Bounding Variable Values and Round-off Effects using Handelman Representations

David Boland and George A. Constantinides

Abstract—The precision used in an algorithm affects the error and performance of individual computations, the memory usage, and the potential parallelism for a fixed hardware budget. However, when migrating an algorithm onto hardware, the potential improvements that can be obtained by tuning the precision throughout an algorithm to meet a range or error specification are often overlooked; the major reason is that it is hard to choose a number system which can guarantee any such specification can be met. Instead, the problem is mitigated by opting to use IEEE standard double precision arithmetic so as to be ‘no worse’ than a software implementation. However, the flexibility in the number representation is one of the key factors that can be exploited on reconfigurable hardware such as FPGAs, whilst on GPUs it is possible to gain significant performance improvements by moving from double to single precision, and hence ignoring this potential significantly limits the performance achievable. As a result, many publications in the reconfigurable computing community try to improve the hardware performance by customising the precision used in an algorithm on an FPGA or by moving from double to single precision on a GPU, but rarely is this move accompanied by a proof of the numerical stability. Instead these works often rely on simulation-based methods [1], but such an approach cannot guarantee the given error estimate, resulting in a potentially unsafe implementation. In order to optimize the performance of hardware reliably, we require a method that can exactly and tractably calculate tight bounds for the error or range of any variable within an algorithm, taking into account both input ranges and finite precision effects. However, currently only a handful of methods can calculate bounds: Interval Arithmetic [2], Affine Arithmetic [3], and more recently, Satisfiability Modulo Theories [4], and these either sacrifice tightness or tractability.

Our work describes a new tool, which is based upon representing the range of values for a variable by a polynomial and finding bounds for this polynomial by searching for Handelman representations [5], to calculate bounds for the error or range of any variable within an algorithm. The bounds our tool calculates are shown, in general, to be tighter in comparison to existing methods; this in turn can be used to tune the hardware to the algorithm specifications. We demonstrate the proposed procedure on several examples, including an iteration of the conjugate gradient algorithm where we achieve proofs of bounds that translate into savings which range from a reduction in silicon area of over 40% when meeting the same error specification found by traditional methods, to proving the existence of ranges that must otherwise be assumed to be unbounded when using competing approaches. We also show it achieves comparable bounds to recent literature [4] in a small fraction of the execution time, with greater scalability.

ACKNOWLEDGMENTS

Supported in part by EU FP7 Project REFLECT.

REFERENCES