Improving the Code Generation of Fixed Point Mathematical Functions with FixM

Daniele Cattaneo*, Michele Chiari*, Gabriele Magnani*, Nicola Fossati*, Stefano Cherubin†, Giovanni Agosta*
*DEIB – Politecnico di Milano, Milano, Italy,
†Codeplay Software Limited, Edinburgh, UK

Approximate Computing is an increasingly popular approach to achieve large performance and energy improvements in error-tolerant applications [1, 7]. This class of techniques aims at trading off computation accuracy for performance and energy.

In particular, among such approximate computing techniques, *precision tuning* trades off the accuracy of arithmetic operations for performance and energy by employing less precise data types. For example, fixed point is used instead of floating point, or standard 32-bit floating point numbers are replaced with, e.g., bfloat16 [6].

Precision tuning is primarily employed in the field of embedded systems, and in general in environments where it is necessary to achieve high performance with limited resources. However, performing precision tuning manually is a non-trivial, error-prone and tedious task, especially when large code bases are involved. To alleviate programmer load, significant research efforts have been spent over the last years to build compiler-based tools to fully or partially automatize this process [3].

However, the current state-of-the-art does not consider the possibility of optimizing mathematical functions whose computation is usually off-loaded to a library. To address this limitation, we extended the TAFFO [5, 4] precision-tuning framework to perform tuning of trigonometric functions as well. We developed a new mathematical function library, which is parameterizable at compile-time depending on the data type, and works natively in the fixed point numeric representation [2]. The parameterized implementations of these functions are then seamlessly inserted by a compiler pass into the program during the precision tuning process.

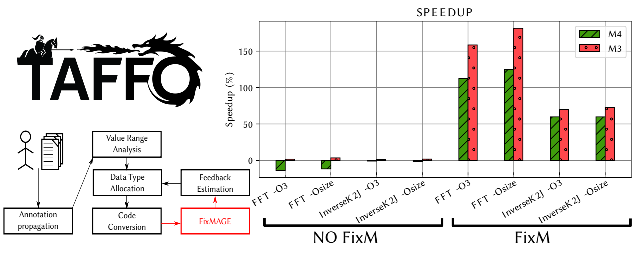

Figure 1: TAFFO workflow with FIXM and key speedup results

We were able to achieve speedups up to approximately 180% on a microcontroller-based embedded system in benchmarks where trigonometric functions represent the majority of the computational effort, with a negligible cost in terms of error. As a result, we achieved energy savings up to 60%. Previous state-of-the-art floating to fixed-point tools are not able to reap any advantages.

We demonstrated our approach on the two most common trigonometric functions, *sin* and *cos*, but it is easily extended to the rest of trigonometric functions and then to hyperbolic functions. Future developments include the implementation of these additional functions, as well as an exploration of different architectural platforms and of alternative computation algorithms.

The integration of FIXM with High Level Synthesis toolchains would also benefit FPGA implementations of algorithms that use trigonometric functions.

1. REFERENCES

[1] A. Agrawal et al. Approximate computing: Challenges and opportunities. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, 2016.

[2] Daniele Cattaneo, Michele Chiari, Gabriele Magnani, Nicola Fossati, Stefano Cherubin, and Giovanni Agosta. Fixm: Code generation of fixed point mathematical functions. *Sustainable Computing: Informatics and Systems*, 29:100478, 2021.

[3] Stefano Cherubin and Giovanni Agosta. Tools for reduced precision computation: a survey. *ACM Computing Surveys*, 53(2), Apr 2020.

[4] Stefano Cherubin, Daniele Cattaneo, Michele Chiari, and Giovanni Agosta. Dynamic precision autotuning with taffo. *ACM Trans. Archit. Code Optim.*, 17(2), May 2020.

[5] Stefano Cherubin, Daniele Cattaneo, Michele Chiari, Antonio Di Bello, and Giovanni Agosta. TAFFO: Tuning assistant for floating to fixed point optimization. *IEEE Embedded Syst. Lett.*, 12(1):5–8, 2019.

[6] IEEE Computer Society Standards Committee. Floating-Point Working group of the Microprocessor Standards Subcommittee. Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

[7] Stanley-Marbell et al. Exploiting errors for efficiency: A survey from circuits to applications. *ACM Computing Surveys*, 53(3), June 2020.

Submitted to IWES 2020