MITSUBISHI ELECTRIC R&D CENTRE EUROPE

1, ALLEE DE BEAULIEU, CS 10806
35708 RENNES CEDEX 7, FRANCE
TEL. (+33) 02.23.45.58.58 – FAX (+33) 02.23.45.58.59

for a greener tomorrow

Internship proposal M/F (6 months)

Machine-checked verification of Floating-Point Units

(reference => FFS_022020)

Supervisor

Mitsubishi Electric R&D Centre Europe: Florian Faissole, Researcher

Overall context

With over 90 years of experience in providing reliable, high-quality products, MITSUBISHI ELECTRIC CORPORATION is a recognized world leader in the manufacture, marketing and sales of electrical and electronic equipment used in information processing and communications, space development and satellite communications, consumer electronics, industrial technology, energy, transportation and building equipment. The company is notably involved in critical applications with very high value added, *e.g.* space transportation or nuclear plants management. Technologies used in these applications are particularly subject to high safety requirements, that could be achieved through the use of formal methods. This internship will take place within Information and Network Systems team of MITSUBISHI ELECTRIC R&D CENTRE EUROPE.

Internship subject

As most current microprocessors incorporate floating-point computing units (FPUs), the use of floating-point numbers on an industrial scale has become widespread, including for the most critical applications. Compared with fixed-point numbers, floating-point numbers give the developer a sense of simplicity, with many and somewhat dangerous misconceptions. There is indeed a belief which consists in thinking that floating-point numbers behave like real mathematical numbers, possibly with a negligible – and deterministically behaved – error. However, history has shown that 1) the miscalculations inherent in the use of finite precision data structures could lead to disasters, *e.g.* the Patriot anti-missile system's failure in 1994 [1], 2) that the clever manipulation of floating-point numbers would require a very high level of expertise and 3) that testing software based on floating-point arithmetic is not sufficient to avoid these disasters [2]. Since 1985, floating-point numbers are governed by the IEEE-754 international standard [3], specifying several floating-point formats and the elementary operations and rounding modes associated to these formats. Recent compilers and architectures generally declare themselves compatible with the IEEE-754 standard. Nonetheless, it has been noticed that most of them may lead to different results for floating-point computations [2]. Formal development efforts have been made to increase confidence in floating-point computations. As an example, the Flocq library [4] for the Coq proof assistant [5] provides a formalization of generic floating-point formats with an important number of associated theorems, *e.g.* about floating-point operations, rounding modes or rounding error analysis. The main target of this library is software verification, for which a mathematical abstraction of floating-point numbers is generally sufficient. Nonetheless, Flocq includes a lower-level description of IEEE-754 binary FP numbers, notably used in the CompCert formally-verified optimizing C compiler [6]. Other formal developments only focus on hardware-level floating-point arithmetic, *e.g.* the FPU's verification frameworks proposed by Russinoff in ACL2 [7] and Jacobi in PVS [8]. Nonetheless, these frameworks rely from the ground up on low-level properties of the IEEE-754 standard and are less adequate to prove properties at the level of software.

With a goal of technological transfer between fundamental research and industrial applications, the formal methods activity at Mitsubishi Electric favours the verification of whole chains of tools. In the context of floating-point arithmetic, it highlights the interest for bridging the gap between proofs of low-level properties of hardware and mathematical properties of software executed on it. For the sake of consistency, we postulate that the toolchain's verification should be performed in a unique framework, without losing formal guarantees at the interface between the components, from the mathematical model of floating-point numbers to the RTL and netlist hardware designs. The goal of this internship is to formalize the hardware implementation of basic operations (*e.g.* rounder, adder, multiplier…) of a simple FPU and verify their compliance with the IEEE-754 standard. For that purpose, we propose to use Coq and connect the development with the floating-point formats of the Flocq library. The trainee must keep in mind future goals of the overall projects and more precisely the possibility to generate correct RTL descriptions written in VHDL from the Coq formalization. Another aspect of the internship will be the implementation of an auxiliary tool to automatically generate an understandable, unambiguous, modular and robust documentation that could serves regulatory certification purposes.

Detailed objectives (and different steps)

The internship will consist in

- Understanding and documenting the precise structure of hardware floating-point units and their interaction with the software
- Formalizing some instructions and components of a simple floating-point unit in Coq and proving their properties
- Implementing an external tool to automatically generate relevant documentation
- Paving the way for the next steps of the project, especially from the hardware synthesis point of view (RTL design, logic synthesis…)

Different steps will be:

- Doing a bibliographic study of formal developments in FP arithmetic and hardware design
- Acquiring the needed skills to use Coq and the Flocq library
- Formalizing a subset of the FPU elementary operations in Coq
- Formally proving the compliance of these FPU operations with the IEEE-754 standard (already formalized in Flocq)
- Generating FPU's documentation from specifications and proofs
- Writing the internship's report

Prerequisites

- Interest in formal methods, more particularly for safety-critical software or hardware design
- A first experience with proof assistants (e.g. Coq, Isabelle, HOL-Light, PVS, ACL2, Lean…) or formal deductive verification tools (Why3, Frama-C WP…) is mandatory
- Good programming skills, preferably with functional languages
- Good mathematical knowledge (real analysis, logic)
- Knowledge or experience in at least one of the following domains would be a plus:
    - Good knowledge of computer arithmetic and IEEE-754 standard for FP arithmetic
    - Experience with Coq and functional languages (OCaml, Haskell…)
    - Hardware design, circuits synthesis, VHDL/Verilog, FPU design
- Autonomy
- English: read and written

Duration: 6 months

Period: February-September 2021

Contact: Magali BRANCHEREAU (jobs@fr.merce.mee.com)

Thank you to provide us an application letter and your CV mentioning the reference of the internship (both in PdF versions).

The signature of an Internship Agreement with your school is mandatory.

References

[1] M. Blair, S. Obenski et P. Bridickas, «Patriot missile defense: Software problem led to system failure at Dhahran.,» Report GAO/IMTEC, 1992.

[2] D. Monniaux, «The pitfalls of verifying floating-point computations,» *ACM Transactions on Programming Languages and Systems (TOPLAS),* vol. 30, n°13, pp. 1-41, 2008.

[3] W. Kahan, IEEE standard 754 for binary floating-point arithmetic, 1985.

[4] S. Boldo et G. Melquiond, Computer Arithmetic and Formal Proofs: Verifying Floating-point Algorithms with the Coq System, Elsevier, 2017.

[5] Y. Bertot et P. Castéran, Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions, Springer Science & Business Media, 2013.

[6] X. Leroy, «A formally verified compiler back-end,» *Journal of Automated Reasoning,* vol. 43, n°14, 2009.

[7] D. M. Russinoff, Formal Verification of floating-point hardware design: a mathematical approach, Springer, 2018.

[8] C. Berg et C. Jacobi, «Formal verification of the VAMP floating point unit,» chez *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, 2001.