

Horizon 2020
Excellent Science
Call: ERC-2014-ADG
Topic: ERC-ADG-2014
Type of action: ERC-ADG
Proposal number: SEP-210208071
Proposal acronym: BeDoP

Table of contents

Section	Title	Action
1	General information	
2	Participants & contacts	
3	Budget	
4	Ethics	
5	Call-specific questions	

How to fill in the forms

The administrative forms must be filled in for each proposal using the templates available in the submission system. Some data fields in the administrative forms are pre-filled based on the previous steps in the submission wizard.



Proposal ID **669971**

Acronym **BeDoP**

1 - General information

Topic ERC-ADG-2014

Type of action ERC-ADG

Call identifier ERC-2014-ADG

Acronym*

Proposal title*

Note that for technical reasons, the following characters are not accepted in the Proposal Title and will be removed: < > " &

Duration in months*

Primary ERC Review Panel*

Secondary ERC Review Panel

(if applicable)

ERC Keyword 1*

Please select, if applicable, the ERC keyword(s) that best characterise the subject of your proposal in order of priority.

ERC Keyword 2

ERC Keyword 3

ERC Keyword 4

Free keywords



Proposal ID **669971**

Acronym **BeDoP**

Abstract

On May 25, 2008, IBM's Roadrunner supercomputer reached the petaflop milestone, i.e., 10 to the power 15 floating-point operations per second (flops). We anticipate that the exaflop milestone of 10 to the power 18 flops will be reached around 2020. Given that current hardware uses a double precision floating-point format of 53 bits corresponding to about 16 decimal digits, and that rounding errors usually increase linearly with the number of operations, we need to reconsider the final accuracy of results obtained on future exascale computers.

We strongly believe that (i) the scientific community has a blind confidence in double precision arithmetic, (ii) even today some scientific computations are fast but wrong, with some potential disastrous consequences, and (iii) in the near future the world will realize that double precision is not enough.

It is therefore our responsibility as computer scientists to (i) warn the scientific community about the limits of double precision for large computations, (ii) design or improve software tools that will enable scientists to go beyond double precision in the parts of their programs that require it, and (iii) make those software tools efficient and robust.

The BeDoP project will address these crucial issues by: (i) demonstrating the limits of double precision on large-scale applications, (ii) making multiple-precision tools easier to use in modern computer languages, and (iii) improving the efficiency and robustness of those tools, in particular by using formal proof techniques.

Our dream with the BeDoP project is that scientific computations on exascale computers will no longer give very fast and very wrong results, but instead give very fast and very accurate results.

Remaining characters

254

In order to best review your application, do you agree that the above non-confidential proposal title and abstract can be used, without disclosing your identity, when contacting potential reviewers?

Yes

No

Has this proposal (or a very similar one) been submitted in the past 2 years in response to a call for proposals under the 7th Framework Programme, Horizon 2020 or any other EU programme(s)?

Yes

No



Proposal ID **669971**

Acronym **BeDoP**

Declarations

1) The Principal Investigator declares to have the explicit consent of all applicants on their participation and on the content of this proposal.*	<input checked="" type="checkbox"/>
2) The information contained in this proposal is correct and complete.	<input checked="" type="checkbox"/>
3) This proposal complies with ethical principles (including the highest standards of research integrity — as set out, for instance, in the European Code of Conduct for Research Integrity — and including, in particular, avoiding fabrication, falsification, plagiarism or other research misconduct).	<input checked="" type="checkbox"/>
4) The Principal Investigator hereby declares that (<i>please select one of the three options below</i>):	
- in case of multiple participants in the proposal, the coordinator has carried out the self-check of the financial capacity of the organisation on http://ec.europa.eu/research/participants/portal/desktop/en/organisations/lfv.html . Where the result was “weak” or “insufficient”, the Principal Investigator confirms being aware of the measures that may be imposed in accordance with the H2020 Grants Manual (Chapter on Financial capacity check) .	<input type="checkbox"/>
- in case of multiple participants in the proposal, the Principal Investigator is exempt from the financial capacity check being a public body including international organisations, higher or secondary education establishment or a legal entity, whose viability is guaranteed by a Member State or associated country, as defined in the H2020 Grants Manual (Chapter on Financial capacity check) .	<input type="checkbox"/>
- in case of a sole participant in the proposal, the applicant is exempt from the financial capacity check.	<input checked="" type="checkbox"/>
5) The Principal Investigator hereby declares that each applicant has confirmed to have the financial and operational capacity to carry out the proposed action. Where the proposal is to be retained for EU funding, each beneficiary applicant will be required to present a formal declaration in this respect.	<input checked="" type="checkbox"/>
The Principal Investigator is only responsible for the correctness of the information relating to his/her own organisation. Each applicant remains responsible for the correctness of the information related to him and declared above. Where the proposal to be retained for EU funding, the coordinator and each beneficiary applicant will be required to present a formal declaration in this respect.	

According to Article 131 of the Financial Regulation of 25 October 2012 on the financial rules applicable to the general budget of the Union (Official Journal L 298 of 26.10.2012, p. 1) and Article 145 of its Rules of Application (Official Journal L 362, 31.12.2012, p.1) applicants found guilty of misrepresentation may be subject to administrative and financial penalties under certain conditions.

Personal data protection

Your reply to the grant application will involve the recording and processing of personal data (such as your name, address and CV), which will be processed pursuant to Regulation (EC) No 45/2001 on the protection of individuals with regard to the processing of personal data by the Community institutions and bodies and on the free movement of such data. Unless indicated otherwise, your replies to the questions in this form and any personal data requested are required to assess your grant application in accordance with the specifications of the call for proposals and will be processed solely for that purpose. Details concerning the processing of your personal data are available on the [privacy statement](#). Applicants may lodge a complaint about the processing of their personal data with the European Data Protection Supervisor at any time.

Your personal data may be registered in the Early Warning System (EWS) only or both in the EWS and Central Exclusion Database (CED) by the Accounting Officer of the Commission, should you be in one of the situations mentioned in:

- the Commission Decision 2008/969 of 16.12.2008 on the Early Warning System (for more information see the [Privacy Statement](#)), or
- the Commission Regulation 2008/1302 of 17.12.2008 on the Central Exclusion Database (for more information see the [Privacy Statement](#)).



Proposal ID **669971**

Acronym **BeDoP**

2 - Administrative data of participating organisations

Host Institution

PIC	Legal name
999547074	INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Short name: INRIA

Address of the organisation

Street Domaine de Voluceau, Rocquencourt

Town LE CHESNAY Cedex

Postcode 78153

Country France

Webpage www.inria.fr

Legal Status of your organisation

Research and Innovation legal statuses

Public body yes

Legal person yes

Non-profit yes

International organisation no

International organisation of European interest no

Secondary or Higher education establishment no

Research organisation yes

Small and Medium-sized Enterprises (SMEs) no

Nace code 72 - Computer & related activities



Proposal ID **669971**

Acronym **BeDoP**

Department(s) carrying out the proposed work

Department 1

Department name

Street

Town

Postcode

Country

Same as organisation address



Proposal ID **669971**

Acronym **BeDoP**

Principal Investigator

The following information of the Principal Investigator is used to personalise the communications to applicants and the evaluation reports. Please make sure that your personal information is accurate and please inform the ERC in case your e-mail address changes by using the call specific e-mail address:

For Advanced Grant Applicants: ERC-2014-AdG-applicants@ec.europa.eu

The name and e-mail of contact persons including the Principal Investigator, Host Institution contact are read-only in the administrative form, only additional details can be edited here. To give access rights and contact details of contact persons, please save and close this form, then go back to Step 4 of the submission wizard and save the changes.

Researcher ID	<input type="text" value="If you have a researcher identifier number (e.g. ResearcherID, ORCID) please enter it here."/>		
Last Name*	<input type="text" value="Zimmermann"/>	Last Name at Birth	<input type="text" value="Zimmermann"/>
First Name(s)*	<input type="text" value="Paul"/>	Gender*	<input checked="" type="radio"/> Male <input type="radio"/> Female
Title	<input type="text" value="Dr."/>	Country of residence*	<input type="text" value="France"/>
Nationality*	<input type="text" value="France"/>	Country of Birth*	<input type="text" value="France"/>
Date of Birth* (DD/MM/YYYY)	<input type="text" value="13/11/1964"/>	Place of Birth	<input type="text" value="SAINT AVOLD"/>

Contact address

Same as organisation address

Current organisation name	<input type="text" value="Inria - Centre de Recherche Nancy Grand Est"/>		
Current Department/Faculty/Institute/ Laboratory name	<input type="text" value="CAMEL Team"/>		
Street*	<input type="text" value="Rue du Jardin Botanique, 615"/>		
Postcode/Cedex*	<input type="text" value="54600"/>	Town*	<input type="text" value="Villers Les Nancy"/>
Phone*	<input type="text" value="+33383593041"/>	Country*	<input type="text" value="France"/>
Phone2 / Mobile	<input type="text" value="+XXXX XXXXXXXXXXXXX"/>		
E-mail	<input type="text" value="paul.zimmermann@inria.fr"/>		

Qualifications

Earliest award (PhD, Doctorate)	Date of award (DD/MM/YYYY)	<input type="text" value="06/03/1991"/>
---------------------------------	----------------------------	---



Proposal ID **669971**

Acronym **BeDoP**

Contact address of the Host Institution and contact person

The name and e-mail of Host Institution contact persons are read-only in the administrative form, only additional details can be edited here. To give access rights and contact details of Host Institution, please save and close this form, then go back to Step 4 of the submission wizard and save the changes. Please note that the submission is blocked without a contact person and e-mail address for the Host Institution.

Organisation Legal Name **INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE**

First name* **Armelle**

Last name* **Demange**

E-Mail* **polecaf-nancy@inria.fr**

Position in org.

Department

Street

Same as organisation address

Town

Postcode

Country

Phone

Phone2/Mobile



Proposal ID **669971**

Acronym **BeDoP**

3 - Budget

Participant Number in this proposal	Organisation Short Name	Organisation Country	Total eligible costs/€ (including 25% indirect costs) ?	Requested grant/€
1	INRIA	FR	2 302 781	2 302 781
Total			2 302 781	2 302 781

Proposal ID **669971**

Acronym **BeDoP**

4 - Ethics issues table

1. HUMAN EMBRYOS/FOETUSES		Page
Does your research involve Human Embryonic Stem Cells (hESCs) ?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Does your research involve the use of human embryos?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Does your research involve the use of human foetal tissues / cells?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
2. HUMANS		Page
Does your research involve human participants?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Does your research involve physical interventions on the study participants?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Does it involve invasive techniques?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
3. HUMAN CELLS / TISSUES		Page
Does your research involve human cells or tissues (other than from Human Embryos/ Foetuses, i.e. section 1)?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
4. PERSONAL DATA (ii)		Page
Does your research involve personal data collection and/or processing?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Does your research involve further processing of previously collected personal data (secondary use)?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
5. ANIMALS (iii)		Page
Does your research involve animals?	<input type="radio"/> Yes <input checked="" type="radio"/> No	

Proposal ID **669971**

Acronym **BeDoP**

6. THIRD COUNTRIES		Page
Does your research involve non-EU countries?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Do you plan to use local resources (e.g. animal and/or human tissue samples, genetic material, live animals, human remains, materials of historical value, endangered fauna or flora samples, etc.)? (v)	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Do you plan to import any material from non-EU countries into the EU? <i>For data imports, please fill in also section 4. For imports concerning human cells or tissues, fill in also section 3.</i>	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Do you plan to export any material from the EU to non-EU countries? <i>For data exports, please fill in also section 4. For exports concerning human cells or tissues, fill in also section 3.</i>	<input type="radio"/> Yes <input checked="" type="radio"/> No	
If your research involves low and/or lower middle income countries , are benefits-sharing measures foreseen? (vii)	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Could the situation in the country put the individuals taking part in the research at risk?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
7. ENVIRONMENT & HEALTH and SAFETY		Page
See legal references at the end of the section. (vi)		
Does your research involve the use of elements that may cause harm to the environment, to animals or plants? <i>For research involving animal experiments, please fill in also section 5.</i>	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Does your research deal with endangered fauna and/or flora and/or protected areas?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Does your research involve the use of elements that may cause harm to humans, including research staff? <i>For research involving human participants, please fill in also section 2.</i>	<input type="radio"/> Yes <input checked="" type="radio"/> No	
8. DUAL USE (vii)		Page
Does your research have the potential for military applications?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
9. MISUSE		Page
Does your research have the potential for malevolent/criminal/terrorist abuse?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
10. OTHER ETHICS ISSUES		Page
Are there any other ethics issues that should be taken into consideration? Please specify	<input type="radio"/> Yes <input checked="" type="radio"/> No	

I confirm that I have taken into account all ethics issues described above and that, if any ethics issues apply, I will complete the ethics self-assessment and attach the required documents.



Proposal ID **669971**

Acronym **BeDoP**

5 - Call specific questions

Eligibility	
I acknowledge that I am aware of the eligibility requirements for applying for this ERC call as specified in the ERC Work Programme 2014, and certify that, to the best of my knowledge my application is in compliance with all these requirements. I understand that my proposal may be declared ineligible at any point during the evaluation or granting process if it is found not to be compliant with these eligibility criteria.*	<input checked="" type="checkbox"/>
Data-Related Questions and Data Protection (Consent to any question below is entirely voluntary. A positive or negative answer will not affect the evaluation of your project proposal in any form and will not be communicated to the evaluators of your project.)	
For communication purposes only, the ERC asks for your permission to publish your name, the proposal title, the proposal acronym, the panel, and host institution, should your proposal be retained for funding.	<input checked="" type="radio"/> Yes <input type="radio"/> No
Some national and regional public research funding authorities run schemes to fund ERC applicants that score highly in the ERC's evaluation but which can not be funded by the ERC due to its limited budget. In case your proposal could not be selected for funding by the ERC do you consent to allow the ERC to disclose the results of your evaluation (score and ranking range) together with your name, non-confidential proposal title and abstract, proposal acronym, host institution and your contact details to such authorities?	<input checked="" type="radio"/> Yes <input type="radio"/> No
The ERC is sometimes contacted for lists of ERC funded researchers by institutions that are awarding prizes to excellent researchers. Do you consent to allow the ERC to disclose your name, non-confidential proposal title and abstract, proposal acronym, host institution and your contact details to such institutions?	<input checked="" type="radio"/> Yes <input type="radio"/> No
The Scientific Council of the ERC has developed a monitoring and evaluation strategy in order to help it fulfil its obligations to establish the ERC's overall strategy and to monitor and quality control the programme's implementation from the scientific perspective. As provided by section 3.10 of the ERC Rules for Submission, a range of projects and studies may be initiated for purposes related to monitoring, study and evaluating the implementation of ERC actions. Do you consent to allow the third parties carrying out these projects and studies to process the content of your proposal including your personal data and the respective evaluation data? The privacy statement on grants (http://erc.europa.eu/document-library) explains further how your personal data is secured.	<input checked="" type="radio"/> Yes <input type="radio"/> No



Proposal ID **669971**

Acronym **BeDoP**

Excluded Reviewers

You can provide up to three names of persons that should not act as an evaluator in the evaluation of the proposal for potential competitive reasons.

ERC Advanced Grant Research Proposal (Part B1)

Beyond Double Precision (BeDoP)

Principal Investigator (PI): Dr Paul Zimmermann
PI's host institution: Inria, France
Proposal full title: Beyond Double Precision
Proposal short name: BeDoP
Project duration: 60 months
Targeted Review Panel: PE6 (Computer Science and Informatics)

Proposal Abstract

On May 25, 2008, IBM's Roadrunner supercomputer reached the petaflop milestone, i.e., 10^{15} floating-point operations per second (flops). We anticipate that the **exaflop milestone** of 10^{18} flops will be reached around 2020. Given that current hardware uses a double precision floating-point format of 53 bits corresponding to about 16 decimal digits, and that **rounding errors** usually increase linearly with the number of operations, we need to reconsider the **final accuracy** of results obtained on future exascale computers.

We strongly believe that (i) the scientific community has a **blind confidence** in double precision arithmetic, (ii) even today some scientific computations are **fast but wrong**, with some potential **disastrous consequences**, and (iii) in the near future the world will realize that **double precision is not enough**.

It is therefore **our responsibility** as computer scientists to (i) **warn the scientific community** about the limits of double precision for large computations, (ii) design and improve software tools that will enable scientists to go **beyond double precision** in the parts of their programs that require it, and (iii) make those software tools **efficient and robust**.

The BeDoP project will address these **crucial issues** by: (i) demonstrating the **limits of double precision** on large-scale applications, (ii) making multiple-precision tools **easier to use** in modern computer languages, and (iii) improving the efficiency and robustness of those tools, in particular by using **formal proof** techniques.

Our dream with the BeDoP project is that scientific computations on exascale computers will no longer give **very fast and very wrong results**, but instead give **very fast and very accurate results**.

Extended Synopsis of the scientific proposal

1 Motivation and State of the Art

Most numerical computations performed nowadays use standard **double precision** arithmetic provided by hardware. Each year, as the hardware becomes faster, and as processors include more computing cores, the number of flops increases. At the same time, since each operation induces a tiny roundoff error, and as we perform larger and larger computations, the relative roundoff error of each computation increases. If nothing changes, in the near future we are at risk of computing **very fast but very wrong**. *We are driving faster and faster cars, but with tyres limited to 100 miles per hour: at some point we will crash.* One example of such a disaster due to insufficient precision is the Patriot anti-ballistic missile failure during Gulf War in 1991, with the death of 28 soldiers [10]: the constant $1/10$ was approximated with a 24-bit register, which resulted in a constant drift which became significant after a few days.

As already pointed out by the European Exascale Software Initiative [1], *the ability to perform floating-point arithmetic with different precisions (e.g., 32-, 64-, and 128-bit) will likely be necessary in Exascale systems. [...] The fundamental challenge of library software design is to develop and provide robust and reliable algorithms and implementations that deliver accurate results or at least compute results with accuracy estimates.* Several applications already require more than double precision [2]: evolution of the solar system over billions of years, supernova simulations, climate modeling, studying the fine structure constant of physics, ... In most of those applications, the need for larger precision is limited to a small part of the program, thus is not time-critical. However, the accuracy of this small part is **critical for the accuracy of the final result**.

The accuracy of floating-point computations heavily relies on small building blocks like the following theorem (which is a simplified version of a more general one):

Theorem. *Let x be a 5-digit decimal floating-point number. Convert x to the nearest q -bit binary floating-point number, say y . Convert back y to the nearest 5-digit decimal floating-point number, say z . Then if $q \geq 18$ we have $z = x$.*

For example, if we convert $x = 3.1415$ to the nearest 18-bit binary floating-point number, we find $y = 205881/2^{16} \approx 3.141495$. Then if we convert back y to the nearest 5-digit decimal floating-point number, we find x again. For $x = 3.1415$, a binary precision of 15 bits is enough in fact. But for $x = 8.0003$, the nearest 17-bit approximation is $y = 32769/2^{12} \approx 8.000244$, which rounds back to 8.0002; this demonstrates that the $q \geq 18$ bound is optimal. Proving such theorems is not enough, since we also have to ensure that the conversions from decimal to binary and from binary to decimal both return the nearest number in the target radix. This is why **formal proof technology** applied to the **real end-user code** becomes crucial.

Several researchers have proposed ways to overcome the **limitations of double precision** and the **dangers of human proofs**. On the one hand, interval arithmetic is a way to be warned when the uncertainty becomes of the same order of magnitude as the value itself; double-double arithmetic enables one to get almost quadruple precision with a slowdown of a factor 2 to 10 with respect to double precision. Finally, arbitrary precision is the ultimate solution. Several software tools provide some of those techniques, but they are not sufficiently used to identify critical parts of current programs where double precision is not enough. On the other hand, formal proof techniques have been used successfully in the last 20 years in several domains, such as proving the four-color theorem and the Feit-Thomson theorem about the classification of finite groups. This was possible thanks to the advent of formal proof environments like HOL, PVS, and Coq. In the domain of computer arithmetic itself, Harrison and Russinoff were hired by Intel and AMD, respectively, to use formal proof techniques to check the design of hardware

chips after the Pentium FDIV bug, which cost Intel \$500 million in 1994 [5, 9]. Boldo and Melquiond designed the Flocq library for proving floating-point algorithms, which is intensively used by the CompCert verified compiler designed by Leroy and his team [4]; people around Filiâtre designed the Why3 platform for deductive program verification.

Using arbitrary precision arithmetic is not enough if your arbitrary precision code is not correct. Using formal proof techniques is not enough if the accuracy of the final result is not sufficient. To avoid future exascale computations giving **very fast but very wrong** results, we must **warn the scientific community** about the limits of double precision, and we need to investigate **arbitrary precision** and **formal proof techniques** in order to build **accurate and bug-free numerical tools for exascale computations**.

2 Grand Challenge and Research Targets

Our Grand Challenge is to investigate next-generation arbitrary precision floating-point algorithms and software tools, which will be required not only to be fast but also bit-accurate and bug-free.

It is very unlikely that hardware will provide more than double precision in the next ten years because the next standard format would be quadruple precision, which would require an increase by a factor of 2 to 4 in the number of gates in the chip, and a corresponding slowdown of arithmetic operations. To fill this gap, only software solutions can avoid a big crash when we hit the **limiting wall of double precision accuracy**, assuming we avoid the reliability wall for Mean Time Between Failures (MTBF) [11]. However, those software tools need to be **efficient**, otherwise they won't be used by the scientific community, and **free of design or implementation bugs** or we will hit the wall in a different way. The BeDoP project will be the first one to address the formal proof of floating-point arbitrary precision.

Our first scientific objective is **formalising low-level arbitrary precision floating-point arithmetic**; this is **Target 1**. Our second scientific objective addresses **quadruple precision floating-point arithmetic**, which is the next standard format beyond double precision; this is **Target 2**. Our third scientific objective is to address **generic arbitrary precision floating-point arithmetic**, which will be useful for all applications where quadruple precision is not enough; this is **Target 3**. Both Targets 2 and 3 will depend on Target 1, since they will use the algorithms and implementations designed in Target 1. However, Targets 2 and 3 will be independent.

Targets 1, 2, and 3 are described in more detail below. Targets 2 and 3 will include validation on large scale computations.

Research Target 1: Formalising Low-Level Arbitrary Precision Floating-Point Arithmetic

The Heartbleed bug, discovered on April 1st, 2014 in the OpenSSL implementation of the Secure Sockets Layer (SSL) protocol, affected about half a million of the world's "secure" web servers. Libraries such as OpenSSL rely on multiple-precision integer arithmetic for cryptographic computations, e.g., RSA signature or encryption. It is thus of utmost importance to check that such multiple-precision implementations are free of bugs, especially "out-of-bound reads" like the Heartbleed bug which might leak some data to the attackers.

- **Design a language (called MPS) of multiple-precision integer routines that is sufficient for floating-point applications. This will include the basic arithmetic**

operations on multiple-precision integers (addition, subtraction, multiplication, division, square root, and radix conversion), with a well-defined interface in the C language.

- **Formally prove, using a proof assistant, the correctness of MPS routines, both at the C-language level for those written in C, and at the assembly language level for those written in assembly.**

Previous work includes a proof in Coq of a square root algorithm by Bertot, Magaud and the PI in 2002 [3]. The formal proof will have to take into account the memory management of input, output, and temporary objects for each routine.

A binary floating-point number can be represented as $m \cdot 2^e$, with m and e integers, m being the *significand* and e the *exponent*. MPS routines will be used as building blocks to compute efficiently on the significands of floating-point numbers in Targets 2 and 3. Therefore, Target 1 is crucial for attacking Targets 2 and 3.

Research Target 2: Formalising Quadruple-Precision Arithmetic

The IEEE 754 standard, revised in 2008, defines three binary floating-point formats: `binary32` with a significand of 24 bits (single precision), `binary64` with a significand of 53 bits (double precision), and `binary128` with a significand of 113 bits (quadruple precision). Only the single and double precision formats are implemented on current hardware. The standard also defines which operations should be *correctly rounded*, i.e., should return the unique number in the output format that is closest to the (infinite precision) exact mathematical result. The following table gives the average number of cycles for different operations in double and quadruple precision on an Intel i5-4570 processor, with MPFR 3.1.2, GMP 6.0.0 and GCC 4.8.2.

precision	add	sub	mul	div	sqrt
double (53 bits)	39	49	37	130	159
quadruple (113 bits)	41	53	56	144	264

Our goal is to save a significant factor on the average number of cycles, using code simplifications for the specific case of quadruple precision.

- **Design a quadruple-precision floating-point library which both returns the best possible — i.e., correctly rounded — results, and is much faster than existing libraries.**
- **Formally prove, using a proof assistant, that this library returns correctly rounded values. Both the algorithms (at the mathematical level) and their implementation (in the C language) will be considered by the proof.**

This quadruple-precision library will be designed on top of the MPS language designed in Target 1. Thus, its proof will rely on the formal proof of the corresponding MPS routines. It is thus crucial for the success of Target 2 that the interface to the MPS routines is fixed once for all. However, the correctness of the MPS routines might be considered as axiomatic here (assuming no flaw is found later on).

Research Target 3: Formalising Arbitrary-Precision Arithmetic

While Target 2 addresses specifically quadruple precision, some large scale applications will require larger precision, or operations with mixed precisions. Therefore, we need a generic arbitrary precision library, allowing mixed-precision operations, which is both efficient and has guaranteed correct rounding.

Implementing such a library leads to several technical difficulties. For example, the significand has to be split among several machine words, which usually store 32 or 64 bits each. Contrary to the hardware case, both the precision and the exponent might be huge here.

- **Design arbitrary-precision floating-point routines on top of MPS routines. This will include basic arithmetic operations (addition, subtraction, multiplication, division), square root, mathematical functions (sin, exp, log).**
- **Formally prove, using a proof assistant, that those arbitrary-precision floating-point routines return correctly rounded values for any input values.**

For most existing arbitrary-precision software tools, which have no formal specification, a formal proof makes no sense. This is the main originality (and difficulty) of this project: to design efficient bit-accurate arbitrary-precision routines, and formally prove their correctness. As in Target 2, the routines we will design here will heavily depend on the MPS routines designed in Target 1, and thus the formal specification of MPS routines will enable to share the formal proof work between Target 1 and Target 3.

3 Methodology and Organisation

The detailed BeDoP work plan (including work packages) is presented in Part B2 of this proposal. We sketch here the dependencies between the Research Targets.

The very first task of the BeDoP project will be to carefully design the MPS language. We need a minimal language that enables to easily implement routines for Targets 2 and 3, but also enables an efficient implementation in C or assembly language. Since the whole project will depend on MPS routines, the necessary time will be dedicated to that task.

Once the MPS language is fixed, all other tasks can start: the real implementation of MPS routines, in C or assembly language, the formal proof of this implementation; the design of quadruple-precision routines and their formal proof (Target 2); the design of arbitrary-precision routines and their formal proof (Target 3).

4 Risk Assessment and Management

The originality of the BeDoP project is to make a bridge between two scientific domains: arbitrary-precision arithmetic and formal proof theory. Given that one of the PI's main research topics is arbitrary-precision arithmetic (both on integers and floating-point numbers), the risk is quite limited in this part of the project.

On the formal proof side, the PI is already well aware of the main theoretical and practical aspects of proof assistant systems [3], and through well chosen cooperations he will keep aware of the latest progress in this very active topic. He will recruit postdoctoral researchers and PhD students who are knowledgeable in formal proof systems and have good skills in computer arithmetic. Two such specialists have already been hired by the host institution (Inria) at the Nancy centre, a research engineer (Stéphane Glondu) and a research associate (Jasmin Blanchette, who will start in 2015). Both of them will contribute to the BeDoP project. Finally, the Coq proof assistant is now a mature tool, and several researchers at Inria are available to help on technical issues related to Coq, among them Xavier Leroy and his group.

5 Expected Impact

Scientific Impact. We expect this project will reveal several **bugs** in existing libraries like GMP, or MPFR. Indeed, when a routine contains a bug, the formal proof will necessarily fail.

Thus, we will investigate why it fails, and at some point we will discover the bug. We also expect this will lead to **new original research** at the interface between computer arithmetic and formal proof, in particular about memory management issues. This will lead to **new algorithms** and **a new formal proof methodology**. And of course, the output of this project will be **reference implementations** that are **formally proven**, and thus can be used in **critical applications on exascale computers**.

Societal Impact. In [6, Table 1], He and Ding report very different results in an ocean circulation model, where they obtain numerical values ranging from 0.32 to 34.41, and conclude: *In fact, we do not know what is the exact correct result [...] On different number of processors, the order of summation is not guaranteed, and the results are not reproducible!*

We want to avoid similar issues. We want to avoid computing **very fast but very wrong**. We want numerical computations to be **reproducible** from one exascale computer to another one. Therefore, we need **quality assurance** about the accuracy of numerical values computed by **exascale computers**. The BeDoP project will raise the awareness of the scientific community that double arithmetic is not enough in some applications, and will investigate bug-free software tools for the **next-generation floating-point arithmetic**.

6 Commitment of the PI

The PI will dedicate at least 75% of his work time to the BeDoP project.

References

- [1] European Exascale Software Initiative: Working group report on numerical libraries, solvers and algorithms. Available from <http://www.eesi-project.eu/>, 2011. 60 pages.
- [2] BAILEY, D. H., BARRIO, R., AND BORWEIN, J. M. High-precision computation: Mathematical physics and dynamics. *Applied Mathematics and Computation* 218 (2012), 10106–10121.
- [3] BERTOT, Y., MAGAUD, N., AND ZIMMERMANN, P. A proof of GMP square root. *Journal of Automated Reasoning* 29 (2002), 225–252. Special Issue on Automating and Mechanising Mathematics: In honour of N.G. de Bruijn.
- [4] BOLDO, S., JOURDAN, J.-H., LEROY, X., AND MELQUIOND, G. A formally-verified C compiler supporting floating-point arithmetic. In *21st IEEE Intern. Symp. on Comp. Arithmetic* (2013), pp. 107–115.
- [5] HARRISON, J. A machine-checked theory of floating point arithmetic. In *Proceedings of 12th International Conference on Theorem Proving in Higher Order Logics* (1999), Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, Eds., pp. 113–130.
- [6] HE, Y., AND DING, C. H. Q. Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications. *The Journal of Supercomputing* 18 (2001), 259–277.
- [7] MÜLLER, N. T., AND UHRHAN, C. Some steps into verification of exact real arithmetic. In *Proceedings of NFM'2012* (2012), no. 7226, pp. 168–173.
- [8] MYREEN, M. O., AND CURELLO, G. Proof Pearl: A verified bignum implementation in x86-64 machine code. In *Proceedings of Certified Programs and Proofs* (2013), vol. 8307, pp. 66–81.
- [9] RUSSINOFF, D. M. A mechanically checked proof of IEEE compliance of the floating point multiplication, division and square roots algorithms of the AMD-K7 processor. *LMS Journal of Computation and Mathematics* 1 (1998), 148–200.
- [10] VUIK, K. Some disasters caused by numerical errors. <http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html>.
- [11] YANG, X., WANG, Z., XUE, J., AND ZHOU, Y. The reliability wall for exascale supercomputing. *IEEE Transactions on Computers* 61, 6 (2012), 767–779.

7 Curriculum Vitae

Personal Information. Paul Zimmermann, born 13/11/1964.
Email Paul.Zimmermann@inria.fr, home page <http://www.loria.fr/~zimmerma>.

Education.

- Habilitation (highest French academic degree), Nancy, France, 2001.
- PhD in Computer Science from École Polytechnique, Palaiseau, France, 1991.
- Master in Computer Science, University Paris VII, France, 1988.
- Engineer from École Polytechnique (a major French high school), Palaiseau, France, 1987.

Positions.

- 1988-present: Researcher at Inria, 1988-1993 in Rocquencourt, 1993-present in Nancy (1988-1998 Junior Researcher, 1998-2008 Research Director, 2008-present Senior Research Director).
- 1994-1995: Sabbatical in the MuPAD group, Paderborn, Germany.
- 1987-1991: Master and PhD Grant from DRET, Inria, Rocquencourt, France.
- 1984-1987: Engineer Student at École Polytechnique, Palaiseau, France.

Fellowships and Awards.

- Holder of “Prix La Recherche”, France, 2012, for the record factorisation of RSA-768.
- Winner of the Many Digits competition, Nijmegen, Netherlands, 2005.

Supervision of Graduate Students and Postdoctoral Fellows.

- 1994-present: supervised 5 PhD students (all as sole advisor). Among the four who already defended, one has been hired by Google, and one is Professor at ÉNS Lyon, France.
- 1997-present: supervised 3 postdoctoral students.

Teaching Activities. 1992-present: about 300 teaching hours at different levels (Master in Computer Science, engineering schools) and different thematics (computer algebra, algorithmic number theory) in Paris and Nancy, France. In particular, the PI created a new course on Algorithmic Number Theory, Coding and Cryptography in the computer science Masters in Nancy (2000-2005), and in 2005-2006 he created a new course “Introduction to Cryptology” in this Masters.

Organisation of Scientific Meetings. Co-organized a workshop on discrete tomography, Pont-à-Mousson, France, 1999; a workshop on open-source computer algebra in Lyon, France, 2002; the RNC’7 (Real Numbers and Computers) conference in Nancy, France, 2006; the Sage Days 10 and the CADO workshop on integer factorisation in Nancy, France, 2008; and the Ninth Algorithmic Number Theory Symposium (ANTS-IX), Nancy, France, 2010. Organized the *Fast Algorithms* track at the workshop *Computing by the Numbers: Algorithms, Precision, and Complexity* for the 60th birthday of Richard Brent, Berlin, Germany, 2006.

Institutional Responsibilities and Research Leadership.

- 2013-present: Scientific Director and Chair of the Projects Committee at the Inria-Nancy research centre (21 research teams and 175 scientists).
- 2011-2014: Elected member of the Inria Scientific Board.
- 2011-2012: Head of a team of 8 tenure track engineers, Inria Nancy, France.
- 1999-2001 and 2005-2007: Elected member of the Inria Evaluation Committee.
- Since his arrival in Nancy in 1993, the PI was at the origin of several research teams on discrete

mathematics and algorithmic number theory. Several full-time researchers were recruited in those teams, where a dozen PhD or postdoctoral students were trained. The PI was in particular at the origin of a joint project involving several French research teams on reliable computer arithmetic (1999-2000).

Commissions of Trust.

- Member of the ARITH program committee in 2001 (ARITH'15), 2003 (ARITH'16), 2005 (ARITH'17), 2007 (ARITH'18) and 2009 (ARITH'19).
- Member of the program committee of AfricaCrypt in 2010, and of ISSAC in 2013.
- Program co-chair of the RNC'7 conference in Nancy, France, 2006.

International Recognition. Invited presentations at:

- *Computational Number Theory* workshop at the *Foundations of Computer Mathematics* conference in Oxford, UK, 1999;
- SCAN conference in Paris, France, 2002 (main conference on interval arithmetic);
- PARI/GP workshop, Paris, France, 2004;
- IEEE 754 revision committee, Silicon Valley, USA, 2005 and 2006;
- *Grand Challenges of Informatics* conference Budapest, Hungria, 2006;
- *Algorithmic Number Theory* conference, Turku, Finland, 2007;
- colloquium in honour of Henri Cohen, Bordeaux, France, 2007;
- *Central European Conference on Cryptography*, Graz, Austria, 2008;
- MSR Talk Series, Microsoft Research, Redmond, USA, 2009;
- International Congress on Mathematical Software, Kobe, Japan, 2010;
- Euroscipy 2013 conference (advanced tutorial), Bruxelles, 2013.

Invited to write an entry on the Elliptic Curve Method in the *Encyclopedia of Cryptography and Security*, Springer, 2005;

Invited to write an article in the *Notices of the American Mathematical Society*, 2011.

Major Collaborations.

- Coordinator in 1997 of a German-French project Procope with the MuPAD group in Paderborn, Germany, led by Prof. Benno Fuchssteiner.
- Head of an associate team co-funded by the University of Canberra and Inria with the group of Richard Brent, 2008-2010.

Past Funding. The development of the MPFR library was supported by Inria in several forms (ARC Fiable 1999-2000, ARC AOC 2000-2002, engineer grants 2003-2005 and 2007-2009, post-doctoral grant 2009-2010) and by the “Conseil Régional de Lorraine” (2002). The PI was a main participant of the CADO and CATREL projects supported by the French National Research Agency (ANR). On-going funding and submitted proposals are to be found in the Appendix.

8 Tenure Track-Record

The publication list reflects the three domains of the PI research: analysis of algorithms and computer algebra [1, 3, 7, 9], computer arithmetic [2, 4, 8, 10], and the applications of number theory to cryptography [5, 6]. References [2, 8] are the more relevant ones for the BeDoP project.

Top 10 Paper Publications. The main publications are written with co-authors, because new ideas are first discussed with colleagues to improve them and only then published. Almost all of these publications correspond to important developments or experimental work. The PI's contribution to all articles is comparable to that of the co-authors, except for [2] where the PI was the main designer and author. The number of citations are from Google Scholar, and were measured on September 24th, 2014, with corresponding *h*-index 26.

1. *Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable*, with B. Salvy, ACM Transactions on Mathematical Software, 1994. 317 citations.
2. *MPFR: A multiple-precision binary floating-point library with correct rounding*, with L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, ACM Transactions on Mathematical Software, 2007. 314 citations.
3. *A calculus for the random generation of labelled combinatorial structures*, with Ph. Flajolet and B. Van Cutsem, Theoretical Computer Science, 1994. 275 citations.
4. *Efficient isolation of polynomial's real roots*, with F. Rouillier, Journal of Computational and Applied Mathematics, 2004. 236 citations.
5. *Factorization of a 768-bit RSA modulus*, with T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele and A. Timofeev, Crypto'2010, 2010. 210 citations.
6. *Factorization of a 512-bit RSA modulus*, with S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, Ch. Putnam and Cr. Putnam, Eurocrypt'2000, 2000. 170 citations.
7. *Automatic average-case analysis of algorithms*, with Ph. Flajolet and B. Salvy, Theoretical Computer Science, 1991. 127 citations.
8. *Modern computer arithmetic*, with R. P. Brent, Cambridge University Press, 2010. 99 citations.
9. *Uniform random generation of decomposable structures using floating-point arithmetic*, with A. Denise, Theoretical Computer Science, 1999. 70 citations.
10. *The middle product algorithm I*, with G. Hanrot and M. Quercia, Applicable Algebra in Engineering, Communication and Computing, 2004. 55 citations.

Top 5 Software Publications. As a computer scientist, the PI cannot consider his research without writing programs or libraries either to experiment with a new idea, as a proof-of-concept of a new algorithm, or as a general tool which will be useful to himself, to his research team, or to other people. Most of his software contributions are distributed under an open-source licence to allow other people to use them in other tools, either free or commercial:

- 1998-present: main designer and main author of MPFR, a library for multiple-precision floating-point arithmetic with correct rounding. Shipped within all Linux distributions. Prerequisite to build the GCC and Gfortran compilers. Used by Magma, Sage, and the MPFI and MPC libraries.
- 2002-present: main designer and main author of MPC, a library for multiple-precision complex floating-point arithmetic with correct rounding. Shipped within all Linux distributions. Prerequisite to build the GCC compiler.
- 2005-present: main designer and main author of CADO-NFS, an integer factorisation program using the Number Field Sieve.
- 1998-present: main designer and main author of GMP-ECM, an integer factorisation program using the Elliptic Curve Method (ECM). GMP-ECM holds the record of the largest prime found with ECM (83 digits).
- several contributions to the GMP library for arbitrary-precision arithmetic: the “Karatsuba square root” algorithm, the REDC-based modular powering code, the original Schönhage-Strassen FFT code (and an improved version with Gaudry and Kruppa in 2007), an improved Toom-Cook multiplication code, and a new n th root code.

Invited Presentations (selection of).

- SCAN conference in Paris, France, 2002 (main conference on interval arithmetic)
- IEEE 754 revision committee, Silicon Valley, USA, 2005 and 2006;
- *Grand Challenges of Informatics* conference Budapest, Hungria, 2006
- *Algorithmic Number Theory* conference, Turku, Finland, 2007
- *Central European Conference on Cryptography*, Graz, Austria, 2008
- MSR Talk Series, Microsoft Research, Redmond, USA, 2009
- International Congress on Mathematical Software, Kobe, Japan, 2010
- Euroscipy 2013 conference (advanced tutorial), Bruxelles, 2013

Conference (Co-)Organisation.

- *Fast Algorithms* track at the conference for the 60th birthday of Richard Brent, Berlin, Germany, 2006
- Sage Days 10, Nancy, France, 2008
- CADO-NFS workshop on integer factorisation, Nancy, France, 2008
- Ninth Algorithmic Number Theory Symposium (ANTS-IX), Nancy, France, 2010

Major contributions to the early careers of excellent researchers. A PhD student supervised by the PI (Laurent Fousse) is now hired by Google at the Mountain View headquarters; another PhD student (Damien Stehlé) is now full Professor in Lyon and obtained an ERC Starting Grant in 2013.

9 Appendix: on-going funding and submitted proposals

The PI is currently involved in the CATREL project *Sieve Algorithms: Theoretical Advances, and Effective Resolution of the Discrete Logarithm Problem*. This project funded by the French National Research Agency (ANR) started in January 2013 and will end in December 2015 (<http://catrel.loria.fr>). The PI involvement in this project is 42% (15 months over 36). This is the only on-going grant of the PI, and there is no submitted proposal.

To be able to dedicate enough time to the BeDoP project, in agreement with Inria's management, the PI will resign from his responsibilities of Scientific Director and Chair of the Projects Committee of the Inria-Nancy research centre as soon as the BeDoP project will start.

ERC Advanced Grant Research Proposal (Part B2)

Beyond Double Precision (BeDoP)

Principal Investigator (PI): Dr Paul Zimmermann
PI's host institution: Inria, France
Proposal full title: Beyond Double Precision
Proposal short name: BeDoP
Project duration: 60 months
Targeted Review Panel: PE6 (Computer Science and Informatics)

Reproducible research is now a big concern in the scientific community (see for example the special issue of Nature [1] and the workshop organised at ICERM in 2012 [39]). Indeed, *numerical reproducibility* and accuracy have been identified as critical concerns by the International Exascale Software Project (IESP) [17] and by the computer arithmetic community [14, 18, 25]:

*Exascale systems will be advanced scientific instruments. As part of the scientific process, scientists need to know how the devices work for **scientific reproducibility and accuracy**. Treating the system software as a black box run by code that cannot be examined or verified does not accomplish this goal.*

In [8] and [27], high-precision arithmetic is considered to be **crucial for applications in climate modelling, astrophysics, mathematical physics and dynamics**. At the European level, the European Exascale Software Initiative (EESI) reports [4]:

*The ability to perform floating-point arithmetic with different precisions (e.g., 32-, 64-, and 128-bit) **will likely be necessary in Exascale systems**. [...] The fundamental challenge of library software design is to develop and provide **robust and reliable algorithms and implementations that deliver accurate results** or at least compute results with accuracy estimates.*

The BeDoP project will precisely address this fundamental challenge. In Section 1 we describe the state of the art concerning technologies and software tools, and we state our Grand Challenge. In Section 2 we detail the three Research Targets we need to focus on, in order to solve this Grand Challenge. The corresponding resources are described in Section 3.

1 State of the art and Scientific Objectives

We review here the main existing technologies relevant to the BeDoP project (binary floating-point formats, double-double arithmetic, interval arithmetic) and corresponding software tools. For an exhaustive description, we refer the reader to the habilitation thesis of Sylvie Boldo [12].

Let us mention that the NSF (National Science Foundation) supports similar projects in the United States, like that of Michela Taufer (University of Delaware) on *Studying the impact of rounding errors on result reproducibility when concurrent executions burst and workflow determinism vanishes in cutting-edge multicore architectures*.

1.1 Binary Floating-Point: the IEEE 754 Standard

The IEEE 754 standard defines how binary and decimal floating-point arithmetic should be performed. We will only focus on binary formats here, which are the formats used in scientific applications. Originally designed in 1985, the IEEE 754 standard was revised in 2008; we refer to this revision as IEEE 754-2008 [24]. The standard defines three main binary formats: `binary32` (single precision), `binary64` (double precision) and `binary128` (quadruple precision), with significands of 24 bits, 53 bits and 113 bits respectively.

The IEEE 754-2008 standard defines five rounding attributes (sometimes also called rounding modes): `(round)TiesToEven`, `TiesToAway`, `TowardPositive`, `TowardNegative`, `TowardZero`. The last three (directed) modes round to the nearest floating-point number in the direction of $+\infty$, $-\infty$ and 0 respectively; `TiesToEven` rounds to the nearest representable floating-point number below or above, and if both such numbers are equally distant, it rounds to the unique one whose last bit is 0; `TiesToAway` is similar except in that special case it rounds away from zero (see Table 1). Current environments (compilers and operating systems) usually provide support for four rounding modes — without `roundTiesToAway` —, as in the C language through the `fenv.h` header. IEEE 754-2008 requires *correct rounding* for each of the available rounding modes and arithmetic operations. Correct rounding means that for a given mathematical function, say `exp`, and a given floating-point input, say x , the implementation returns the closest floating-point number y to the infinite precision value $\exp(x)$ according to the given rounding mode, for example $y = \text{TiesToEven}(\exp x)$. Therefore, there is a *unique* possible value y , which is called the *correct*

t	<code>TiesToEven(t)</code>	<code>TiesToAway(t)</code>	<code>TowardPositive(t)</code>	<code>TowardNegative(t)</code>	<code>TowardZero(t)</code>
2.4	2	2	3	2	2
-2.5	-2	-3	-2	-3	-2
2.6	3	3	3	2	2

Table 1: Correct rounding with a target of 2 bits for the IEEE 754-2008 rounding modes.

rounding of $\exp(x)$. For mathematical functions, correct rounding is only *recommended* by IEEE 754-2008; some libraries (for example the GNU `libc`) have started to provide correct rounding for some functions, but we are far from correct rounding for all supported mathematical functions and all possible inputs. In current processors, the only binary IEEE 754 formats that have been implemented in hardware are single precision (`binary32`) and double precision (`binary64`). The `x86` and `x86_64` processors also provide a 80-bit format called *double extended* (`long double` in the C language) with a 64-bit significand.

GCC Quadruple-Precision Library. Since 2008, GCC provides a quadruple precision data type (`__float128`) with basic arithmetic operations ($+$, $-$, \times , \div). Since 2011, mathematical functions are available for the `__float128` type through the `libquadmath` library, which originates from the FDLIBM library developed by Sun around 1993. While GCC seems to guarantee correct rounding for the four basic arithmetic operations and the fused-multiply-add, it does not do so for the mathematical functions. For example, with version 4.9.1 of GCC, the square-root function `sqrtq` with input 2.0 gives an output which is off by one ulp (unit in last place). For other functions, similar discrepancies from correct rounding can be found, ranging from one ulp

to hundreds of thousands ulps. Table 2 compares the speed of different kinds of arithmetic and different precisions.

53 bits	double:	0.54s	MPFR: 43.5s (ratio 81 over double)
64 bits	long double:	2.9s	MPFR: 53.2s (ratio 18 over long double)
113 bits	__float128:	38.2s	MPFR: 47.1s (ratio 1.2 over __float128)

Table 2: Time for multiplying two 1000×1000 matrices, with GCC -O3 (version 4.9.1) on a 3.2Mhz Intel Core i5-4570.

Arbitrary-Precision. Arbitrary-precision floating-point arithmetic is available in most computer algebra systems (Maple, Mathematica, Sage) and in special-purpose libraries (MPFUN, Pari/GP, ARPREC, MPF within GMP, NTL, CLN, MPFR, ARB). MPFR guarantees correct rounding for all mathematical functions it implements, following IEEE 754-2008 [19]. ARB provides *ball arithmetic*, also known as mid-rad arithmetic, i.e., it returns tight intervals $[y - \varepsilon, y + \varepsilon]$ which are guaranteed to contain the exact result. A comparison of the efficiency of those libraries for 100, 1000 and 10000 digits shows that MPFR is usually the fastest library [40].

1.2 Double-Double Arithmetic

Double-double arithmetic enables the accuracy of the hardware double precision format to be doubled, with a slowdown of a factor 2 to 10 with respect to hardware [15, 34]. A double-double number is the simplest case of a *floating-point expansion* [32], which is the approximation of a real number x by a sum $t_1 + t_2 + \dots + t_n$ of floating-point numbers, ordered by decreasing magnitude. In the double-double case, $x \approx t_1 + t_2$, where t_1, t_2 are double-precision numbers. The precision of a double-double number is at least twice that of double precision, i.e., 106 bits. Efficient algorithms exist to add, subtract, multiply and divide two double-double numbers, using the hardware for efficiency, such as Knuth’s TwoSum algorithm [26] and Dekker’s TwoProd algorithm [16].

The QD package, developed by the group of David Bailey (Lawrence Berkeley National Laboratory), includes routines to perform “double-double” and “quad-double” arithmetic. However, QD is mainly designed for speed, and no guarantee is given concerning the maximal roundoff error made in each arithmetic operation. Moreover, since it relies on double-precision arithmetic, double-double arithmetic (and thus QD) suffers from the exponent limitation of this arithmetic, and therefore cannot handle numbers larger than about 10^{308} or smaller than 10^{-324} , which is a severe limitation, especially in combinatorics, for example.

1.3 Interval Arithmetic

In interval arithmetic, each arithmetic operation is performed on an interval $[a, b]$ instead of a scalar, where both a and b are floating-point numbers. For example, when adding two intervals $[a, b]$ and $[c, d]$, interval arithmetic guarantees that the result $[e, f]$ satisfies the *inclusion property*:

$$\forall x \in [a, b], \quad \forall y \in [c, d], \quad e \leq x + y \leq f.$$

Interval arithmetic always return an interval containing the exact mathematical result (as would be obtained with infinite precision), and in most cases it is enough to prove the desired properties.

For example if the result interval $[e, f]$ is $[0.3, 0.7]$, we are sure that the corresponding value $x + y$ is positive. However, interval arithmetic is known to produce wide intervals, in particular when there are dependencies between variables in the same expression. A simple example is the expression $(x - 1)(x + 1)$ on the interval $[-0.4, 0.6]$. If evaluated as such, one obtains $[-1.4, -0.4] \times [0.6, 1.6]$, which yields $[-2.24, -0.24]$. However, if computed as $x^2 - 1$, the same expression gives $[-1, -0.64]$, which is the smallest interval satisfying the inclusion property. A working group sponsored by IEEE is currently developing a standard for interval arithmetic [2]. An implementation of interval arithmetic in arbitrary-precision is MPFI, also available within the Sage computer algebra system. As for issues with interval arithmetic in parallel computations, we refer the reader to [33].

1.4 Formalisation of Floating-Point Arithmetic

Hardware Level. After the Pentium FDIV bug, which cost Intel \$500 million in 1994, both Intel and AMD hired engineers, respectively Harrison and Russinoff, to produce a formal proof of the algorithms implemented on chip before manufacturing them. Harrison used HOL Light [22], whereas Russinoff used ACL2 [35]. Harrison also developed a complete proof of a full implementation of the exponential function in double precision.

More recently, Russinoff verified the SRT quotient and square root algorithms [36], and while doing so he revealed several errors in previous work by Kornerup published in 2005 in *IEEE Transactions on Computers*.

Software Level. Several tools exist at the software level, we mention here the most relevant ones for the BeDoP project. The Frama-C platform, designed by CEA and Inria, is an extensible and collaborative platform dedicated to source-code analysis of C software (part of the STANCE multi-disciplinary initiative [37]). Astree is a C program analyzer using abstract interpretation. Fluctuat is another tool, developed by CEA (France), and dedicated to the analysis of floating-point programs. Sollya is a library for safe floating-point code development. Gappa is a tool intended to help verify and formally prove properties of numerical programs dealing with floating-point and fixed-point arithmetic. Flocq (Floats for Coq) is a floating-point formalisation for the Coq system; it provides a comprehensive library of theorems on multi-radix multi-precision arithmetic; it also supports efficient numerical computations inside Coq. Melquiond formalised IEEE 754 arithmetic in the Coq system, and was able to prove such theorems [29]:

```
Theorem Fdiv_correct :
  forall radix mode prec (x y : float radix), 1 < radix ->
  Fdiv mode prec x y = round radix mode prec (x / y).
```

Table 3 gives a summary of the status of current implementations with respect to correct rounding and formal proof. For double precision, usually only the four basic operations are implemented in hardware, and the mathematical functions (exp, log, sin, ...) are computed in software (for example by the GNU libc for the C language under Linux). For quadruple precision with the `__float128` type in the C language, the basic arithmetic operations are provided by the GCC runtime (the `libgcc` library), and seem to be correctly rounded. The mathematical functions provided by `libquadmath` are not correctly rounded: some errors up to hundreds of thousands units in last place were observed. The MPFR library provides correct rounding for all operations and mathematical functions, but only a paper proof is provided [41].

implementation	precision	correct rounding	formal proof
hardware $+$, $-$, \times , \div	53 bits	yes	yes
libc math. functions	53 bits	no	no
libgcc $+$, $-$, \times , \div	113 bits	yes	no
libquadmath math. functions	113 bits	no	no
MPFR $+$, $-$, \times , \div	arbitrary	yes	no
MPFR math. functions	arbitrary	yes	no

Table 3: Status of current implementations with respect to correct rounding and formal proof.

In short, the Grand Challenge of the BeDoP project is to replace all “no” entries by “yes” in Table 3 beyond double precision, i.e., for quadruple and arbitrary-precision. Several difficulties have to be overcome to reach this Grand Challenge, which will require (i) expertise of researchers in computer arithmetic, (ii) expertise of researchers in formal proof, and (iii) fruitful interactions between both domains of research. The detailed steps that we believe will allow us to solve this Grand Challenge are described in the next section.

Our Grand Challenge is to investigate next-generation arbitrary-precision floating-point algorithms and software tools, which will be required not only to be fast, but also bit-accurate and bug-free.

2 Methodology: The BeDoP Scientific Roadmap

It is very unlikely that hardware will provide more than double precision in the next ten years because the next standard format (quadruple precision) would require an increase by a factor of 2 to 4 in the number of gates in the chip, and a corresponding slowdown of arithmetic operations. To fill this gap, only software solutions can avoid a big crash when we hit the **limiting wall of double precision accuracy**. However, those software tools need to be **efficient**, otherwise they won’t be used by the scientific community, and **free of design or implementation bugs** or we will hit the wall in a different way. The BeDoP project will be the first one to address the formal proof of floating-point arbitrary-precision.

Our first scientific objective is **formalising low-level arbitrary-precision floating-point arithmetic**; this is **Target 1**. Our second scientific objective addresses **quadruple precision floating-point arithmetic**, which is the next standard format beyond double precision; this is **Target 2**. Our third scientific objective is to address **generic arbitrary-precision floating-point arithmetic**, which will be useful for all applications where quadruple precision is not enough; this is **Target 3**.

Below, we describe in more detail the three Research Targets we need in order to address our Grand Challenge. Both Targets 2 and 3 will depend on Target 1, since they will use the algorithms and implementations designed in Target 1. However, Targets 2 and 3 will be independent. We give at the end a timing roadmap over the 5 years of the project.

2.1 Research Target 1: Formalising Low-Level Arbitrary-Precision Floating-Point Arithmetic

When adding or multiplying two integers, the result depends on all input bits. This is no longer the case when adding or multiplying floating-point numbers. For example, when adding a large number with a tiny number, the latter might contribute only in the last bit of the result. When multiplying two floating-point numbers of one million bits, if their product is wanted to only 100 bits, there is no need to read all the input bits. Therefore, low-level routines for integer arithmetic — like in the `mpn` layer of GMP [21] — are not directly suitable for floating-point arithmetic. This is why we propose to design a new language (called MPS for Multiple-Precision Significand) of low-level routines for arbitrary-precision floating-point arithmetic. This is Research Target 1. This will be completely pioneering, since such a language does not exist so far. We expect this language will also form a foundation for the design of other multiple-precision floating-point libraries, with or without correct rounding.

We propose to split the work on Research Target 1 into the following tasks that we describe in more detail below:

- Task RT1-1: Design the MPS Language Interface
- Task RT1-2: Efficient Implementation of the MPS Language
- Task RT1-3: Formally Prove the Correctness of the MPS Implementation

Task RT1-1: Design the MPS Language Interface

Design, with a well-defined interface in the C language, a new language (called MPS) of low-level routines operating on floating-point significands and exponents, the former being represented by arrays of machine integers, the latter by machine integers.

The MPS language might contain for example the `mps_add` routine, which adds (in place) to the significand array starting at pointer a (with precision of p bits) the significand starting at pointer b (with precision of q bits) shifted by k bits toward the least significant bits. In addition, `mps_add` puts into `rnd` and `stck` the values of the round and sticky bits, respectively.¹

```
void mps_add (void *a, long p, const void *b, long q, long k, int *rnd, int *stck)
```

Similar routines will be needed to deal with floating-point exponents (dealing with underflows and overflows), and with correct rounding (taking as input a rounding bit and a sticky bit).

Task RT1-2: Efficient Implementation of the MPS Language

Design an efficient implementation of MPS routines in the C language, and also assembly language for x86_64 targets when needed. This implementation should reach the efficiency of the GMP reference library [21].

With the running example of the `mps_add` function above, its implementation might use the `mpn_add` function of GMP to add the significand of `b` to that of `a`, possibly after a shift using the GMP `mpn_lshift` or `mpn_rshift` functions. For computing the round and sticky bits, some bit manipulation functions at the machine integer level will be required. To ease the implementation and the proof, those bit manipulations routines will also be formalised in Task RT1-1.

¹The round bit is the next bit after the most significant p bits of $a + b/2^k$, and the sticky bit is zero if all further bits are zero, and one otherwise. Knowing them is sufficient to compute a correct rounding for all IEEE 754 rounding modes.

Task RT1-3: Formally Prove the Correctness of the MPS Implementation

Prove formally the correctness of the code designed in Task RT1-2. The theorem corresponding to the above routine `mps_add` could be as follows in the Coq proof assistant:

```
Theorem mps_add_correct :
  forall a p b q k rnd stck mem mem',
  mem' = eval mem (mps_add a p b q k rnd stck) ->
  let c_exact = value mem a p + value mem b q / 2^k in
  no_overlap a p b q -> value mem' a p = round (c_exact p) /\
  is_round_bit (value mem' rnd) c_exact /\
  is_sticky_bit (value mem' stck) c_exact.
```

In the above theorem, the variables `mem` and `mem'` represent the memory state before and after the call to `mps_add`, respectively. One of the main difficulties will be to prove that no change was made to memory, except of course in this example to the memory pointed by `a`, and to the variables `rnd` and `stck`. This will ensure that no out-of-bound read or write occurs.

Here, as in Tasks RT2-2 and RT3-2 below, different ways to obtain a formal proof are possible. A first strategy is to formalise the implementation in a proof assistant, and to prove the correctness of this formalisation; the issue here is to ensure that the formalisation really corresponds to the actual code [6, 31]. A second strategy is to extract the code directly from a proof assistant; in this way, no discrepancy can happen between the formalisation and the actual code, but specific knowledge is required to obtain efficient extracted code. Also, code extraction in the Coq system currently only works for Objective Caml, Haskell and Scheme, whereas our target is the C language. Some interaction with the Coq developers will be needed at this point. A third strategy is to annotate the source code, and to use a proof assistant (for example the Why3 platform [11]) which can read those annotations and prove lemmas and theorems from them. We expect all three strategies will need to be tested and compared.

Task RT1-3 will overlap in time with Task RT1-2, since interaction between both tasks will be required to ensure the implementation designed in Task RT1-2 can be formally proven.

2.2 Research Target 2: Formalising Quadruple-Precision Arithmetic

The quadruple-precision format is fully specified by IEEE 754: the significand has 113 bits, and valid numbers are of the form (in binary)

$$(-1)^s \cdot b_0.b_1\dots b_{112} \cdot 2^e,$$

with $s, b_i \in \{0, 1\}$, and $-16382 \leq e \leq 16383$. This format can represent numbers as large as about 10^{4932} , and as small (in absolute value) as about 10^{-4966} . Quadruple-precision numbers can be represented in 128 bits: one bit for the sign s , 15 bits for the exponent, and the remaining 112 bits for the significant (most numbers can be normalised so that $b_0 = 1$, which is then implicit). Additional special numbers are NaN (Not-a-Number), $+\infty$ and $-\infty$.

To formalise quadruple-precision arithmetic, we will split the work for Research Target 2 into the following tasks which are detailed below:

- Task RT2-1: Efficient Quadruple-Precision Routines
- Task RT2-2: Formal Proof of the Quadruple-Precision Routines
- Task RT2-3: Validate Quadruple-Precision Routines on Large-Scale Applications

Task RT2-1: Efficient Quadruple-Precision Routines

Design efficient quadruple-precision routines on top of MPS routines (which might contain special-purpose code for 113-bit significands). This will be split in two subtasks: basic arithmetic routines (addition, subtraction, multiplication, division, square root, radix conversion), and mathematical functions from the C11 standard. For basic arithmetic routines, the correct rounding can usually be obtained directly [13, Chapter 10], and the corresponding formal proof will be similar to the one obtained for the integer square root by the PI and collaborators [10]. However, for most mathematical functions, a single-pass computation is possible only when it is possible to determine the *hardest-to-round* inputs for the given format and function. This is known as the *Table Maker's Dilemma* (TMD). In general, the hardest-to-round inputs for a format of p bits require $2p + o(1)$ bits of working precision; the main difficulty is to determine exactly the $o(1)$ term. Either the TMD is solved, and one then knows an exact upper bound, say $2p + 17$, on the internal precision needed; otherwise one has to resort to Ziv's onion peeling strategy [13]: compute a first approximation with say $p + 10$ bits of precision if p is the target precision, and if this approximation does not enable one to compute the correct rounding, compute again with say $p + 20$ bits of precision, and so on. Since current methods can solve the TMD only up to double-extended precision [38], we will have to use Ziv's strategy here.

Two ways of reaching this milestone will be considered: either building on the existing `libquadmath` library from GCC, or building a quadruple precision library on top of MPFR. In the former case, the BeDoP project will contribute to the `libquadmath` library, by making the mathematical functions always return a correctly rounded value, for example:

```
__float128 sinq (__float128 x)
```

In the MPFR case, some specific code will need to be developed in the case where all operands are quadruple precision numbers. Note that in both cases, Ziv's strategy will require to implement an internal library with about 256-bit accuracy. The general framework of the code for a given mathematical function (say the sine function here) might be as follows: given a quadruple precision number x (with a significand of 113 bits),

1. set the working precision p to 128,
2. compute an approximation y of $\sin x$ with working precision p , and maximal error ε ,
3. if $y \pm \varepsilon$ are both rounded to the same quadruple precision value z , return z ,
4. otherwise $p \leftarrow p + 64$, and goto Step 1.

Task RT2-2: Formal Proof of the Quadruple-Precision Routines

Prove formally the correctness of the code designed in Task RT2-1. In particular, we will prove that the obtained results are *correctly rounded*, and that the code does not loop forever. We expect several interactions will be needed between Tasks RT2-1 and RT2-2, until we obtain an implementation that is both efficient and easy to prove correct. The expected output will be theorems as follows:

```
Theorem sinq_correct :
  forall (x : binary128) rnd_mode,
    sinq x rnd_mode = round (sin x) binary128 rnd_mode.
```


Consider the general framework outlined above to compute $\sin x$ in quadruple precision. The first subtask is to prove that the approximation y and the error bound ε in Step 2 really satisfy $|y - \sin x| \leq \varepsilon$. A second subtask is to prove that the rounding routine in Step 3 is correct (this routine will be shared by all mathematical functions). Finally, we have to prove that if the algorithm terminates, the returned value z is the correct rounding of $\sin x$ with the given rounding mode. As for termination, this is a non-trivial mathematical problem, which is (to the best of our knowledge) still unsolved for some mathematical functions. Therefore, we might have to assume termination in some cases.

As in Task RT1-3, different proof strategies will be tested and compared: manual proof, code extraction, and code annotation.

Task RT2-3: Validate Quadruple-Precision Routines on Large-Scale Applications

All the developments within the BeDoP project will be systematically validated on large-scale applications. Such applications will be identified at the beginning of the project, and new developments will be continuously exercised on large computers. For quadruple precision, we will focus on dense linear algebra using parallel hierarchical linear solvers [7].

2.3 Research Target 3: Formalising Arbitrary-Precision Arithmetic

We propose to split that Research Target into the following tasks:

- Task RT3-1: Efficient Arbitrary-Precision Routines
- Task RT3-2: Formal Proof of Arbitrary-Precision Routines
- Task RT3-3: Validate Arbitrary-Precision Routines on Large-Scale Applications

Task RT3-1: Efficient Arbitrary-Precision Routines

Design efficient arbitrary-precision routines on top of MPS routines. As for Task RT2-1, there will be two subtasks: first to implement basic arithmetic routines, then mathematical functions. For the set of functions considered, we will focus on those in the C11 standard, but will also consider other functions, for example the hyperbolic sine integral, also called the “Shi function”. If implemented in MPFR, this would yield the following function `mpfr_shi`, that computes the hyperbolic sine integral of the input `x`, and stores into the output `y` the correctly rounded result according to the rounding mode `rnd_mode`:

```
int mpfr_shi (mpfr_t y, const mpfr_t x, mpfr_rnd_t rnd_mode)
```

Task RT3-2: Formal Proof of Arbitrary-Precision Routines

Prove formally the correctness of the code designed in Task RT3-1. As in Tasks RT1-3 and RT2-2, several proof strategies will be tested and compared: manual proof, code extraction, and code annotation.

The expected outcome of this task will be theorems like the following, which guarantees that for any arbitrary-precision input `x`, the hyperbolic sine integral implementation `mpfr_shi` returns the correctly rounded output `y`:

Theorem `shi_correct` :

```
forall (x : arbitrary_fp) (y : arbitrary_fp) rnd_mode mem mem',
mem' = eval mem (mpfr_shi y x rnd_mode) ->
y = round (shi x) (precision y) rnd_mode.
```

Task RT3-3: Validate Arbitrary-Precision Routines on Large-Scale Applications

As for quadruple precision, the multiple-precision routines will be systematically validated on large-scale applications. Arbitrary-precision is required, for example, to compute periodic orbits of dynamical systems [5], to study multi-body systems [9], or dynamical systems [20], for climate modelling and ocean circulation models [23].

In Table 4, we summarize the distribution of the work over the 5 years of the BeDoP project, according to the dependencies and relationships between the different tasks.

2016	2017	2018	2019	2020
Task RT1-1				
	Task RT1-2			
		Task RT1-3		
		Task RT2-1		
			Task RT2-2	
			Task RT3-1	
				Task RT3-2
				Task RT2-3, Task RT3-3

Table 4: The BeDoP Roadmap.

2.4 Expected Outcome: Beyond Double Precision

According to the U.S. Department of Energy [3], high-precision arithmetic is one of the nine areas in mathematics and algorithms where advances are needed for exascale computing:

Some, and possibly many, exascale applications will require high-precision arithmetic facility, yet there is little prospect for vendor support. While some software packages are available, they have shortcomings. What is needed is a simple-to-use facility that infallibly converts large application programs for high precision, yet results in only a modest inflation in run time. Such a facility is possible but not yet available.

Let us recall that it is **our responsibility** as computer scientists to (i) **warn the scientific community** about the limits of double precision for large computations, (ii) design and improve software tools that will enable scientists to go **beyond double precision** in the parts of their programs that require it, and (iii) make those software tools **efficient and robust**.

The expected outcome of the BeDoP Project will address these **crucial issues**: we will perform original research in the field of **formal proof for arbitrary-precision floating-point arithmetic**, and we will develop new libraries for quadruple- and arbitrary-precision floating-point arithmetic, which will be **efficient and robust**, and will allow **reproducible research**.

Addressing these issues will be a real challenge, but we strongly believe it is the right time to address them, and that we have a unique opportunity to bring together researchers from the formal proof and the computer arithmetic communities.

3 Resources for the BeDoP project

3.1 The Research Environment

The PI and his research team (Caramel) are part of Inria, the French National Institute for Research in Mathematics and Informatics. The Caramel group is located in the “Nancy-Grand Est” Inria research centre (21 research teams). The Caramel group maintains long-term tight collaborations with several research groups in computer arithmetic and algorithmic number theory, in particular the group of Richard Brent (now retired) in Australia, the group of Arjen Lenstra at EPFL, and the GMP developers around Torbjörn Granlund (KTH, Stockholm). Locally in Nancy, the team VeriDis of Stephan Merz brings together specialists of verification, model checking, SMT solvers, and will hire Jasmin Blanchette in early 2015. Connections with other Inria teams in France already exist: with the team of André Seznec in Rennes focusing on hardware, with the AriC team of Jean-Michel Muller in Lyon specialising in computer arithmetic, with the team of Xavier Leroy in Paris working on certified computation, with the Toccata team in Saclay concerning formal proof, and that of Yves Bertot in Sophia Antipolis. We also expect connections with the FASTRELAX project supported by the French National Research Agency (ANR) and coordinated by Bruno Salvy from the AriC team.

3.2 The BeDoP Research Team

ERC Funded People

- the Principal Investigator, with a 75% commitment to the BeDoP project. He will manage the BeDoP Research Team, and work on the design and implementation of the MPS routines (RT1-1, RT1-2), on the implementation and formal proof of quadruple-precision (RT2-1, RT2-2), on the formal proof of arbitrary-precision routines (RT3-2), and on the validation of arbitrary-precision routines on large-scale applications (RT3-3);
- one full-time Junior Researcher over 5 years, with an initial training in computer arithmetic or in formal proof, ideally in both domains. She/he will work on the design and formal proof of the MPS low-level arithmetic (RT1-1, RT1-3), and on the implementation and formal proof of arbitrary-precision arithmetic within the MPFR library, with a focus on mathematical functions (RT3-1, RT3-2);
- one postdoctoral researcher (Postdoc 1) for 2 years. Postdoc 1 will work on the implementation and formal proof of quadruple precision arithmetic within the `libquadmath` library (RT2-1, RT2-2), as well on the validation on large-scale applications (RT2-3);
- one postdoctoral researcher (Postdoc 2) for 2 years. Postdoc 2 will work on the implementation and formal proof of arbitrary-precision arithmetic (RT3-1, RT3-2);
- one PhD student (PhD 1) for 3 years. PhD 1 will work on the design and formal proof of the MPS language (RT1-1, RT1-3);

- one PhD student (PhD 2) for 3 years. PhD 2 will work on the implementation and formal proof of quadruple precision arithmetic within the MPFR library (RT2-1, RT2-2);
- one PhD student (PhD 3) for 3 years. PhD 3 will work on the implementation and formal proof of arbitrary-precision arithmetic within the MPFR library, with a focus on basic arithmetic (RT3-1, RT3-2);
- one software engineer for 4 years, who will contribute to all design and implementation tasks (RT1-1, RT1-2, RT2-1, RT3-1), and to the validation of quadruple and arbitrary-precision on large-scale applications (RT2-3, RT3-3);
- several visiting professors (4 months per year in total), in particular we expect Norbert Müller (Trier, Germany), Torbjörn Granlund (Stockholm, Sweden) and Marco Bodrato (Italy) to be regular visitors of the BeDoP team.

A detailed schedule is given in Table 5. Apart from those people funded by ERC, several other people will also contribute partly to the BeDoP team in Nancy: Stéphane Glondu, a research engineer hired by the host institution, who is a specialist of the Coq formal proof system; Jasmin Blanchette, a junior researcher hired by VeriDis team, specialist of first-order and higher-order theorem provers.

2016	2017	2018	2019	2020
Principal Investigator: RT1-1, RT1-2, RT2-1, RT2-2, RT3-2, RT3-3				
Junior Researcher: RT1-1, RT1-3, RT3-1, RT3-2				
PhD 1: RT1-1, RT1-3				
PhD 2: RT2-1, RT2-2				
Postdoc 1: RT2-1, RT2-2, RT2-3				
PhD 3: RT3-1, RT3-2				
			Postdoc 2: RT3-1, RT3-2	
Development Engineer: RT1-1, RT1-2, RT2-1, RT2-3, RT3-1, RT3-3				
Workshop 1		Workshop 2		Workshop 3

Table 5: Detailed schedule of the BeDoP project.

External Collaborators. Other Inria tenure-track researchers outside the Nancy site will be very helpful: Lefèvre in Lyon, Melquiond and Boldo in Saclay, Théry in Sophia-Antipolis are all specialists of formal proof and/or computer arithmetic. A past collaboration of the PI with Norbert Müller in Trier will be renewed [30]. BeDoP will also build on the work of Reynald Affeldt and Magnus Myreen on the formal proof of multiple-precision assembly code [6, 31], and on the work of Krebbers and Spitters on efficient real arithmetic in Coq [28].

3.3 Other Direct Costs

BeDoP Workshops. We will organise three BeDoP workshops, one in 2016, one in 2018, and one in 2020, with partial support from the host institution. We aim to organise those workshops in Dagstuhl, which is an exceptional place for scientific exchanges. The 2016 workshop will bring together researchers from the formal proof and computer arithmetic communities, to which we

will present the main goals of the project, and the preliminary results obtained so far, especially in Research Target 1; it will be an excellent opportunity to get feedback from the scientific community, and to adjust the project schedule if needed. The 2018 workshop will focus on quadruple-precision arithmetic, and on the results obtained in Research Target 2. The 2020 workshop will focus on arbitrary-precision arithmetic, on the results obtained in Research Target 3, and will include researchers working on large-scale applications.

Travel and conferences. 147.5 k€ for 5 years to visit our partners, and attend international conferences in computer arithmetic and formal proof (ARITH, ISSAC, LICS, SSV, POPL, ...).

3.4 Available Resources

The Caramel team has access to a large cluster of 48 Intel Xeon nodes, each node having 16 cores, thus with a total of 768 cores. In addition, the members of the BeDoP team will have access to Grid5000, a French scientific instrument supporting experiment-driven research in all areas of computer science, including high performance computing, distributed computing, networking and big data, with more than 5000 high-performance cores available.

Cost Category		Total in Euro	
Direct Costs	Personnel	PI (75%)	525,000
		Junior Researcher	281,725
		Postdoctoral Researchers	192,000
		PhD Students	351,000
		Software Engineer	230,000
	Total Direct Costs for Personnel		1,579,725
	Travel		262,500
	Equipment		
	Other goods and services	Consumables	
		Publications	
Other			
Total Other Direct Costs		262,500	
A - Total Direct Costs		1,842,225	
B - Indirect Costs (overheads)		460,556	
C1 - Subcontracting Costs (no overheads)			
C2 - Other Direct Costs with no overheads			
Total Estimated Eligible Costs (A+B+C)		2,302,781	
Total Requested Grant		2,302,781	

References

- [1] Challenges in irreproducible research. <http://www.nature.com/nature/focus/reproducibility/>. Special focus, 2013. [1]
- [2] IEEE interval standard working group - P1788. <http://grouper.ieee.org/groups/1788/>. [4]
- [3] Modeling and simulation at the exascale for energy and the environment. <http://science.energy.gov/~media/ascr/pdf/program-documents/docs/Townhall.pdf>, 2007. [10]

- [4] European Exascale Software Initiative: Working group report on numerical libraries, solvers and algorithms. Available from <http://www.eesi-project.eu/>, 2011. 60 pages. [1]
- [5] ABAD, A., BARRIO, R., AND DENA, A. Computing periodic orbits with arbitrary precision. *Phys. Rev. E* 84 (2011), 016701. [10]
- [6] AFFELDT, R. On construction of a library of formally verified low-level arithmetic functions. *Innovations Syst. Softw. Eng.* 9 (2013), 59–77. [7, 12]
- [7] AGULLO, E., FAVERGE, M., GIRAUD, L., GUERMOUCHE, A., RAMET, P., AND ROMAN, J. Toward parallel scalable linear solvers suited for large scale hierarchical parallel platforms. In *WCCM-ECCM-ECFD* (Barcelone, Spain, July 2014). [9]
- [8] BAILEY, D. H., BARRIO, R., AND BORWEIN, J. M. High-precision computation: Mathematical physics and dynamics. *Applied Mathematics and Computation* 218 (2012), 10106–10121. [1]
- [9] BAILEY, D. H., AND FROLOV, A. M. Universal variational expansion for high-precision bound-state calculations in three-body systems. Applications to weakly bound, adiabatic and two-shell cluster systems. *Journal of Physics B: Atomic, Molecular and Optical Physics* 35, 20 (2002), 4287. [10]
- [10] BERTOT, Y., MAGAUD, N., AND ZIMMERMANN, P. A proof of GMP square root. *Journal of Automated Reasoning* 29 (2002), 225–252. Special Issue on Automating and Mechanising Mathematics: In honour of N.G. de Bruijn. [8]
- [11] BOBOT, F., FILLIÂTRE, J.-C., MARCHÉ, C., MELQUIOND, G., AND PASKEVICH, A. *The Why3 platform, version 0.82*. LRI, CNRS & Univ. Paris-Sud & INRIA Saclay, Dec. 2013. <http://why3.lri.fr/download/manual-0.82.pdf>. [7]
- [12] BOLDO, S. *Deductive Formal Verification: How to Make Your Floating-Point Programs Behave*. Habilitation thesis, University of Paris-Sud, 2014. [1]
- [13] BRISEBARRE, N., DE DINECHIN, F., JEANNEROD, C.-P., LEFÈVRE, V., MELQUIOND, G., MULLER, J.-M., REVOL, N., STEHLÉ, D., AND TORRES, S. *Handbook of Floating-Point Arithmetic*. Birkhäuser, 2009. 572 pages. [8]
- [14] CORNEA, M. Precision, accuracy, and rounding error propagation in exascale computing. In *Proceedings of 21st IEEE Symposium on Computer Arithmetic (ARITH)* (2013), pp. 231–234. [1]
- [15] DE DINECHIN, F., AND VILLARD, G. High precision numerical accuracy in physics research. *Nuclear Instruments and Methods in Physics Research A* 559 (2005), 207–210. [3]
- [16] DEKKER, T. J. A floating-point technique for extending the available precision. *Numer. Math.* 18 (1971), 224–242. [3]
- [17] DONGARRA, J., AND BECKMAN, P. The International Exascale Software Roadmap. *International Journal of High Performance Computer Applications* 25, 1 (2011). [1]
- [18] DOU, Y., LEI, Y., WU, G., GUO, S., ZHOU, J., AND SHEN, L. FGPA accelerating double/quad high precision floating-point applications for exascale computing. In *Proceedings of ICS'10* (Tsukuba, Ibaraki, Japan, June 2-4 2010), pp. 325–335. [1]
- [19] FOUSSE, L., HANROT, G., LEFÈVRE, V., PÉLISSIER, P., AND ZIMMERMANN, P. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.* 33, 2 (2007). [3]
- [20] GELFREICH, V., AND SIMÓ, C. High-precision computations of divergent asymptotic series and homoclinic phenomena. *Discrete Contin. Dynam. Syst. Ser. B* 10 (2008), 511–536. [10]
- [21] GRANLUND, T., AND THE GMP DEVELOPMENT TEAM. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.0.0 ed., 2014. <http://gmp1ib.org/>. [6]

- [22] HARRISON, J. A machine-checked theory of floating point arithmetic. In *Proceedings of 12th International Conference on Theorem Proving in Higher Order Logics* (1999), Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, Eds., pp. 113–130. [4]
- [23] HE, Y., AND DING, C. H. Q. Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications. *The Journal of Supercomputing* 18 (2001), 259–277. [10]
- [24] IEEE standard for floating-point arithmetic, 2008. Revision of ANSI-IEEE Standard 754-1985, approved June 12, 2008: IEEE Standards Board. [2]
- [25] KAHAN, W. Desperately needed remedies for the undebuggability of large floating-point computations in science and engineering. <http://www.digiteo.fr/seminaire-7-oct-Kahan>, 2014. [1]
- [26] KNUTH, D. E. *The Art of Computer Programming*, third ed., vol. 2 : Seminumerical Algorithms. Addison-Wesley, 1998. <http://www-cs-staff.stanford.edu/~knuth/taocp.html>. [3]
- [27] KOGGE, P. Exascale computing study: Technology challenges in achieving exascale systems. <http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf>, 2008. DARPA Report. [1]
- [28] KREBBERS, R., AND SPITTERS, B. Type classes for efficient exact real arithmetic in Coq. *Logical Methods in Computer Science* 9, 1:1 (2013), 1–27. [12]
- [29] MELQUIOND, G. Floating-point arithmetic in the Coq system. *Information and Computation* 216 (2012), 14–23. [4]
- [30] MÜLLER, N. T., AND UHRHAN, C. Some steps into verification of exact real arithmetic. In *Proceedings of NFM'2012* (2012), no. 7226, pp. 168–173. [12]
- [31] MYREEN, M. O., AND CURELLO, G. Proof Pearl: A verified bignum implementation in x86-64 machine code. In *Proceedings of Certified Programs and Proofs* (2013), vol. 8307, pp. 66–81. [7, 12]
- [32] PRIEST, D. M. Algorithms for arbitrary precision floating point arithmetic. In *Proceedings of the 10th Symposium on Computer Arithmetic* (1991), P. Kornerup and D. Matula, Eds., pp. 132–144. [3]
- [33] REVOL, N., AND THÉVENY, P. Numerical reproducibility and parallel computations: Issues for interval algorithms. *IEEE Transactions on Computers* 63, 8 (2014), 1915–1924. [4]
- [34] ROBEY, R. W., ROBEY, J. M., AND AULWES, R. In search of numerical consistency in parallel programming. *Parallel Computing* 37 (2011), 217–229. [3]
- [35] RUSSINOFF, D. M. A mechanically checked proof of IEEE compliance of the floating point multiplication, division and square roots algorithms of the AMD-K7 processor. *LMS Journal of Computation and Mathematics* 1 (1998), 148–200. [4]
- [36] RUSSINOFF, D. M. Computation and formal specification of SRT quotient and square root digit selection tables. *IEEE Transactions on Computers* 62, 5 (2013), 900–913. [4]
- [37] STANCE: A SOURCE CODE ANALYSIS TOOLBOX FOR SOFTWARE SECURITY ASSURANCE. <http://www.stance-project.eu/>. [4]
- [38] STEHLÉ, D., LEFÈVRE, V., AND ZIMMERMANN, P. Worst cases and lattice reduction. In *Proceedings of the 16th IEEE Symposium on Computer Arithmetic* (2003), J.-C. Bajard and M. Schulte, Eds., pp. 142–147. [8]
- [39] STODDEN, V., BAILEY, D. H., BORWEIN, J., LEVEQUE, R. J., RIDER, W., AND STEIN, W. Setting the default to reproducible - reproducibility in computational and experimental mathematics. <http://icerm.brown.edu/tw12-5-rcem>, 2012. 19 pages. [1]
- [40] THE MPFR TEAM. Comparison of multiple-precision floating-point software. <http://www.mpfr.org/mpfr-current/timings.html>. [3]
- [41] THE MPFR TEAM. The MPFR library: Algorithms and proofs. <http://mpfr.org/algorithms.pdf>. [4]

Rocquencourt, 3rd October 2014

Commitment of the host institution for ERC Calls 2014

The Inria (Institut National de Recherche en Informatique et en Automatique) which is the applicant legal entity, confirms its intention to sign a supplementary agreement with **Paul Zimmermann** in which the obligations listed below will be addressed should the proposal entitled **BeDoP: Beyond Double Precision** be retained.

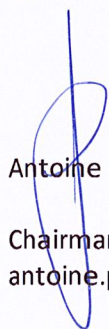
Performance obligations of the applicant legal entity that will become the beneficiary of the grant agreement, should the proposal be retained and the preparation of the grant agreement be successfully concluded:

The applicant legal entity commits itself to engage the principal investigator for the duration of the grant to:

- a) ensure that the work will be performed under the scientific guidance of the principal investigator who is expected to devote at least 30% of his total working time to the ERC- funded project and spend at least 50% of his total working time in an EU Member State or associated country.
- b) carry out the work to be performed, as it will be identified in Annex 1 of the ERC Grant Agreement, taking into consideration the specific role of the principal investigator;
- c) establish a supplementary agreement with the principal investigator which specifies that the applicant legal entity shall:
 - i) support the principal investigator in the management of the team and provide reasonable administrative assistance to the principal investigator, in particular as regards:
 - a) the timeliness and clarity of financial information,
 - b) the general management and reporting of finances,
 - c) the advice on internal applicant legal entity management practices,
 - d) the organisation of project meetings as well as the general logistics of the project.
 - ii) provide research support to the principal investigator and his team members throughout the duration of the project in accordance with Annex 1 ERC Grant Agreement, in particular as regards infrastructure, equipment, products, access rights and other services as necessary for the conduct of the research;
 - iii) ensure that the principal investigator and his team members enjoy, on a royalty- free basis, access rights to the background and the results needed for their activities under the project as specified in Annex 1 ERC Grant Agreement;
 - iv) ensure that the principal investigator enjoys adequate contractual conditions, in particular as regards:
 - a) the provisions for annual, sickness and parental leave,
 - b) occupational health and safety standards,
 - c) the general social security scheme, such as pension rights.

- v) guarantee the necessary scientific independence of the principal investigator, in particular as regards:
- a) the selection and supervision of other team members, hosted and engaged by the applicant legal entity or other legal entities, in line with profiles needed to conduct the research, including the appropriate advertisement, and in accordance with the beneficiary's usual management practices;
 - b) the use of the budget to achieve the scientific objectives;
 - c) the preparation of scientific reports to the ERC Executive Agency;
 - d) the authority to publish as senior author and invite as co-authors only those who have contributed substantially to the reported work.
- vi) inform the principal investigator of any circumstances affecting the implementation of the project or leading potentially to a suspension or termination of the ERC Grant Agreement;
- vii) subject to the observance of applicable national law and to the agreement of the ERC Executive Agency, the transfer of the grant agreement as well as any pre- financing of the grant not covered by an accepted cost claim to a new legal entity, should the principal investigator request to transfer the entire project or part of it to this new legal entity. The applicant legal entity shall submit a substantiated request for amendment or notify the ERC Executive Agency in case of its objection to the transfer.

Inria

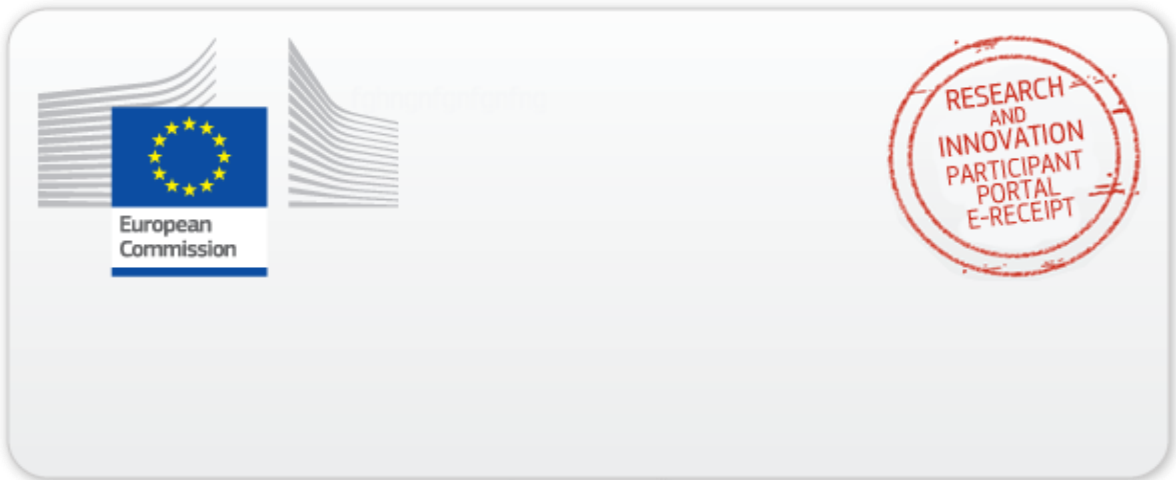


Antoine PETIT

Chairman & CEO

antoine.petit@inria.fr





This electronic receipt is a digitally signed version of the document submitted by your organisation. Both the content of the document and a set of metadata have been digitally sealed.

This digital signature mechanism, using a public-private key pair mechanism, uniquely binds this eReceipt to the modules of the Participant Portal of the European Commission, to the transaction for which it was generated and ensures its full integrity. Therefore a complete digitally signed trail of the transaction is available both for your organisation and for the issuer of the eReceipt.

Any attempt to modify the content will lead to a break of the integrity of the electronic signature, which can be verified at any time by clicking on the eReceipt validation symbol.

More info about eReceipts can be found in the FAQ page of the Participant Portal. (<http://ec.europa.eu/research/participants/portal/page/faq>)