High-Performance Code Generation for FIR Filters and the Discrete Wavelet Transform Using SPIRAL*

Aca Gačić† Markus Püschel† José M. F. Moura†

The discrete wavelet transform (DWT) and filtering algorithms, together with certain classes of trigonometric transforms, are among the most frequently used methods in digital signal processing (DSP) applications. For example, the DWT is used in detection, identification, denoising, prediction, channel coding, and compression such as in the JPEG2000 standard. In many of these applications, filters and the DWT are the computationally most intensive components, and their performance is crucial for the overall efficiency of the application.

Creating high-performance software implementations of fast DSP algorithms is a difficult problem. For example, in many cases the data access and data flow patterns of the implementation play a more important role than the computational cost of the algorithm. As a consequence, the best implementation is platform-dependent, and, when the platform is changed, the application has to be reimplemented to retain optimal performance. In many cases, the best implementation on one platform is far suboptimal on another. Because of these problems, there is a need for a systematic and automated way to create, modify and adapt implementations across different platforms.

The SPIRAL system [1] offers a solution to this problem for the class of linear DSP transforms by automatically generating implementations that are highly optimized and adapted to the given software platform. SPIRAL's current version generates code for the discrete Fourier transform (DFT), the discrete cosine and sine transforms (DCTs and DSTs), and several other transforms. This paper extends SPIRAL to generate high performance code for FIR filters and the DWT.

SPIRAL

SPIRAL is a system that automatically generates platform-adapted libraries for DSP transforms [1]. The system uses a high level algebraic notation to represent, generate, and manipulate various algorithms for a user-specified transform. Using a SPIRAL proprietary compiler, these algorithms are automatically translated into C or Fortran code. SPIRAL then tunes the implementation to the platform by intelligently searching in the space of different algorithms and their implementation options for the fastest on the given platform.

SPIRAL uses a small set of mathematical constructs and primitives to represent fast algorithms for DSP transforms as mathematical *formulas*. Examples for constructs include the tensor or Kronecker product, direct sum, diagonal and permutation matrices. These constructs capture the structure of the algorithm and reveal implementation choices when mapped into code. For a given, formally specified transform, the possible formulas, or algorithms, are recursively generated using a small set of *rules*, which represent the various methods of recursively computing a transform. An example for a rule is the famous Cooley-Tukey FFT for the discrete Fourier transform. The degrees of freedom when applying these rules yield a large space of different formulas for a given transform—the search space used by SPIRAL for optimization.

FIR Filter and DWT Algorithms

SPIRAL provides a suitable setting for automatic generation of platform-adapted code for DSP transforms. Hence the strong motivation to include the FIR filter and the DWT transforms into SPIRAL. To accomplish this, two main problems have to be solved: (1) identifying and including into SPIRAL the appropriate mathematical constructs necessary to express their algorithms; and (2) identifying and including into SPIRAL the available rules for filters and the DWT that generate the algorithm space for these transforms.

*This work was supported by DARPA through research grant DABT63-98-1-0004 administered by the Army Directorate of Contracting and by NSF through award 0234293.

†Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, {agacic,pueschel,moura}@ece.cmu.edu .

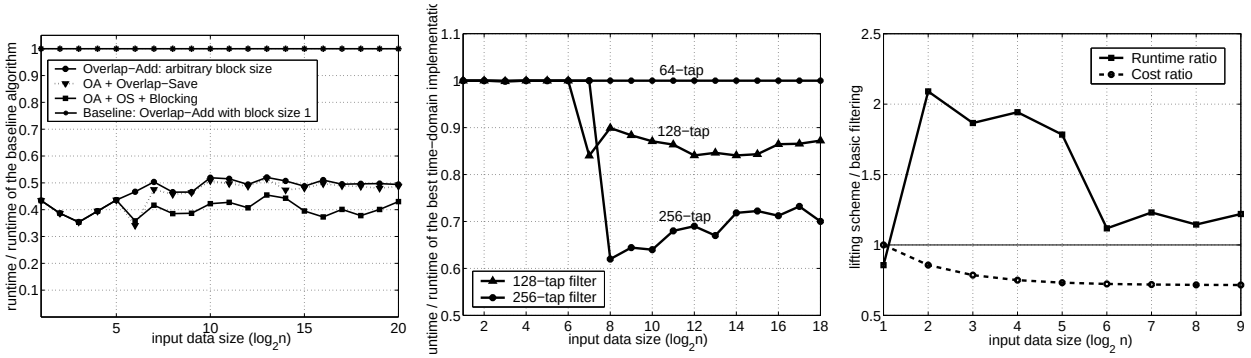

Figure 1: left: time-domain algorithms for FIR filter; middle: time vs. frequency domain methods for varying filter lengths; right: runtime and cost comparison of lifting scheme and filtering for the DWT.

For the FIR filter transform we formulate a set of rules that capture the essence of some popular computational algorithms, namely, overlap-save, overlap-add, FFT techniques using real DFT-like transforms, as well as register-sensitive multi-level blocking techniques [2].

Using the FIR filter rules as a basis, we apply the same approach to derive a set of DWT rules in order to generate fast DWT algorithms in SPIRAL. As a baseline rule we use Mallat's scheme for computing the DWT, where at each stage we use low-pass and high-pass filtering to produce coefficients for that scale. Other important rules include FFT techniques, as well as the lifting scheme [3].

The advantage of using the formalism of rules is the flexibility of the representation and the fast generation of many algorithm possibilities, including mixed methods (e.g., lifting scheme combined with methods for FIR filters, such as the blocking technique). Using SPIRAL's search the different methods span a rich implementation space, where various methods compete against each other for the fastest solution.

Experimental Results

We performed experiments on FIR filters and the DWT using a variety of methods. Some of the findings are presented in Figure 1; the machine is a Pentium 4, 2.53 GHz, running Linux. All the runtimes shown are for code automatically generated using the dynamic programming search provided by SPIRAL.

First we consider an FIR filter with 33 taps (filter length) for inputs of 2-power lengths 2^n, $1 \leq n \leq 20$. The left graph in Figure 1 shows the comparison of different time-domain methods, all having the same arithmetic cost. The runtimes are shown relative to the baseline method (constant = 1), a single loop implementation over the columns of an FIR filter transform. Including the overlap-add method in the search, improves runtime by about a factor of 2. The results further improve with the inclusion of overlap-save and the blocking rule, respectively. The flexible blocking techniques proves to be best, showing 70% improvement over the baseline.

The middle plot shows the comparison of the best frequency-domain and the best time-domain methods for various input sizes and various filter lengths. Surprisingly, the frequency-domain methods (using a real FFT) show improvement only for filter lengths greater than 64; a comparison of arithmetic cost suggests a smaller number.

Finally, the rightmost graph shows the comparison of the lifting scheme method (used in the JPEG2000 standard) and the direct filtering method for one stage of the DWT using the Daubechies 4 (D4) wavelet on an Intel Xeon (1.7 GHz) machine running Linux. Even though there is a clear advantage of using lifting scheme in terms of the arithmetic cost (up to a factor of 2), the runtimes show its inferiority to the baseline method in this short wavelet case.

In the full paper we will include a larger range of DWT experiments including multi-level blocking and FFT methods.

References

[1] 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," *to appear in Journal of High Performance Computing and Applications*, 2004. `http://www.ece.cmu.edu/~spiral`.

[2] A. Gačić, M. Püschel, and J. M. F. Moura, "Fast Automatic Software Implementations of FIR Filters," in *Proc. International Conference on Acoustics Speech and Signal Processing ICASSP'03*, (Hong Kong), 2003.

[3] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 245–267, 1998.