LOW-DRIFT FIXED-POINT 8X8 IDCT APPROXIMATION WITH 8-BIT TRANSFORM FACTORS

Yuriy A. Reznik, De Hsu, Prasanjit Panda, Brijesh Pillai*

QUALCOMM Incorporated
5775 Morehouse Drive, San Diego, CA 92121

ABSTRACT

We describe an efficient algorithm for computing the Inverse Discrete Cosine Transform (IDCT) for image and video coding applications.

This algorithm was derived by converting an 8-point IDCT factorization of C. Loeffler, A. Ligtenberg, and G. Moschytz into a scaled form, leaving 8 multiplications by irrational factors inside the transform. The key advantage of such a modification is that these factors can be sufficiently accurately represented by 8-bit integer values, resulting in a very small dynamic range of variables inside the transform. Our scaled 1D transform can be implemented either by using 8 multiplications, 26 additions and 6 shifts or (in a multiplier-less fashion) by using only 44 additions and 18 shifts.

This implementation fully complies with the new MPEG IDCT precision standard (ISO/IEC 23002-1, replacement of former IEEE 1180 specification), and shows remarkably low drift in decoding of H.263, MPEG-2, and MPEG-4 bitstreams produced by reference software encoders (employing 64-bit floating-point DCT and IDCT implementations).

Index Terms— DCT, IDCT, factorization, multiplier-less algorithms

1. INTRODUCTION

The implementation of many existing image and video coding standards (such as JPEG, H.261, H.263, MPEG-1, MPEG-2, and MPEG-4 part 2) require the implementation of integer-output approximations of the 8x8 inverse discrete cosine transform (IDCT), and forward discrete cosine transform (DCT).

All these standards require decoders to use an approximation that is within a specified degree of precision relative to the integer valued IDCT function, defined as follows:

$$\hat{f}_{yx} = \lfloor f_{yx} + 1/2 \rfloor \ , \qquad (1)$$

where

$$f_{yx} = \sum_{u=0}^{7}\sum_{v=0}^{7} F_{vu}\frac{c_u c_v}{4}\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\ ,$$

$$c_w = \begin{cases} 1/\sqrt{2}, & \text{if } w = 0 \\ 1, & \text{if } w \neq 0 \end{cases}\ ,$$

$F_{v\,u}$ – input DCT coefficients ($u, v = 0...7$), and

$\hat{f}_{y\,x}$ – reconstructed pixel values ($y, x = 0...7$).

In order to satisfy precision requirements of MPEG and ITU video standards, an IDCT approximation must pass several tests. Most of these tests, defined in IEEE 1180 [3] and ISO/IEC 23002-1 [2] specifications, produce sequences of random-generated test matrices F^i_{vu} ($i = 1\ldots10000$), and measure distances between reference and approximate IDCT values (reconstructed matrices $\hat{f}^i_{yx}$ and $\hat{g}^i_{yx}$ correspondingly), using the following set of metrics:

$p = \max_{y,x,i} |\hat{f}^i_{yx} - \hat{g}^i_{yx}|$ – peak pixel error ($p \leqslant 1$),

$d_{yx} = \sum_i \hat{f}^i_{yx} - \hat{g}^i_{yx}$ – mean pixel error ($\max|d_{yx}| \leqslant 0.015$),

$M = \sum_{y,x} d_{y\,x}$ – mean error ($|m| \leqslant 0.0015$),

$e_{yx} = \sum_i (\hat{f}^i_{yx} - \hat{g}^i_{yx})^2$ – mean square error ($\max e_{yx} \leqslant 0.06$),

$N = \sum_{y,x} e_{y\,x}$ – overall mean square error ($n \leqslant 0.02$),

Here, the values in brackets specify the minimum necessary IDCT precision required for use in MPEG and ITU-T video coding standards.

However, passing these tests does not yet guarantee high quality of decoded video, particularly in situations with low quantization noise and long runs of predicted (P-type) frames or macroblocks. This is why, in selecting an IDCT design, it is always a good practice to use additional tests, such as ones measuring the *drift* (difference between reconstructed video frames in encoder and decoder) caused by the use of this approximate IDCT design in the decoder.

In this paper we describe the design of a low-complexity fixed-point 8x8 IDCT algorithm, that passes all formal requirements of MPEG and ITU-T video coding standards and

*Corresponding author. Email: yreznik@ieee.org

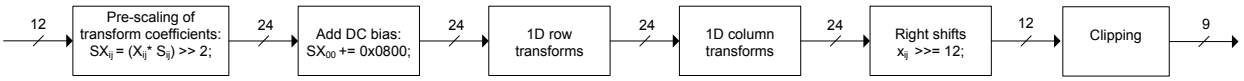

Fig. 1. Fixed-point 8x8 IDCT architecture.

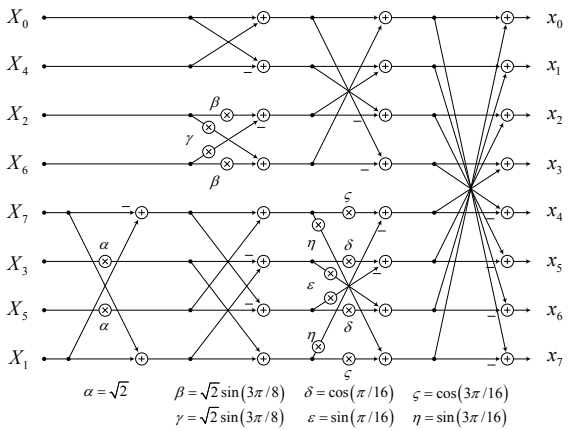

Fig. 2. LLM IDCT factorization.

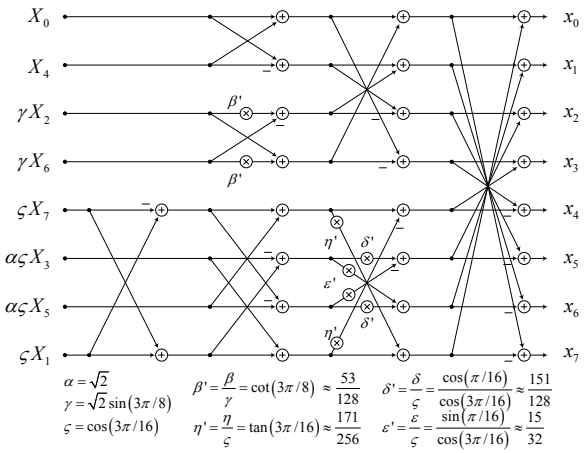

Fig. 3. LLM factorization converted to a scaled form.

shows a remarkably low drift in decoding of H.263, MPEG-2, and MPEG-4 SP bitstreams.

A particular feature of our proposed algorithm is a very compact representation of fixed point factors inside the transform – they need at most 8 bits of precision. This property helps with reducing dynamic range of all intermediate values inside the transforms and yields very efficient multiplier-based and multiplier-less implementations.

2. DESCRIPTION OF FIXED-POINT IDCT DESIGN

As underlying 8-point IDCT factorization for our fixed point design we have chosen a well-known factorization of C. Loeffler, A. Ligtenberg, and G. S. Moschytz [1]. In its original form (see Fig. 2), it uses 3 plane rotations and two direct multiplications, resulting in 14 multiplications by irrational factors. Overall, this factorization employs 7 unique irrational factors, denoted in Fig. 2, as $\alpha, \beta, \gamma, \delta, \epsilon, \zeta$, and η.

In our fixed point design (see Fig. 3), we further modify this factorization by moving factors α, γ, and ζ outside the transform. In effect, this creates a scaled transform with only 8 multiplications, where modified irrational factors $\beta', \delta', \epsilon'$ and η' are such that they can be very accurately approximated by using 8-bit integer fractions (see Fig. 3).

Scaling in this algorithm is done by combining 1D scale factors for rows and columns. This results in a 2D matrix of scale factors with 10 unique values, denoted as $A - J$:

$$S = \begin{pmatrix} A & B & C & D & A & D & C & B \\ B & E & F & G & B & G & F & E \\ C & F & H & I & C & I & H & F \\ D & G & I & J & D & J & I & G \\ A & B & C & D & A & D & C & B \\ D & G & I & J & D & J & I & G \\ C & F & H & I & C & I & H & F \\ B & E & F & G & B & G & F & E \end{pmatrix} \quad (2)$$

In order to achieve sufficient precision of scaling, the above values are scaled by 2^{11} before conversion to integers. In turn, the scaling step:

$$SX_{yx} = (X_{yx} * S_{yx}) >> 2\,. \quad (y, x = 0, \ldots, 7) \quad (3)$$

leaves 9 bits of precision to serve as a fixed-point "mantissa" for subsequent processing.

Following scaling, we add bias to the DC term:

$$SX'_{00} = SX_{00} + 2^{12-1}\,, \quad (4)$$

which ensures proper rounding at the final stage of the transform, where we simply downshift everything by 12 bits (where 9 bits is our fixed-point "mantissa" + 3 bits are added by LLM transform stages).

The overall structure of our 8x8 IDCT algorithm is depicted in Fig. 1. It can be seen that we need at most 24 bits of precision for all intermediate values between transform stages.

Table 1. Constant factors and multiplier-less algorithms used in our IDCT approximation.

Factor	Value	**Algorithms:** $x = x * [A, \ldots, J] \gg 2$; $x = x * [\beta', \delta']$; $y = x * [\epsilon', \eta']$;	Additions	Shifts
A	2048	$x \ll 9;$	0	1
B	1703	$x2 = -x + (x \ll 6), x3 = x2 + (x2 \ll 3), x4 = (x + x3) \ll 1, x5 = (x3 + x4) \gg 2;$	4	4
C	2676	$x2 = -x + (x \ll 3), x3 = -x + (x2 \ll 5), x5 = x3 + (x3 \ll 1);$	3	3
D	2408	$x2 = x + (x \ll 5), x3 = x2 \ll 1, x4 = x + x3, x5 = x3 + (x4 \ll 3);$	3	3
E	1416	$x2 = x \ll 4, x3 = x2 \ll 3, x4 = x3 - x2, x5 = x + x4, x6 = x3 + (x5 \ll 1);$	3	3
F	2225	$x2 = x \ll 5, x3 = x + (x2 \ll 2), x4 = x3 \ll 4, x5 = x2 + x3, x6 = (x4 + x5) \gg 2;$	3	4
G	2003	$x2 = x \ll 4, x3 = x2 - x, x4 = -x3 + (x2 \ll 5), x5 = (x3 \gg 2) + x4;$	3	3
H	3496	$x2 = x \ll 3, x3 = x2 - x, x4 = x3 \ll 1, x5 = x2 + x4, x6 = -x5 + (x4 \ll 6);$	3	3
I	3147	$x2 = x \ll 9, x3 = -x + (x \ll 4), x4 = x2 + x3, x5 = x2 + x4, x6 = (x5 \gg 2) + x4;$	4	3
J	2832	$x2 = x \ll 2, x3 = x2 - x, x4 = x3 \ll 6, x5 = x2 + x4, x6 = x5 + (x2 \ll 7);$	3	3
β'	$53/128$	$y = x, x2 = x \gg 2, x3 = x + x2, x4 = x2 - x, y = (x3 \gg 5) - (x4 \gg 1);$	3	3
δ'	$151/128$	$x2 = x - (x \gg 4), x3 = x2 + (x \gg 7), y = x2 \gg 2; x = y + x3;$	3	3
ϵ'	$15/32$			
η'	$171/256$	$x2 = (x \gg 3) - x; x3 = x + (x2 \gg 3); y = x3 - (x3 \gg 2);$	3	3

Table 2. Accuracy measurements and complexity of our IDCT approximation.

Implementation	Precision					Complexity	
	$\mathbf{p}$ (1)	$\max \mathbf{e}_{yx}$ (0.06)	$\mathbf{N}$ (0.02)	$\max \lvert\mathbf{d}_{yx}\rvert$ (0.015)	$\mathbf{M}$ (0.0015)	1D	2D
Multiplier-based	1	0.0278	0.0191	0.0043	0.000223	8m, 26a, 6s	(K+128)m, 417a, (K+160)s
Multiplier-less	1	0.0277	0.0191	0.0044	0.000205	44a, 18s	Km, 705a, (K+352)s

Numerical values of scale-factors, factors inside the transform, and possible multiplier-less algorithms for computation of products are presented in Table 1.

It can be seen that all factors used inside the transform are just 8-bit quantities, which reduces dynamic range of intermediate values inside the 1D transforms, and enables its efficient programming and execution on many of the existing DSP platforms.

The multiplier-less factorizations, presented in Table 1, show that each of such multiplications need at most 3 additions and 3 shifts to compute, making the entire 8-point transform computable using just 44 additions and 18 shifts. This modification may be appealing for custom circuit designs.

3. PERFORMANCE OF OUR PROPOSED ALGORITHM

As required by MPEG standards, we measure the performance of our IDCT approximation by computing the IEEE 1180 - ISO/IEC 23002-1 metrics described earlier. In Table 2, we report the worst case results for each of the metrics measured across all tests. We report these metrics for both multiplier-based and multiplier-less implementations of our approximation. As shown in the table, all results fall within the tolerances acceptable for use in MPEG video coding standards. It can also be noted that multiplier-based and multiplier-less implementations have almost the same performance.

We estimate implementation costs of our algorithms in Table 2 in terms of 1D and 2D complexities. 1D complexity means complexity of executing our 8-point scaled LLM transform, and 2D complexity includes complexities of 16 iterations of 1D transforms plus complexity of scaling and right shifts at the end of the transform. In all cases the numbers of multiplications denoted by 'm', the numbers of additions – by 'a', and the numbers of shifts – by 's'.

Parameter 'K' in Table 2, denotes the number of non-zero DCT coefficients on the input of the IDCT. Since typically, in video decoding process, input data to IDCT originate from a list of non-zero coefficients in the 8x8 block, it is often possible (and convenient) to execute scaling only for those non-zero coefficients.

Furthermore, in many image and video codecs, it is also possible to simply merge factors involved in IDCT scaling with the factors used by the corresponding inverse-quantization process. In such cases scaling can be executed in-place, without taking any extra resources.

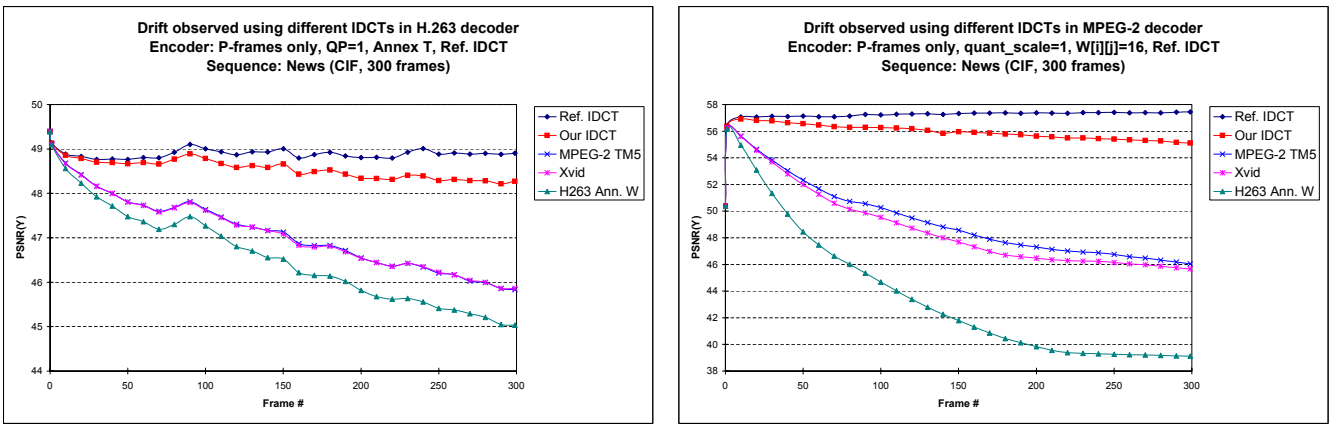

Fig. 4. IDCT drift study using H.263 (left) and MPEG-2 (right) video codecs.

4. DRIFT PERFORMANCE

In order to measure IDCT drift performance we have used reference software encoders (employing floating-point DCTs and IDCTs) of H.263, MPEG-2, and MPEG-4 P2 standards. In order to emphasize IDCT drift effects, we have also:

- forced all frames after the first one to be P-frames;
- disabled Intra-macroblock refreshes;
- forced $QP = 1$ ($quant\_scale = 1$, and $w[i, j] = 16$ in MPEG 2,4) for all frames;

In decoder we have used our IDCT approximations, and for comparison, we have also run tests for the following existing IDCT implementations:

- MPEG-2 TM5 IDCT - fixed-point implementation included in MPEG-2 reference software [6],
- XVID IDCT - a high-accuracy fixed-point implementation of IDCT in XVID opens source implementation of MPEG-4 P2 codec [7], and
- H.263 Annex W IDCT - idct algorithm specified in Annex W of ITU-T Recommendation H.263 [5].

The results of our tests for sequence "News", using H.263 and MPEG-2 codecs, are shown in Fig. 4 (data shown only for our multiplier-based implementation; but both multiplier-based and multiplier-less algorithms are almost undistinguishable drift-wise).

It can be seen, that even under such extreme test conditions the magnitude of drift using our IDCT algorithm is approximately 0.5dB in H.263 test and approximately 2dB in MPEG-2 test. In comparison, MPEG-2 TM5 IDCT shows close to 3dB and 12dB drift correspondingly. The results collected using other standard test sequences (Foreman, Bus, Mobile) show consistent drift behavior.

5. SUMMARY

In the paper we have presented the design of a low-complexity, low-drift fixed-point 8x8 IDCT design, suitable for use in existing MPEG and ITU-T video standards. Our 1D scaled transform needs 8 multiplications by 8-bit integer factors, and it is convenient for both software and hardware implementations.

6. REFERENCES

[1] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, Practical fast 1-D DCT algorithms with 11 multiplications, *in Proc. IEEE International Conference on Acoustic, Speech, and Signal Proc. (ICASSP)*, vol. 2, pp. 988-991, Feb. 1989.

[2] ISO/IEC JTC1/SC29/WG11 N7815 [23002-1 FDIS] Information technology - MPEG video technologies - Part 1: Accuracy requirements for implementation of integer-output 8x8 inverse discrete cosine transform.

[3] CAS Standards Committee of the IEEE Circuits and Systems Society, IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform, 1991.

[4] ISO/IEC 14496-2:2001 Information technology - Coding of audio-visual objects - Part 2: Visual, July, 2001.

[5] ITU-T Recommendation H.263: Video Coding for Low Bit Rate Communication, 01/2005.

[6] MPEG-2 TM5 source code and documentation: `http://www.mpeg.org/MPEG/MSSG/tm5/`

[7] XVID open source implementation of MPEG-4 ASP: `http://downloads.xvid.org/`