, 2001 to November 30, 2005]
Ilia Toli [Inria Post-doctoral Fellow since May 1, 2005]

PhD Students
Sylvie Boldo [ÉNS student *Allocataire-monitrice* ÉNS (PhD defense on November 22, 2004), until August 31, 2005]
Francisco Cháves [*European Marie Curie grant* Mathlogaps, 2nd year]
Jérémie Detrey [ÉNS student *Allocataire-moniteur* INSA, 3rd year]
Christoph Lauter [*Allocataire-moniteur* MESR ÉNS, 1st year]
Guillaume Melquiond [ÉNS student *Allocataire-moniteur* INSA, 3rd year]
Romain Michard [Inria grant, 2nd year]
Saurabh Kumar Raina [Grant from the *Région Rhône-Alpes*, 3rd year]
Nicolas Veyrat-Charvillon [*Allocataire-moniteur* MESR ÉNS, 2nd year]

2. Overall Objectives

2.1. 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.

The conception of 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 minimize energy consumption.

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.

Thanks to past and recent efforts, 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, this represents a key point for improving numerical reliability. Standardization is also related to properties of floating-point arithmetic—we mean invariants that operators or sequences of operators may satisfy. We work on establishing new properties such as exact rounding or the representation of roundoff errors. An important objective is further to have these results progressively integrated into the future standards of floating-point arithmetic.

For certifying the properties that we identify, and the compliance of the numerical programs we develop with their specifications, we rely on formal proving. Proofs are checked using Coq [58] as well as PVS [63] proof assistants. In particular, this is made possible by a careful specification of the arithmetic operators that are involved.

An increasingly growing demand exists for certified numerical results, we mean for computing with known or controlled errors. Our answer is the conception of modern and efficient error-measurement tools. We first concentrate on two types of errors: roundoff errors, polynomial and power series approximation errors. Our objective here is the conception and the implementation of automatic tools for computing certified (exact) error bounds.

When conventional floating-point arithmetic does not suffice, we use other kinds of arithmetics. Especially in the matter of error bounds, we work on interval arithmetic libraries, including arbitrary precision intervals. Intervals give an “exact” answer when the problem is to bound the result of a computation. Here a main domain of application is global optimization. Original algorithms dedicated to this type of arithmetic must be designed in order to get accurate solutions or sometimes simply to avoid divergence, 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).

To conclude we emphasize that the research directions mentioned above are supported by the development and diffusion of corresponding libraries, either in hardware or in software. The features of our libraries are the quality of the implemented algorithms, and code optimization for performance. See §5. for details about each of these libraries.

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 (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, subnormal numbers, etc).

3.3.1. Formal Specifications and Proofs

Very mediatized problems (the Pentium bug, or the fact that $2001!/2000! = 1$ in Maple 7) show that arithmetic correctness is sometimes difficult to obtain on a computer. Few tools handle rigorous proofs on floating-point data. However, thanks to the IEEE-754 standard, the arithmetic operations are completely specified, which makes it possible to build proofs of algorithms and properties. But it is difficult to present a proof including the long list of special cases generated by these calculations. The formalization of the standard, begun with our collaboration with the Lemme project (ARC AOC) in year 2000, makes it possible to use a proof assistant such as Coq [58] to guarantee that each particular case is considered and handled correctly. Thanks to funding from CNRS and NASA, the same specification is now also available in PVS.

Systems such as Coq and PVS make it possible to define new objects and to derive formal consequences of these definitions. Thanks to higher order logic, we establish properties in a very general form. The proof is built in an interactive way by guiding the assistant with high level tactics. At the end of each proof, Coq builds an internal object, called a proof term, which contains all the details of derivations and guarantees that the theorem is valid. PVS is usually considered less reliable because it builds no proof term.

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 rounding also requires the study of algorithms in large (but not arbitrary) precision, as well as the study of more general methods for the three stages of the evaluation of elementary functions: argument reduction, approximation, and reconstruction of the result.

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 in the last few years, substantially reducing 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 reduce problems to matrix multiplication in a systematic way, and to relate bit complexity to algebraic complexity. Another direction is to make these algorithms fast in practice as well, especially since applications yield very large matrices that are either sparse or structured. 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 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

4.1. 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** and **PVS** are used to formally **bound values and roundoff errors** for **safety critical** applications such as flight control. Our automatic tool (see §5.13) 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 **reproducibility** of floating-point computations. Exponentials and logarithms, for instance, are routinely used in accounting systems for interest calculation, where roundoff errors have a financial meaning. Our current focus is on bounding the worst-case time for such computations, which is required to allow their use in **safety critical** applications.
- 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**.
- The design of faster algorithms for matrix polynomials provides faster solutions to various problems in **control theory**, especially those involving multivariable linear systems.

5. Software

5.1. Introduction

Arénaire proposes various software and hardware realizations that are accessible from the web page http://www.ens-lyon.fr/LIP/Arenaire/Ware. We describe below only those which progressed in 2005.

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

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

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

The CRlibm project aims at developing a mathematical library (`libm`) which provides implementations of the double precision C99 standard elementary functions,
- correctly rounded in the four IEEE-754 rounding modes,
- with a comprehensive proof of both the algorithms used and their implementation,
- sufficiently efficient in average time, worst-case time, and memory consumption to replace existing `libms` transparently.

In 2005, we released the stable version 0.8, with implementations of exponential, natural logarithm, hyperbolic sine and cosine, arctangent, and the trigonometric functions (sine, cosine and tangent), and complete proofs of these implementations. The infrastructure was also improved in many ways, especially concerning testing and validation.

Then we released the interim versions 0.10beta, then 0.11beta with three more functions (logarithms in base 2 and 10, and arcsine), and a complete rewrite of the previous logarithms and exponential: the new versions use double-extended hardware when available, and also reduce the worst-case execution time by a factor 10 using either double-double-extended [10] or triple-double [52], [22].

In the beta versions, the code is complete, working and validated by an extensive self-test procedure, but the proof of correct rounding is not complete yet. These new releases include machine-assisted proofs using the Gappa tool [48].

The library includes an extensive documentation and proof which makes an excellent tutorial on elementary function software development.

The library has been downloaded more than 1000 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 / **Programming Language:** C / **URL:** http://www.ens-lyon.fr/LIP/Arenaire

5.3. 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 developed 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

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

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

Participants: J. Detrey, F. de Dinechin.

FPLibrary is a VHDL library that describes arithmetic operators (addition, subtraction, multiplication, division, and square root) for two formats of representation of real numbers: floating-point, and logarithmic number system [18]. These formats are parametrized in precision and range.

It has been extended in 2005 by the addition of floating-point logarithm and exponential operators [3], which exhibit 10x speedup when compared with the processor. FPLibrary is the first hardware library to offer parametrized hardware architectures for such elementary functions.

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

5.5. HOTBM: a VHDL Generator for the Higher-Order Table-Based Method

Keywords: *FPGA*, *fixed-point*, *function evaluation*.

Participants: J. Detrey, F. de Dinechin.

HOTBM is a VHDL generator for fixed-point function evaluation operators using the Higher-Order Table-Based Method [34]. The key features of this method are:

- piecewise polynomial approximation,
- parallel computation of all the terms,
- ad-hoc powering units,
- optimized look-up tables,
- small multipliers,
- and guaranteed faithful rounding.

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

5.6. LinBox: High Performance Software for Matrix Computations

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

Participant: G. Villard.

This software library is developed 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 [59], offers connections to external softwares like Maple, and provides online servers. LinBox 1.0 (July 2005) is available. This is the first non-beta and stable release. The algorithmic solutions provided to the user have undergone a major revision and extension with the purpose of making the linear algebra functions easily invoked. Also the example directory contains illustrative uses of each function. The new algorithms released concern especially Dixon's rational solving, the characteristic polynomial, block Krylov solvers, and a hybrid algorithm for the Smith normal form. A Linbox/Maple interface has been developed by P. Giorgi (who left Arénaire early 2005) during his postdoctoral stay at the Symbolic Computation Group (Waterloo University, Canada). LinBox is part of the Roxane project (see §8.1).

Status: Stable / **License:** LGPL / **OS:** Unix, Linux, Windows (Cygwin), WinNT / **Programming Language:** C++ / **Dependencies:** GMP, ATLAS / **URL:** http://www.linalg.org

5.7. 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 [8]. For efficiency and portability reasons, it is based on GMP and MPFR and the implementation takes advantage of these specific libraries. Modifications made this year mainly concern the compatibility with new releases of MPFR.

Status: stable (alpha for the C++ interface) / **Target:** x86, PPC / **License:** GPL / **OS:** Unix, Windows (Cygwin) / **Programming Language:** C, C++ / **Dependencies:** GMP v4.1.0 or higher, MPFR v2.2.0 or higher / **URL:** http://www.ens-lyon.fr/LIP/Arenaire

5.8. MPCheck: Testing the Quality of Elementary Functions

Keywords: *elementary function*, *mathematical library*, *quality*, *rounding*.

Participant: N. Revol.

The MPCheck program, version 1.1.0, designed with P. Zimmermann and P. Pélissier (Spaces project, Inria Lorraine), is freely distributed. It tests the elementary functions available in binary floating-point mathematical libraries (correct rounding, output range, monotonicity, symmetry), by performing numerous evaluations of these functions on random arguments.

Status: stable (available only for the double precision) / **Target:** x86, PPC, Sparc, Itanium / **License:** GPL / **OS:** Unix / **Programming Language:** C / **Dependencies:** gcc 3.3, GMP 4.1.0, MPFR 2.1.0, or higher / **URL:** http://www.loria.fr/~zimmerma/mpcheck/

5.9. Boost Interval Arithmetic Library

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

Participant: G. Melquiond.

In collaboration with H. Brönnimann (Polytechnic U. Brooklyn, 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 1.32 release has been downloaded more than 130,000 times.

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

5.10. MEPLib : Machine-Efficient Polynomials Library

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

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

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

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

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

5.11. PFF: Formal Proofs about Floats

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

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

Our library of theorems and proofs about floating-point arithmetic is based on the library originated by M. Daumas, L. Rideau and L. Théry during the ARC AOC. 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.

Four 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

5.12. A part of the NASA Langley PVS Libraries

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

Participants: S. Boldo, M. Daumas.

The Coq formalization of floating-point numbers originated in the ARC AOC was ported in PVS by S. Boldo and C. Muñoz. This formalization and new results (see §6.3) are now part of the NASA Langley PVS Libraries which is one of the most well-known and most complete PVS libraries available.

Status: stable / **License:** free / **Programming Language:** PVS / **URL:** http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html

5.13. Gappa: a Tool for Certifying Numerical Programs

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

Participants: M. Daumas, G. Melquiond.

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

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

Various people use this tool in order to certify their numerical code. For example, in CRlibm, floating-point elementary functions are proved with its help (see §6.4).

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

5.14. FLIP: Floating-point Library for Integer Processors

Keywords: *VLIW processor*, *addition*, *division*, *floating-point arithmetic*, *multiplication*, *single precision*, *software library*, *square root*.

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

FLIP is a C library for the software support of single precision floating-point arithmetic on processors without floating-point hardware units such as VLIW (Very Large Instruction Word) or DSP (Digital Signal Processor) processors for embedded applications. The current target architecture is the VLIW ST200 family from STMicroelectronics. This research project is funded by Région Rhône-Alpes.

The library provides the five basic operations: addition, subtraction, multiplication, division and square-root for the single-precision IEEE 754 FP format. It also provides some running modes with relaxed characteristics: no subnormal numbers nor restricted rounding modes for instance. The latest version (Flip-0.2) also includes the following optimized additional operators: square, FMA, reciprocal and square root reciprocal.

Status: Beta release (Flip-0.2) / **Target:** VLIW processors (ST200 family from STMicroelectronics)/ **License:** LGPL / **OS:** Linux, Windows / **Programming Language:** C / **URL:** http://lipforge.ens-lyon.fr/www/flip

6. New Results

6.1. Hardware Arithmetic Operators

Keywords: *ASIC*, *FPGA*, *arithmetic operators*, *circuit generator*, *division*, *function evaluation*, *integrated circuit*, *low-power consumption*.

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

6.1.1. Evaluation of Functions

J. Detrey and F. de Dinechin have worked on general methods for the hardware evaluation of fixed-point elementary functions. The Higher-Order Table-Based Method is a generalization of previous first-order methods [11] to arbitrary polynomials. It provides implementations both smaller and faster thanks to the use of small multipliers and powering units [34].

The previous fixed-point function generators have been used to build the first floating-point elementary function library for FPGAs: J. Detrey and F. de Dinechin have studied the floating-point logarithm [32], then the exponential [33]. Although the basic floating-point operators in FPGAs are usually much slower than their processor counterparts, both elementary functions exhibit 10x speedup when compared with the processor, thanks to specific algorithms. This work will be published in the Special Issue on FPGA-based Reconfigurable Computing of the Journal of Microprocessors and Microsystems [3].

R. Michard, A. Tisserand and N. Veyrat-Charvillon have proposed a new method for the approximation of functions without multipliers. This method uses degree-2 or degree-3 polynomial approximations with at most

3 non-zero bits for the coefficients and low precision estimations of the powers of x. The first implementation on FPGAs leads to very small operators by replacing the costly multipliers by a small number of additions [41] (best paper award of the ASAP 2005 Conference).

R. Michard, A. Tisserand and N. Veyrat-Charvillon have worked on an efficient FPGA implementation of a shift-and-add algorithm, for polynomial and rational approximation of functions. These operators are high-radix iterations of the E-method proposed by M. Ercegovac. The results show high performances by mixing the simple architecture of shift-and-add algorithms and the generic nature of polynomial and rational approximations [40].

M. Ercegovac, J.-M. Muller and A. Tisserand have worked on the approximation of the reciprocal and the square root reciprocal in hardware. The proposed method is based on degree-1 polynomial approximation with specific coefficients and a table [35]. These approximations can be used to speed up the division and square root software iterations.

6.1.2. Division

R. Michard, A. Tisserand and N. Veyrat-Charvillon have developed a software for the generation of division circuits [39]. This software, called `divgen`, allows the comparison of various parameters (sizes, radix, algorithm type, optimizations...) for architecture exploration. This work has been done within a collaboration between Inria and CEA-Léti. This software is released in version 0.11.

6.1.3. Low-Power Arithmetic Operators

R. Michard, A. Tisserand and N. Veyrat-Charvillon have done a statistical study of the activity due to the selection function in the polynomial approximation algorithm called E-method and proposed by M. Ercegovac. The latitude in the choice of the result digits in the selection function, when using a redundant representation, allows to consider a reduced electrical activity in some cases. Power consumption benefits can be expected [42].

6.1.4. Hash Functions

R. Glabb, a Phd student at ATIPS laboratory at University of Calgary, and N. Veyrat-Charvillon studied the SHA-2 family of hash functions. They implemented more efficient stand-alone versions of the separate operators, and devised a multi-mode operator able to compute all algorithms in a single architecture with a very-high level of hardware sharing between the modes. This collaboration took place during September at LIP, and in October at ATIPS.

6.1.5. 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 [55]. 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.

In 2004, J.-L. Beuchat, N. Sendrier, A. Tisserand, and G. Villard proposed a first hardware implementation which allows to sign a document in 0.86 second on an XCV300E-7 FPGA, hence making this scheme practical [54]. However, N. Sendrier modified the first step of the algorithm to prevent a possible flaw. This step involved a multiplication by a matrix stored in the memory blocks of the FPGA. Since the new version does not require this matrix anymore, intermediate results can now be stored in the memory blocks and more configurable logic is available to implement computing units. Therefore, we decided to study a new architecture from scratch. Our second architecture reduces the signature time (place-and-route results), while improving the security. We plan to design a prototype on a ZestSC1 FPGA USB board (see http://www.orangetreetech.com for details) and to publish our results.

6.1.6. Iterative Modular Multiplication

Modular multiplication is often implemented in a parallel-serial fashion: the n-bit operand Y is stored in a register and X is processed digit by digit. At each step, a partial product $2^i x_i Y$ is formed and added (modulo

M) to the previous intermediate result. The well-known Montgomery's algorithm [61] allows to design LSDF (Least Significant Digit First) algorithms. MSDF (Most Significant Digit First) schemes are based on Horner's rule.

6.1.6.1. Survey and Practical Aspects

J.-L. Beuchat, J.-M. Muller, M. Neve (UCL Crypto Group), and E. Peeters (UCL Crypto Group) studied and implemented most of iterative schemes published in the open literature. They plan to carry out a fair comparison of these algorithms on FPGA and to write a survey on this topic.

6.1.6.2. High-Radix Carry-Save Algorithm Based on Horner's Rule

MSDF algorithms are based on the following iteration:

$$Q[i] = (2Q[i+1]+x_i Y) \mod M,$$

where $Q[r] = 0$ and $Q[0] = XY \mod M$. Several improvements of this algorithm have been proposed. The basic idea consists in computing a number congruent with $Q[i]$ modulo M, which requires less hardware than a modulo M addition.

Public key cryptography often involves modular multiplication of large operands (160 up to 2048 bits). Several researchers have proposed iterative algorithms whose internal data are carry-save numbers. This number system is unfortunately not well suited to today's Field Programmable Gate Arrays (FPGAs) embedding dedicated carry logic.

J.-L. Beuchat, J.-M. Muller, R. Beguenane (Université du Québec à Chicoutimi), and S. Simard (Université du Québec à Chicoutimi) proposed to perform modular multiplication in a high-radix carry-save number system, where the *sum bit* of the well-known carry-save representation is replaced by a *sum word* [24]. The originality of this approach is to analyze the modulus in order to select the most efficient high-radix carry-save representation. Place-and-route results show that this approach reduces the area up to 50% and does not increase the critical path compared to previously published algorithms based on Horner's rule.

6.1.7. RN-Codings

A property of the original Booth recoding is that the first non-zero digit following a 1 is necessarily -1 and vice versa. This allows to prove that truncating the Booth recoding of a number X is equivalent to rounding x to the nearest. P. Kornerup and J.-M. Muller investigated the positional, radix β, number systems sharing this rounding property and called them RN-codings [37] (where "RN" stands for "Round to Nearest"). J.-L. Beuchat and J.-M. Muller studied addition, multiplication, and squaring algorithms for radix 2 RN-codings (i.e. Booth recodings) [26], [25].

6.1.8. Publication of Previous Works

The work done in 2001–2003 by F. de Dinechin and A. Tisserand on the multipartite table method has been published in *IEEE Transactions on Computers* [11].

The work done in 2002–2004 by N. Boullis and A. Tisserand on the generation of optimized circuits for the problem of multiplication by constants has been published in *IEEE Transactions on Computers* [13].

6.2. Software Division

Keywords: *DSP*, *SRT*, *VLIW*, *division*, *division by constant*, *floating-point division*, *high-radix SRT*, *shift-and-add algorithms*.

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

6.2.1. Algorithms for Floating-Point Arithmetic on Integer Processors

In [36] we present floating-point division algorithms and implementations. We compare all standard algorithms and propose a high-radix digit-recurrence algorithm which gives a speed-up factor of about 3.

6.2.2. Division by Constant

The work done in 2001–2003 by J.-M. Muller, A. Tisserand, B. Dupont de Dinechin (STMicroelectronics) and C. Monat (STMicroelectronics) on the division by constant for the ST100 DSP processor has been presented in ARITH17 [43].

6.3. Properties and Proofs on Floating-Point Arithmetic

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

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

6.3.1. 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.

N. Brisebarre and J.-M. Muller have shown that the `fma` instruction can often 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 [29]. These results have been presented at ARITH17. The authors are currently working on an extended and improved version of [29]. They prove that if a larger precision (one additional bit does suffice) than the target precision is available then the methods developed in [29] become simpler and faster.

S. Boldo and J.-M. Muller have given examples of some floating-point functions (such as $\mathrm{ulp}(x)$ or $\mathrm{Nextafter}(x, y)$), or some useful tests, that are easily computable using a fused-mac. They have also proved that the error of a fused-mac instruction is exactly representable as the sum of two floating-point numbers and given an algorithm that computes that error [28].

6.3.2. Formalization and Proved Results in Floating-Point Arithmetic Using PVS

During and after her stay at the National Institute for Aerospace (NIA, Hampton, VA) between February and April 2005, S. Boldo has been working on floating-point arithmetic properties in PVS. This collaboration was initiated by M. Daumas who worked out its funding and its scientific framework. This work is both a part of the former Coq formalization (see §5.11) and a new result about polynomial evaluation that is formally proved to be faithful on mild assumptions (met for example when computing elementary functions) [49]. All the developments are part of the NASA Langley PVS Libraries (see §5.12).

6.3.3. Double Rounding

Double-rounding consists in a first rounding in an intermediate extended precision and then in a second rounding in the working precision [56]. The natural question is then of the accuracy and correctness of the final result. S. Boldo and G. Melquiond proved an efficient algorithm [27] 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.

6.4. Correct Rounding of Elementary Functions

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

Participants: N. Brisebarre, F. de Dinechin, C. Lauter, G. Melquiond, J.-M. Muller.

6.4.1. Double Precision Correctly Rounded Elementary Functions

F. de Dinechin, A. Ershov (Intel Corporation) and N. Gast (ÉNS) demonstrated that correct rounding of elementary function in double precision entailed no overhead in term of average case speed, worst-case speed, and memory consumption for processors with double-extended hardware [10]. C. Lauter then extended this result to all processors with double precision hardware, thanks to a redundant triple-double format [52]. A range of implementations of correctly-rounded logarithm functions will be published in a special issue of the Journal of Theoretical Informatics [22].

As the main difficulty in such work is the proof of the correct rounding property, F. de Dinechin, G. Melquiond and C. Lauter developed and used the Gappa tool, a high-level proof assistant which helps building machine-checkable proofs of numerical properties [48].

Meanwhile, the CRlibm library was developed further, with the addition of `log2`, `log10` and `asin` functions, a complete rewrite of `exp`, a version of `log` optimized for IA32 and IA64 instruction sets, the addition of a self-test, bits of Gappa proofs for various functions, experimental code for interval functions, and many other improvements. The stable version 0.8 was released in April, and the next stable version (0.11) is scheduled for early 2006.

`crlibm` is used by the LHC@Home project at CERN, where it allows to manage the distribution of the computation on a network of heterogeneous computers. An article on the subject was submitted by F. de Dinechin, E. McIntosh (CERN) and F. Schmidt (CERN).

F. de Dinechin and G. Villard were invited by nuclear physicists to present a survey on the subject of quadruple precision at the Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT'05) [23].

6.4.2. Correct Rounding of Algebraic Functions

In [15], N. Brisebarre and J.-M. Muller explicit the link between the computer arithmetic problem of providing correctly rounded algebraic functions and some diophantine approximation issues. This allows to get bounds on the accuracy with which intermediate calculations must be performed to correctly round these functions.

6.4.3. Publication of Previous Works

A former work on range reduction has been published in [14]. The proposed algorithm is fast for most cases and accurate over the full range. Furthermore, the statistical distribution of these cases has been determined.

In [30], S. Chevillard and N. Revol present an algorithm for the evaluation of the error functions erf and erfc in arbitrary precision with correct rounding.

6.5. Approximation

Keywords: *Chebyshev polynomials, Fourier coefficient, Hankel function, automatic generation, circle method, floating-point arithmetic, linear programming, minimax approximation, modular function, modular invariant, polynomial approximation, polytopes*.

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

6.5.1. Efficient Polynomial Approximation

In [1], N. Brisebarre, J.-M. Muller and A. Tisserand provide a general and efficient method for finding the best polynomial approximation under constraints of form and size in bits of the coefficients. The method described in [1] is currently implemented in the C library MEPLib (cf. §5.10).

6.5.2. Modular Functions

The modular invariant j is an important function in number theory [64]. G. Philibert (LARAL, Saint-Étienne) and N. Brisebarre established in [16] precise upper and lower bounds for the Fourier coefficients of j^m for all $m \in \mathbb{N} \smallsetminus \{0\}$. These results improve on previously known results, especially those of Mahler [60] and Herrmann [57].

6.6. Certified Numerical Codes, Interval Arithmetic and Taylor Models

Keywords: *Coq, PVS, Taylor models, formal proof, interval arithmetic*.

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

6.6.1. PVS-Guaranteed Proofs using Interval Arithmetic

With C. Muñoz (National Institute of Aerospace), we have designed a set of tools [31] for mechanical reasoning using interval arithmetic in the PVS proof assistant [63]. 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.

6.6.2. 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 [65]. Taking into account all the potential problems is a tedious and error-prone work if done 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 [38]. 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 [20]. These filters have been added to the latest release of the CGAL software http://www.cgal.org/.

6.6.3. Formal Proofs on Taylor Models Arithmetic

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

Defining operations in a proof assistant is usually very simple. However, work is needed since we have to prove that the operators implement what they are supposed to implement. We have proven the arithmetic operations and a few elementary functions on Taylor models using the proof assistant system PVS.

6.6.4. Efficient and Accurate Computations on Taylor Models with Floating-Point Arithmetic

When arithmetic on Taylor models is implemented using floating-point arithmetic for the coefficients of the Taylor models, roundoff errors due to the representation and to previous computations are also accounted for in the interval remainder [7]. Using the properties of the IEEE-754 floating-point arithmetic and algorithms proposed by Rump [62], accurate algorithms have been proposed for the arithmetic operations on Taylor models using floating-point arithmetic [44].

6.7. Algorithms and Software for High Performance Linear Algebra

Keywords: *Padé approximants, asymptotic complexity, determinant, inversion, matrix fraction, matrix multiplication, nullspace, polynomial matrix, rank, reduced form.*

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

We have pursued our study of asymptotically fast algorithms for the most basic operations on polynomial matrices, with an emphasis on reductions to the multiplication problem. The target matrices are typically $n \times n$ of degree d with univariate entries in $\mathsf{K}[x]$ for K an arbitrary commutative field. Among the studied operations are computing the *rank*, a *nullspace* basis, the *determinant* or a *reduced form*; there is also the task of computing the matrix of fractions which is equal to the *inverse* of a generic polynomial matrix [4]. In [51] we highlight the role played by two problems when designing asymptotically fast algorithms for any of the operations above: *computing minimal bases* of some matrix Padé approximants, and *expanding/reconstructing polynomial matrix fractions*. We show that reducing these two problems to polynomial matrix multiplication implies the same kind of reductions for all the other operations, hence cost estimates in $O(n^\omega d \log^\alpha n \log^\beta d)$ operations in K where ω is the exponent of scalar matrix multiplication, and with α and β two real constants. The latter type of reduction has been developed in [45] for establishing that a polynomial nullspace basis (almost minimal) can be computed using about the same number of operations as for multiplying two polynomial matrices. These studies, and an algorithm we propose for computing the Kalman form [53], may be useful for the treatment of multivariable linear systems in control.

Part of these theoretical developments yield software components in the LinBox library (see §5.6).

7. Contracts and Grants with Industry

7.1. Région Rhône-Alpes Grant

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

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

2005 has been the second year of a 3 year joint project with STMicroelectronics (fall 2003-fall 2006). It is supported by both the *Région Rhône-Alpes* and STMicroelectronics. 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. ANR GECKO Project

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

Participant: G. Villard.

The GECKO project (*Geometrical Approach to Complexity and Applications*, end of 2005-2008) is funded by ANR and headed by B. Salvy (ALGO project, Inria Rocquencourt). Other teams participating are at the École polytechnique, Université de Nice Sophia-Antipolis, and Université Paul Sabatier in Toulouse. The project is at the meeting point of numerical analysis, effective methods in algebra, symbolic computation and complexity theory. The aim is to improve significantly solution methods for algebraic or linear differential equations by taking geometry into account.

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

8.1.2. Ministry Grant ACI "Cryptology"

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

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

The OPAC project (*OPérateurs Arithmétiques pour la Cryptographie*, 2002-2005), is a collaboration with the team *Arithmétique Informatique* of the Lirmm laboratory and the GTA team at Université de 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.

8.1.3. 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 project (*Opérateurs Cryptographiques et Arithmétique Matérielle*, 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 (see §6.1).

8.1.4. Ministry Grant ACI "New Interfaces of Mathematics"

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

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

The GAAP project (*étude et outils pour la Génération Automatique d'Approximants Polynomiaux efficaces en machine*, 2004-2007) is a collaboration with the LArAl laboratory of Université de 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 target applications are software and hardware implementations, such as embedded systems for instance.

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 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 was headed by S. Lesecq (Lag, INPG Grenoble) and N. Revol.

8.1.6. "Adaptive and Hybrid Algorithms", Imag-Inria project

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

Participants: N. Revol, G. Villard.

The AHA project (*Adaptive and Hybrid Algorithms*, March 2005-2007) is headed by J.-L. Roch (Laboratoire ID-Imag), and supported by Imag Grenoble and Inria. Our motivation is the conception of algorithms that may adapt themselves automatically to the execution context. Arénaire is involved for building reliable algorithms (e.g. adaptive precision, general algorithms versus algorithms specific to interval arithmetic, ...). Other partners of the project will focus on parallel developments for problems in optimization and vision. Using the AHA approach our objective is to improve the performance of softwares such as LinBox (§5.6) and Roxane (§8.1).

8.1.7. Roxane Initiative

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

Participants: 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 research teams. Roxane integrates, in a homogeneous environment, tools to build dedicated and efficient components for solving real problems, mainly in computer algebra. The promotion of Roxane is done via http://www-sop.inria.fr/galaad/logiciels/roxane, and schools, software distribution CDs, etc.

8.2. European Initiatives

8.2.1. Mathlogaps Marie Curie Early Stage Training

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

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

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).

F. Cháves has started a PhD in the Arénaire project in November 2004. He will develop the use and certification of interval arithmetic with the PVS automatic proof checker (see the related result in §6.6). M. Hofmann acts as a distant expert and a future host in Munich for this PhD.

8.3. International Initiatives

8.3.1. Contributions to Standardization Bodies (ANSI-IEEE 754R and ISO/IEC JTC1/SC22/WG21)

The Department of Development and Industrial Relations (DirDRI) at Inria has supported our participation to the ongoing revision of the IEEE Standard for Binary Floating-Point Arithmetic (ANSI-IEEE 754). Since the first visit of M. Daumas, we have taken many opportunities to raise the impact of formal results from our project and lately from the Spaces project that was also supported in 2005. In particular, correctly rounded floating-point elementary functions are now considered for inclusion in the next revision of the IEEE-754 standard. G. Melquiond participated to the meeting of the revision committee in June 2005. Extended scientific reports are available on the intranet of DirDRI.

The challenges encountered when developing the Boost library made clear how an interval arithmetic library and the C++ language have to interact. As a consequence, G. Melquiond wrote, in collaboration with H. Brönnimann (Polytechnic U. Brooklyn, NY USA) and S. Pion (Géométrica team, Sophia Antipolis), a proposal [50] for integrating interval arithmetic to the C++ standard library which was submitted to the C++ Standards committee (ISO/IEC JTC1/SC22/WG21). H. Brönnimann and S. Pion participated to the Fall meeting of the revision committee in 2005.

8.3.2. LinBox Initiative

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

Participant: G. Villard.

LinBox is an ongoing collaborative research project for efficient algorithms and a software library in exact linear algebra (see §5.6 and §6.7). About thirty researchers from nine institutions in Canada, the USA and France are participating—http://www.linalg.org.

8.3.3. 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 (postponed due to the pregnancy of N. Revol).

8.3.4. Certifications of Properties of Floating-Point Arithmetic (CNRS-NASA)

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

Participants: S. Boldo, F. Cháves, 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 Aerospace 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 Arénaire project. He animated a series of seminal workshops.

In 2005, this collaboration with NIA participated to a long post-doctoral visit of S. Boldo, and a short visit of F. Cháves and F. Kirchner (*École Polytechnique*). In the meantime, G. Melquiond worked with colleagues from University of California at Berkeley and presented a seminar at Intel in Portland.

8.3.5. Collaboration ATIPS–LIP

Keywords: *cryptography*, *efficient implementation*, *hardware arithmetic operator*, *hash function*.

Participants: J.-L. Beuchat, A. Tisserand, N. Veyrat-Charvillon.

J.-L. Beuchat and A. Tisserand have been invited one month (June 2005) in the ATIPS laboratory, University of Calgary, Canada, to work on the efficient implementation of arithmetic operators (dedicated modular arithmetic support in current architecture and FPGA implementations).

R. Glabb (ATIPS) has been invited one month (September 2005) in Lyon and N. Veyrat-Charvillon has been invited one month (October 2005) in Calgary on the implementation of an efficient multi-mode operator computing all algorithms of the SHA-2 family of hash functions into a single architecture.

9. Dissemination

9.1. Conferences, Edition

- N. Brisebarre is member of the program committee of RNC7 (*7th Conference on Real Numbers and Computers*, Nancy, 2006).
- M. Daumas is co-program chair of the *2005 French Symposium on Computer Architecture* (SympA) held at Croisic.
- M. Daumas is organizing the 2006 French joint conference Renpar, Sympa and CFSE to be held near Perpignan.
- M. Daumas and J.-M. Muller are members of the steering committee of the *French Symposium on Computer Architecture* (SympA).
- 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 2006.
- J. Detrey and G. Melquiond organized the *Journées Arinews* (Lyon, France, May 2005).
- C.-P. Jeannerod has been in charge of the tutorials at the 2005 *International Symposium on Symbolic and Algebraic Computation* (ISSAC'05) and, with A. Enge (Inria, LIX) and A. Sedoglavic (UST Lille, LIFL), of the *Journées Nationales de Calcul Formel 2005* (Luminy, November 21-25, 2005).
- J.-M. Muller is member of the steering committee of the *IEEE Symposium on Computer Arithmetic* (ARITH). He has been a member of the program committee of ARITH17.
- N. Revol co-organizes the Dagstuhl seminar *Reliable Implementation of Real Number Algorithms: Theory and Practice*, January 2006.
- A. Tisserand organized the ARCHI05 winter school on *Computer Architecture*.
- G. Villard is chair of the steering committee of the *International Symposium on Symbolic and Algebraic Computation* (2003-2006). He was member of the program committee of *Computer Algebra in Scientific Computing* 2006; he is member of the program committee of *Transgressive Computing* 2006.

General public meetings:

- M. Daumas visited a junior school during the 2005 solar eclipse. He interacted with two classes, first on the black board and later for a safe seeing.
- N. Revol visited high-schools in the region of Lyon. She gave an interview for the on-line magazine *L'Internaute* (http://www.linternaute.com/femmes/carriere/0501metiers-d-homme/index.shtml).

9.2. Doctoral School Teaching

- 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 / 2006).
- M. Daumas gives a 15h Master course at the Université Montpellier 2 with D. Defour "Algorithms and Architectures of Computer Arithmetic" (2005 / 2006).
- F. de Dinechin gives a 30h ÉNSL Master course "Hardware Arithmetic Operators" (2004 / 2005, 2005 / 2006).
- C.-P. Jeannerod gives a 30h ÉNSL Master course "Algorithms for Computer Arithmetic" (2005 / 2006).
- C.-P. Jeannerod and N. Revol organized a course of the Doctoral School MATHIF, "Applications of Computer Science to Research and Technological Development".
- J.-M. Muller gives a 30h ÉNSL Master course "Floating-point Arithmetic" (2005 / 2006).
- J.-M. Muller gives a 30h Master course "Computer Arithmetic" at *Université Claude Bernard - Lyon 1* (2004 / 2005 / 2006).
- A. Tisserand gives a 30h ÉNSL Master course "Digital Integrated Circuits" (2004 / 2005).
- G. Villard has been the head of the ÉNSL Master 2 *Informatique Fondamentale* until July 2005.

9.3. Other Teaching and Service

- S. Boldo, N. Boullis, J. Detrey, G. Melquiond and N. Veyrat-Charvillon are teaching assistants—*moniteurs*—they give courses at the ÉNS and INSA.
- F. de Dinechin teaches *Computer Architecture* and *Computer Science for Non-Computer Scientists* in Licence and *Compilation* in Master at ÉNSL.
- C.-P. Jeannerod has been examiner for the ÉNS admissions.

9.4. Leadership within Scientific Community

- M. Daumas is a member of the board of the CNRS Nationwide Initiative GDR ARP to become ASR.
- M. Daumas received from CNRS-STIC an incentive grant to promote emerging subjects at the interface between mathematics and computer science. Participating laboratories were Lirmm, Lix, Liens, LP2A, Lip, Lip6, Laco, A2X, Loria and Inria Sophia-Antipolis. A call for projects has been issued and after discussion, the grant was spent to support five projects in 2005-2006.
- J.-M. Muller is head of the Lip laboratory (joint laboratory (UMR) of CNRS, École Normale Supérieure de Lyon, Inria and Université Claude bernard/Lyon 1 - about 90 persons).
- A. Tisserand installed and maintained the computers and softwares of CAD tools for the Lip laboratory up to September 2005.
- G. Villard will be the vice-head of the Lip laboratory starting mid-2006.

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.
- G. Villard was in the PhD Advisory Committee as *rapporteur* for the work of S. Graillat (U. Perpignan, Nov. 2005), and of D. Stehlé (U. Nancy, Dec. 2005).

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

The team members regularly give talks at the Department Seminar and at other French Institutions Seminars, in 2005: Beuchat (LIRMM, Montpellier); Brisebarre (Univ. Montpellier and Univ. Lille); Melquiond (École polytechnique and Université de Perpignan); Villard (Inria ALGO Rocquencourt).

National meetings:

- J.-L. Beuchat, N. Brisebarre, G. Melquiond, J.-M. Muller gave a talk at the *Journées Arinews*, Lyon, France, May 2005.
- F. Cháves, J. Detrey, F. de Dinechin, C. Lauter, S.-K. Raina and N. Veyrat-Charvillon gave a talk at the *Journées Arinews*, Perpignan, France, November 2005.
- G. Villard gave a talk at the *33rd Theoretical Computer Science Spring School, Computational Complexity*, Montagnac-les-truffes, France, June 2005.

International:

- Both J.-L. Beuchat and J. Detrey gave a talk at the Los Alamos National Laboratory, New Mexico, USA, November 2005.
- N. Brisebarre gave a talk at the Slovak Academy of Sciences.
- F. de Dinechin was an invited researcher for three month (Feb-Apr 2005) at the Intel Nizhniy Novgorod Lab (Russia), where he gave two seminars. He was also invited to give a talk at CERN (Centre Européen de Recherche Nucléaire) in Geneva, January 2005. He was also invited to give a talk at the ACAT workshop (Advanced Computing and Analysis Techniques in Physics Research) in Zeuthen, May 2005.
- G. Melquiond gave an invited talk at the Intel Hillsboro Lab, Oregon, U.S.A., June 2005.
- N. Revol was an invited speaker at the 76th Annual Meeting of GAMM (Gesellschaft für Angewandte Mathematik und Mechanik) in Luxemburg, April 2005 (and had to cancel due to pregnancy).
- A. Tisserand has an invited tutorial on “Algorithms and Number Systems for Hardware Computer Arithmetic” at the ISSAC 2005 Conference, Beijing, China, July 2005.
- G. Villard has participated to 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] N. BRISEBARRE, J.-M. MULLER, A. TISSERAND. *Computing machine-efficient polynomial approximations*, in "ACM Transactions on Mathematical Software", to appear.

[2] 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^o 1-4, dec 2004, p. 367–375.

[3] J. DETREY, F. DE DINECHIN. *Parameterized floating-point logarithm and exponential functions for FPGAs*, in "Journal of Microprocessors and Microsystems", to appear.

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

[5] E. KALTOFEN, G. VILLARD. *On the complexity of computing determinants*, in "Computational Complexity", vol. 13, n^o 3-4, 2005, p. 91–130.

[6] J.-M. MULLER. *Elementary Functions: Algorithms and Implementation*, Second, Birkhäuser, Boston, 2006.

[7] N. REVOL, K. MAKINO, M. BERZ. *Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY*, in "Journal of Logic and Algebraic Programming", vol. 64, 2005, p. 135–154.

[8] N. REVOL, F. ROUILLIER. *Motivations for an arbitrary precision interval arithmetic and the MPFI library*, in "Reliable Computing", vol. 11, n^o 4, 2005, p. 275–290.

[9] A. TISSERAND. *Low-Power Arithmetic Operators*, in "Low Power Electronics Design", C. PIGUET (editor). , chap. 9, CRC Press, November 2004.

[10] F. DE DINECHIN, A. ERSHOV, N. GAST. *Towards the post-ultimate libm*, in "17th Symposium on Computer Arithmetic", IEEE Computer Society Press, June 2005.

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

Articles in refereed journals and book chapters

[12] B. BECKERMANN, G. LABAHN, G. VILLARD. *Normal forms for general polynomial matrices*, in "Journal of Symbolic Computation", to appear, 2005.

[13] N. BOULLIS, A. TISSERAND. *Some Optimizations of Hardware Multiplication by Constant Matrices*, in "IEEE Transactions on Computers", vol. 54, n^o 10, October 2005, p. 1271–1282.

[14] N. BRISEBARRE, D. DEFOUR, P. KORNERUP, J.-M. MULLER, N. REVOL. *A new range reduction algorithm*, in "IEEE Transactions on Computers", vol. 54, n^o 3, 2005, p. 331–339.

[15] N. BRISEBARRE, J.-M. MULLER. *Correct Rounding of Algebraic Functions*, in "RAIRO Theoretical Informatics and Applications", to appear, 2005.

[16] N. BRISEBARRE, G. PHILIBERT. *Effective lower and upper bounds for the Fourier coefficients of powers of the modular invariant j*, in "J. Ramanujan Math. Soc.", vol. 20, n^o 4, 2005, p. 255–282.

[17] A. DARTE, R. SCHREIBER, G. VILLARD. *Lattice based memory allocation*, in "IEEE Transactions on Computers", vol. 54, n^o 10, 2005, p. 1242–1257.

[18] 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, 2005.

[19] P. KORNERUP, J.-M. MULLER. *Choosing Starting Values for Certain Newton-Raphson Iterations*, in "Theoretical Computer Science", 2004.

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

[21] 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", 2004.

[22] F. DE DINECHIN, C. Q. LAUTER, J.-M. MULLER. *Fast and correctly rounded logarithms in double-precision*, in "Theoretical Informatics and Applications", to appear, 2005.

[23] F. DE DINECHIN, G. VILLARD. *High precision numerical accuracy in physics research*, in "Nuclear Inst. and Methods in Physics Research, A", to appear, 2006.

Publications in Conferences and Workshops

[24] R. BEGUENANE, J.-L. BEUCHAT, J.-M. MULLER, S. SIMARD. *Modular Multiplication of Large Integers on FPGA*, in "39th Asilomar Conference on Signals, Systems & Computers", IEEE Signal Processing Society, November 2005.

[25] J.-L. BEUCHAT, J.-M. MULLER. *Multiplication Algorithms for Radix-2 RN-Codings and Two's Complement Numbers*, in "Proceedings of the 16th IEEE International Conference on Application-Specific Systems, Architectures, and Processors", S. VASSILIADIS, N. DIMOPOULOS, S. RAJOPADHYE (editors). , IEEE Computer Society, 2005, p. 303–308.

[26] J.-L. BEUCHAT, J.-M. MULLER. *RN-codes : algorithmes d'addition, de multiplication et d'élévation au carré*, in "SympA'2005: 10 è m e édition du SYMPosium en Architectures nouvelles de machines", April 2005, p. 73–84.

[27] S. BOLDO, G. MELQUIOND. *When double rounding is odd*, in "Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics, Paris, France", 2005.

[28] S. BOLDO, J.-M. MULLER. *Some Functions Computable with a Fused-mac*, in "Proc. 17th IEEE Symposium on Computer Arithmetic (ARITH-17), Cape Cod, USA", P. MONTUSCHI, E. SCHWARZ (editors). , 2005, p. 52–58, http://arith17.polito.it/program.html.

[29] N. BRISEBARRE, J.-M. MULLER. *Correctly rounded multiplication by arbitrary precision constants*, in "Proc. 17th IEEE Symposium on Computer Arithmetic (ARITH-17)", IEEE Computer Society Press, June 2005.

[30] S. CHEVILLARD, N. REVOL. *Computation of the error functions erf and erfc in arbitrary precision with correct rounding*, in "17th IMACS Conf. on Scientific Computation, Applied Math. and Simulation, Paris, France", July 2005.

[31] M. DAUMAS, G. MELQUIOND, C. MUÑOZ. *Guaranteed proofs using interval arithmetic*, in "Proceedings of the 17th IEEE Symposium on Computer Arithmetic, Cape Cod, Massachusetts, USA", P. MONTUSCHI, E. SCHWARZ (editors). , 2005, p. 188–195.

[32] J. DETREY, F. DE DINECHIN. *A Parameterizable Floating-Point Logarithm Operator for FPGAs*, in "39th Asilomar Conference on Signals, Systems & Computers", IEEE Signal Processing Society, November 2005.

[33] J. DETREY, F. DE DINECHIN. *A Parameterized Floating-Point Exponential Function for FPGAs*, in "IEEE International Conference on Field-Programmable Technology (FPT'05)", IEEE Computer Society Press, December 2005.

[34] J. DETREY, F. DE DINECHIN. *Table-based polynomials for fast hardware function evaluation*, in "16th Intl Conference on Application-specific Systems, Architectures and Processors", IEEE Computer Society Press, July 2005.

[35] M. D. ERCEGOVAC, J.-M. MULLER, A. TISSERAND. *Simple Seed Architectures for Reciprocal and Square Root Reciprocal*, in "Proc. 39th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, U.S.A.", October 2005.

[36] C.-P. JEANNEROD, S.-K. RAINA, A. TISSERAND. *High-Radix Floating-Point Division Algorithms for Embedded VLIW Integer Processors*, in "Proc. 17th World Congress on Scientific Computation, Applied Mathematics and Simulation IMACS, Paris, France", July 2005.

[37] P. KORNERUP, J.-M. MULLER. *RN-coding of numbers: definition and some properties*, in "Proceedings of the 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Paris", July 2005.

[38] G. MELQUIOND, S. PION. *Formal certification of arithmetic filters for geometric predicates*, in "Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics, Paris, France", 2005.

[39] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *Divgen: a divider unit generator*, in "Proc. Advanced Signal Processing Algorithms, Architectures and Implementations XV, San Diego, California, U.S.A.", F. T. LUK (editor). , vol. 5910, SPIE, August 2005, 59100M.

[40] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *Evaluation de polynômes et de fractions rationnelles sur FPGA avec des opérateurs à additions et décalages en grande base*, in "10ième SYMPosium en Architectures nouvelles de machines (SYMPA), Le Croisic", April 2005, p. 85–96.

[41] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *Small FPGA polynomial approximations with 3-bit coefficients and low-precision estimations of the powers of x*, in "Proc. 16th International Conference on Application-specific Systems, Architectures and Processors (ASAP), Samos, Greece", S. VASSILIADIS, N. DIMOPOULOS, S. RAJOPADHYE (editors). , Best Paper Award, IEEE Computer Society, July 2005, p. 334–339.

[42] R. MICHARD, A. TISSERAND, N. VEYRAT-CHARVILLON. *Étude statistique de l'activité de la fonction de sélection dans l'algorithme de E-méthode*, in "5ième journées d'études Faible Tension Faible Consommation (FTFC), Paris", May 2005, p. 61–65.

[43] J.-M. MULLER, A. TISSERAND, B. DUPONT DE DINECHIN, C. MONAT. *Division by Constant for the ST100 DSP Microprocessor*, in "Proc. 17th Symposium on Computer Arithmetic (ARITH), Cape Cod, MA., U.S.A", P. MONTUSCHI, E. SCHWARZ (editors). , IEEE Computer Society, June 2005, p. 124–130.

[44] N. REVOL. *Bounding roundoff errors in Taylor models arithmetic*, in "17th IMACS Conf. on Scientific Computation, Applied Math. and Simulation, Paris, France", July 2005.

[45] A. STORJOHANN, G. VILLARD. *Computing the rank and a small nullspace basis of a polynomial matrix*, in "Proc. International Symposium on Symbolic and Algebraic Computation, Beijing, China", ACM Press, July 2005, p. 309–316.

[46] A. TISSERAND. *Algorithms and Number Systems for Hardware Computer Arithmetic*, in "International Symposium on Symbolic and Algebraic Computation (ISSAC), Beijing, China", Invited tutorial, July 2005.

[47] G. VILLARD. *Efficient algorithms in linear algebra*, 33rd Theoretical Computer Science Spring School, Computational Complexity, Montagnac-les-truffes, May 2005.

[48] F. DE DINECHIN, C. Q. LAUTER, G. MELQUIOND. *Assisted verification of elementary functions using Gappa*, in "Symposium on Applied Computing", to appear, 2005.

Internal Reports

[49] S. BOLDO, C. MUÑOZ. *A formalization of floating-point numbers in PVS*, Technical report, National Institute for Aerospace, 2005.

[50] H. BRÖNNIMANN, G. MELQUIOND, S. PION. *A Proposal to add Interval Arithmetic to the C++ Standard Library*, Technical report, n° 5646, C++ standardization committee, 2005, http://www.inria.fr/rrrt/rr-5646.html.

[51] C.-P. JEANNEROD, G. VILLARD. *Asymptotically fast polynomial matrix algorithms for multivariable systems*, Research report, n° ccsd-00008211, ArXiv cs.SC/0505030, Laboratoire de l'Informatique du Parallélisme, ÉNS Lyon, August 2005, http://hal.ccsd.cnrs.fr/ccsd-00008211.

[52] C. Q. LAUTER. *Basic building blocks for a triple-double intermediate format*, Technical report, n° RR2005-38, LIP, September 2005, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2005/RR2005-38.pdf.

[53] C. PERNET, A. RONDEPIERRE, G. VILLARD. *Computing the Kalman form*, Research report, n° ccsd-00009558, ArXiv cs.SC/0510014, IMAG, Grenoble, October 2005, http://hal.ccsd.cnrs.fr/ccsd-00009558.

Bibliography in notes

[54] 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.

[55] 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.

[56] D. GOLDBERG. *What every computer scientist should know about floating-point arithmetic*, in "ACM Computing Surveys", vol. 23, n° 1, 1991, p. 5-47, http://www.acm.org/pubs/articles/journals/surveys/1991-23-1/p5-goldberg/p5-goldberg.pdf.

[57] O. HERRMANN. *Über die Berechnung der Fourierkoeffizienten der Funktion $j(\tau)$*, in "J. Reine Angew. Math.", vol. 274/275, 1975, p. 187–195.

[58] 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.

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

[60] K. MAHLER. *On the coefficients of transformation polynomials for the modular function*, in "Bull. Austral. Math. Soc.", vol. 10, 1974, p. 197–218.

[61] P. L. MONTGOMERY. *Modular Multiplication without Trial Division*, in "Mathematics of Computation", vol. 44, n° 170, April 1985, p. 519–521.

[62] T. OGITA, S. RUMP, S. OISHI. *Accurate Sum and Dot Product*, in "SIAM Journal on Scientific Computing (SISC)", vol. 26, n° 6, 2005, p. 1955–1988.

[63] 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.

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

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