A Software Tool for Designing Fixed-Point Implementations of Computational Data Paths for Embedded and Reconfigurable Computational Environments.

Dr. David M. Buehler	Dr. Gregory W. Donohoe
dbuehler@cambr.uidaho.edu	donohoe@cambr.uidaho.edu
208-885-6707	208-885-6707
University of Idaho	University of Idaho

Abstract Submitted for the MAPLD-2004 Conference

1 Introduction

In computational environments where the cost of floating-point circuitry is prohibitive, such as deeply embedded computing and reconfigurable computing, computations requiring values from the mathematical field of reals can often be performed using fixed-point representations. However, designing fixed-point implementations of computational algorithms has a history of being a difficult, time-consuming and error-prone task which can result in sub-optimal implementations due to unnecessarily high computational error or even incorrect results.

In conjunction with the Reconfigurable DataPath Processor project, the software tool SIFOpt has been developed which helps algorithm designers create fixed-point implementations of computational algorithms. SIFOpt performs a static analysis of the computation, determining scaling factors for computed variables, alignment operations for additions and subtractions, rescaling operations for multiplications and determining optimal representations for the constants which are used in the computation.

2 Related Research

Recent research into creating fixed-point implementations of computations have focused on dynamic approaches in which the computation is implemented in a floating-point environment which has been instrumented to collect statistics at runtime about the computed values.

Researchers from Seoul National University [4], University of Toronto [1] and Aachen University of Technology [3] use this approach to perform automatic floating-point to fixed-point conversion, taking a C program which includes floating-point variables and constants and generating a C program which uses only integer variables and constants.

Researchers at the University of Washington [2] and MIT [5] use this approach to minimize the data path width of FPGA and ASIC implementations of fixed-point computations.

3 Key Elements of the Paper

SIFOpt takes a high-level description of a data path computation as input. The designer annotates the computation's inputs with partitioning information indicating which bits of the run-time integer values represent Sign, Integer and Fractional (SIF) information. The designer can optionally provide value range information for the inputs. In addition, the designer can provide data path width specifications for any variable in the computation.

SIFOpt performs a static analysis of the computation to be performed, estimating the number of mantissa bits and the integer value range for the runtime integer values in the computation. Limiting the growth of the data path width is achieved by tracking the value range of runtime integers and allocating only as many mantissa bits as necessary for each variable. The designer can also use design knowledge or runtime analysis to restrict the growth of the data path width by specifying that any particular mathematical operator does not result in a carry out.

Because SIFOpt tracks which bits in runtime values represent mantissa values and also tracks scaling factors, it can determine which right-shift operations cause truncation of mantissa bits, and determine the worst-case maximum absolute error that might result from any such operation. This value is tracked and a maximum absolute error value is determined for every variable in the computation.

SIFOpt reports complete design information for the computational steps determined, including: the scaling factor, estimated value range and maximum truncation error for every variable in the computation and all alignment and prescaling operations inserted, indicating any truncation those steps cause. In addition, SIFOpt creates integer-only C code as output.

4 Conclusions

SIFOpt has been used effectively to create fixed-point implementations of computational data paths for the Reconfigurable DataPath Processor. We have demonstrated applications including FIR filtering, image pixel non-uniformity correction, and discrete Fourier transform algorithms. In each case, we were able to use SIFOpt to create a fixed-point implementation in which runtime values utilize the full range of integer bit positions.

References

[1] Tor Aamodt and Paul Chow. Embedded ISA support for enhanced floating-point to fixed-point ANSI C compilation. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embeded Systems*, pages 128–137, 2000.

[2] Mark L. Chang and Scott Hauck. Precis: A design-time precision analysis tool. In *Proceedings of the 2002 IEEE Symposium on Field-Programmable Custom Computing Machines*, 2002.

[3] Holger Keding, Frank Hürtgen, Markus Willems, and Martin Coors. Transformation of floating-point into fixed-point algorithms by interpolation applying a statistical approach. In *Proceedings of the 9th International Conference On Signal Processing Applications and Technology 1998*, 1998.

[4] Ki-Il Kum, Jiyang Kang, and Wonyong Sung. AUTOSCALER for C: An optimizing floating-point to integer C program converter for fixed-point digital signal processors. *IEEE Tr. Circuits and Systems II*, 47(9):840–848, September 2000.

[5] Mark Stephenson, Johathan Babb, and Saman Amarasinghe. Bitwidth analysis with application to silicon compilation. In *Proceedings of the ACM SIGPLAN '2000 Conference on Programming Language Design and Implementation (PLDI)*, June 2000.