

# *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

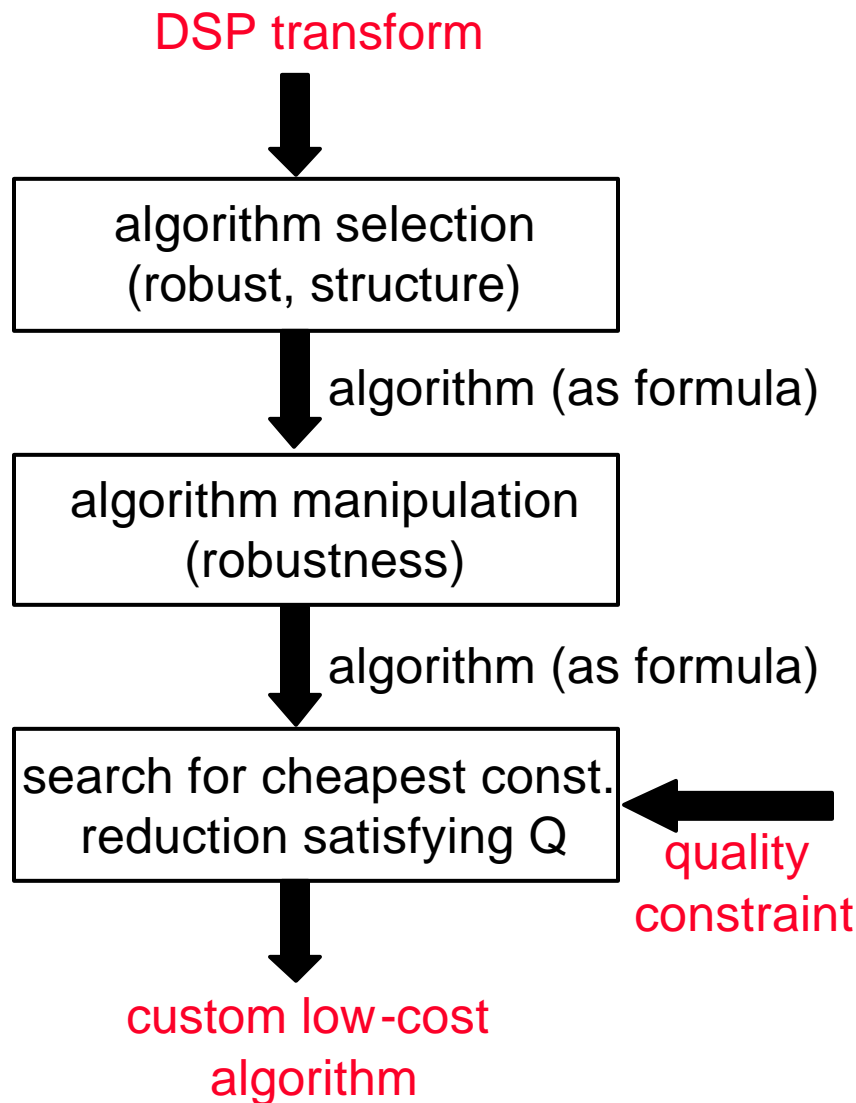
- e.g. DFT, DCTs, WHT, DWTs, ....
- ubiquitously used, often in computation intensive kernels
- comprised of additions and multiplication-by-constant
- **applications: multimedia, bio-metric, image/data processing . . . . .**

## ✍ Light-weight hardware implementations

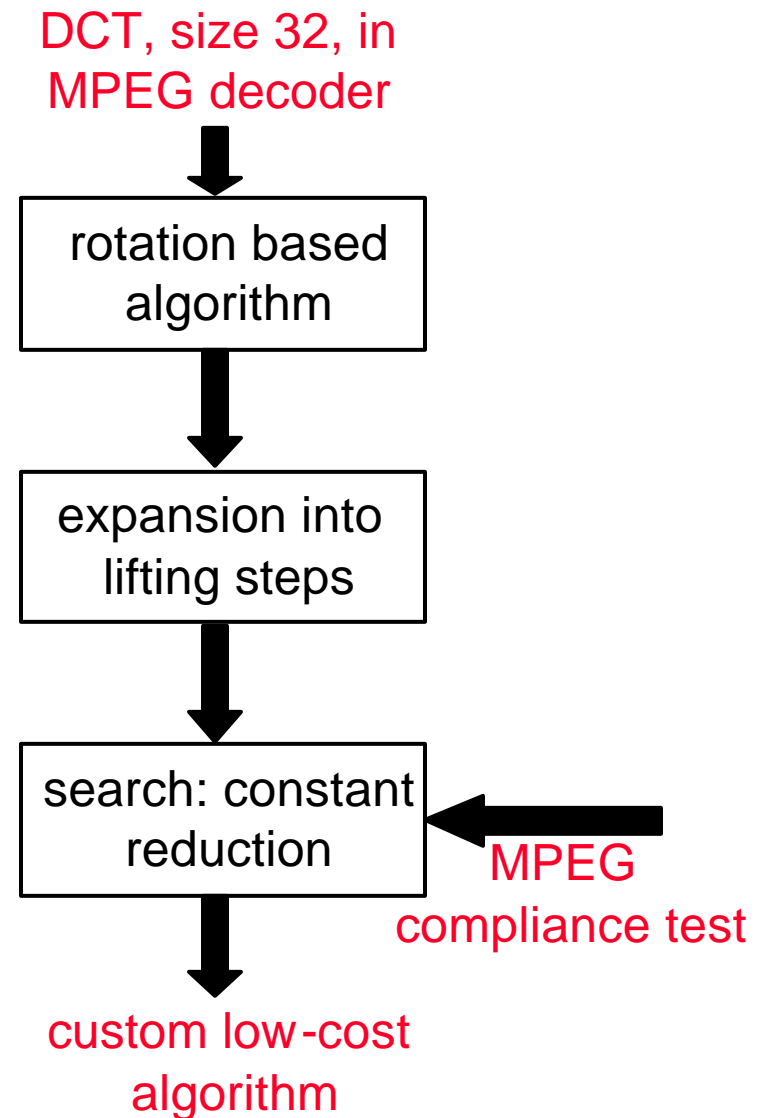
- fixed-point data format
- multiplierless: mult-by-constant as shifts and adds
- **problem 1:** output quality reduced by cost-saving measures  
*(reducing the bitwidth of data and constants)*
- **problem 2:** different applications have vastly different quality metric and requirements  
*? need application specific tuning*

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

# Our Automatic Flow



Example



# Related Work

---

- ✍ Liang/Tran, “Fast Multiplierless Approximation of the DCT with the Lifting Scheme,” *IEEE Trans. Sig. Proc.*, 49(12) 2001, pp. 3032-3044
  - examined arithmetic cost reduction for DCT size 8
  - steps performed by hand, exhaustive search
- ✍ Fang/Rutenbar/Püschel/Chen, “Toward Efficient Static Analysis of Finite-Precision Effects in DSP Applications via Affine Arithmetic Modeling,” *Proc. DAC 2003*
  - 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](http://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):

1	1	1	1	0	1	0	0	1	0	0	1	1	0	0	0	0
1	$i$	1	$-i$	0	1	0	1	0	1	0	0	1	0	0	1	0
1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	0	0
1	$-i$	1	$i$	0	1	0	1	0	0	1	0	1	0	1	0	1

Fourier transform

Diagonal matrix (twiddles)

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

Kronecker product

Identity

Permutation



allows for computer generation/manipulation  
(provided by SPIRAL)

# Example: DCT size 8

[(2,5)(4,7)(6,8),8]

$(diag(1,1/\sqrt{2}) \cdot R_{3\pi/8} \cdot R_{15\pi/16} \cdot R_{21\pi/16})$

$(I_2 \cdot (DFT_2 \cdot I_3) \cdot I_2) \cdot (5,6),8]$

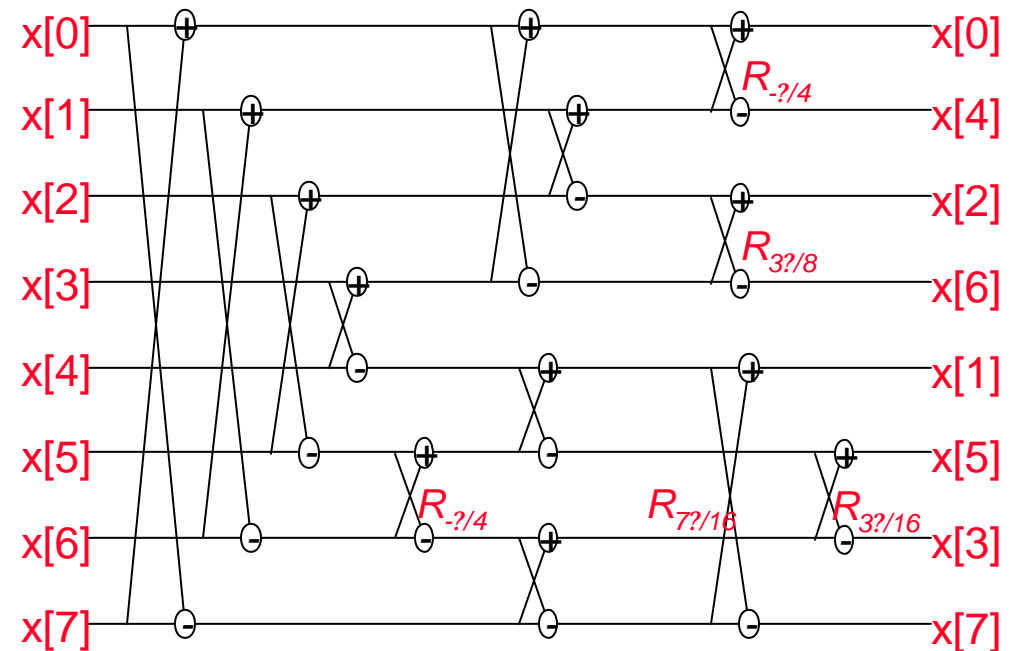
$(I_4 \cdot 1/\sqrt{2} \cdot DFT_2 \cdot I_2) \cdot (2,3,4,5,8,6,7),8]$

$(I_2 \cdot ((DFT_2 \cdot I_2) \cdot (2,3),4) \cdot (I_2 \cdot DFT_2))$

$(1,8,6,2)(3,4,5,7),8]$

as formula

(generated by SPIRAL)



as data flow diagram

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  $x$ :  $\|x - \tilde{x}\|$

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

✍ Error in transform:  $\|T - \tilde{T}\|$

- 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):  $\begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}$  or  $\begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix}$

- 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

$$R_\theta = \begin{pmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & p & 0 \\ 0 & 1 & u \\ 0 & 0 & 1 \end{pmatrix}$$

target for approximation

$$p = \frac{1 - \cos \theta}{\sin \theta} = \tan \frac{\theta}{2}, \quad u = \sin \theta$$



rotation based algorithms can be automatically expanded into LS

# Error Analysis

- rounding error in the first lifting step (third LS analogous)

$$\tilde{R}_\theta \approx R_\theta \begin{bmatrix} 1 & \theta & 0 & p \\ 0 & 1 & \theta & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx R_\theta \begin{bmatrix} \cos \theta & \sin \theta & 0 & ? \\ -\sin \theta & \cos \theta & 0 & ? \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{not magnified}$$

- rounding error in the second lifting step

$$\tilde{R}_\theta \approx R_\theta \begin{bmatrix} 1 & p & 0 & p \\ 0 & 1 & \theta & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \approx R_\theta \begin{bmatrix} \tan \frac{\theta}{2} & \tan^2 \frac{\theta}{2} \\ -\tan \frac{\theta}{2} & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\theta$  is magnified, unless  $\theta$  in  $[0, \theta/2]$  or  $[3\theta/2, 2\theta]$

**Solution:** angle manipulation

$$R_\theta \approx R_{\theta/2} R_{\theta/2} \approx R_{\theta/2} R_{\theta/2} \approx R_{\theta/2} R_{\theta/2} \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

# *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:

$y ? x_1 ? x_2$       additions

$y ? cx$       multiplication by constant

Example:

simple       $c=0.10111011 =$             5 adds (5 shifts)

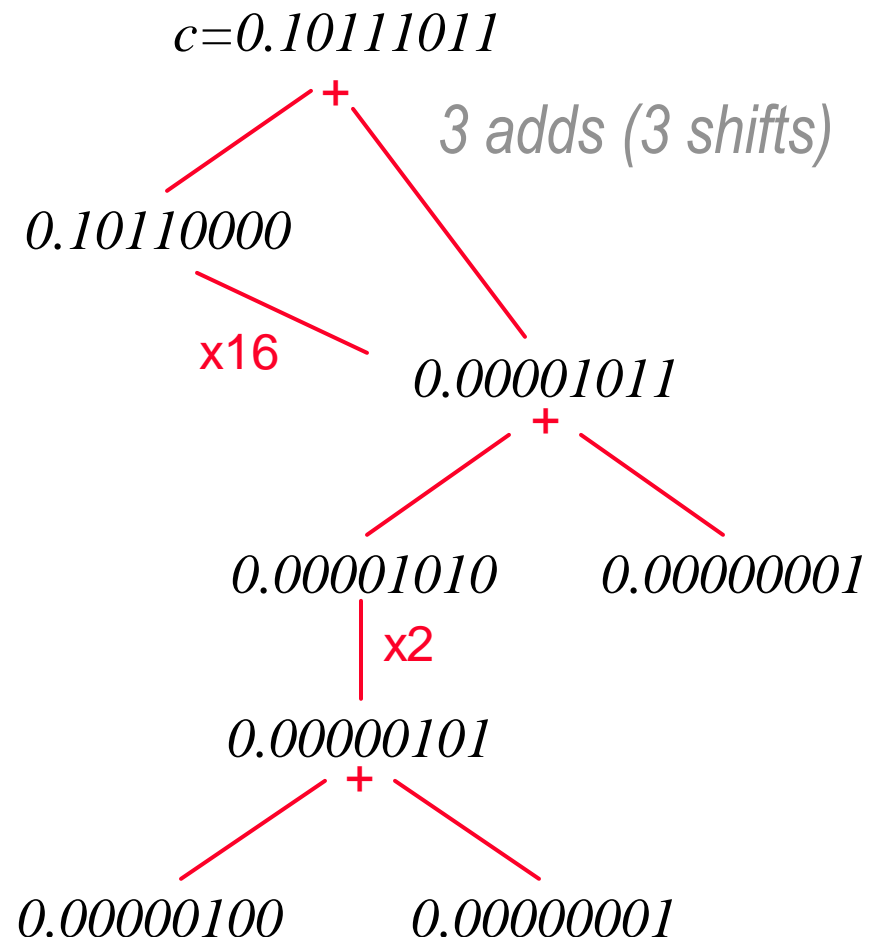
SD recoding 1       $c=0.1100\bar{1}10\bar{1}$             4 adds (3 shifts)

SD recoding 2       $c=0.11000\bar{1}0\bar{1}$             3 adds (3 shifts)

*SD recoding is not optimal*

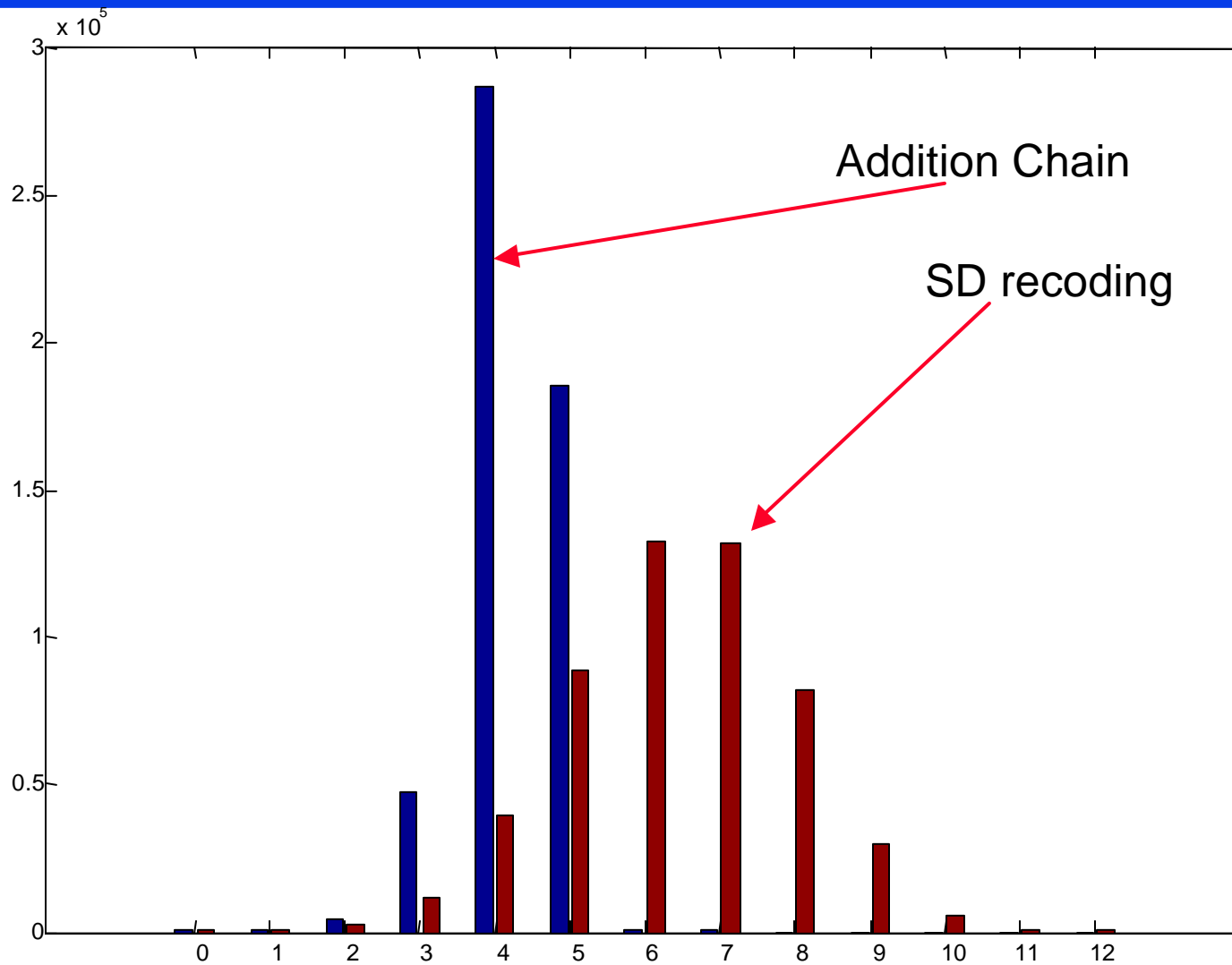
# Addition/Subtraction Chain

- ✍ 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)



\*Sebastian Egner, Philips Research, Eindhoven

# *SD recoding vs. Addition Chains*



Histogram of addition cost for all constants between 1 and  $2^{19}$

# 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  $Q$ 
  - $\|T - \tilde{T}\|$  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	= 1.00 $\bar{1}$ 01	? 2 adds
4-bit	= .1110	= 1.00 $\bar{1}$ 0	? 1 adds
<del>3-bit</del>	<del>= .111</del>	<del>= 1.00<math>\bar{1}</math></del>	<del>? 1 adds</del>
<del>2-bit</del>	<del>= .11</del>	<del>= 0.11</del>	<del>? 1 adds</del>
1-bit	= .1	= 0.1	? 0 adds
0-bit	= 0	= 0	? 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

<i>Transform</i>	<i>Quality Threshold</i>
8-pt. DCT-II	8.82 dB coding gain (cg)
16-pt. DFT	Convolution error = 1
32-pt. DCT-II	Limited Compliance (LC) MP3 decoder?
18x36 IMDCT	LC MP3 decoder?

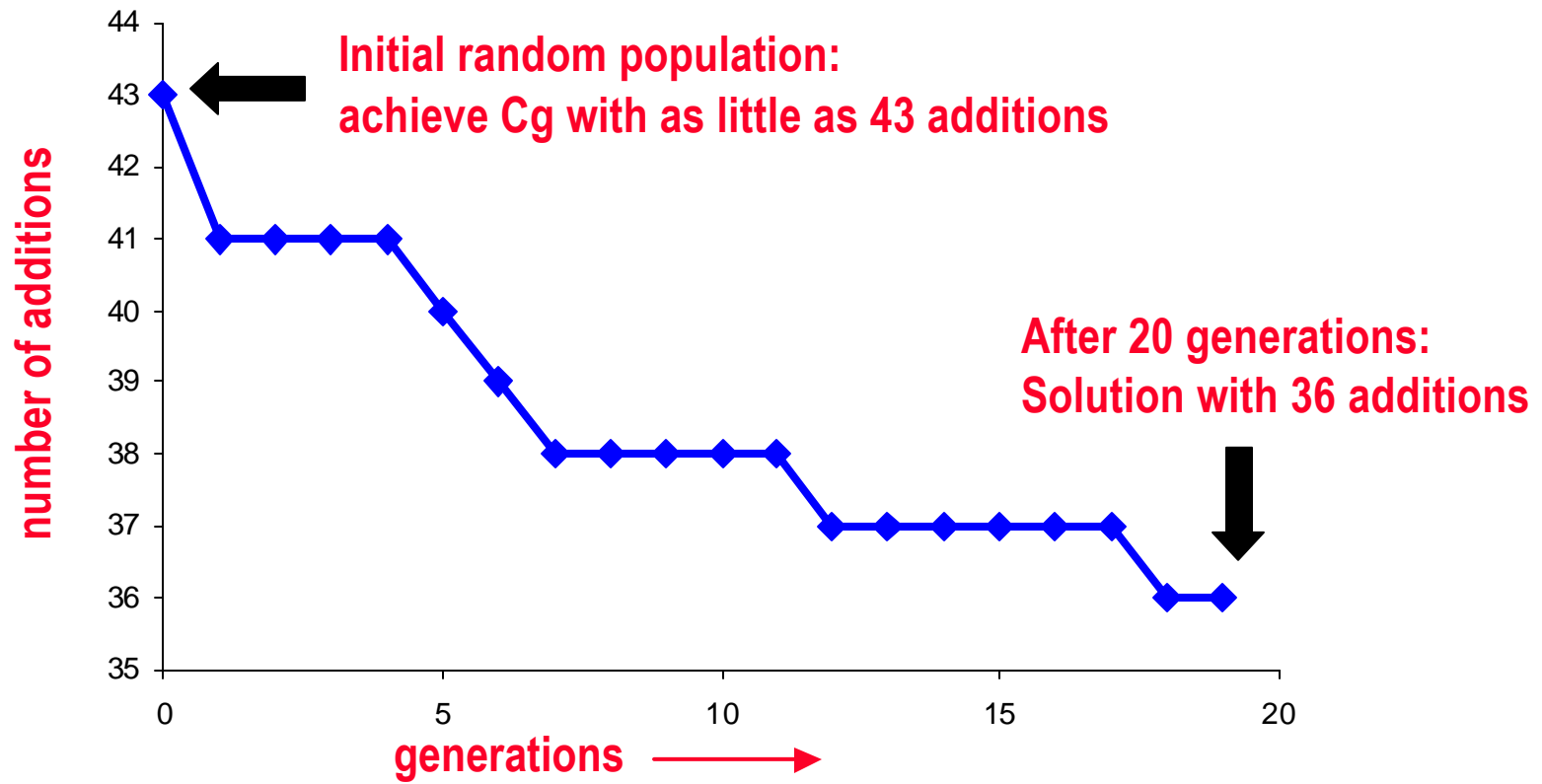
- 3 searches methods were compared
- entire framework implemented as part of SPIRAL ([www.spiral.net](http://www.spiral.net))

? *MAD Decoder by Robert Mars, <http://www.underbit.com/products/mad>*

# 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

	Number of Additions (fewer is better)			
	8 pt. DCT-II (8.82 dB cg)	16 pt. DFT (conv. err = 1)	32 pt. DCT-II (LC MP3)	18x36 IMDCT (LC MP3)
initial (31 bits)	126	500	1222	643
global	40	168	408	182
evol.	36	185	490	212
greedy (top-down)	56	158	417	170
greedy (bottom-up)	57	154	n/a	n/a

*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.

$$\text{PSNR} = 20 \log_{10} \frac{255}{\text{RMSE}}$$

$$\text{RMSE} = \sqrt{\frac{1}{512 \times 512} \sum_i \sum_j |D(i, j) - O(i, j)|^2}$$

- 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<sup>?</sup> requires 261 additions and produces a Lena image with a PSNR of 37.6462 dB.

<i>Method</i>	# Additions	PSNR
global	37	30.0354
evolutionary	67	36.5323
greedy (t-d)	<b>28</b>	<b>32.4503</b>

- 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, 1, -1, -1/4, 1/2, -1/4 ]
  - Greedy: [ 3/2, 1, 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

<sup>?</sup> Base on source from Independent JPEG Group (IJG), <http://www.ijg.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