Solving the Table Maker's Dilemma by reducing divergence on GPU

Pierre Fortin and Mourad Gouicem and Stef Graillat

UPMC Univ Paris 06 and CNRS UMR 7606, LIP6,
4 place Jussieu, F-75252, Paris cedex 05, France
{pierre.fortin, mourad.gouicem, stef.graillat}@lip6.fr

Keywords: Table Maker's Dilemma, Graphical Processing Unit, correct rounding, elementary functions

The IEEE 754-2008 standard recommends correctly rounding elementary functions. However, these functions are transcendental and their results can only be approximated with error $\epsilon > 0$. If $\circ_p$ is a rounding function at precision p, there may exist some arguments x, called (p, ϵ) *hard-to-round* arguments, such that $\circ_p(f(x) - \epsilon) \neq \circ_p(f(x) + \epsilon)$, inducing an uncertainty on the rounding of $f(x)$. Finding an error ε such that there are no (p, ε) hard-to-round arguments is known as the Table Maker's Dilemma (TMD).

There exist two major algorithms to solve the TMD for elementary functions which are Lefvre's and SLZ algorithms [2, 3]. The most computationally intensive step of these algorithms is the (p, ϵ) *hard-to-round* argument search since its complexity is exponential in the size of the targeted format. It takes for example several years of computation to get all of them for the classic exponential function in double precision and the same holds for all other classical elementary functions. Hence, getting (p, ϵ) *hard-to-round* arguments is a challenging problem. In order to obtain these (p, ϵ) *hard-to-round* arguments for larger formats (extended precision, quadruple precision), the implemented algorithms should be able to efficiently operate on petaflops systems. In the long term, we would expect to require the correct rounding of some functions in the next versions of the IEEE 754 standard, which will allow to completely specify all the components of numerical software.

High-performance computing systems increasingly rely on many-core chips such as Graphical Processing Units (GPU), which present a partial SIMD execution (Single Instruction Multiple Data). However, when the control flows of the threads on a SIMD unit diverge, the execution paths are serialized. Hence, in order to efficiently use GPU, one has thus to avoid divergence, i.e. manage to have regular control flow within each group of threads executed on the same

SIMD unit.

This work is a first step for solving the TMD on many-core architectures. We focused on Lefèvre's algorithm [2] as it is efficient for double precision. Also, it is embarrassingly parallel and fine-grained which makes it suitable for GPU. We first deployed this algorithm on GPU using the most efficient (to our knowledge) implementation techniques [5]. Then we redesigned it using the concept of continued fractions. This made it possible to obtain a better understanding of Lefèvre's algorithm and a new algorithm which is much more regular. More precisely, we strongly reduce two major sources of divergence of Lefèvre's algorithm: loop divergence and branch divergence. Compared to the reference implementation of Lefèvre's algorithm on a single high-end CPU core, the deployment of Lefèvre's algorithm on an NVIDIA Fermi GPU offers a speedup of 15x whereas the new algorithm enables a speedup of 52x.

References:

[1] J.M. Muller, N. Brisebarre, F. de Dinechin, C.P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, S. Torres, *Handbook of Floating-point Arithmetic*, Birkhauser, 2009.

[2] V. Lefèvre, New Results on the Distance Between a Segment and $\mathbb{Z}^2$. Application to the Exact Rounding, *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, 2005, pp. 68–75.

[3] D. Stehlé, V. Lefèvre, Paul Zimmermann, Searching worst cases of a one-variable function using lattice reduction, *IEEE Transactions on Computers*, 54 (2005), pp. 340–346.

[4] A. Ziv, Fast evaluation of elementary mathematical functions with correctly rounded last bit, *ACM Trans. Math. Softw.*, 17 (1991), pp. 410–423.

[5] P. Fortin, M. Gouicem, S. Graillat, Towards solving the Table Maker's Dilemma on GPU, *Proceedings of the 20th International Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2012, pp. 407–415.