AUTOMATIC COST MINIMIZATION FOR MULTIPLIERLESS IMPLEMENTATIONS OF DISCRETE SIGNAL TRANSFORMS

Adam C. Zelinski, Markus Püschel, Smarahara Misra, and James C. Hoe

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, U.S.A.

ABSTRACT

The computation of linear DSP transforms consists entirely of additions and multiplications by constants, which, in a hardware realization, can be implemented as a network of wired shifts and additions. Thus, a light weight fixed point implementation that approximates an exact transform can be built from only adders. This paper presents an automatic approach for minimizing the number of additions required for a given transform under the constraint of a particular quality measure. We present an evaluation of our approach. For example, one experiment shows that the IMDCT transform within an MP3 decoder can be reduced from 572 additions to 260 additions while maintaining Limited Accuracy as defined by the MP3 ISO standard.

1. INTRODUCTION

Discrete signal transforms are key components in virtually every digital signal processing (DSP) application and require efficient implementations in software and in hardware. Mathematically, a transform is composed of additions and multiplications by constants; the particular data flow depends on the chosen fast algorithm for this transform. When implemented in hardware, the multiplications by constants are often implemented by a sequence of additions and shifts which is less expensive in terms of chip area and power consumption. These implementations of transforms are referred to as *multiplierless* and confront the designer with two major problems: 1) the choice of a numerically robust algorithm for the transform to ensure the best possible quality when implemented in finite precision; and 2) the large space of different trade-offs between implementation cost, i.e., the total number of additions required, and the output quality, both of which are determined by the particular choice of precisions for the occurring constants. We note that these precisions can be chosen independently for each constant, which leads to a large space of possible alternatives. For example, [1] explores these trade-offs for the DCT of type II and size 8 using coding gain as quality measure.

Automatic custom cost minimization. We present an entirely automatic method to minimize the cost of a multiplierless transform implementation (measured by the number of additions required) while satisfying an arbitrarily given quality constraint. Our approach consists of the following three high-level steps: 1) For the given transform we generate a numerically robust algorithm represented as a sparse matrix factorization in a symbolical mathematical notation. 2) We formally manipulate the algorithm to increase its robustness to finite precision approximation. 3) We search in the space of all possible independent constant precisions for this algorithm for the solution that produces the lowest implementation cost while still satisfying the given quality constraint. The size of the search space is determined by the number of constants in the algorithm and the maximum precision considered.

This approach combines ideas from [1] and the SPIRAL code generation system [2, 3] but generalizes [1] to be automatic and applicable for all transforms and quality measures. We implemented the optimization procedure as part of SPIRAL.

Paper Organization. In Section 2 we present the mathematical framework we use to generate, represent, and manipulate transform algorithms. Section 3 explains how to obtain a low cost implementation of a constant multiplication. Possible quality measures for transforms are discussed in Section 4. The different search strategies to find the best approximation in the large space of alternatives are introduced in Section 5. We conclude with various experimental results in Section 6, which show the success of our approach and the large cost savings possible when custom optimizing a transform. For example, we show that we can reduce the cost of the IMDCT within an MP3 decoder from 572 to 260 additions while still maintaining Limited Accuracy as defined by the MP3 ISO standard [4].

2. DSP TRANSFORM ALGORITHMS

In this section we present the framework that we use to generate, represent, and manipulate fast transform algorithms to improve their numerical robustness. For further details we refer to [2, 3].

Transforms and algorithms. A (discrete, linear) *transform* is a class of matrices parameterized by their size. In this paper we consider the $n \times n$ discrete cosine transforms of type II ($\mathrm{DCT}_n^{(\mathrm{II})}$) and type IV ($\mathrm{DCT}_n^{(\mathrm{IV})}$) and the $2n \times n$ inverse modified discrete cosine transform (IMDCT_n). They are defined (without normalizing factors), respectively, by

$$\begin{aligned}
\mathrm{DCT}_n^{(\mathrm{II})} &= \big[\cos \frac{k(2\ell+1)\pi}{2n}\big]_{0\le k,\ell<n},\\
\mathrm{DCT}_n^{(\mathrm{IV})} &= \big[\cos \frac{(2k+1)(2\ell+1)\pi}{4n}\big]_{0\le k,\ell<n},\\
\mathrm{IMDCT}_n &= \big[\cos \frac{(2k+1)(2\ell+1+n)}{4n}\big]_{0\le k<2n,0\le \ell<n}.
\end{aligned}$$

A *rule* decomposes a transform into a product of structured sparse matrices containing smaller transforms. Examples of rules for the above transforms include

$$\begin{aligned}
\mathrm{DCT}_n^{(\mathrm{II})} &= P_n \cdot (\mathrm{DCT}_{n/2}^{(\mathrm{II})} \oplus (\mathrm{DCT}_{n/2}^{(\mathrm{IV})})^{P'_n}) \cdot (\mathbf{1}_{n/2} \otimes \mathrm{F}_2)^{P''_n},\\
\mathrm{IMDCT}_n &= S_n \cdot \mathrm{DCT}_n^{(\mathrm{IV})},\\
\mathrm{DCT}_n^{(\mathrm{IV})} &= T_n \cdot \mathrm{DCT}_n^{(\mathrm{II})} \cdot D_n,
\end{aligned}$$

This work was supported by NSF through awards 0234293, 0310941, and 0325687.

and many others. The symbols P_n, P'_n, P''_n denote permutation matrices, D_n a diagonal matrix, S_n, T_n certain sparse matrices, $\mathrm{F}_2 = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$, and $\mathbf{1}_n$ the $n \times n$ identity matrix. The exact form is not given due to lack of space. Further, $\oplus$ denotes the direct sum and $\otimes$ the tensor or Kronecker product of matrices, defined by

$$A \oplus B = \begin{bmatrix} A & \\ & B \end{bmatrix}, \quad A \otimes B = [a_{k,\ell} \cdot B],\ A = [a_{k,\ell}].$$

Recursive application of rules until all occurring transforms are fully expanded yields a *formula* that represents an algorithm for the original transform. An example of a formula is

$$\begin{aligned}
&\mathrm{DCT}_8^{(\mathrm{II})} = \\
&[(2,5)(4,7)(6,8),8] \cdot (\mathrm{diag}(1, \tfrac{1}{\sqrt{2}}) \oplus \mathrm{R}_{\frac{3}{8}\pi} \oplus \mathrm{R}_{\frac{15}{16}\pi} \oplus \mathrm{R}_{\frac{21}{16}\pi}) \\
&\cdot[(2,4,7,3,8),8] \cdot ((\mathrm{DFT}_2 \otimes \mathbf{1}_3) \oplus \mathbf{1}_2) \cdot (\mathbf{1}_4 \oplus \mathrm{R}_{\frac{3}{4}\pi} \oplus \mathbf{1}_2) \\
&\cdot[(2,3,4,5,8,6,7),8] \cdot (\mathbf{1}_2 \otimes ((\mathrm{DFT}_2 \oplus \mathbf{1}_2) \\
&\cdot[(2,3),4] \cdot (\mathbf{1}_2 \otimes \mathrm{DFT}_2))) \cdot [(1,8,6,2)(3,4,5,7),8],
\end{aligned}$$

which represents Chen's DCT algorithm [5]. We used the following additional notation: $[\sigma, n]$ is an $n \times n$ permutation matrix for the permutation σ given in cycle notation, e.g., $\sigma = (1,3,2)$ signifies the mapping $1 \to 3 \to 2 \to 1$; $\mathrm{diag}(a_1, \ldots, a_n)$ represents a diagonal matrix with diagonal $a_1 \ldots, a_n$; and

$$\mathrm{R}_\alpha = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$$

is a rotation matrix for angle α.

SPIRAL. The above framework is the foundation for the code generation system SPIRAL [2, 3]. For a given transform, SPIRAL generates one out of many possible formulas (arising from different choices of rules upon expansion), translates it into code, measures its runtime, and, in a feedback loop, triggers the generation of different formulas, thus *searching* for the best match between algorithm and target platform. SPIRAL provides us with the infrastructure to generate algorithms of a desired structure, to formally manipulate them, to translate them into code, and to evaluate their quality, e.g., in the context of an application.

Numerical robustness and lifting steps. In a multiplierless implementation of a transform, every constant multiplication in the chosen algorithm is replaced by a fixed point approximation. Thus it is important to choose a numerically robust algorithm for implementation. We ensure robustness by 1) generating for the given transform an algorithm based on rotations (as in Chen's DCT algorithm above); and 2) expanding the rotations into lifting steps as explained below.

A lifting step is a 2×2 matrix of the form

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}.$$

Lifting steps have the following properties which make them a desirable algorithmic structure for fixed point implementations. 1) A lifting step is invertible, independent of the contained constant; the inverse is again a lifting step, namely

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}. \quad (1)$$

2) Moreover, if the constant a in a lifting step is approximated to require few additions, then so does $-a$ in the inverse.

Lifting steps can be obtained from rotations via the identity

$$\mathrm{R}_\alpha = \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ v & 1 \end{bmatrix} \begin{bmatrix} 1 & u \\ 0 & 1 \end{bmatrix}, \quad (2)$$

where $u = \tan(\alpha/2)$ and $v = -\sin\alpha$. We analyze (2) for robustness with respect to approximation of the occurring constants. Assume we replace the first occurrence of u in (2) by $\tilde{u} = u + \epsilon$. Elementary calculation shows that for the obtained matrix $\widetilde{\mathrm{R}}_\alpha$,

$$\widetilde{\mathrm{R}}_\alpha = \mathrm{R}_\alpha + \begin{bmatrix} -\epsilon\sin\alpha & \epsilon\cos\alpha \\ 0 & 0 \end{bmatrix},$$

i.e., the error is not magnified. A similar observation holds for the second occurrence of u in (2). Replacing v by $\tilde{v} = v + \epsilon$ gives a very different result, namely

$$\widetilde{\mathrm{R}}_\alpha = \mathrm{R}_\alpha + \begin{bmatrix} \epsilon\tan(\alpha/2) & \epsilon\tan^2(\alpha/2) \\ \epsilon & \epsilon\tan(\alpha/2) \end{bmatrix},$$

i.e., the error magnifies if $|\tan(\alpha/2)| > 1$ or $\alpha \in [\pi/2, 3\pi/2]$. These angles can be avoided in an algorithm at no arithmetic cost by the manipulation

$$\mathrm{R}_\alpha = \mathrm{R}_{\alpha-\pi/2} \cdot \mathrm{R}_{\pi/2} = \mathrm{R}_{\alpha-\pi/2} \cdot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Summary. In preparation to developing a multiplierless approximation of a transform algorithm, we 1) generate an algorithm based on rotations; 2) formally manipulate the angles to assure $\alpha \notin [\pi/2, 3\pi/2]$; and 3) expand the rotations into lifting steps.

3. MULTIPLIERLESS IMPLEMENTATIONS

In this section we provide the necessary background for multiplierless implementations and, in particular, for multiplications by fixed point constants.

Multiplierless implementation. In a multiplierless implementation of a transform algorithm, the occurring constants c are replaced by a fixed point approximation

$$c \approx k/2^n,$$

where n denotes the number of fractional bits. The multiplication by a fixed point number c, in turn, can be implemented by additions and shifts. For example $y = 1.01 \cdot x$ may become $y = x + (x \gg 2)$, costing one addition and one shift. To the first order, the hardware cost of such an implementation is dominated by the cost of additions, since fixed shifts are simply wiring permutations in hardware. Given a transform T and an algorithm for T, reducing the precision of the multiplicative constants in this algorithm yields an approximate algorithm representing an approximation $\widetilde{T}$ of T. Our goal is to find the approximation with the lowest cost that still satisfies a user-specified measure of quality.

Multiplication Methods. Different methods exist to realize a multiplication by a constant using additions and shifts. We briefly discuss two standard methods before presenting a third *addition chain* method employed in our approach. For the sake of clarity, we restrict the discussion below to multiplications by fixed point constants between 0 and 1 with n fractional bits, which are represented as a sequence of binary digits $c = 0.b_1b_2....b_{n-1}b_n$ where $b_i \in \{0,1\}$. Thus, the product of c and x can be computed as

$$c \cdot x = \sum_{i=1}^{n} b_i 2^{-i} x.$$

Hence, if the binary representation of c has k non-zero digits, the direct method of multiplication corresponds to summing k shifted

versions of x (i.e., $2^{-i}x$ for each non-zero b_i). The multiplication cost in terms of the number of additions is $k-1$.

Signed digit (SD) recoding is commonly used, in both hardware and software, to reduce the number of additions required when multiplying by a constant [6]. An SD-recoded constant is interpreted as

$$c = \sum_{i=0}^{n} b_i 2^{-i},$$

where $b_i \in \{\overline{1}, 0, 1\}$ and $\overline{1}$ stands for -1. The most salient aspect of SD recoding is in replacing a sequence of s consecutive '1' digits in a normal binary representation by an SD sequence of $s-1$ '0' digits with a prefix '1' and a suffix '$\overline{1}$', i.e.,

$$...0\underbrace{111...11}_{s}0... \rightarrow ...1\underbrace{000...00}_{s-1}\overline{1}0...$$

The above recoding implies that the $s-1$ additions corresponding to the original sequence of '1' digits can be reduced to a single subtraction. A well-known algorithm exists to convert a standard binary representation into a canonical signed digit (CSD) representation such that no two consecutive digits are non-zero (i.e., $b_i b_{i+1} = 0$). This ensures that the cost of multiplying by a n-bit constant is less than $n/2$ additions/subtractions.

In our approach we invoke a method based on *addition chains*. This approach differs from the above methods by potentially reusing intermediate results of the computation, e.g., later additions can use shifted results from earlier additions. For example, the addition chain to multiply x by $c = \frac{10021}{2^{14}} = 0.10011100100101$ is

$$\begin{aligned} s_1 &= (x \gg 14) + (x \gg 12) \\ s_2 &= (x \gg 6) - s_1 \\ s_3 &= (s_1 \ll 11) - s_2 \\ s_4 &= (x \gg 9) + s_3 \end{aligned}$$

In this example, the addition chain method only requires 4 additions; in contrast, the direct method requires 6 additions and the CSD method requires 5 additions.

It has been shown empirically that at most 5 additions are needed to multiply x by any constant representable by up to 19 bits [6]. Unfortunately, finding the optimal addition chain of a given constant has been proven to be NP-hard [7]. In our approach, we employ an approximate algorithm based on dynamic programming [8]. Out of the 2^{19} constants representable in 19 bits, this near-optimal algorithm only requires more than 5 additions for 225 constants and in the worst case requires 7 additions.

4. QUALITY MEASURES

When minimizing the cost of a transform algorithm via approximation, it is important that the *quality* of the algorithm is maintained. Clearly, the measure of quality and an acceptable threshold depends on the application context and ultimately has to be chosen by the user.

We consider two types of quality measures, which are discussed below, and introduce the measures we have used in our experiments (presented in Section 6). As before, a transform is denoted by T, and an approximation by $\widetilde{T}$.

Model based. These quality measures evaluate the quality of an approximate transform based on a mathematical model, independent of any application context. An example is coding gain, which assesses the compression quality of a transform with respect to an input model and is defined as:

$$C_g = 10\log_{10}\frac{\sigma_x^2}{(\prod_{i=0}^{M-1}\sigma_{x_i}^2\|f_i\|^2)^{\frac{1}{M}}}$$

where M is the number of subbands, σ_x^2 the variance of the input, $\sigma_{x_i}^2$ the variance of the i-th subband, and $\|f_i\|^2$ the norm of the i-th synthesis filter [1]. As in [1], we evaluate an approximation of the $\mathrm{DCT}^{(\mathrm{II})}$ assuming an $AR(1)$ model with zero-mean and correlation $\rho = 0.95$. The coding gains of the $\mathrm{DCT}_8^{(\mathrm{II})}$ and the 8-point Karhunen-Loeve Transform (KLT) are 8.8259 dB and 8.8462 dB, respectively.

Application based. The most accurate but also most time-consuming way to evaluate the quality of an approximate transform $\widetilde{T}$ is to insert $\widetilde{T}$ into the respective application and measure the application's overall quality. In our experiments, we considered two applications, an MP3 decoder [9] and a JPEG decoder [10], and their application-level quality metrics.

- The MP3 ISO standard prescribes a compliance test based on the decompression of a reference bitstream [4]. Depending on the root-mean-square difference between a MP3 decoder's output and the reference decoded output, a MP3 decoder is classified as Non-Compliant, Limited Accuracy (LA) or Fully Compliant.
- A commonly used quality measure for JPEG compression is the Peak Signal-to-Noise Ratio (PSNR) of a compressed and then decompressed reference image relative to the original uncompressed input image. In practice, a PSNR ≥ 30 dB is considered acceptable in many imaging applications.

5. OPTIMIZATION THROUGH SEARCH

For a given exact algorithm, there exists a variety of approximated implementations—where the precision of the multiplicative constants has been individually varied—that can be searched to find a low-cost, high-quality trade-off. In this section, we describe three search strategies that we implement in our automatic optimization approach. Using the framework discussed thus far, we can address two complementary optimization problems involving cost and quality. First, for a transform T, we can minimize the number of additions needed in an approximate implementation $\widetilde{T}$ to satisfy a quality threshold. Conversely, given an upper bound on the number of additions allowed, we can maximize the quality achievable by an approximate implementation. This paper focuses on the former problem, but our approach and conclusions can be extended to both optimizations.

Search Space. Given an algorithm comprised of k n-bit constants, there are n^k possible ways to choose the individual bitwidth of the k constants. However, not all n^k configurations are meaningful. Our addition chain example of multiplying x by the constant 0.10011100100101 requires just four additions. For this 14-bit constant, at most five bitwidth settings need to be considered, namely the five most precise approximations of C corresponding to exactly 0, 1, 2, 3 and 4 additions. Based on [6], we know that an exhaustive search for an algorithm with 19-bit constants would require evaluating at most 6^k configurations. Unfortunately, this means an exhaustive search is infeasible for sizes of interest. For example, a $\mathrm{DCT}^{(\mathrm{II})}$ of size 32 requires at least 80 multiplications.

Search Strategy. To cover the extremely large search space efficiently, we implemented three search strategies, described below in increasing order of complexity.

- **Global Search**. Our fastest search strategy adjusts the bit width of *all* constants in a transform by the same amount. Starting

Table 1. Transform and quality threshold considered

Experiment	Transform	Quality Threshold
E_1	DCT-II$_8$	$C_g \geq 8.80$ dB
E_2	DCT-II$_{32}$	MP3, LA
E_3	18x36 IMDCT	MP3, LA
E_4	DCT-II$_8$ (inv only)	JPEG, PSNR ≥ 30 db

Table 2. Summary of experimental results

	E_1	E_2	E_3	E_4
initial (31 bits)	165	1056	572	165
global	62	420	279	**26**
evolutionary	**38**	504	317	28
greedy (top-down)	49	**396**	**260**	28

with a maximum default bit-width, the global bit-width is reduced until the quality threshold is violated. Hence, the quality measure is evaluated for at most $n + 1$ different approximate transforms.

- **Greedy Search**. A top-down greedy search begins from the maximum bit-width configuration. In each step, the greedy search selectively reduces the bit width of one constant by one bit such that the quality degradation is minimized. The greedy search terminates when the quality threshold is violated. We did not consider a bottom-up greedy search in this paper.
- **Evolutionary Search** The evolutionary search strategy mimics the natural selection process, gradually evolving the design towards an optimal configuration. Each iteration of the evolutionary search begins with a population of configurations. A new generation of a population is generated nondeterministically by mutating the previous generation's configurations (e.g., modifying the precision of one constant) and by "cross-breeding" features of the previous generation's configurations (e.g, exchanging precisions for one constant). From the so expanded population, the highest quality configurations are selected as the next population. The search terminates after a preset number of generations.

6. EXPERIMENTAL RESULTS

To evaluate our approach, we experimented with four different combinations of transforms and quality measures (summarized in Table 1). E_1 attempts to minimize the number of additions required by the $\mathrm{DCT}_8^{(\mathrm{II})}$, constrained by the model-based metric C_g; E_2, E_3 and E_4 minimize the cost of some specific transforms in an application under the constraint of application-based overall quality metrics. For example, E_4 approximates the $\mathrm{DCT}_8^{(\mathrm{II})}$ within the Independent JPEG Group's reference JPEG implementation such that a PSNR ≥ 30 dB is achieved after compressing and decompressing the *Lena* image.

Table 2 summarizes the results of our experiments. For each test combination, the number of additions resulting from all three search strategies is given. (The evolutionary search iterated for 25 generations in all cases.) The number of additions required by a default implementation using 31-bit constants is also provided for reference.

In all four experiments, the number of additions was significantly reduced from those required by the default 31-bit implementations. The greedy search strategy produced the best results in the MP3 application-driven optimizations, but failed in the model-based E_1. Also notice that the simple global search strategy produced the best result for E_4 and highly competitive results for E_2 and E_3 but not for E_1.

Fig. 1 provides a closer examination of the behavior of the evolutionary search in E_1. Fig. 1 plots the number of additions for the best configuration found in each of the first 14 generations, after which the cost of the best configuration remained constant. In these 14 generations, the number of additions was reduced from 165 (the default implementation) to 56 (after the first generation) to 38, while preserving $C_g \geq 8.80$.

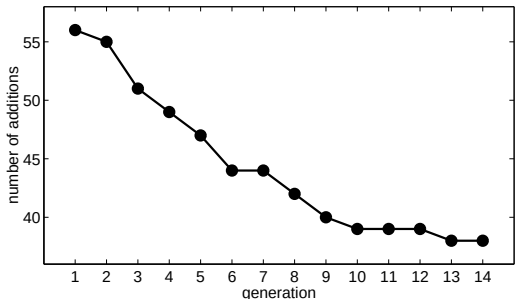

Fig. 1. The progress of the evolutionary algorithm during E_1.

7. REFERENCES

[1] J. Liang and T.D. Tran, "Fast Multiplierless Approximations of the DCT with the Lifting Scheme," *IEEE Trans. on Signal Processing*, vol. 49, no. 12, pp. 3032–3044, 2001.

[2] M. Püschel and J.M.F. Moura, "Generation and manipulation of DSP transform algorithms," in *10th Digital Signal Processing Workshop*, 2002, pp. 344–349.

[3] 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.spiral.net.

[4] ISO/IEC, *Information Technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s – Part 4: Compliance testing*, 1995.

[5] W.-H. Chen, C.H. Smith, and S.C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. on Communications*, vol. COM-25, no. 9, pp. 1004–1009, 1977.

[6] O. Gustafsson, A.G. Dempster, and L. Wanhammar, "Extended results for minimum-adder constant integer multipliers," in *Circuits and Systems, IEEE International Symposium on*. IEEE, May 2002, vol. 1, pp. I–73–I–76.

[7] P.R. Cappello and K. Steiglitz, "Some Complexity Issues in Digital Signal Processing," *IEEE Trans. on Acous., Speech, Signal Processing*, vol. 32, no. 5, pp. 1037–1041, Oct. 1984.

[8] S. Egner, "Automatic addition graphs for multiplication with a constant," 2003, personal communication.

[9] R. Leslie, "MAD MPEG Audio Decoder Software," online, http://www.underbit.com/products/mad.

[10] Independent JPEG Group, "JPEG Image Compression Software," online, http://www.ijg.org.