Arbitrary and Variable Precision Floating-Point Arithmetic Support in Dynamic Binary Translation

Marie Badaroux
Univ. Grenoble Alpes, CNRS,
Grenoble INP†, TIMA
Grenoble, France
marie.badaroux@univ-grenoble-alpes.fr

Frédéric Pétrot
Univ. Grenoble Alpes, CNRS,
Grenoble INP†, TIMA
Grenoble, France
frederic.petrot@univ-grenoble-alpes.fr

Floating-point hardware support has more or less been settled 35 years ago by the adoption of the IEEE 754 standard which is now the dominating floating-point representation. However, given the huge spectrum of software that relies on numerical computations, the IEEE 754 standard cannot satisfy all needs. The current applications have very different requirements. On one side, in neural network, representations on a small number of bits are mandatory. On the other side, scientific applications in nuclear physics, astronomy, geoscience, etc, are seeking kernels able to solve their equations deterministically, with high precision and numerical stability. As a result, the representation of floating-point numbers is again in debate today, as some sensitive applications are gaining momentum. To reach a good performance/accuracy trade-off, developers use variable precision, requiring *e.g.* more accuracy as the computation progresses. Some quite useful applications need very high precision, at least during some specific algorithmic phases. To that aim, several arbitrary floating-point libraries have been developed and among them MPFR is used in widely distributed tools. It is the de-facto standard for that kind of computations. Hardware accelerators for this kind of computations do not exist yet, and independently of the actual quality of the underlying arithmetic computations, defining the right instruction set architecture, memory representations, etc, for them is a challenging task.

For this work, our goal is to design and prototype a simulation environment which helps co-designing the hardware support for large floating-point number representations and variable precision arithmetic. We investigate how the introduction of representations that do not fit in classical bounded size registers can be modeled in practice, so as to be able to rapidly perform functional design space exploration of the hardware/software interface. We are well aware that doing the actual implementation of an arbitrary/multi-precision hardware acceleration unit with unlimited register size is not an option, but being able to simulate without constraints might help making educated decisions. The definition of instruction set architecture (ISA) extensions is a complex matter that requires knowledge in hardware design, compiler design, application needs, and so on. As a modest contribution to this topic, we aim at providing a fast simulation environment that can be relatively simply retargeted to test new instructions, new representations, new compiler support. We base our work on the QEMU dynamic binary translator, using the RISC-V target architecture, and demonstrate how to handle large number and variable precision by using as backend the MPFR library.

To demonstrate the practicality of our approach, we defined an Arbitrary Precision ISA that extends the existing RISC-V ISA with new AP registers and AP instructions and we proposed support for this extension in the QEMU dynamic binary translation framework, using MPFR as arithmetic library. Given the fact that we do not have yet a compiler targeting our AP extensions, we wrote the computing kernels directly in assembly to evaluate the simulation performance. Because of that, our benchmark is rather limited, it however covers different computation types. Our experiments demonstrate that computing kernels are slowed down by one order of magnitude compared to partially hardware accelerated floating-point emulation. Even though this is not negligible and makes the approach unfit for production, it is suitable to make ISA/compiler design space exploration and thorough test of compiler back-ends. This is however a purely functional simulator. The next step is thus estimating the performances on the target, taking into account the memory hierarchy and also possibly the out-of-order effects, as intrinsically, AP numbers have a variable size and AP computations are multi-cycle.

†Institute of Engineering Univ. Grenoble Alpes