Approach based on instruction selection for fast and certified code generation

Christophe Mouilleron, Mohamed Amine Najahi, Guillaume Revy

Approach based on instruction selection for fast and certified code generation

Christophe Mouilleron[1,2,3] Amine Najahi[1,2,3] Guillaume Revy[1,2,3]

[1] Univ. Perpignan Via Domitia, DALI, F-66860, Perpignan, France
[2] Univ. Montpellier II, LIRMM, UMR 5506, F-34095, Montpellier, France
[3] CNRS, LIRMM, UMR 5506, F-34095, Montpellier, France

{amine.najahi,christophe.mouilleron,guillaume.revy}@univ-perp.fr

Keywords: fixed-point arithmetic, automatic code generation, instruction selection, numerical certification

Abstract: Nowadays floating-point arithmetic [1] has become ubiquitous in the specification and implementation of programs, including those targeted at embedded systems. However, for the sake of chip area or power consumption constraints, some of these embedded systems are still shipped with no floating-point unit. In this case, only integer arithmetic is available at the hardware level. Hence, in order to run floating-point programs, we need either to use a library that emulates floating-point arithmetic in software (like the FLIP[1] library), or to rewrite the programs so as to rely on fixed-point arithmetic instead [2]. Yet both approaches require the design of fixed-point routines, which appears to be a tedious and error prone task especially since it is still partly done by hand. Thus, one of the current challenges is to design automatic tools to generate fixed-point programs as fast as possible while satisfying some accuracy constraints. In this sense, we have developed the CGPE[2] software tool, dedicated to the generation of fast and certified codes for evaluating bivariate polynomials in fixed-point arithmetic. This tool, based on the generation of several fast evaluation codes combined with a systematic numerical verification step, is well suited for VLIW integer processors using only binary adders and multipliers. We propose here an extension of CGPE, which consists in adding a step based on instruction selection [4, §8.9] in order to improve the speed and the accuracy of the generated codes for more advanced architectures.

Given an instruction set architecture, *instruction selection* is the compilation process that aims at finding a sequence of instructions implementing "at best" a given program. It works on a target-independent intermediate representation

[1] Floating-point Library for Integer Processors (see `http://flip.gforge.inria.fr`).
[2] Code Generation for Polynomial Evaluation (see `http://cgpe.gforge.inria.fr` and [3]).

of this program, represented as a tree or a directly acyclic graph (DAG), and is usually used to optimize the code size or latency on the target architecture, while no guarantee is provided concerning the accuracy of the generated code. The general problem of instruction selection has been well studied and, although it has been proven to be NP-complete even for simple machines in the case of DAGs [5], several algorithms exist to tackle this problem (see [5] and the references therein).

In the context of CGPE, where we represent polynomial evaluation expressions with DAGs, we can benefit from this work on instruction selection by combining it with the numerical verification step already implemented. The advantage of our new approach is twofold. First it is much more flexible than writing a generation algorithm for each available processor. Indeed, it mainly needs to work on the DAG representation of the expression to be implemented, which is independent of the target architecture, and thus it makes easier to handle various architectures shipping different kind of instructions. Second it allows us to generate automatically codes optimized for a given target and satisfying various criteria like accuracy and performance, as well as code size. So far, this approach has been tested on the evaluation of polynomials, where it allows us to write efficient codes using at best some advanced architecture features like a fused-multiply-add operator.

References

[1] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standard 754-2008, 2008.

[2] D. Menard, D. Chillet, and O. Sentieys. Floating-to-fixed-point conversion for digital signal processors. In *EURASIP Journal on Applied Signal Processing*, pp. 1–15, 2006.

[3] C. Mouilleron and G. Revy. Automatic Generation of Fast and Certified Code for Polynomial Evaluation. In *Proc. of the 20th IEEE Symposium on Computer Arithmetic*, pp. 233–242, Tuebingen, Germany, 2011. IEEE Computer Society.

[4] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[5] D. R. Koes and S. C. Goldstein. Near-optimal instruction selection on DAGs. In *Proc. of the 6th IEEE/ACM international Symposium on Code Generation and Optimization*, pp. 45–54, New York, NY, USA, 2008. ACM.