Adaptive Mapping of Linear DSP Algorithms to Fixed-Point Arithmetic*

Lawrence J. Chang† Yevgen Voronenko† Markus Püschel†

Introduction

Embedded DSP (digital signal processing) applications are typically implemented using fixed point arithmetic; in hardware to reduce area requirements and increase throughput, but also in software since most embedded processors do not provide floating point arithmetic. Consequently, the developer is confronted with the difficult task of deciding on the fixed point format, i.e., the number of integer and fractional bits to avoid overflow and ensure sufficient accuracy. For software implementations, the entire bitwidth is fixed, typically at 32, which means that increasing the representable range (number of integer bits) reduces the available accuracy (number of fractional bits) and vice-versa.

In this paper we present a compiler that translates a floating point C implementation of a linear DSP kernel, such as a discrete Fourier or wavelet transform, into a high accuracy fixed point C implementation. The inputs to the compiler are a floating point arithmetic C program and the range of the input vector elements. First, the compiler statically analyzes the program in a single pass using a recently developed tool that uses affine arithmetic modeling [1]. Then, in the *global mode*, the compiler determines the global fixed point format with the least number of integer bits (and thus the highest accuracy) that guarantees to avoid overflow and outputs the corresponding code. More interesting is the *local mode*, in which the compiler determines the best format *independently* for each variable, thus further pushing the possible accuracy. The compiler is currently limited to straightline code; an extension to loop code is in development.

Further, we used the SPIRAL code generator [2] to generate numerically robust implementations as input to our compiler, thus automating the entire design flow of creating high accuracy fixed point implementations for linear DSP kernels. Experiments with different transforms show that by choosing the formats independently (local mode) the accuracy can be improved by a factor of up to 5 in terms of a norm-based error measure.

Adaptive Fixed-Point Mapping for High Accuracy

Our approach to generating a high accuracy fixed point implementation for a DSP transform T consists of the following two high-level steps; the second step is our main contribution.

- We generate a numerically robust initial *floating point* implementation for T using SPIRAL.
- We translate this implementation into a high accuracy fixed point implementation using the input range as additional information.

Generating a Robust Initial Implementation. SPIRAL is a generator for fast, platform-adapted implementations of DSP transforms and filters. SPIRAL operates in a feedback loop that generates, for a given transform, alternative algorithms and implementations to find the best match to the given platform. The feedback loop is driven by the measured runtimes of the generated codes; by replacing it with a norm-based accuracy measure, we use SPIRAL to generate numerically robust code.

Adaptive Translation into Fixed Point Code. To translate a floating point implementation into fixed point format, the crucial task is to determine the maximal range of each occurring variable. The tool in [1] uses affine arithmetic modeling to achieve this statically with a single pass through the code. The basic idea is to represent each variable x by an *affine expression*

$$\hat{x} = x + \sum x_i \epsilon_i, \quad x_i > 0,$$

*This work was supported by NSF through awards ACR-0234293, SYS-0310941, and ITR/NGS-0325687.

†Dept. of Electrical and Computer Engineering, Carnegie Mellon University, {lchang,yvoronen,pueschel}@ece.cmu.edu .

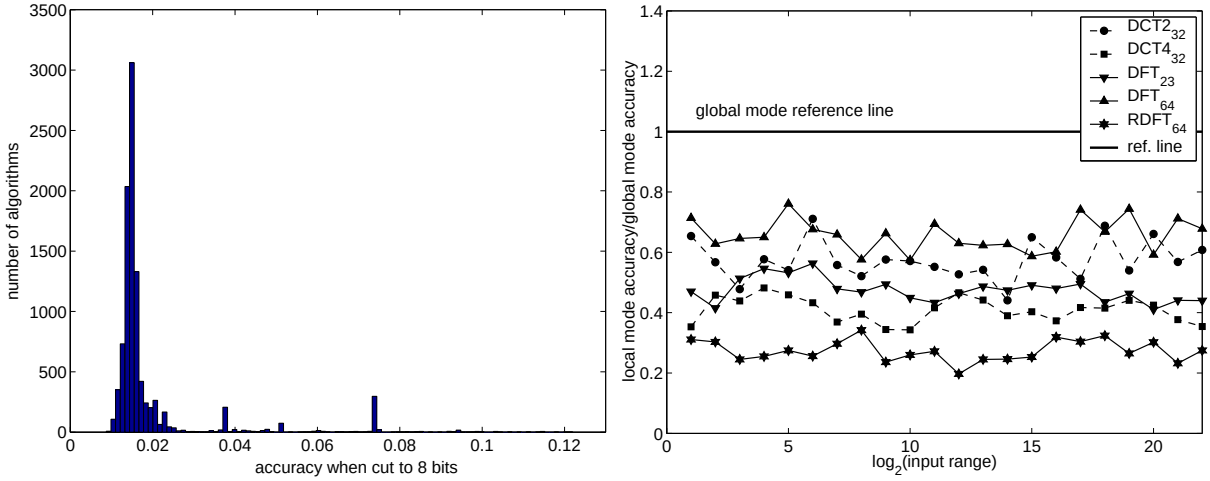

Figure 1: Left: accuracy histogram of 10,000 SPIRAL generated algorithms for a DCT, type 2, size 32. Right: relative accuracy of "local method" vs. "global method" (lower is better).

where the ϵ_i are random variables uniformly distributed in $[-1, 1]$. Intuitively, some of the ϵ_i's capture the range of each variable and others the uncertainty due to finite precision effects in the computation. Starting from the input, these affine expressions are computed for every variable in the code; each finite precision operation introduces a new error variable. For example, a global input range of $[-N, N]$ corresponds to affine expressions of the form $0 + N\epsilon_1$, i.e., at the input the entire uncertainty is due to range. For further details on the method see [1].

From the affine expression for a variable, its maximal range is obtained by setting all ϵ_i to 1 and -1. In the global mode, we determine the number of integer bits through the largest occurring range among all variables. In the local mode, the format of each variable is chosen independently.

Results

Figure 1 (left) shows a robustness histogram of 10,000 SPIRAL generated algorithms for a DCT (type 2) of size 32. The robustness measure compares a floating point implementation to an 8-bit fixed-point implementation for each algorithm.[1] Most algorithms are within a factor of 2–3. Using SPIRAL's search mechanism we generate a robust algorithm as input to our compiler.

Figure 1 (right) shows the benefit of choosing independent fixed point formats (local mode) versus choosing a global format. Each line represents a transform; the x-axis shows the logarithm of the chosen input range (e.g., 10 means a range of $[-2^{10}, 2^{10}]$); the y-axis shows the relative accuracy of local vs. global. The best improvement of a factor of 3–5 is achieved for a real DFT (RDFT).

Conclusion. Our compiler achieves two main goals in the targeted domain. First, we free the developer from choosing a suitable fixed-point format by hand. Second, we obliterate the need for extensive simulations, since the generated code provably avoids overflow by construction. By using our compiler as backend to SPIRAL, the design flow is fully automated.

References

[1] Claire F. Fang, Rob A. Rutenbar, and Tsuhan Chen. Fast, Accurate Static Analysis for Fixed-Point Finite Precision Effects in DSP Designs. In *Proc. ICCAD*, 2003.

[2] M. Püschel, B. Singer, J. Xiong, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, and R. W. Johnson. SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithms. *International Journal of High Performance Computing Applications*, 18(1):21–45, 2004. http://www.spiral.net.

[1] More precisely, we apply both implementations to all standard base vectors to create the (almost) exact transform matrix M and the approximation $\tilde{M}$ and measure the matrix infinity norm $||M - \tilde{M}||_\infty$ of the difference.