Floating-Point Verification

M.J.C. Gordon
Computer Laboratory
University of Cambridge

CASE FOR SUPPORT

Part I

Previous Research and Record

The Cambridge Automated Reasoning Group consists of researchers applying interactive theorem provers to a variety of problems, from hardware and software systems to pure mathematics. The group dates from the early 1980s.[1] The two main theorem provers in use are Isabelle and HOL. Research with Isabelle has concentrated on generic tools to provide powerful proof support for a wide variety of logics. Research with HOL has concentrated on applications, especially hardware verification, though it has also been applied to software and protocol verification, the study of embedded formalisms, and even pure mathematics. HOL has been stable for several years and it is maintained rather than developed. It provides a secure platform for applied research. It is one of the most widely used theorem proving systems worldwide, having users at universities, industrial sites and Government organizations. Its users have had conferences for the last 8 years, with the participants coming from both academic and industrial institutions from all around the world.

Funding for the development and application of HOL has come not only from the UK, but also from Government agencies in the USA and Australia. Past and present research grants at Cambridge include the following.

- *Higher Order Logic supported design for Complex Data Processing Systems*, Esprit Basic Research Action 3215 with IMEC (Inter-university Micro-Electronics Centre, Belgium) and Philips Research Laboratories (Nertherlands), 1 August 1989 – 31 January 1992.
- *Foundations and Tools for Formal Verification*, IED project 1563. Joint project with Martin Hyland of the University of Cambridge Department of Pure Mathematics and Mathematical Statistics, and partners ICL, PVL and the University of Kent, 1 May 1989 – 30 April 1993.
- *HOL Verification of ELLA Designs*, IED project 1129, with Praxis Systems plc and British Aerospace, 1 August 1989 – 31 July 1992.
- *Demonstration of the Possibility of Totally Verified Systems (SAFEMOS)*, IED Project 1036, with INMOS Limited, SRI International and the Oxford University Programming Research Group, 1 October 1989 – 30 September 1992.

---

[1] Current research is described in the Automated Reasoning Group's World Wide Web page `http://www.cl.cam.ac.uk/Research/HVG/`

- *A Verified Vista Implementation*, MOD Research Grant 2029/250, 1 January 1990 – 31 December 1992.
- *Improved Interface for HOL*, Research grant from the Defence Science and Technology Organization (DSTO), Australia, 1 January 1992 – 30 April 1993.
- *Representation and Validation of Mechanically Generated Proofs*, Joint SERC/MOD Research Grant, 1 January 1992 – 31 December 1994.
- *Continuation of VIPER Verification*, MoD Research Grant, 1 January 1992 – 31 December 1992.
- *Verification of VHDL Designs*, USAF Research Grant, 1991-92.
- *Representation and Validation of Mechanically Generated Proofs*, SERC Research Grant, 1 January 1992 – 31 December 1994.
- *Formal Verification of Aspects of an Asynchronous Transfer Mode Network*, SERC Research Grant, 1 April 1993 – 30 September 1994.
- *A Semantically Based Design Environment for ASICs*, SERC Research Grant, 16 November 1993, 15 November 1996.
- *HCMP Network/Euroform: Formal Methods for Correct System Design*, EC research grant reference CHRX-CT92-0061.
- *'ESPRIT' - Provably Correct Systems (PROCOS)*, EC research grant reference ESPRIT/8694.
- *Hierarchical Formal Verification of Communication Networks*, EPSRC Grant, 1 October 1994 – 31 December 1996.

Ex members of the hardware Verification Group include faculty at universities in the UK, North America and Hong Kong, researchers in government facilities in the USA and Australia and technical staff of several companies (including Hewlett Packard, IBM and Digital).

The bulk of the research proposed here would be carried out by John Harrison. Prior to commencing his PhD, he worked for two years on a project to support the ELLA hardware description language in HOL [1]. As part of that work he carried out a simple verification of a floating point square-root algorithm. His subsequent PhD research involved developing substantial parts of useful mathematics (real analysis etc.) inside HOL [3]. This has, we believe, laid a foundation for the verification of industrially significant floating point algorithms.

Part II
Description of Proposed Research

A Abstract

This project aims to demonstrate that it is practical, using existing theorem proving technology, to formally verify industrially significant floating point algorithms and their implementations.

Models of such algorithms will be mechanically verified with the HOL theorem proving system against precise specifications, often based on real numbers.

Industry is sceptical about the value of formal verification. It is hoped that our studies will help convince manufacturers that the potential benefits far outweigh the costs. This could have a tremendous impact on the industrial uptake of 'formal methods'.

B Scientific/Technological Relevance

In most circumstances, even intelligent testing and simulation can still leave considerable doubts as to the correctness of computer systems. This makes formal verification appealing. There are well-rehearsed arguments over the value of verification for safety-critical systems, such as fly-by-wire aircraft, antilock braking systems in cars, radiotherapy machines and nuclear reactor controllers. However there are also strong commercial arguments for formally verifying hardware, simply because the cost of a recall, redesign, refabrication etc.

Two main approaches to mechanical correctness proofs can be identified:

1. *Model-checking* is used in situations where proof can be reduced to a (usually large) exhaustive search. A classic example is proving correctness of certain combinational logic circuits, where it can be reduced to tautology-checking.

2. *Theorem proving* usually eschews such a brute-force approach, and attempts to prove correctness more in the manner of conventional mathematical proofs, following some kind of formal deductive rules that ensure the correctness of the reasoning.

It is probably fair to say that model-checking has made much more impact on real-world verification tasks. That this is so is mainly because appropriate algorithms and data representations (e.g. Binary Decision Diagrams [2]) allow it to be applied in situations with surprisingly large numbers of states. And when it *is* applicable, the great attraction is that a proof is completely automatic; the user just has to wait for the model-checking algorithm to terminate.

Theorem proving is usually more demanding of the user. This is so even if the prover features a high degree of automation, since it must still be guided by a manually-chosen series of automatically-proved lemmas. However theorem proving has two big advantages over model-checking. First, it is applicable even in situations where the model-checking process is infeasibly complex, or even impossible (in continuous or hybrid systems, the state space may well be infinite). Second, it allows the use of conventional mathematical abstractions and thus allows specifications to be written using full mathematical resources rather than being shoe-horned into a restrictive language such as propositional expressions.

It is our belief that the area of floating point verification is particularly suitable for verification based on a theorem-proving approach. This is so because the natural interpretation of floating-point numbers is as real numbers. The correctness of floating point operations is naturally thought of as properties that hold of certain real numbers, e.g. 'z is the closest representable number to x/y', rather than as bitstrings. All the tools are at hand to tackle floating point verification in HOL. Some simple examples such as a floating-point square root circuit have already been verified, and other more complex examples are in progress at time of writing.

The verification of floating point implementations is also particularly timely, since a bug in the division algorithm of Intel Pentium's floating point unit has been getting unprecedented publicity.[2] This appears to have been caused by omissions in tables of precomputed constants. After initial reluctance, Intel has agreed to a policy of full replacements and has written off $306 million to cover the costs.

C Beneficiaries and Collaborators

The beneficiaries of the project are industrial collaborators who cooperate with us (e.g. providing details of their own systems as verification targets). Our work may actually identify bugs for them, or at least increase their confidence in correctness.

Initially, we have established informal contact with Tim Leonard of Digital Equipment Corporation and Albert Camilleri of Hewlett Packard. These are both ex members of the Cambridge Hardware Verification Group and are familiar with HOL, which they use for their own work. We thus hope that any useful results we get can be smoothly transferred to them for evaluation of industrial usefulness. Both these contacts are with business units of the respective companies, not with research laboratories. We hope this will ensure that the feedback we get paints an accurate picture of the true industrial value of our work.

---

[2] See 'Science' vol. 267, Jan 95, pp. 332-3 for a quick summary of the bug and its relevance to verification.

D Dissemination and Exploitation

The main scientific outputs of the project will be placed in the public domain using standard academic mechanisms, including conference presentations, journal papers, anonymous `ftp`, and the World Wide Web.

We anticipate that links with commercial companies may dictate confidentiality over certain details, but it should be possible to publish general lessons even about specific commercial designs. In any case, the companies concerned will be able to exploit our research for their own ends.

E Project Programme

We believe most of the necessary general theoretical and theorem-proving infrastructure will already be in place at Cambridge by the time the project starts. We will, however, need to adjust some of our existing HOL theories, which are based on ad-hoc floating point formats, to match current industrial standards (e.g. the IEEE standard [5]). We also plan to look carefully at some existing models of floating point operations which may help provide useful abstractions (e.g. Karlsruhe arithmetic). This preliminary work will be completed within the first year of the project. Any further general infrastructure development will only be undertaken if the case studies need it.

A crucial part of the project is to choose tractable examples to demonstrate that verification of floating point algorithms by machine checked formal proof is practical. Based on past experience, we anticipate that getting actual industrial examples will be quite hard, due to the proprietary nature of designs and the associated commercial secrecy. The first part of the project will therefore concentrate on selecting examples, both from the open literature and via our contacts in industry.

Detailed plan

Floating point numbers, of whatever hue, form a large but finite set, F. There is canonical mapping $V : F \to \mathbb{R}$ which associates with each floating point number the abstract real number which it is intended to denote. For example, breaking apart a floating point number f into bit patterns representing an integer exponent e and 'significand' (mantissa) m, we might have:

$$V(f) = 2^{e-e_0}m$$

The IEEE floating point standard, among others, complicates this picture slightly. First, because floating point numbers may be denormalized, the denotation function is a bit more involved. Second, there are various special values like infinities, NaNs (Not a Number) and even positive and negative zeros. However these cases can all be identified by suitable predicates like $Isnormalized : F \rightarrow \mathit{bool}$ and $Isnan : F \rightarrow \mathit{bool}$, and treated specially. In what follows, we will neglect them, for clarity of exposition.

By far the most attractive way of verifying floating point hardware is with reference to the abstract real numbers denoted. For example, the specification of a floating point multiplier whose implementation is represented by $MUL : F \times F \rightarrow F$ might be:

$$\forall x,\ y,\ z \in F.\ |V(MUL(x, y)) - V(x)V(y)| \leq |V(z) - V(x)V(y)|$$

or in words, there is no floating point number z whose value comes closer to the true mathematical value than that already calculated by the multiplier. Similar statements can be expressed in a quite uniform way for a wide variety of operations, including the calculation of transcendental functions like *sin* and *exp*. By contrast, other forms of specification, e.g. using bit patterns, are both more opaque and do not scale well to sophisticated functions.

The above specification may be too strong, and in some cases a more modest assurance may be all that is wanted, e.g. a certain bound on the error in calculation.

$$\forall x,\ y \in F.\ |V(MUL(x, y)) - V(x)V(y)| \leq 10^{-12}$$

The first goal of the project is to write a formal specification in HOL of the IEEE binary floating point standard [5]. Though it creates some additional complications compared with a 'toy' idealization, it is the intention to verify operations for IEEE standard numbers. It may be possible to devise generic ways of hiding the complexities of the standard and making any proofs of correctness easier to modify to conform to different floating point formats. This possibility will be investigated actively.

The bulk of the early work will consist of doing verification examples. We are keen to move as soon as possible to sophisticated functions like *sin*, which have not been tackled before. We want to focus on verifying *algorithms* without undue stress on whether they are implemented in hardware (microcoded or otherwise), software or a mixture of both. Of course, some algorithms are much more suited to hardware implementation, others to software implementation. Nevertheless the eventual transcription to hardware is a well-known problem amply studied by other researchers, and we do not want to get too heavily involved in hardware verification per se. On the other hand we are prepared to get our hands dirty occasionally, e.g. in verifying heavily pipelined implementations. Algorithms will be expressed inside the HOL system using a simple embedded programming language, probably imperative. Some experiments using a program refinement tool written by von Wright [6] have already been done.

It is our intention to verify realistic designs. However some simplified cases will be a useful starting point — in particular there are publications giving hand proofs of floating point algorithms, and an interesting starting point would be to mechanize some of these. There are also some patented algorithms for computer arithmetic, and it would be fascinating to attempt to verify those to see if they contain bugs. If possible, we want to move eventually to real industrial designs, in cooperation with partners.

In the later stages of the project, we hope to build on earlier work in a few directions. Most notably, floating point arithmetic lacks the simple compositionality properties of ordinary arithmetic, so passing from correctness of the underlying operations to precise error bounds on high-level algorithms (e.g. from numerical analysis) which use these operations is not easy. We want to look at ways of avoiding the complexity of ad-hoc arguments. Perhaps work on Karlsruhe arithmetic [7] and asymptotic correctness [4] hold important lessons.

Milestones

The anticipated schedule is as follows:

1. By end of Year 1: examples selected; IEEE fully formalized; some pedagogical examples completed and a tutorial paper written.
2. By end of Year 2: at least one industrially significant example completed and written up.
3. During year 3: further examples; work on correctness of higher level algorithms, review of progress and adjustment of infrastructure.

Criteria for success

The project will be a success if it achieves the following objectives:

1. Demonstrates the potential of theorem proving methods in the verification of floating point hardware.
2. Performs genuinely useful, nontrivial verifications of floating point hardware, of the kind which have not been done before.
3. Leads to greater enthusiasm for verification and formal methods in general in industry.

F Management and Resources

The project will be managed by Mike Gordon.

Staff

John Harrison has agreed be the RA on the project if it is funded. His PhD thesis (which is expected to be complete by the time the project starts) concerns the theoretical and theorem proving infrastructure needed for floating point verification. Mike Gordon will provide supervision and technical guidance. We aim to have some input from industrial contacts.

We request 10% of the cost of a system support officer.

Equipment and other costs

HOL is implemented in both Lisp and Standard ML and is computationally demanding. We plan to use an existing Sun workstation, but are requesting funds to upgrade its processor and buy more memory. We also request funds for a colour X-terminal, so we can benefit from the new graphic interfaces to HOL that are available, and for a laptop for use on trips. Some network interface equipment and a new disc are also requested, so that the workstation can be better integrated into the Computer Laboratory's computing infrastructure.

Travel and subsistence

We request funds to attend some of the following conferences: CADE (Automated Deduction), CAV (Computer Aided Verification), Designing Correct Circuits (DCC), TPCD (Theorem Provers in Circuit Design), LICS (Logic in Computer Science), FME (Formal Methods Europe), DAC (Design Automation Conference), and HUG (HOL Users Group meetings). We are also requesting funds for visit industrial sites in an attempt to explain our work and acquire examples.

References

[1] R. Boulton et al. Experience with embedding hardware description languages in HOL. In Victoria Stavridou, Thomas F. Melham, and R. T. Boute, editors, *Proceedings of the IFIP TC10/WG 10.2 International Conference on Theorem Provers in Circuit Design: Theory, Practice and Experience*, volume A-10 of *IFIP Transactions A: Computer Science and Technology*, pages 129–156, Nijmegen, The Netherlands, 1993. North-Holland.

[2] Randall E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24:293–318, 1992.

[3] John Harrison. Constructing the real numbers in HOL. *Formal Methods in System Design*, 5:35–59, 1994.

[4] D. N. Hoover and D. McCullough. Verifying launch interceptor routines with the asymptotic method. Technical report TM-93-0034, Odyssey Research Associates, 301 Dates Drive, Ithaca NY 14850-1326, USA, 1993.

[5] IEEE. Standard for binary floating point arithmetic. ANSI/IEEE Standard 754-1985, The Institute of Electrical and Electronic Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA, 1985.

[6] J. von Wright, J. Hekanaho, P. Luostarinen, and T. Langbacka. Mechanizing some advanced refinement concepts. *Formal Methods in System Design*, 3:49–82, 1993.

[7] Peter J. L. Wallis. *Improving floating-point programming.* Wiley, 1990.

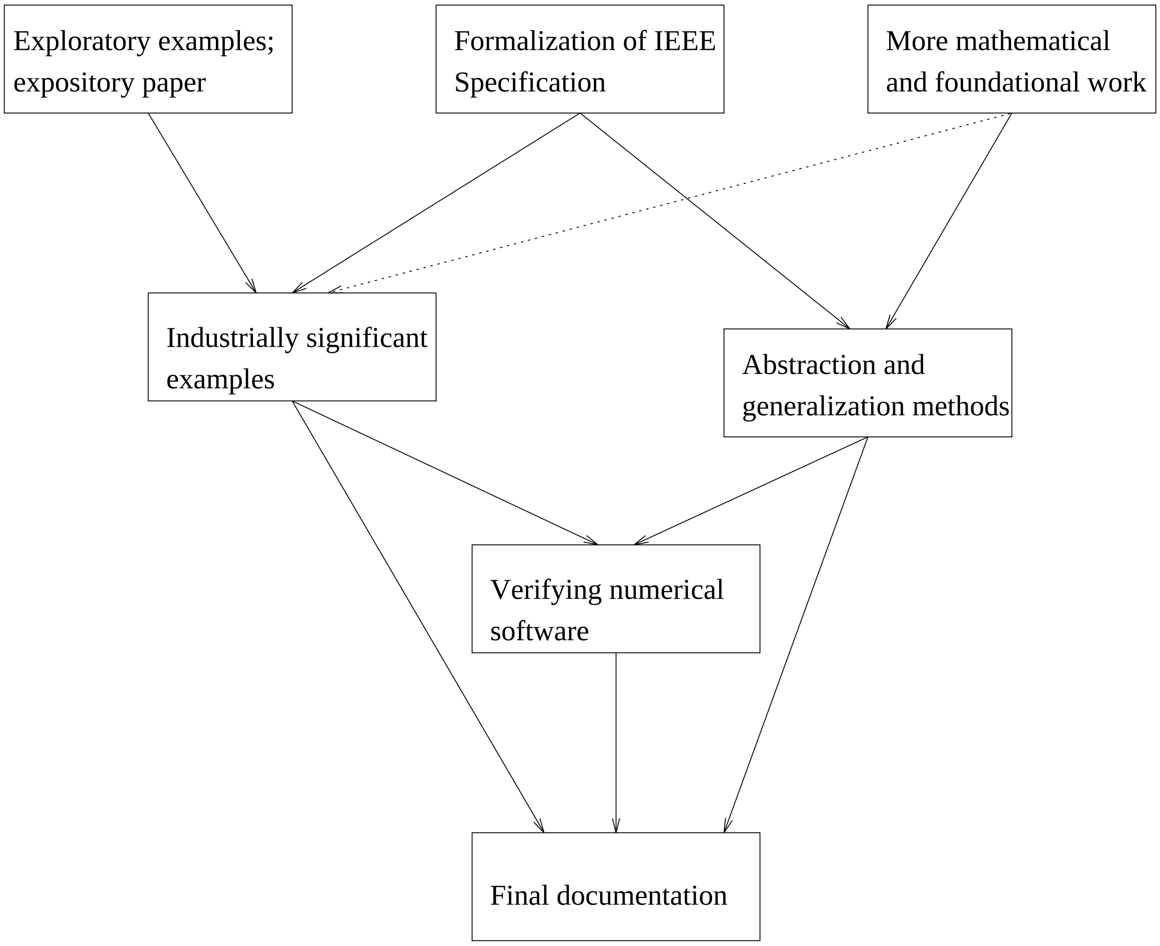

Figure 1: Diagrammatic project plan