Efficient Fixed-Point Approximations of the 8x8 Inverse Discrete Cosine Transform

Yuriy A. Reznik*, Arianne T. Hinds†, Cixun Zhang‡, Lu Yu‡, and Zhibo Ni‡

*QUALCOMM Incorporated, 5775 Morehouse Dr., San Diego, CA 92121; USA
†IBM, Incorporated, 6300 Diagonal Hwy, Boulder, CO 80301; USA
‡Zhejiang University, Hangzhou, Zhejiang, 310027; China

ABSTRACT

This paper describes fixed-point design methodologies and several resulting implementations of the Inverse Discrete Cosine Transform (IDCT) contributed by the authors to MPEG's work on defining the new 8x8 fixed point IDCT standard – ISO/IEC 23002-2. The algorithm currently specified in the Final Committee Draft (FCD) of this standard is also described herein.

Keywords: DCT, IDCT, factorizations, diophantine approximations, multiplierless algorithms, MPEG, JPEG, ITU-T, H.261, H.263, MPEG-1, MPEG-2, MPEG-4

1. INTRODUCTION

The Discrete Cosine Transform (DCT)[1] is a fundamental operation used by the vast majority of today's image and video compression standards, such as JPEG, MPEG-1, MPEG-2, MPEG-4 (P.2), H.261, H.263, and others [2–8]. Encoders in these standards apply such transforms to each 8x8 block of pixels to produce DCT coefficients, which are then subject to quantization and encoding. The Inverse Discrete Cosine Transform (IDCT) is used in both the encoder and decoder to convert DCT coefficients back to the spatial domain.

At the time when the first image and video compression standards were defined, the implementation of DCT and IDCT algorithms was considered a major technical challenge, and therefore, instead of defining a specific algorithm for computing it, ITU-T H.261, JPEG, and MPEG standards have included *precision specifications* that must be met by IDCT implementations conforming to these standards.[9] This decision has allowed manufacturers to use the best optimized designs for their respective platforms. However, the drawback of this approach is the impossibility to guarantee exact decoding of MPEG-encoded videos on across different decoder implementations.

In early 2005, prompted by the expiration and withdrawal of a related IDCT precision specification (IEEE Standard 1180-1990[9]), MPEG has decided to improve its handling of this matter, and produce

- a new precision specification ISO/IEC 23002-1, replacing the IEEE 1180 standard, and harmonizing the treatment of precision requirements across different MPEG standards,[10] and
- a new voluntary standard, ISO/IEC 23002-2, providing specific (deterministically defined) examples of fixed-point 8x8 IDCT and DCT implementations.

The Call for Proposals[11] for the ISO/IEC 23002-2 standard was issued at the August 2005 meeting in Poznań, Poland, and during several subsequent meetings MPEG has received a number of contributions with specific algorithm proposals, as well as general optimization techniques, analyses of different factorizations, drift problem, implementation studies, and other informative documents. Many of the successive submissions have benefited from earlier ideas contributed to MPEG by other proponents, converging on key design aspects.[10] The

Further author information: (please send correspondence to Yuriy A. Reznik)
Yuriy A. Reznik: yreznik@ieee.org, phone: +1 (858) 658-1866, Arianne T. Hinds: arianne@us.ibm.com, Cixun Zhang: cixunzhang@hotmail.com, Lu Yu: yul@zju.edu.cn, Zhibo Ni: hzjimmy@hotmail.com.
The work of Lu Yu, Cixun Zhang, and Zhibo Ni was supported by NSFC under contract No. 60333020 and Project of Science and Technology Plan of Zhejiang Province under contract No. 2005C1101-03.

Committee Draft (CD) of this standard, containing a single algorithm was issued at the October 2006 meeting in Hangzhou, China. The Final Committee Draft (FCD) of this standard was reached at the April 2007 meeting in San Jose, CA,[13] and the Final Draft International Standard (FDIS) is now expected to be issued in October of 2007.

This paper describes fixed-point design methodologies and several resulting IDCT implementations contributed by the authors to this MPEG project. The algorithm currently specified in the Final Committee Draft (FCD) of this standard is also described.

Our paper is organized as follows. In Section 2, we provide background information, including definitions of the DCT and IDCT, examples of their factorizations, and review of some basic techniques used for their fixed-point implementations. In Section 2, we also explain several ideas that we have proposed for improving performance of fixed-point designs: introduction of floating factors between sub-transforms, the use of fast algorithms for computation of products by groups of factors, and techniques for minimizing rounding errors in algorithms using right shift operations. In Section 3, we show how these techniques were applied to design our proposed IDCT approximations. Finally, in Section 4, we provide a detailed description of the algorithm in the FCD of the ISO/IEC 23002-2 standard. Appendices A and B contain supplemental information and proofs of our claims.

2. BACKGROUND INFORMATION & MAIN IDEAS USED IN THIS WORK

2.1 Definitions

The order-8, one-dimensional (1D) type II[1] Discrete Cosine Transform (DCT), and its corresponding Inverse DCT (IDCT) are defined as follows:

$$F_u = \frac{c_u}{2} \sum_{x=0}^{7} f_x \cos \frac{(2x+1)u\pi}{16}, \quad u = 0, \ldots, 7, \tag{1}$$

$$f_x = \sum_{u=0}^{7} \frac{c_u}{2} F_u \cos \frac{(2x+1)u\pi}{16}, \quad x = 0, \ldots, 7, \tag{2}$$

where $c_u = 1/\sqrt{2}$, when $u = 0$, and $c_u = 1$ otherwise.

The definitions for the two-dimensional (2D) versions of these transforms are:

$$F_{vu} = \frac{c_u c_v}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} f_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}, \quad v, u = 0, \ldots, 7, \tag{3}$$

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

where $f_{y\,x}$ $(y, x = 0...7)$ denote input spatial domain values (for image and video coding – values of pixels or prediction residuals), and $F_{v\,u}$ $(u, v = 0...7)$ denote transform domain values, or transform coefficients. When used in image or video coding applications, the pixel values are normally assumed to be in the range of $[-256, 255]$, while the transform coefficients are in the range of $[-2048, 2047]$.

Mathematically, these are linear, orthogonal, and separable transforms. That is, a 2D transform can be decomposed into a cascade of 1D transforms applied successively to all rows and then to all columns in the matrix. This property of separability is often exploited by implementors to mitigate the complexity of an entire 2D transform into a much simpler set of 1D operations.

2.2 Precision Requirements for IDCT Implementations in MPEG and ITU-T Standards

As described previously, specifications of MPEG, JPEG, and several ITU-T video coding standards, do not require IDCTs to be implemented exactly as specified in (4). Rather, they require practical IDCT implementations to produce integer output values $\tilde{f}_{yx}$ that fall within certain specified tolerances from outputs of an ideal IDCT rounded to the nearest integer:

$$\hat{f}_{yx} = \text{sign}\,(f_{yx})\ \lfloor |f_{yx}| + 1/2 \rfloor\ . \tag{5}$$

The precise specification of these error tolerances, and how they are to be measured for a given IDCT implementation under test, is defined by the MPEG IDCT precision standard: ISO/IEC 23002-1.[14] This standard includes several tests using a pseudo-random input generator originating from the former IEEE Standard 1180-1990,[9] as well as additional tests required by MPEG standards.

A summary of the error metrics defined by the IEEE 1180 | ISO/IEC 23002-1 specification is provided in Table 1. Here, the variable $i = 1, \ldots, Q$ indicates the index of a pseudo random input 8x8 matrix used in a test, and $Q = 10000$ (or in some tests, $Q = 100000$) denotes the total number of sample matrices. The tolerance for each metric is provided in the last column of this table.

Table 1. The IEEE 1180 | ISO/IEC 23002−1 pseudo-random IDCT test metrics and conditions

Error metric			Description	Test condition		
$\mathbf{p}$	=	$\max_{y,x,i} \lvert\hat{f}^i_{yx} - \tilde{f}^i_{yx}\rvert$	Peak pixel error	$\mathbf{p}$	$\leqslant$	1
$\mathbf{d}_{yx}$	=	$\frac{1}{Q}\sum_i \hat{f}^i_{yx} - \tilde{f}^i_{yx}$	Pixel mean error	$\max_{y,x} \lvert\mathbf{d}_{yx}\rvert$	$\leqslant$	0.015
$\mathbf{m}$	=	$\frac{1}{64}\sum_{y,x} d_{y\,x}$	Overall mean error	$\lvert\mathbf{m}\rvert$	$\leqslant$	0.0015
$\mathbf{e}_{yx}$	=	$\frac{1}{Q}\sum_i (\hat{f}^i_{yx} - \tilde{f}^i_{yx})^2$	Mean square error	$\max_{y,x}\ \mathbf{e}_{yx}$	$\leqslant$	0.06
$\mathbf{n}$	=	$\frac{1}{64}\sum_{y,x} e_{y\,x}$	Overall mean square error	$\mathbf{n}$	$\leqslant$	0.02

It should be noted that IEEE 1180 | ISO/IEC 23002 error metrics satisfy the following chain of inequalities

$$\begin{array}{lcl} \max \mathbf{e}_{yx} & \geqslant & \max |\mathbf{d}_{yx}| \quad \geqslant \quad |\mathbf{m}|\,, \\ \max \mathbf{e}_{yx} & \geqslant & \mathbf{n} \end{array} \tag{6}$$

which implies that max $\mathbf{e}_{yx}$ or *peak mean square error (pmse)* metric is the strongest one in this set.

Among the additional (informative) tests provided in the ISO/IEC 23002-1/FPDAM1 specification,[15] the so-called "linearity test"* is of notable interest. This test requires the reconstructed pixel values produced by an IDCT implementation under test $\tilde{f}_{yx}$ to be symmetric with respect to the sign reversal of its input coefficients:

$$\tilde{f}_{yx}\,(-F_{vu}) = -\tilde{f}_{yx}\,(F_{vu})\ , \qquad F_{vu} = \left[\begin{array}{ll} z, & v = t\,,\ u = s \\ 0, & \text{otherwise} \end{array}\right. , \quad \begin{array}{l} z = 1, 3, \ldots, 527\,, \\ v, u, t, s = 0, \ldots, 7\,. \end{array} \tag{7}$$

This test was motivated by the observation that in the decoding of static regions within consecutive video frames, the decoder will reconstruct small (zero-mean, symmetrically distributed) differences, that will normally negate each other over time (across the sequence of frames). If the IDCT implementation does not satisfy this property (7), then the mismatch between IDCT outputs may instead accumulate, eventually producing a remarkable visible degradation in the quality of the reconstructed video.[16,17]

2.3 DCT/IDCT Factorizations

Much of the original research for designing fast implementations of DCT transforms was focused on finding DCT *factorizations*, resulting in the minimum number of multiplications by irrational factors. Many factorizations have been derived by utilizing other known fast algorithms, such as the classic Cooley-Tukey FFT algorithm, or by applying systematic approaches, such as a decimation in time, or a decimation in frequency.[1] The formal

*Considering general definition of linearity $f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$, for some operator $f(.)$, this test only considers a case when $\alpha = 0$, and $\beta = -1$. Therefore, this is rather a "sign-symmetry test".

setting of this problem and an upper bound for the multiplicative complexity of transforms of orders 2^n can be found in E. Feig and S. Winograd.[20]

In one special case of the order-8 two-dimensional DCT/IDCT, the least complex direct 2D factorization is described by E. Feig and S. Winograd.[21] Their implementation requires 96 multiplication and 454 addition operations for the computation of the complete set of 2D outputs. The same paper further describes an efficient *scaled* 8x8 DCT implementation, that requires only 54 multiplication, 462 addition, and 6 shift operations.[21] The latter of these transforms is refered to as a *scaled* transform because all of its outputs must be scaled (i.e. multiplied by fixed, possibly irrational, constants) so that each output will equate to the relative output of a nonscaled DCT. In some applications, such as JPEG, and several video coding algorithms, this process of scaling can be implemented jointly with the process of quantization (by factoring together the scale constants with the corresponding quantization values), thereby resulting in significant computational savings.

Some of the most efficient and well-known 1D DCT factorizations include the scaled factorization of Y. Arai, T. Agui and M. Nakajima (AAN),[25] (only 5 multiplication and 29 addition operations), and the non-scaled factorizations of W. Chen, C.H. Smith and S.C. Fralick,[23] B.G. Lee,[24] M. Vetterli and A. Ligtenberg (VL),[26] and C. Loeffler, A. Ligtenberg and G. Moschytz (LLM).[27] The VL and LLM algorithms are the least complex among known non-scaled designs, and require only 11 multiplication and 26 addition operations.

We note that the suitability of each of these factorizations to the design of fixed point IDCT algorithms has been extensively analyzed in the course of work for the ISO/IEC 23002-2 standard.[28–31]

2.4 Fixed-Point Approximations

As described previously, implementations of the DCT/IDCT require multiplication operations with irrational constants (i.e. the cosines). Clever factorizations can only reduce the number of such "essential" multiplications, but not eliminate them altogether. Hence, in the design of implementations of the DCT/IDCT, one is usually tasked with finding ways of approximately computing products of these irrational factors by using fixed-point arithmetic.

One of the most common and practical techniques for converting floating-point to fixed-point values is based on the approximations of irrational factors α_i by dyadic fractions:

$$\alpha_i \approx a_i / 2^k \,, \tag{8}$$

where both a_i and k are integers. In this way, multiplication of x by factor α_i permits the implementation of a very simple approximation in integer arithmetic as follows:

$$x\alpha_i \approx (x * a_i) >> k \,; \tag{9}$$

where $>>$ denotes the bit-wise right shift operation.

In some transform designs, right shift operations in approximations (9) can be delayed to later stages of the implementation, or done at the very end of the transform, but the more complex operations, such as multiplications for each non-trivial constant α_i still need to be performed in the algorithm.

The key variable that affects the precision and complexity of these dyadic rational approximations (8) is the number of precision bits k. In software designs, this parameter is often constrained by the width of registers (e.g. 16 or 32) and the consequence of not satisfying such a design constraint can easily result in the doubling of execution time for the transform. In hardware designs, the parameter k affects the number of gates needed to implement adders and multipliers. Hence, one of the basic goals in fixed point designs is to minimize the total number of bits k, while maintaining sufficient accuracy of approximations.

2.5 Improving Precision of Dyadic Rational Approximations

Without placing any specific constraints on values for α_i, and assuming that for any given k, the corresponding values of nominators a_i are chosen such that:

$$\left|\alpha_i - a_i/2^k\right| = 2^{-k} \left|2^k\alpha_i - a_i\right| = 2^{-k} \min_{z\in\mathbb{Z}} \left|2^k\alpha_i - z\right| \,,$$

we can conclude that the absolute error of approximations in (8) should be inversely proportional to 2^k:

$$\left|\alpha_i - a_i/2^k\right| \leqslant 2^{-k-1}\,.$$

That is, each extra bit of precision (i.e. incrementing k), should reduce the error by half.

Nevertheless, it turns out that this rate can be significantly improved if the values $\alpha_1, \ldots, \alpha_n$ that we are trying to approximate can be simultaneously scaled by some additional parameter ξ.

We claim the following (the proof for which is provided in Appendix A):

LEMMA 2.1. *Let $\alpha_1, \ldots, \alpha_n$ be a set of n irrational numbers ($n \geqslant 2$). Then, there exist infinitely many $n+2$-tuples $a_1, \ldots, a_n, k, \xi$, with $a_1, \ldots, a_n \in \mathbb{Z}$, $k \in \mathbb{N}$, and $\xi \in \mathbb{Q}$, such that*

$$\max\left\{\left|\xi\,\alpha_1 - a_1/2^k\right|, \ldots, \left|\xi\alpha_n - a_n/2^k\right|\right\} < \frac{n}{n+1}\,\xi^{-1/n}\,2^{-k(1+1/n)}\,. \tag{10}$$

In other words, if the algorithm can be altered such that all of its irrational factors $\alpha_1, \ldots, \alpha_n$ can be pre-scaled by some parameter ξ, then we might be able to find approximations whose absolute error decreases as fast as $2^{-k(1+1/n)}$. For example, when n=2, this means 50% higher effectiveness in the usage of bits. For large sets of factors $\alpha_1, \ldots, \alpha_n$, however, this gain will be smaller.

These observations suggest that *we can significantly improve the precision* of a fixed-point IDCT design by splitting it into a set of smaller blocks (or sub-transforms) with alterable common factors, and then adjust these factors such that they yield high-accuracy solutions predicted by Lemma 2.1.

2.6 Reducing Complexity of Multiplications

The dyadic approximations shown in (8, 9) already reduce the problem of computing products by irrational constants to multiplications by integers. However, integer multiplications can still be computationally "expensive" to use on many existing platforms, and in such cases it becomes desirable to find ways to compute these products without using general purpose multipliers.

To illustrate this idea, consider a multiplication by an irrational factor $1/\sqrt{2}$, using its 5-bit dyadic approximation: 23/32. By looking at the binary bit pattern of $23 = 10111$ and substituting each "1" with an addition operation, we can compute a product by 23 as follows:

$$x * 23 = (x << 4) + (x << 2) + (x << 1) + x\,.$$

This approximation requires 3 addition and 3 shift operations. By further noting that the last 3 digits form a series of "1"s, we can instead use:

$$x * 23 = (x << 4) + (x << 3) - x\,. \tag{11}$$

which reduces the complexity to just 2 shift and 2 addition operations.

In engineering literature, the sequences of operations "+" associated with isolated digits "1", or "+" and "-" associated with beginnings and ends of runs "1 . . . 1" are commonly referred to as a "Canonical Signed Digit" (CSD) decomposition.[34] This is a well known and frequently used tool in the design of multiplierless circuits.[38] However, CSD decompositions do not always produce results with the lowest numbers of operations. For example, considering an 8-bit approximation of the same factor $1/\sqrt{2} \approx 181/256 = 10110101$, we find that its CSD decomposition:

$$x * 181 = (x << 7) + (x << 5) + (x << 4) + (x << 2) + x\,,$$

needs 4 addition and 4 shift operations. But, by rearranging the computations and *reusing intermediate results*, a more efficient algorithm can be constructed:

$$\begin{array}{lcll} x2 & = & x + (x << 2); & // \quad 101 \\ x3 & = & x2 + (x << 4); & // \quad 10100 \\ x4 & = & x3 + (x2 << 5); & // \quad 10110101 \quad = x * 181 \end{array}$$

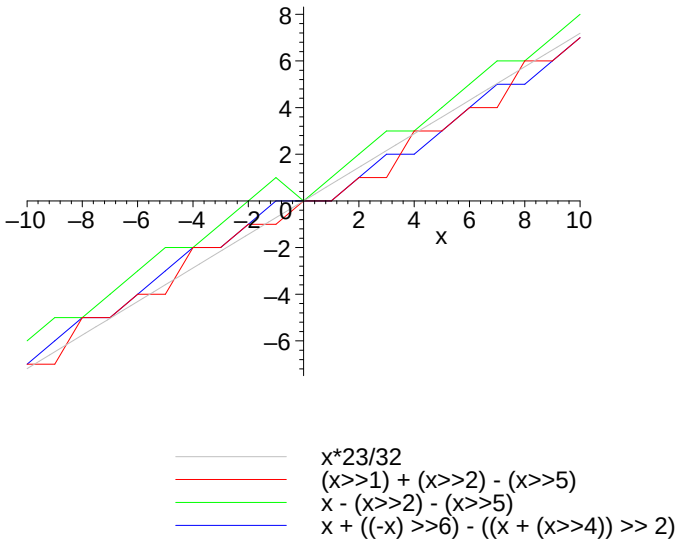

Figure 1. Errors in multiplierless implementations of a product by 23/32.

This approximation requires only 3 addition and 3 shift operations.

An even more dramatic reduction in complexity can be achieved by performing *joint factorization of simultaneous products by multiple integer constants*. For example, consider the task of computing products by two constants: $99 = 1100011$ and $239 = 11101111$. The use of CSD decompositions

$$\begin{array}{rcl} x * 99 & = & (x << 6) + (x << 5) + (x << 1) + x; \\ x * 239 & = & (x << 8) - (x << 4) - x; \end{array}$$

results in a total complexity of 5 addition and 5 shift operations. At the same time, by using a *joint factorization* of these two products, the same task can be simplified by the following implementation:

$$\begin{array}{rcll} x2 & = & x + (x << 5); & // \quad 100001 \\ x3 & = & x2 << 2; & // \quad 10000100 \\ x4 & = & x3 - x2; & // \quad 1100011 \quad = x * 99 \\ x5 & = & x3 + x4 + (x << 3); & // \quad 11101111 \quad = x * 239 \end{array}$$

which needs only 4 addition and 3 shift operations.

In the context of the IDCT design, such algorithms can be used for simultaneous computation of products by pairs of factors in transform butterflies. Moreover, since in each butterfly there are typically two variables that need to be multiplied by the same factors, such computations can easily be done in parallel.

In passing, we should note that finding optimal (i.e. with fewest numbers of additions and/or shifts) algorithms for computing multiplications by integer constants has been an area of active and fruitful research during the last few decades[33–40]. It has been established that this problem is NP-complete,[36] and numerous fast heuristic algorithms have been proposed for solving it approximately.[34, 37, 39, 40]

2.7 Minimizing Errors in Multiplierless Algorithms using Right Shifts

Another family of techniques for computation of products by dyadic fractions (8) can be derived by allowing the use of right shifts as elementary operations.

For example, considering a factor $1/\sqrt{2} \approx 23/32 = 0.10111$, and using right shift and addition operations according to its CSD decomposition, we obtain[†]:

$$x * 23/32 \ \sim \ (x >> 1) + (x >> 2) - (x >> 5). \tag{12}$$

[†]Hereafter, by writing $a(x) \sim b(x)$ for some functions $a(.)$ and $b(.)$, we imply that there exists a constant $\delta \geqslant 0$, such that for all x: $|a(x) - b(x)| \leqslant \delta$.

or (by further noting that $1/2 + 1/4 = 1 - 1/4$):

$$x * 23/32 \;\sim\; x - (x >> 2) - (x >> 5). \tag{13}$$

Yet another (although, somewhat less obvious) way of computing product by the same factor is:

$$x * 23/32 \;\sim\; x - ((x + (x >> 4)) >> 2) + ((-x) >> 6)\,. \tag{14}$$

We present plots of values produced by these algorithms in Figure 1. It can be noted that they all compute values that approximate products by fraction 23/32, however, the errors in each of these approximations are different. For example, the algorithm (13) produces all positive errors, with a maximum magnitude of 55/32. The algorithm (12) has more balanced errors, with the magnitude of oscillations within $\pm 65/64$. Finally, the algorithm (14) produces perfectly sign-symmetric errors with oscillations in $\pm 7/8$.

The sign-symmetry property of an algorithm $A_{a_i,b}(x) \rightsquigarrow xa_i/2^b$ means that for any ($x \in \mathbb{Z}$):

$$A_{a_i,b}(-x) = -A_{a_i,b}(x)\,,. \tag{15}$$

It can be shown that sign-symmetry also implies that for any such algorithm with $A_{a_i,b}(0) = 0$, and any N:

$$\sum_{x=-N}^{N} \left[A_{a_i,b}(x) - x\frac{a_i}{b}\right] = 0\,, \tag{16}$$

that is, zero-mean error for any symmetric range of input values.

This property is very important in the design of signal processing algorithms, as it minimizes the probability that rounding errors introduced by fixed-point approximations will accumulate. Below we will establish the existence of right-shift-based sign-symmetric algorithms for computing products by dyadic fractions and provide upper bounds for their complexity.

Given a set of dyadic fractions $a_1/2^b, \ldots, a_m/2^b$, we define an algorithm

$$A_{a_i,\ldots,a_m,b}(x) \rightsquigarrow \left(xa_1/2^b, \ldots, xa_m/2^b\right) \tag{17}$$

as the following sequence of steps:

$$x_1, x_2, \ldots, x_t\,, \tag{18}$$

where $x_1 := x$, and where subsequent values x_k ($k = 2, \ldots, t$) are produced by using one of the following elementary operations:

$$x_k := \left[\begin{array}{lll} x_i >> s_k; & 1 \leqslant i < k,\ s_k \geqslant 1; & \text{or} \\ -x_i; & 1 \leqslant i < k; & \text{or} \\ x_i + x_j; & 1 \leqslant i, j < k; & \text{or} \\ x_i - x_j; & 1 \leqslant i, j < k,\ i \neq j. & \end{array}\right. \tag{19}$$

The algorithm terminates when there exists indices $j_1, \ldots, j_m \leqslant t$, such that:

$$x_{j_1} \sim x * a_1/2^b, \ldots, x_{j_m} \sim x * a_m/2^b\,. \tag{20}$$

We state the following (the proofs for which are provided in Appendix B):

THEOREM 2.2. *For any $m, b \in \mathbb{N}$ and $a_i, \ldots, a_m \in \mathbb{Z}$, there exist algorithms $A_{a_i,\ldots,a_m,b}$ (17-20), which are sign-symmetric. That is, for any $x \in \mathbb{Z}$: $A_{a_i,\ldots,a_m,b}(-x) = -A_{a_i,\ldots,a_m,b}(x)$.*

THEOREM 2.3. *The lowest possible number of shifts in algorithms (17-20) satisfying the sign-symmetry property is at most twice the lowest possible number of shifts in algorithms without this property.*

THEOREM 2.4. *The lowest possible total number of instructions in algorithms (17-20) satisfying the sign-symmetry property is at most four times the lowest possible total number of instructions in algorithms without this property.*

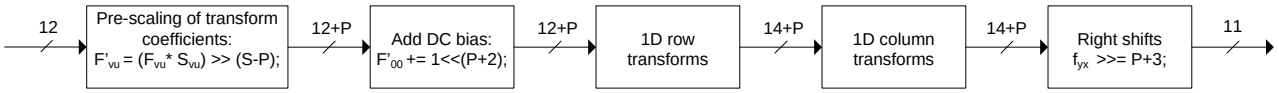

Figure 2. Fixed-point 8x8 IDCT architecture.

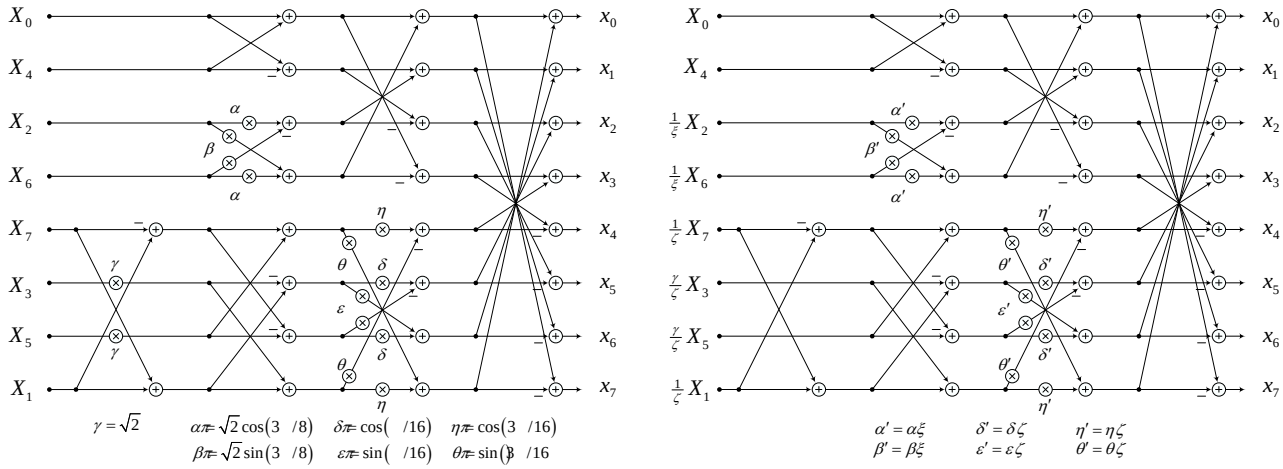

Figure 3. Loeffler-Ligtenberg-Moschytz IDCT factorization (left), and its generalized scaled form (right). Parameters (ξ, ζ) represent additional "floating factors" that we use for finding efficient fixed-point approximations.

We should point out that these are very simple and rather coarse complexity bounds, and that in many cases, the complexity overhead for achieving sign-symmetry is not that high. Moreover, when complexity considerations are paramount, one can pick algorithms that are sign-symmetric for most, but not all values of x in the expected range of this variable. In many cases, such "almost symmetric" algorithms can also be the least complex for a given set of factors.

In the design of our IDCTs we have used an exhaustive enumeration process for searching for the best algorithms (17-20) with symmetric (or at least well-balanced) rounding errors. As additional criteria for selection of such algorithms, we have used estimates of mean, variance, and magnitude (maximum values) of errors that they produce. In assessing their complexity, we have counted the numbers of operations, as well as the longest execution path, and maximum number of intermediate registers needed for computations.

3. DESIGN OF FIXED-POINT APPROXIMATIONS OF THE 8X8 IDCT

The overall architecture used in the design of the proposed fixed-point IDCT algorithms is shown in Figure 2, which can be characterized by its separable and scaled features. The scaling stage is performed with a single 8x8 matrix that is precomputed by factoring the 1D scale factors for the row transform with the 1D scale factors for the column transform. The scaling stage is also used to pre-allocate P bits of precision to each of the input DCT coefficients thereby providing a fixed-point "mantissa" for use throughout the rest of the transform. Other key features of this architecture include simplicity, compactness / cache-efficiency, and flexibility of its interface, by allowing the potential for merging of scaling and quantization logic in video and image codec implementations.

As the underlying basis for scaled 1D transform design, we use a variant of the well-known factorization of C. Loeffler, A. Ligtenberg, and G.S. Moschytz[27] with 3 planar rotations and 2 independent factors $\gamma = \sqrt{2}$ (see Figure 3). This choice has been made empirically based on an extensive analysis of fixed-point designs derived from other known algorithms, including variants of AAN,[25] VL,[26] and LLM[27] factorizations.[31]

In order to allow efficient rational approximations of constants α, β, δ, ϵ, η, and θ within the LLM factorization, we introduce two floating factors ξ and ζ, and apply them to two sub-groups of these constants as follows (see also Figure 3, right flowgraph):

$$\begin{array}{ll} \xi: & \alpha' = \xi\alpha, \quad \beta' = \xi\beta; \\ \zeta: & \delta' = \zeta\delta, \quad \epsilon' = \zeta\epsilon, \quad \eta' = \zeta\eta, \quad \theta' = \zeta\theta; \end{array} \tag{21}$$

We invert these multiplications by ξ and ζ in the scaling stage by multiplying each input DCT coefficient with the respective reciprocal of ξ and ζ. That is, we pre-compute a vector of scale factors for use in the scaling stage prior to the first in the cascade of 1D transforms.

$$\sigma = (1, 1/\zeta, 1/\xi, \gamma/\zeta, 1, \gamma/\zeta, 1/\xi, 1/\zeta)^T \ . \tag{22}$$

These factors are subsequently merged into a scaling matrix which is precomputed as follows:

$$\Sigma = \sigma\, \sigma^T 2^S = \begin{pmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{A} & \mathrm{D} & \mathrm{C} & \mathrm{B} \\ \mathrm{B} & \mathrm{E} & \mathrm{F} & \mathrm{G} & \mathrm{B} & \mathrm{G} & \mathrm{F} & \mathrm{E} \\ \mathrm{C} & \mathrm{F} & \mathrm{H} & \mathrm{I} & \mathrm{C} & \mathrm{I} & \mathrm{H} & \mathrm{F} \\ \mathrm{D} & \mathrm{G} & \mathrm{I} & \mathrm{J} & \mathrm{D} & \mathrm{J} & \mathrm{I} & \mathrm{G} \\ \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{D} & \mathrm{A} & \mathrm{D} & \mathrm{C} & \mathrm{B} \\ \mathrm{D} & \mathrm{G} & \mathrm{I} & \mathrm{J} & \mathrm{D} & \mathrm{J} & \mathrm{I} & \mathrm{G} \\ \mathrm{C} & \mathrm{F} & \mathrm{H} & \mathrm{I} & \mathrm{C} & \mathrm{I} & \mathrm{H} & \mathrm{F} \\ \mathrm{B} & \mathrm{E} & \mathrm{F} & \mathrm{G} & \mathrm{B} & \mathrm{G} & \mathrm{F} & \mathrm{E} \end{pmatrix} \tag{23}$$

where $A - J$ denote unique values in this product:

$$A = 2^S,\ B = \frac{2^S}{\zeta},\ C = \frac{2^S}{\xi},\ D = \frac{\gamma 2^S}{\zeta},\ E = \frac{2^S}{\zeta^2},\ F = \frac{2^S}{\xi\zeta},\ G = \frac{\gamma 2^S}{\zeta^2},\ H = \frac{2^S}{\xi^2},\ I = \frac{\gamma 2^S}{\xi\zeta},\ J = \frac{\gamma^2 2^S}{\zeta^2},$$

and S denotes the number of fixed-point precision bits allocated for scaling.

This parameter S is chosen such that it is greater than or equal to the number of bits P for the mantissa of each input coefficient. This allows scaling of the coefficients F_{vu}, to be implemented as follows:

$$F'_{vu} = (F_{vu} * S_{vu}) >> (S - P)\,, \tag{24}$$

where $S_{vu} \approx \Sigma_{vu}$ denote integer approximations of values in matrix of scalefactors (23).

At the end of the last transform in the series of 1D transforms, the P fixed-point mantissa bits (plus 3 extra bits accumulated during executions of each of the 1D stages[‡]) are simply shifted out of the transform outputs by right shift operations:

$$f_{yx} = f'_{yx} >> (P + 3)\,. \tag{25}$$

To ensure a proper rounding of the computed value in (25), we add a bias of 2^{P+2} to the values f'_{yx} prior to the shifts. This rounding bias is implemented by perturbing the DC coefficient prior to executing the first 1D transform:

$$F"_{00} = F'_{00} + 2^{P+2}\,.$$

Using this architecture, the task of finding fixed point IDCT implementations is now reduced to finding sets of integer approximations of factors

- $A, B, C, D, E, F, G, I, J$ – the coefficients in the matrix of scale factors (23), and
- α', β', δ', ϵ', η', θ' – the factors inside the 1D transforms

and algorithms for computing products by them. Global parameters that can also be adjusted are:

- P – the number of fixed-point mantissa bits;
- S – the number of bits used to implement the scaling stage such that $S \geqslant P$;
- k – the number of bits used for the approximations of factors within 1D transforms.

[‡]The LLM factorization naturally causes all quantities on the output to be multiplied by a factor of $2\sqrt{2}$.[27] This results in 1.5 bits mantissa expansion during row- and column- passes, and 3 bits accumulated at the end of the 2D transform.

Table 2. Fixed-Point IDCT Approximations Derived by Using our Design Framework.

Details \ Algorithm		L16	L1	L0	L2	Z0a 23002-2	Z1	Z4
2D scale factors	A	16384	2048	2048	2048	1024	1024	8192
	B	15852	1703	2275	2275	1138	867	8037
	C	22930	2729	2556	2446	1730	1278	11051
	D	22418	2408	3218	3218	1609	1226	11366
	E	15337	1416	2528	2528	1264	734	7885
	F	22185	2269	2840	2718	1922	1082	10842
	G	21690	2002	3575	3574	1788	1038	11151
	H	32090	3637	3190	2923	2923	1595	14908
	I	31374	3209	4016	3844	2718	1530	15333
	J	30675	2832	5056	5055	2528	1468	15770
Transform factors	α'	519/512	151/128	113/128	113/128	41/128	111/256	6573/16384
	β'	413/2048	15/32	45/256	45/256	99/128	67/64	31737/32768
	δ'	55/64	1	1533/2048	1533/2048	113/128	18981/16384	16379/16384
	ϵ'	147/256	171/256	1/2	1/2	719/4096	59/256	1629/8192
	η'	99/256	13/32	111/256	29/64	1533/2048	16091/16384	27771/32768
	θ'	239/256	251/256	67/64	35/32	1/2	21/32	4639/8192
Bit-usage	S	14	11	11	11	10	10	13
	P	**3**	10	10	10	10	10	13
	k	11	**8**	11	11	12	14	15
Complexity	1D	$50a, 20s$	$44a, 18s$	$\mathbf{42}a, 20s$	$48a, \mathbf{12}s$	$44a, 20s$	$48a, 26s$	$56a, 32s$
	2D	$801a, 384s$	$705a, 352s$	$673a, 384s$	$769a, 256s$	$705a, 384s$	$769a, 480s$	$897a, 576s$
	2D+S	$1017a, 624s$	$901a, 552s$	$829a, 576s$	$925a, 448s$	$865a, 576s$	$925a, 680s$	$1125a, 792s$
Precision	$\mathbf{p}$	1	1	1	1	1	1	1
	$\max \mathbf{e}_{yx}$	0.022800	0.022600	0.019400	0.019800	0.024800	0.007500	0.001300
	$\mathbf{n}$	0.018577	0.017923	0.015397	0.016094	0.017866	0.005384	0.000425
	$\max \lvert\mathbf{d}_{yx}\rvert$	0.003800	0.006000	0.004000	0.007500	0.004300	0.001800	0.000700
	$\lvert\mathbf{m}\rvert$	0.000573	0.000245	0.000425	0.000342	0.000166	0.000153	0.000053
Linearity test		fail	fail	fail	fail	pass	pass	pass
Ext. dynamic range test		fail	fail	fail	fail	pass	pass	pass

Notably, this list of parameters does not include the values for our "floating factors" – ξ and ζ. The reason for their exclusion is that these factors are needed only for establishing the relationship between the values of the factors inside the transform (21) and the values for the scale factors (22). The actual values for ξ and ζ are absorbed by the rational fractions assigned to each factor.

This design framework has been used for the design of several IDCT approximations submitted to MPEG.[30,31] The search for the above parameters and algorithms has been organized such that for each candidate transform approximation we were able to measure: (a) the IDCT precision in terms of accuracy metrics, and (b) the number of operations needed for its implementation. This approach allowed us to identify transforms with the best achievable complexity and precision tradeoffs.

3.1 Examples of IDCT Designs

We summarize the values of parameters and performance characteristics of several algorithms designed using this framework in Table 2. These algorithms have the following particular features:

L16 – an algorithm passing all normative ISO/IEC 23002-1 precision tests using *the lowest achievable number of mantissa bits:* $P = 3$. This implies that this algorithm is implementable on 16-bit platforms.

L1 – an ISO/IEC 23002-1 compliant IDCT approximation with *the lowest achievable number of bits in approximations of transform factors:* $k = 8$.

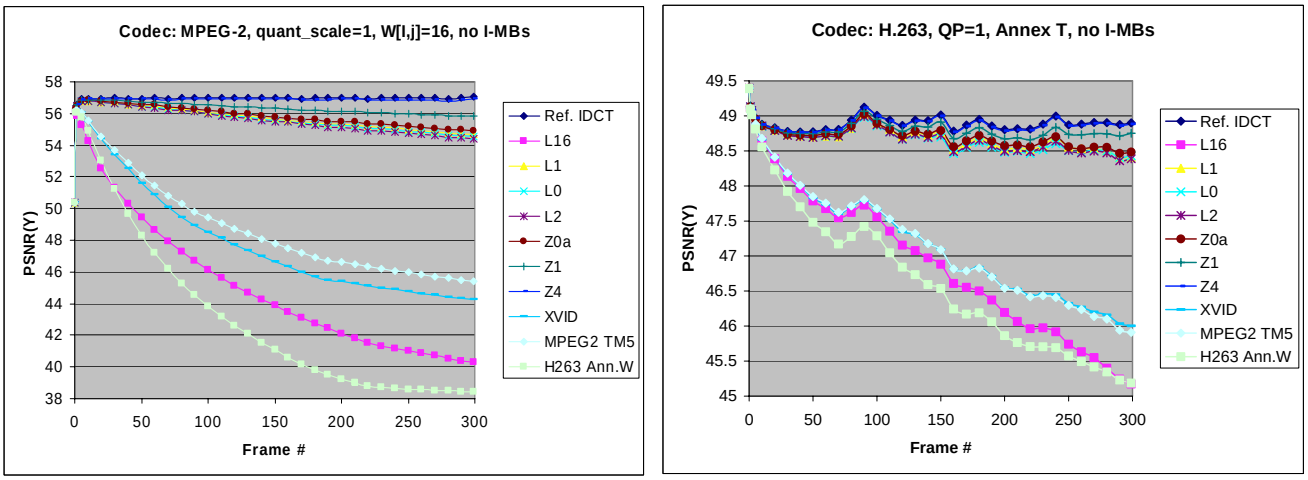

Figure 4. Drift performance of our IDCT approximations using sequence "News" (CIF, 300 frames).

L0 – an ISO/IEC 23002-1 compliant IDCT approximation with the *lowest achievable number of additions:* 42 *additions per 1D transform.* Since the underlying factorization contains 26 additions + 12 multiplications, this means that each multiplication in algorithm L0 is implemented by using only $1 + \frac{1}{3}$ additions.

L2 – an ISO/IEC 23002-1 compliant IDCT approximation with the *lowest achievable number of shifts (*12 *shifts per 1D transform).* Since the underlying factorization contains 12 multiplications, this means that each multiplication in algorithm L2 is implemented by using only 1 shift operation.

Z0a – a higher-accuracy (linearity-test compliant) algorithm, selected for the Final Committee Draft (FCD) of the ISO/IEC 23002-2 standard.[13]

Z1 – an algorithm that was originally selected for the Committee Draft (CD) of ISO/IEC 23002-2 standard.[12] This algorithm is considerably more complex than the FCD design (Z0a).

Z4 – an ultra-high precision IDCT approximation.

In characterizing IDCT precision, Table 2 lists worst-case values of ISO/IEC 23002-1 metrics, collected over all normative pseudo-random tests.[14] In describing complexity, letters "a" are used to denote the numbers of additions and letters "s" – to denote the numbers of shifts necessary to implement these algorithms. The "1D" complexity section provides the numbers of operations necessary to implement each scaled one-dimensional transform. The "2D" complexity section shows the total numbers of operations necessary to implement the scaled 2D transform. Finally, the "2D+S" complexity section shows the total numbers of operations necessary to implement the complete 2D IDCT transform, including scaling (assuming that all input coefficients are non-zero).

The collection of algorithms L16, L0, L1, and L2 illustrates extremes that can be reached if the goal is to simply pass the basic set of precision requirements for IDCT implementations in MPEG standards. Algorithms Z0a, Z1, and Z4 strive to go beyond this basic goal and have some nice additional properties. For example, they all pass the linearity test,[16,17] pass extended dynamic range tests,[15] and perform better in so-called IDCT-drift tests described in the next section.

3.2 Drift Performance Analysis

The IEEE 1180 | ISO/IEC 23002-1 tests define mandatory requirements for IDCT implementations in MPEG and ITU-T video coding standards. However, passing them does not always guarantee high quality of the decoded video, particularly in situations with low quantization noise and long runs of predicted (P-type) frames or macroblocks.[42] This is why, in evaluating an IDCT design, it is important to use additional tests, such as those measuring *drift* (difference between reconstructed video frames in encoder and decoder) caused by the use of this approximate IDCT design in the decoder.

In order to measure the drift performance of our IDCTs 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 the 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,[43]
- XVID IDCT - a high-accuracy fixed-point implementation of IDCT in XVID (MPEG-4 P2) codec,[44] and
- H.263 Annex W IDCT - a 16-bit IDCT algorithm specified in Annex W of ITU-T Recommendation H.263.[7]

The results of our tests for sequence "News", using H.263 and MPEG-2 codecs, are shown in Figure 4. It can be observed, that the high-precision algorithm Z4 has virtually no drift. Then algorithms Z1 and Z0a follow with their worst case accumulated drift contained approximately within 0.5dB in H.263 tests, and within 2dB in MPEG-2 tests. Algorithms L0, L2, L1, then follow with their worst case drift being slightly worse (approximately 0.625dB in H.263 and 2.25dB in MPEG-2 tests). The rest of the algorithms, however, perform much worse.

The MPEG-2 TM5 and XVID implementations show approximately 3dB drift in H.263 test, and almost 12dB drift in the MPEG-2 environment. Even worse is the drift behavior of the 16-bit algorithms in our tests L16 and H.263 Annex W: they both show approximately 4dB drift in H.263 test, and 18-20dB drift in the MPEG-2 test.

These results illustrate that IDCT drift performance can be significantly affected by the choice of the fixed-point architecture, and its parameters. In particular, in testing numerous implementations produced using our scaled, LLM-based framework, we have observed that drift performance is most significantly affected by our "mantissa" parameter P. For the majority of algorithms: L0, L1, L2, Z0a, and Z1, reducing the mantissa by $2, 3$, sometimes even by 4 bits had almost no effect on most of the IEEE 1180 | ISO/IEC 23002-1 precision metrics, and yet, each such bit had a major effect (about 1-2dB per bit difference) in drift tests. The algorithm L16 is an extreme example of such a mantissa reduction process (leaving only $P = 3$), and it is obviously unacceptable in terms of drift performance. For this reason, we have retained at least $P = 10$ bits of mantissa in the design of most of our algorithms proposed to MPEG.

4. THE ISO/IEC 23002-2 FCD FIXED POINT IDCT ALGORITHM

The overall architecture and 1D factorization flowgraph used by ISO/IEC 23002-2 FCD algorithm are depicted in Figure 2 and Figure 3 correspondingly. All integer factors and parameters used in this algorithm are listed in Table 2 under the column "23002-2".

This transform allocates $P = 10$ bits for the fixed-point mantissa, and uses the same number of bits for specifying the scale factors $S = 10$. This cancels out right shifts in the processing of input coefficients (24), and makes the scaling stage of this transform particularly simple:

$$F'_{vu} = F_{vu} * S_{vu}\,, \quad v, u = 0, \ldots, 7\,, \tag{26}$$

$$F''_{00} = F'_{00} + 2^{12}\,, \tag{27}$$

where F_{vu} are input coefficients, and the DC-term adjustment (27) is done to ensure proper rounding at the end of the transform:

$$f_{yx} = f'_{yx} >> 13\,, \quad y, x = 0, \ldots, 7\,.$$

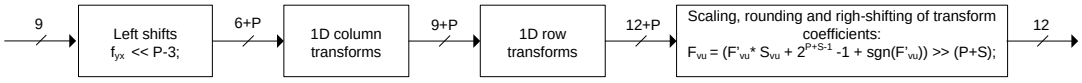

Figure 5. ISO/IEC 23002-2 FDCT architecture.

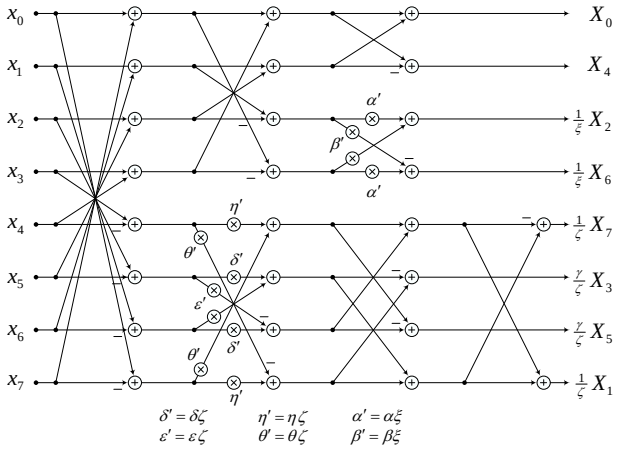

Figure 6. Factorization employed in ISO/IEC 23002-2 FDCT design.

The maximum total number of bits needed by all variables in this transform is 26 bits, which assumes full 12-bit dynamic range of reconstructed pixel values, which is sufficient to cover even extreme cases of quantization noise expansion, as described in [41] .

There are three groups of rational dyadic factors processed by this algorithm (see Figure 3, and Table. 2):

- $\alpha' = 41/128$ and $\beta' = 99/128$ – in the butterfly with coefficients X_2 and X_6,
- $\delta' = 113/128$ and $\epsilon' = 719/4096$ – in the butterfly with coefficients X_3 and X_5, and
- $\eta' = 1533/2048$ and $\theta' = 1/2$ – in the butterfly with coefficients X_1 and X_7.

The computation of products by these factors is performed as follows:

$x2$	=	$x + (x >> 5);$	//	1.00001	
$x3$	=	$x2 >> 2;$	//	0.0100001	
$x4$	=	$x3 + (x >> 4);$	//	0.0101001	$\sim x * 41/128 = x * \alpha'$
$x5$	=	$x2 - x3;$	//	0.1100011	$\sim x * 99/128 = x * \beta'$
$x2$	=	$(x >> 3) - (x >> 7);$	//	0.0001111	
$x3$	=	$x2 - (x >> 11);$	//	0.00011101111	
$x4$	=	$x2 + (x3 >> 1);$	//	0.001011001111	$\sim x * 719/4096 = x * \epsilon'$
$x5$	=	$x - x2;$	//	0.1110001	$\sim x * 113/256 = x * \delta'$
$x2$	=	$(x >> 9) - x;$	//	-0.111111111	
$x3$	=	$x >> 1;$	//	0.1	$\sim x/2 = x * \theta'$
$x4$	=	$(x2 >> 2) - x2;$	//	0.10111111101	$\sim x * 1533/2046 = x * \eta'$

The combined complexity of all these operations is only 9 addition and 10 shift operations. Therefore, the average complexity for computing a single multiplication in this algorithm is only $9/6 = 1.5$ addition and $10/6 \approx 1.66$ shift operations. In comparing this with traditional fixed point-point implementation of products by factors:

$$x * \eta' \sim (x * 1533 + 1024) >> 11$$

which includes an addition (for proper rounding) and a shift, we conclude that the effective cost of each integer multiplication in ISO/IEC 23002-2 FCD algorithm is only 0.5 addition + 0.66 shift operations.

The total complexity of computing each scaled 1D transform in this algorithm is 44 addition and 20 shift operations. The description of a complete 1D transform in C programming language requires only 50 lines.[13] Extra C-code needed to describe the full 2D version takes only 20 lines.

The scaling of transform coefficients can be done either outside of the transform, e.g. in the quantization stage, thereby taking advantage of the sparseness of the input matrix of coefficients, or inside the transform, by executing multiplications (26).

This algorithm passes all normative ISO/IEC 23002-1 precision tests,[14] as well as many additional tests that have been created in the process of evaluating fixed point designs in MPEG. These additional tests include MPEG-2 and MPEG-4, and T.83 (JPEG) conformance tests, drift tests with H.263, MPEG-2, and MPEG-4 encoders and decoders, as well as linearity test, and extended dynamic range tests.[15]

4.1 ISO/IEC 23002-2 FCD FDCT Design

The design of the corresponding fixed-point forward ISO/IEC 23002-2 DCT is fully symmetric relative to the IDCT design. Its overall architecture and 1D factorization are presented in Figure 5 and Figure 6 correspondingly. All integer factors and algorithms for computing products in this FDCT design are exactly the same as in the IDCT, with the only difference being simply the order in which they are executed.

The two elements in the FDCT design that are implemented differently when compared to the IDCT design are: the reservation of mantissa bits and scaling. The allocation of mantissa bits is done at the very beginning of the FDCT transform as follows:

$$f'_{yx} = f_{yx} << 7\,, \quad y, x = 0, \ldots, 7\,, \tag{28}$$

and the scaling is done at the very end, by using

$$F_{vu} = \left(F'_{vu} * S_{vu} + 2^{19} - 1 + \mathrm{sgn}(F'_{vu})\right) >> 20\,, \quad v, u = 0, \ldots, 7\,, \tag{29}$$

where

$$\mathrm{sgn}(x) = \begin{cases} 0, & \text{if} \quad x \geqslant 0 \\ 1, & \text{if} \quad x < 0 \end{cases}\,. \tag{30}$$

The use of the term (30) in rounding (29) assures that FDCT scaling is done in a sign-symmetric fashion, with a slightly wider deadzone around 0.

We note that the scaled architecture of ISO/IEC FDCT design makes it also possible to combine the final scaling stage (29-30) with the quantization process in video or image encoders, thereby enabling further complexity reductions.

5. CONCLUSIONS

In this paper we have described our proposed fixed-point IDCT design methodologies and several resulting algorithms achieving different precision/complexity characteristics. We have explained choices of the parameters in such designs, and their connection to IDCT precision and drift performance.

The fixed-point 8x8 IDCT and DCT algorithms adopted in ISO/IEC 23002-2 FCD standard are also described. Their architecture has benefited from the ideas contributed to the MPEG standardization process by multiple proponents and yielded a remarkably efficient implementation, surpassing all IEEE 1180 | ISO/IEC 23002-2 precision requirements, with low implementation complexity (requiring only 44 addition and 20 shift operations per scaled 1D transform), and performing very well in linearity, extended dynamic range, and IDCT drift tests.

6. ACKNOWLEDGEMENTS

The authors wish to thank Honggang Qi, Wen Gao, Debin Zhao, Siwei Ma, and other participants in the MPEG IDCT project for their contributions influencing the design of the ISO/IEC 23002-2 FCD algorithm. The authors also wish to thank Joan Mitchell and Gary Sullivan for their helpful comments on the manuscript of this paper.

REFERENCES

1. K. R. Rao, and P.Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, San Diego, 1990.
2. W. B. Pennebaker and J. L. Mitchell, JPEG Still Image Compression Standard, Van Nostrand Reinhold, New York, 1993.
3. J. L. Mitchell, W. B. Pennebaker, D. Le Gall, and C. Fogg, MPEG Video Compression Standard, Chapman & Hall, New York, 1997.
4. ITU-T Recommendation T.81 | ISO/IEC 10918-1: *Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines*, 1992.
5. ISO/IEC 11172: *Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2 - Video*, August, 1993.
6. ITU-T Recommendation H.262 | ISO/IEC 13818-2: *Information technology – Generic coding of moving pictures and associated audio information: Video*, 1994.
7. ITU-T Recommendation H.263: *Video Coding for Low Bit Rate Communication*, 1995.
8. ISO/IEC 14496-2: *Information technology – Coding of audio-visual objects – Part 2: Visual*, July, 2001.
9. ANSI/IEEE 1180-1990, Standard Specifications for the Implementations of 8 8 Inverse Discrete Cosine Transform, December 1990 (withdrawn by ANSI 9 September, 2001; withdrawn by IEEE 7 February, 2003).
10. G. J. Sullivan, Standardization of IDCT approximation behavior for video compression: the history and the new MPEG-C parts 1 and 2 standards, *SPIE Applications of Digital Image Processing XXX, Proc. SPIE, Vol. 6696*, August 28–31, 2007 (this conference).
11. ISO/IEC JTC1/SC29/WG11 (MPEG), *Call for Proposals on Fixed-Point IDCT and DCT Standard*, Moving Picture Experts Group (MPEG) output document N7335, Poznan, Poland, July 2005.
12. ISO/IEC JTC1/SC29/WG11 (MPEG), ISO/IEC CD 23002-2: *Fixed-point IDCT and DCT*, Moving Picture Experts Group (MPEG) output document N8479, Hangzhou, China, October 2006,
13. ISO/IEC JTC1/SC29/WG11 (MPEG), ISO/IEC FCD 23002-2: *Information technology MPEG video technologies, Part 2: Fixed-point 8x8 IDCT and DCT*, Moving Picture Experts Group (MPEG) output document N8983, San Jose, CA, April 2007.
14. ISO/IEC 23002-1: *Information technology – MPEG video technologies – Part 1: Accuracy requirements for implementation of integer-output 8x8 inverse discrete cosine transform*, December 2006.
15. ISO/IEC JTC1/SC29/WG11 (MPEG), ISO/IEC 23002-1/FPDAM1: *Information technology – MPEG video technologies – Part 1: Accuracy requirements for implementation of integer-output 8x8 inverse discrete cosine transform. Amendment 1: Software for Integer IDCT Accuracy Testing*, Moving Picture Experts Group (MPEG) output document N8981, San Jose, CA, April 2007.
16. M. A. Isnardi, Description of Sample Bitstream for Testing IDCT Linearity, Moving Picture Experts Group (MPEG) document M13375, Montreux, Switzerland, April 2006.
17. C. Zhang and L. Yu, On IDCT Linearity Test, Moving Picture Experts Group (MPEG) document M13528, Klagenfurt, Austria, July 2006.
18. Z. Ni and L. Yu, Drift Problem of Fixed-Point IDCT on News Sequence, Moving Picture Experts Group (MPEG) document M13912, Hangzhou, China, October 2006.
19. G. Sullivan, Video IDCT static-content pathological drift: Analysis and encoder techniques, Moving Picture Experts Group (MPEG) document M14077, Marrakech, Morocco, January 2007.
20. E. Feig and S. Winograd, On the multiplicative complexity of discrete cosine transforms (Corresp.), *IEEE Trans. Info. Theory*, vol. IT-38, pp. 1387–1391, July 1992.
21. E. Feig, and S. Winograd, "Fast algorithms for the discrete cosine transform", *IEEE Trans. Signal Processing*, vol. 40, pp. 2174–2193, September 1992.
22. N. Ahmed, T. Natarajan and K. R. Rao, Discrete cosine transform, *IEEE Trans. Comput*, vol.C-23, pp. 90–93, January 1974.
23. W. Chen, C. H. Smith and S. C. Fralick, A Fast Computational Algorithm for the Discrete Cosine Transform, *IEEE Trans. Comm.*, vol. com-25, No. 9, pp. 1004–1009, September 1977.
24. B. G. Lee, A new algorithm to compute the discrete cosine transform, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp.1243–1245, May 1984.

25. Y. Arai, T.Agui and M. Nakajima, A Fast DCT-SQ Scheme for Images, *Transactions of the IEICE* E71(11): 1095, November 1988. (in Japaneze)
26. M. Vetterli and A. Ligtenberg, A Discrete Fourier-Cosine Transform Chip, *IEEE Journal on Selected Areas in Communications*, Vol. 4, No. 1, pp. 49–61, January 1986.
27. C. Loeffler, A. Ligtenberg, and G. S. Moschytz, Practical fast 1-D DCT algorithms with 11 multiplications, *in Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'89)*, vol. 2, pp. 988-991, February 1989.
28. Y.A. Reznik, A.T. Hinds, N. Rijavec, Low-Complexity Fixed-Point Approximation of Inverse Discrete Cosine Transform, *In Proc. IEEE Int. Conf. Acoust., Speech, and Sig. Proc. (ICASSP'07)*, Honolulu, HI, April 15-20, 2007.
29. A. Navarro, A. Silva, Y. Resnik, A Full 2D IDCT with Extremely Low Complexity, *SPIE Applications of Digital Image Processing XXX, Proc. SPIE, Vol. 6696*, August 28–31, 2007 (this conference).
30. Y. Reznik, A. Hinds, C. Zhang, L. Yu, and Z. Ni, Response to CE on Convergence of Scaled and Non-Scaled IDCT Architectures, Moving Picture Experts Group (MPEG) document M13650, Klagenfurt, Austria, July 2006.
31. Y. Reznik, Analysis of hybrid scaled/non-scaled IDCT architectures, Moving Picture Experts Group (MPEG) document M13705, Klagenfurt, Austria, July 2006.
32. J. W. S. Cassels, *An Introduction to Diophantine Approximations*, Cambridge University Press, 1957.
33. D. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, vol. 2. Addison-Wesley, 1969.
34. A. Avizienis, Signed-digit number representations for fast parallel arithmetic, *IRE Transactions on Electronic Computers*, Vol. EC- 10, pp. 389-400. 1961.
35. A. Karatsuba and Y. Ofman, Multiplication of multidigit numbers on automata, *Soviet Phys. Doklady*, Vol. 7, No. 7, pp. 595–596, January 1963.
36. P. R. Cappello and K. Steiglitz, Some complexity issues in digital signal processing, *IEEE Trans. Acoustics, Speech Signal Proc.*, ASSP-32, 5, pp. 1037–1041, 1984.
37. R. Bernstein, Multiplication by Integer Constants, *Software - Practice and Experience*, Vol 16, No. 7, pp 641-652, 1986.
38. R.M.Hewlitt and E.S. Swartzlantler, Canonical signed digit representation for FIR digital filters, *in Proc. IEEE Workshop on Signal Processing Systems (SiPS 2000)*, pp. 416-426, 2000.
39. V. Lefèvre, *Moyens arithmétiques pour un calcul fiable*, PhD thesis, École Normale Supérieure de Lyon, Lyon, France, January 2000.
40. Y. Voronenko and M. Püschel, Multiplierless multiple constant multiplication, *ACM Trans. Algorithms*, 3, 2, pp. 11-, May, 2007.
41. M. Zhou, and J. De Lameillieure, IDCT output range in MPEG video coding, *Signal Processing: Image Communication*, Vol. 11, No. 2, pp. 137–145, Dec. 1997.
42. A.T. Hinds, Z.Ni, C.Zhang, L.Yu, and Y.A. Reznik, On IDCT Drift Problem, *SPIE Applications of Digital Image Processing XXX, Proc. SPIE, Vol. 6696*, August 28–31, 2007 (this conference).
43. MPEG-2 TM5 source code and documentation:
`http://www.mpeg.org/MPEG/MSSG/tm5/`
44. XVID open source implementation of MPEG-4 ASP:
`http://downloads.xvid.org/`

APPENDIX A. SOME FACTS FROM DIOPHANTINE APPROXIMATION THEORY AND PROOF OF LEMMA 2.1

Let θ be a real number, and let us try to approximate it by a rational fraction

$$\theta \approx p/q.$$

Here, p and q are integers and, without loss of generality, it can be assumed that $q > 0$.

Given a fixed q, it can be noted that the precision of best approximation p/q satisfies:

$$|\theta - p/q| = q^{-1}\,|q\theta - p| = q^{-1} \min_{z\in\mathbb{Z}} |q\theta - z| = q^{-1}\,\|q\theta\|\,,$$

where $\|\theta\|$ denotes the distance of θ to the nearest integer. Based on the above, it appears that the magnitude of error should decrease inverse proportional to q.

Nevertheless such approximations can be much more precise. We quote the following result from [32, p. 11, Theorem V] (in this context, irrational numbers p, q are called "equivalent" if $p = \frac{rq+s}{uq+v}$, where r, s, u, v are integers such that $rv - us = \pm 1$).

THEOREM A.1. *Let θ be irrational. Then there are infinitely many q such that*

$$q \, \|q\theta\| < 5^{-1/2}.$$

If θ is equivalent to $\frac{1}{2}\left(5^{-1/2} - 1\right)$ then the constant $5^{-1/2}$ cannot be replaced by any smaller constant. If θ is not equivalent to $\frac{1}{2}\left(5^{-1/2} - 1\right)$, then there are infinitely many q such that:

$$q \, \|q\theta\| < 2^{-3/2}.$$

Even more notable is the result concerning the achievable precision of simultaneous approximations of irrational numbers $\theta_1, \ldots, \theta_n$ by fractions $p_1/q, \ldots, p_n/q$, with common denominator q (see [32, p. 14, Theorem III]):

THEOREM A.2. *There are infinitely many integers q such that*

$$q^{1/n} \max\left\{\|q\,\theta_1\|, \ldots, \|q\,\theta_n\|\right\} < \frac{n}{n+1}\,.$$

This means that there exist sets of integers $(p_1, \ldots, p_n, q)$ such that:

$$\max\left\{|\theta_1 - p_1/q|, \ldots, |\theta_n - p_n/q|\right\} < \frac{n}{n+1}\, q^{-1-1/n}\,.$$

It can be seen that our Lemma 2.1 is a simple consequence of the above fact, where we additionally introduce parameter $k \in \mathbb{N}$, and set $\xi := q\,2^{-k}$. That is, by multiplying both sides of the last formula by ξ we obtain:

$$\max\left\{\left|\xi\,\theta_1 - p_1/2^k\right|, \ldots, \left|\xi\theta_n - p_n/2^k\right|\right\} < \frac{n}{n+1}\, \xi^{-1/n}\, 2^{-k(1+1/n)}\,.$$

APPENDIX B. SIGN-SYMMETRIC RIGHT SHIFT OPERATOR AND PROOFS OF THEOREMS 2.2-2.4

For the purpose of our analysis we will need to introduce the following operation.

DEFINITION B.1. *The Sign-Symmetric Right Shift (SSRS) $x \overset{\text{sym}}{>>} b$ of an integer variable x by $b \geqslant 1$ bits is computed as follows:*

$$x \overset{\text{sym}}{>>} b := (x >> (b+1)) - ((-x) >> (b+1))\,, \tag{31}$$

where $>>$ denotes the ordinary (arithmetic) right shift operation.

Based on its definition, it is easy to see that

$$(-x) \overset{\text{sym}}{>>} b = -\left(x \overset{\text{sym}}{>>} b\right)\,, \tag{32}$$

which implies that it satisfies the sign-symmetry property.

The proof of Theorem 2.2 follows by construction: we take any existing non-sign symmetric algorithm, and replace all its right shifts with SSRS operators. The complexity of the SSRS operator is 2 shifts, 1 addition, and 1 negation. The total complexity is 4 operations. Theorems 2.3 and 2.4 follow.