Custom Reduction of Arithmetic in Linear DSP Transforms

S. Misra, A. Zelinski, J. C. Hoe, and M. Püschel
Dept. of Electrical and Computer Engineering
Carnegie Mellon University

Research Overview

- Linear DSP transforms
  - e.g. DFT, DCTs, WHT, DWTs, ....
  - ubiquitously used, often in computation intensive kernels
  - comprised of additions and multiplication-by-constant
  - applications: multimedia, bio-metric, image/data processing . . . .

- Light-weight hardware implementations
  - fixed-point data format
  - multiplierless: mult-by-constant as shifts and adds
  - problem 1: output quality reduced by cost-saving measures
  - problem 2: different applications have vastly different quality
    metric and requirements
                  ⇒ need application specific tuning

Our Goal: automatic, custom reduction of arithmetic
(additions) w.r.t. a given application’s requirements
Our Automatic Flow

 DSP transform
  ↓
 algorithm selection (robust, structure)
  ↓
 algorithm (as formula)
  ↓
 algorithm manipulation (robustness)
  ↓
 algorithm (as formula)
  ↓
 search for cheapest const. reduction satisfying Q
  ↓
 custom low-cost algorithm

Example

 DCT, size 32, in MPEG decoder
  ↓
 rotation based algorithm
  ↓
 expansion into lifting steps
  ↓
 search: constant reduction
  ↓
 custom low-cost algorithm
  ↓
 MPEG compliance test

Related Work

  - examined arithmetic cost reduction for DCT size 8
  - steps performed by hand, exhaustive search

  - efficient static analysis of output error (hard and probabilistic)
  - range of input values used/needed
  - analysis assumes a common global bitwidth

- Püschel/Singer/Voronenko/Xiong/Moura/Johnson/Veloso/Johnson, “SPIRAL system”, www.spiral.net
  - automatic generation of custom runtime optimized DSP transform software
  - provides implementation environment for our approach (in particular algorithm generation and manipulation)
Outline

- DSP transform algorithms
- Algorithm manipulation for robustness
- Multiplication by constants
- Search Methods
- Results

DSP Algorithms as Formulas:
Example DFT size 4

Cooley/Tukey FFT (size 4):

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & -i & -1 & i \\
-1 & 1 & i & -1 \\
-1 & i & 1 & i
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1 & -1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & 0 & 0 & i \\
0 & 0 & 1 & -1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Fourier transform  Diagonal matrix (twiddles)

\[DFT_4 = (DFT_2 \otimes I_2) \cdot \text{diag}(1,1,i,1) \cdot (I_2 \otimes DFT_2) \cdot [(2,3),4]\]

Kronecker product  Identity  Permutation

allows for computer generation/manipulation
(provided by SPIRAL)
Example: DCT size 8

\[ [(2,5)(4,7)(6,8),8] \]
\[ \cdot (\text{diag}(1,1/\sqrt{2}) \oplus R_{1/8} \oplus R_{15/16} \oplus R_{31/16}) \]
\[ \cdot [(2,4,7,3),8] \cdot ((\text{DFT}_2 \oplus I_2) \oplus I_2) \cdot [(5,6),8] \]
\[ \cdot (R_1 \oplus 1/\sqrt{2} \cdot \text{DFT}_2 \oplus I_2) \cdot [(2,3,4,5,8,6,7),8] \]
\[ \cdot (I_2 \oplus ((\text{DFT}_2 \oplus I_2) \cdot (2,3,4) \cdot (I_2 \oplus \text{DFT}_2))) \]
\[ \cdot [(1,8,6,2)(3,4,5,7),8] \]

Basic building blocks:
- 2 x 2 rotations, DFT_2’s (butterflies), permutations, diagonal matrices (scaling)

Algorithm is orthogonal = robust to input errors (from fixed point representation)

Outline

- DSP transform algorithms
- Algorithm manipulation for robustness
- Multiplication by constants
- Search Methods
- Results
Fixed Point Error: Data vs. Transform

Implementing a transform $x \mapsto Tx$ in fixed point arithmetic produces two type of errors:

- **Error in input $x$:** $||x - \widetilde{x}||$
  - from rounding of the input coefficients $x$ to the fix-point data representation $\widetilde{x}$
  - for robustness: choose orthogonal algorithms

- **Error in transform:** $||T - \widetilde{T}||$
  - from finite precision multiplication by constants
  - further approximation is a source of savings in multiplierless implementations
  - for robustness: translate algorithm into lifting steps

Lifting Steps

- **Lifting step (LS):**
  $\begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} 1 & 0 \\ y & 1 \end{bmatrix}$
  - invertible (det = 1) independent of approximation of $x$, $y$
  - inverse of LS is also LS (with $-x$, $-y$)
  - if LS is cheap, then so is its inverse

- **Rotation as lifting steps**
  $R_\alpha = \begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix} = \begin{bmatrix} 1 & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}
  u = \sin \alpha$
  \[
p = \frac{1 - \cos \alpha}{\sin \alpha} = \tan \frac{\alpha}{2}
  \]

  rotation based algorithms can be automatically expanded into LS

rotation target for approximation
**Error Analysis**

- Rounding error in the first lifting step (third LS analogous)
  \[
  \tilde{R}_\alpha = R_\alpha + \begin{bmatrix} 1 & \varepsilon & 1 & 0 & 1 & p \\ 0 & 1 & \varepsilon & 1 & 0 & 1 \end{bmatrix} = R_\alpha + \begin{bmatrix} \varepsilon & \sin \alpha & \cos \alpha & 0 \\ 0 & \varepsilon & \varepsilon & 0 \end{bmatrix}
  \]

  \(\varepsilon\) is not magnified

- Rounding error in the second lifting step
  \[
  \tilde{R}_\alpha = R_\alpha + \begin{bmatrix} 1 & p & 1 & 0 & 1 & p \\ 0 & 1 & \varepsilon & 1 & 0 & 1 \end{bmatrix} = R_\alpha + \begin{bmatrix} \varepsilon & \tan \frac{\alpha}{2} & \varepsilon \tan^2 \frac{\alpha}{2} & \varepsilon \\ \varepsilon \tan \frac{\alpha}{2} & \varepsilon & \varepsilon \tan^2 \frac{\alpha}{2} & \varepsilon \end{bmatrix}
  \]

  \(\varepsilon\) is magnified, unless \(\alpha\) in \([0, \pi/2]\) or \([3\pi/2, 2\pi]\)

**Solution:** Angle manipulation

\[
R_\alpha = R_{\alpha-\pi/2} \cdot R_{\pi/2} = R_{\alpha-\pi/2} \cdot \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}
\]

---

**Ensuring Robustness**

Steps to ensure robustness

- Choose algorithms based on rotations
- Manipulate angles of rotations
- Expand into lifting steps

**Done automatically as formula manipulation**
Outline

- DSP transform algorithms
- Algorithm manipulation for robustness
- Multiplication by constants
- Search Methods
- Results

Multiplication by Constants

Operations in transforms:
\[ y = x_1 + x_2 \quad \text{additions} \]
\[ y = cx \quad \text{multiplication by constant} \]

Example:

- **Simple**: \( c = 0.10111011 \) = → 5 adds (5 shifts)
- **SD recoding 1**: \( c = 0.1100\overline{1}10\overline{1} \) → 4 adds (3 shifts)
- **SD recoding 2**: \( c = 0.11000\overline{1}0\overline{1} \) → 3 adds (3 shifts)

\textbf{SD recoding is not optimal}
Addition/Subtraction Chain

- Provide optimal solution for constant mult using adds and shifts
- Finding the optimal addition chain is a hard problem
- A near optimal table of solutions can be computed using dynamic programming methods*
- For all constants up to $2^{19}$
  - only 225 constants require more than 5 additions
    (214@6, 11@7)

SD recoding vs. Addition Chains

![Histogram of addition cost for all constants between 1 and $2^{19}$](image)

*Sebastian Egner, Philips Research, Eindhoven
Outline

- DSP transform algorithms
- Algorithm manipulation for robustness
- Multiplication by constants
- Search Methods
- Results

Optimization Problem

Given a linear DSP transform and quality measure $Q$

1. Find the multiplierless implementation with the least arithmetic cost $C$ (number of additions) that satisfies a given $Q$ threshold

2. Find the multiplierless implementation with the highest quality $Q$ for a given arithmetic cost $C$ threshold
Quality Measures of Transforms

For an approximation $\tilde{T}$ of a transform $T$.

- **Transform independent $Q$**
  - $\| T - \tilde{T} \| \text{ for some norm } \| \cdot \|$

- **Transform dependent $Q$**
  - coding gain for DCT
  - convolution error for DFT

- **Application-based $Q$**
  - MPEG standard compliance test

Search Space: approximating multiplicative constants

- For each multiplication-by-constant in the transform choose custom bitwidth $i \in [0...k-1]$
  - Given $n$ constants, $k^n$ configurations are possible

- But, for a given constant, not all $k$ configurations lead to different cost,
  
  e.g., given 5-bit constant 0.11101, SD recoding gives

  - 5-bit = .11101 = 1.00101 $\Rightarrow$ 2 adds
  - 4-bit = .1110 = 1.0010 $\Rightarrow$ 1 adds
  - 3-bit = .111 = 1.001 $\Rightarrow$ 1 adds
  - 2-bit = .11 = 0.11 $\Rightarrow$ 1 adds
  - 1-bit = .1 = 0.1 $\Rightarrow$ 0 adds
  - 0-bit = 0 = 0 $\Rightarrow$ 0 adds

Recall all constants up to 19-bits can be reduced to 5 adds
Search Methods

- Global Bitwidth
  - all constant assigned the same bitwidth
  - very fast (small search space), but only works well in some cases
- Greedy Search
  - starting with maximum bitwidth, in each round, choose one constant
    to be reduced by 1-bit that minimizes quality loss
    (also go bottom-up instead of top-down)
  - local minima traps are possible
- Evolutionary Search
  - start with a population of random configurations
  - in each round
    1. breed a new generation by crossbreeding and mutations
    2. select from generation the fittest members
    3. repeat new round
  - local minima traps

Outline

- DSP transform algorithms
- Algorithm manipulation for robustness
- Multiplication by constants
- Search Methods
- Results
Interaction between Transforms, Q and Search

- Goal: given a transform and a required Q threshold, find an approximation to the transform that requires the fewest additions
- Transforms and Q tested

<table>
<thead>
<tr>
<th>Transform</th>
<th>Quality Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-pt. DCT-II</td>
<td>8.82 dB coding gain (cg)</td>
</tr>
<tr>
<td>16-pt. DFT</td>
<td>Convolution error = 1</td>
</tr>
<tr>
<td>32-pt. DCT-II</td>
<td>Limited Compliance (LC) MP3 decoder*</td>
</tr>
<tr>
<td>18x36 IMDCT</td>
<td>LC MP3 decoder*</td>
</tr>
</tbody>
</table>

- 3 searches methods were compared
- entire framework implemented as part of SPIRAL (www.spiral.net)


Example: Evolutionary Search

Evolutionary Search DCT of size 8 with 12 constants
- Q = cg > 8.82, exact DCT has 8.8259
- constant bit length in [0..31]

Choosing 31 bits for all constants: 126 additions
**Summary of Search Comparison**

<table>
<thead>
<tr>
<th>Number of Additions (fewer is better)</th>
<th>8 pt. DCT-II (8.82 dB cg)</th>
<th>16 pt. DFT (conv. err = 1)</th>
<th>32 pt. DCT-II (LC MP3)</th>
<th>18x36 IMDCT (LC MP3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial (31 bits)</td>
<td>126</td>
<td>500</td>
<td>1222</td>
<td>643</td>
</tr>
<tr>
<td>global</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greedy (top-down)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greedev (bottom-up)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

One search method alone is not sufficient — each search performs differently depending on transform and quality measure.

**Approximation of DCT within JPEG**

- Approximate DCT-II inside JPEG while retain images of reasonable quality
  - \( Q \) = Peak Signal to Noise Ratio (decibels) of decompressed JPEG image against the original uncompressed input image.

  \[
  \text{PSNR} = 20 \times \log_{10} \left( \frac{255}{\text{RMSE}} \right)
  \]

  \[
  \text{RMSE} = \sqrt{\frac{1}{512 \times 512} \sum_{i=1}^{512} \sum_{j=1}^{512} [D(i, j) - O(i, j)]^2}
  \]

- \( Q \) Threshold
  - Test Image: Lena, 512x512 pixel, 8-bit grayscale
  - PSNR must be at least 30 decibels or image becomes noticeably lossy.
Approximation of DCT within JPEG

- Before approximating, the original DCT requires 261 additions and produces a Lena image with a PSNR of 37.6462 dB.

<table>
<thead>
<tr>
<th>Method</th>
<th># Additions</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>global</td>
<td>37</td>
<td>30.0354</td>
</tr>
<tr>
<td>evolutionary</td>
<td>67</td>
<td>36.5323</td>
</tr>
<tr>
<td>greedy (t-d)</td>
<td>28</td>
<td>32.4503</td>
</tr>
</tbody>
</table>

- Compare constants global vs. greedy search:
  - Global: [3/2, 3/2, 3/2, 3/2, 3/2, 3/2, 3/2, 1/2, -1/2, 1, -1/2, -1/2, 1/2, -1/2, -1/4, 1/2, -1/4]
  - Greedy: [3/2, 1, 1, 1, 1, 1, 1/2, -1/2, 1, -1/2, 0, 1/2, 0, -1, 1, -1, 0, 1/2, -1/4]
- Greedy succeeds in zeroing 3 constants that affect the high frequency (HF) outputs 'thrown away' by JPEG

*Base on source from Independent JPEG Group (IJG), http://www.iijg.org*

Summary

- Application specific tuning yields ample opportunities for optimization
- The optimization flow can be automated
  - algorithm selection and manipulation
  - arithmetic reduction through search
  - arbitrary quality measures supported
- Details of the arithmetic reduction is non-trivial
  - non-monotonic relation between Q and C
  - different search methods succeed in different scenarios
- The results of this study needs to be combined with other aspects of DSP domain-specific high-level synthesis