14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

ON IMPROVING THE PERFORMANCE OF MULTIPLIERLESS DCT ALGORITHMS

Raymond K.W. Chan *Moon-Chuen Lee*

Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
phone: +852-26098416, fax: +852-26035024, email: {chankw, mclee} @.cse.cuhk.edu.hk

ABSTRACT

This paper investigates a number of issues having an impact on the performance of an approximated multiplierless DCT, which include: types of inverse transforms; types of normalizations; algorithm structures; and assignment of signed digits for approximating constants. Based on our experiment results, we have the following findings: (1) a transform based on a reversible inverse generally performs better than a version based on a traditional inverse; (2) a transform with a delayed or post normalization can achieve a much better performance; (3) uniform normalization can be considered a useful feature; (4) a lifting structure transform can achieve better accuracy than a non-lifting structure version; (5) an optimized configuration for the assignment of signed digits to the constants could help to boost the performance of the approximated DCT. It is believed that such findings should provide useful insights into making the proper design choices when converting a fast DCT into a multiplierless version.

1. INTRODUCTION

The Discrete Cosine Transform (DCT) [1] can be considered the most widely used transform in image and video compression. It has been adopted in various compression standards such as JPEG, MPEG, H.264, etc. In signal processing, 1-D and 2-D DCT have been used extensively. Many fast algorithms have been proposed for both 1-D and 2-D DCTs [2-6]. Since 2-D DCT is separable, it can be implemented in 1-D only, using the row-column approach. Though a direct 2-D DCT [7] implementation is more efficient than the one based on the row-column approach, it requires much effort to derive the required fast algorithms.

Recently, the approximation or implementation of multiplierless DCT algorithms has attracted much research interest [8-10]; since a multiplierless implementation involves only shift-and-add operations, it requires a more simple hardware design; and thus such algorithms could reduce power consumption in mobile devices. By using the method proposed in [8], any fast DCT can be converted into a multiplierless version by replacing each multiplication constant with a sum-of-power-of-two (SOPOT) representation. However, the conversion could introduce approximation errors. In [11], Chokchaitam, *et. al.* provide an analysis of approximation error in multiplierless DCTs.

Chan and others [8-10] proposed similar methods to approximate fast DCT algorithms. Their methods could approximate only DCT algorithms with planer rotation (or butterfly) structures; they all first decompose each planer rotation structure to several lifting structures, and then convert the multiplications in each lifting structure into shift-and-add operations. By changing a planar rotation structure into lifting structures, the approximation error of the multiplierless algorithm could be reduced.

The normalization in a DCT algorithm may belong to one of two types: (i) Uniform normalization – all constants used have the same value or factor. The underlying fast DCT algorithms are said to be of *non-scalded type*. (ii) Non-uniform normalization – some or all of the constants used are different. Then the fast DCT algorithms are said to be of the *scaled type*. If uniform normalization constants are used, the normalization step can be delayed in both the forward and inverse transforms of a 2-D DCT if it is implemented in 1-D DCT based on the row-column approach. The normalization constants are usually $1/\sqrt{8}$. Thus, after combining the normalization constants in the row-column approach, they would become 1/8. Then the constants need not be approximated since the normalization can be replaced simply by a right shift operation. However, in the case of *scaled type* DCTs, the non-uniform normalization constants multiplication could not be delayed when implementing a 2-D DCT in 1-D DCT using the row-column approach since matrix vector multiplication is not commutative. Thus, after applying 1-D DCT to each row of the input matrix; we must approximate the non-uniform constants and perform the normalization to obtain a temporary DCT matrix; and after applying 1-D DCT to each column of the temporary DCT matrix, another normalization should be performed using the approximated constants. Clearly, approximation errors could be introduced in the above two normalizations. Thus, the approximation of the constants in the multiplierless DCT algorithms can introduce a certain amount of errors.

Tran [9] converted two sample fast DCT algorithms, with uniform normalization constants, into their multiplierless versions; the butterfly structures were replaced by lifting steps, and the resulting approximated fast algorithms become having a non-uniform normalization. Thus the approximate fast 1-D DCT cannot be used to implement 2-D DCT with its normalization step delayed. Though the conversion of butterfly structures into to lifting structures could improve the approximation of the algorithms in some way, the resulting non-uniform normalization could make further improvement in approximation precision difficult.

Another issue having an impact on the performance of the approximated DCT is the assignment of the number of non-zero digits to the constants in their SOPOT representation. In general, using more digits to represent a constant could reduce the approximation error. However, using more digits will increase the algorithm complexity. So there should be a trade-off between accuracy and algorithm complexity. Chan *et. al*. [8] proposed a random assignment method to determine the number of digits for the constants; however, this method is not systematic. A more systematic method known as the quasi-coordinate descent algorithm has been proposed in [10]. However, it is not clear how optimal an assignment, obtained by using the foregoing methods, could be for minimizing the approximation error. The section below introduces how the choice of transform inverses could also affect the approximation errors.

2. INVERSE AND REVERSE TRANSFORMS

A fast forward DCT algorithm involves a polyphase matrix multiplication of a number of matrices obtained by the factorization of the forward transform matrix. Thus, the forward transform can be written as:

$$C = NC_{n-1}C_{n-2}...C_1 \tag{1}$$

where $C_1...C_{n-1}$ are the factorized matrices which together form the kernel of the transform, and N is the normalization matrix related to the normalization step in a fast algorithm.

As DCT is an orthogonal transform, the transform matrix should have its inverse equal to its transpose. Mathematically, if the forward transform matrix is C, then $C^{-1} = C^T$. We can have two different versions of inverse transforms for a fast DCT algorithm, derived respectively from the inverse, and the transpose of the forward transform matrix. We have:

$$C^T = (NC_{n-1}C_{n-2}...C_1)^T = C_1^T...C_{n-2}{}^TC_{n-1}^TN^T \tag{2}$$

and $$C^{-1} = (NC_{n-1}C_{n-2}...C_1)^{-1} = C_1^{-1}...C_{n-2}^{-1}C_{n-1}^{-1}N^{-1} \tag{3}$$

In general $N^T \neq N^{-1}$ and $C_i^T \neq C_i^{-1}$. The inverses of all the factorized matrices should exist since C is unitary and orthogonal. In this paper, the inverse of a forward DCT algorithm, obtained based on (2), is referred to as a *traditional inverse transform*; and the inverse obtained based on (3) is referred to as a *reversible inverse transform* since the inverse, when applied to the transformed matrix, has the effect of reversing the operations performed by the forward transform matrix. Consider the multiplication CC^{-1} involving forward transform matrix C, defined in (1), and the inverse C^{-1}, defined in (3), the first operation in (3) reverses the last operation of (1) and thus cancel out each other. Similarly, other operations in (3) can reverse the corresponding operations in (1).

It makes no difference in using either one of the two inverse transforms in floating point based fast DCT algorithms since they would still yield the same result as there is almost no loss in precision in the underlying floating point computations. However, in multiplierless fast DCT algorithms, the choice of inverse transform could have an impact on the approximation error. It can be shown that using the reversible inverse transform could give less error.

A number of research workers [8-10] proposed recently different methods to convert fast DCT algorithms into multiplierless versions, which involve replacing the butterfly structures by lifting structures. A lifting structure can be formed via the decomposition of invertible matrices which represent a butterfly structure. The inverse of the lifting structure also has two forms: the traditional inverse transform version and the reversible inverse transform version. In section 4, we shall show a performance comparison between traditional inverse transform and reversible inverse transform.

Since DCT is a separable transform, 2-D DCT can be implemented in 1-D DCT based on the row-column approach. There are two normalization steps in both the forward and inverse transforms. It is desirable to delay the normalization step as much as possible in order to avoid the amplification of the approximation error caused by error propagation during the transform. However, only those multiplierless DCT algorithms with uniform normalization constants can have the two normalization steps combined into a single step which can be delayed and performed as the last step of the transform. We cannot combine the normalizations with non-uniform constants. In this case, the normalization steps cannot be delayed. The forward 2-D DCT can be written as:

$$Y = CXC^T \tag{4}$$

and its inverse transform can be written as:

$$X' = C^TYC \tag{5}$$

or $$X' = C^{-1}YC \tag{6}$$

where *X*, *C*, *Y*, and *X'* are the original input matrix, the transform matrix, the transformed matrix, and the reconstructed input matrix respectively. From (1), (2) and (4), we have:

$$\begin{aligned} Y &= (NC_{n-1}C_{n-2}...C_1)X(NC_{n-1}C_{n-2}...C_1)^T \\ &= (NC_{n-1}C_{n-2}...C_1)X(C_1^T...C_{n-2}^TC_{n-1}^TN^T) \end{aligned} \tag{7}$$

If N is uniform, we can combine N and N^T, and (7) becomes

$$Y = ((C_{n-1}C_{n-2}...C_1)X(C_1^T...C_{n-2}^TC_{n-1}^T))NN^T \tag{8}$$

Equation (8) is said to have a *delayed normalization* since the two normalization steps can be combined and carried out after the row and the column computations during the transform. Similarly, from (1), (2), and (5), we can obtain the following:

$$X' = ((C_1^T...C_{n-2}^TC_{n-1}^T)Y(C_{n-1}C_{n-2}...C_1))NN^T \tag{9}$$

And from (1), (3), and (6), the following derivations can be performed:

$$\begin{aligned} X' &= C^{-1}YC \\ &= (C_1^{-1}...C_{n-2}^{-1}C_{n-1}^{-1}N^{-1})Y(NC_{n-1}C_{n-2}...C_1) \\ &= ((C_1^{-1}...C_{n-2}^{-1}C_{n-1}^{-1})Y(C_{n-1}C_{n-2}...C_1))NN^{-1} \\ &= ((C_1^{-1}...C_{n-2}^{-1}C_{n-1}^{-1})Y(C_{n-1}C_{n-2}...C_1))I \end{aligned} \tag{10}$$

where I is the identity matrix. In this case, we need not approximate the normalization constants for the multiplierless transform.

The normalizations in (8) and (9) can be further combined to form (NN^TNN^T); this combined normalization, which can be performed after the forward and inverse transforms, is referred to as a *post normalization*. Similarly, from (8) and (10) for a transform using a reversible inverse, we can obtain the post normalization (NN^TNN^{-1}) which can be simplified to NN^T . With post normalization, we can avoid any propagation of errors while performing the transforms by the multiplierless DCT. However, such a multiplierless DCT is not compatible with a standard 2-DDCT. The section below introduces an optimized assignment of signed digits for approximating transform matrix constants, which could have a great impact on the performance of the resulting transform.

3. ASSIGNMENT OF SIGNED DIGITS TO CONSTANTS

The accuracy of an approximated DCT could be more sensitive to the approximation errors of some of its constants. So we should assign more digits to represent the *sensitive* constants in SOPOT representations in order to reduce their errors. However, assigning more digits to a constant can increase the complexity of the approximated algorithm. So we should develop a method to find an optimized assignment of digits to the constants in order to minimize the approximation error of the algorithm for a given complexity. We present here a method to assign a number of non-zero digits to each constant (approximated by a sum of power-of-two terms), based on the constraint on the total number of additions for a given transform. We use MSE, defined in [9], as the metric for measuring the performance of the approximated DCT. The algorithm below finds an optimized signed digits configuration for the constants of an approximated algorithm.

Suppose there are n constants C_1, C_2, …and C_n in a fast DCT algorithm and they are assigned d_1, d_2, …and d_n signed digits respectively. Then (d_1, d_2, …d_n) is called a *signed digits configuration* of the algorithm being approximated. The total number of the signed digits is defined as the *length* of the configuration. The complexity of the algorithm depends partly on the *length* of its signed digits configuration. Two different configurations should correspond to the same complexity of an algorithm as long as they have the same length. However, two different configurations with the same length may lead to different MSEs. The reason is that the MSE of an algorithm can be more sensitive to some constants and less sensitive to others. If a configuration assigns more digits to the sensitive constants and fewer to those less sensitive ones, we can get a smaller MSE. It is not easy to know which constants are sensitive, however, they could be searched by examining all possible configurations of a given length and compute the MSEs. Then we can find an optimized configuration with the smallest MSE. However, this exhaustive search approach could be infeasible especially when there are many constants. The algorithm outlined below can be shown capable of finding an optimized configuration.

Algorithm for finding an optimized configuration with length *L* for a DCT algorithm having *n* constants

```
1. Let $P$=($d_1$, $d_2$, …$d_n$) be the initial signed digits configuration
   and its length is $L$; $e$ = $MSE$ of algorithm with configuration $P$;
   /* Number of additions contributed by configuration $P$ would
   be $L$-$n$. Assume the target complexity of the algorithm allows
   only $L$-$n$ additions from the signed digits configuration. */
2. for each $i \in \{1...N\}$ ; $Q_i$ = $P$; increment $d_i$ of $Q_i$ by 1; compute
   MSE $e_{P,i}$ of algorithm with $Q_i$;   /* no. of additions from $Q_i$ =
   $L$-$n$+1. */
3. find the minimum $e_{P,i}$ ; let $Q$ = $Q_i$;
4. for each $i \in \{1...N\}$ ; let $Q_i$=$Q$; decrement $d_i$ of $Q_i$ by 1; com-
   pute MSE $e_{Q,i}$ of algorithm with $Q_i$;   /* no. of additions from
   $Q_i$ : $L$-$n$ */
5. find the minimum $e_{q,i}$ ; let $P$ = $Q_i$;
6. if ($e$ not equal to $e_{Q,i}$ ) then {$e$ = $e_{Q,i}$ ; goto 2;}
7. exit   /* minimum MSE: $e$; optimized configuration: $P$ with
   length $L$ */
```

4. SAMPLE MULTIPLIERLESS DCT

To convert floating-point based fast DCT algorithms into their multiplierless versions, we applied the conversion method proposed by Chan [8] and Chen [10]. Chan *et. al.* [8] proposed a method to convert each multiplication constant into a sum-of-power-of-two (SOPOT) representation. Since each power-of-two term is dyadic, we can implement the multiplication by shift-and-add operations only. Then the complexity can be measured by the number of additions and the number of shifts. As the shift cost is relatively low, algorithm complexity could then be measured by the number of additions. Chen *et. al.* [10] proposed a method to convert planer rotation structures into lifting structures. This conversion makes the constants become less than unity; so the subsequent approximation of the constants will introduce a smaller error. In summary, in designing a multiplierless DCT algorithm for achieving improved performance, we need to consider the following choices: (i) lifting or non-lifting structures; (ii) traditional inverse or reversible inverse transform; (iii) uniform normalization or non-uniform normalization; (iv) non-delayed normalization, delayed normalization, or post normalization; (v) optimized or non-optimized signed digits configuration.

We shall use the well known LLM's [3] fast DCT algorithms to illustrate the above important design issues having an impact on the performance of an approximated multiplierless DCT. To address the above issues, we consider different versions of the transform as outlined below. Concerning algorithm structures, the fast DCT has two versions: original transform (non-lifting structures) and lifting structure transform. Each forward transform could have two versions for the two different inverses: traditional inverse transform and reversible inverse transform. Therefore the fast DCT has six versions altogether, namely: original forward (OF) transform, original inverse (OI) transform, original reversible inverse (OR) transform, lifting structure based forward (LF) transform, lifting structure based inverse (LI) transform, and lifting structured based reversible inverse (LR) transform. While the original LLM algorithms (with non-lifting structures) have the uniform normalizations, the lifting structure versions have non-uniform normalizations. So the above non-uniform normalization versions can be converted into various uniform normalization versions: uniform normalization based forward transform (UF), uniform normalization based inverse (UI) transform, and uniform normalization based reversible inverse transform (UR). To make a non-uniform normalization version become a uniform normalization version, we can replace the non-uniform normalization matrix N_i by the multiplication of $(N_i N_u^{-1})$ and N_u, where N_u is the target uniform normalization matrix. Further, for each the above uniform normalization versions, we can create two other versions: delayed normalization and post normalization versions.

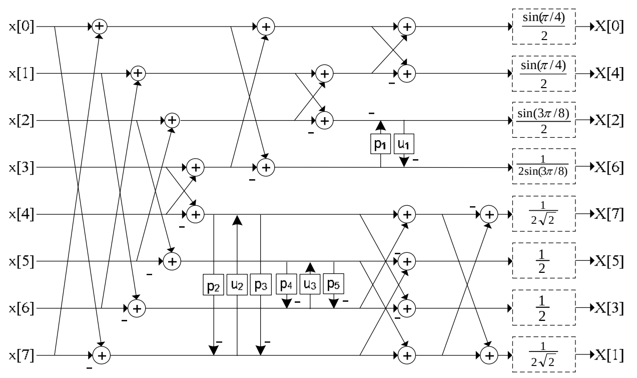

Figure 1 – LLM's forward DCT algorithm with lifting steps

The original LLM DCT algorithms have their uniform normalization constants all equal to 1/√8. After converting the original version into a lifting structure based version [9], the normalization constants become non-uniform.

Figure 1 shows the signal flow diagram of LLM's lifting version of the forward fast DCT algorithm. The four normalizations constants for X[0], X[4], X[7] and X[1] all have the same value 1/√8. We can further change the non-uniform normalization constants for X[2], X[3], X[5], X[6] to 1/√8, and obtain a uniform normalization based lifting version. Then the number of multiplications in the kernel becomes 12 which is smaller than the total number of multiplications (equal to 14) in the original lifting version.

Table 1 shows the optimized signed digits configurations, obtained via the algorithm presented in section 3, for LLM's lifting structure based forward DCT algorithm with different complexities.

5. RESULTS

We used the Lena image to assess the performance, in terms of PSNR, of various versions of LLM's DCT algorithms. The results are as shown in Table 2. As expected, a higher PSNR can be obtained when using a higher algorithm complexity.

By comparing the PSNRs in rows (OF1 OI1) and (LF1 LI1) of Table 2, the lifting structures based fast DCT outperformed the non-lifting structures based version. The reason could be that the constants in the lifting structure version are smaller than 1 in magnitude, which could be approximated with a smaller error when the same number of signed digits is used.

Table 1: The assignment of signed digits for each constant in the lifting version of the LLM forward transform and the corresponding MSE.

Total	Number of non-zero digits														
Adds	ρ_1	μ_1	ρ_2	μ_2	ρ_3	ρ_4	μ_3	ρ_5	N_0	N_4	N_2	N_6	N_7	N_1	MSE
⌖28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	8.29e-2
*28	1	0	1	1	1	1	0	0	3	1	1	1	1	2	3.19e-3
⌖42	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3.56e-3
*42	2	2	1	2	2	1	2	2	4	2	1	2	2	3	1.63e-4
⌖54	2	2	3	3	3	3	3	3	3	3	3	3	3	3	2.75e-4
*54	3	3	2	3	2	2	2	3	5	3	2	3	3	4	1.08e-5
⌖66	3	3	3	3	4	4	4	4	4	4	4	4	4	4	2.92e-5
*66	4	4	3	3	3	3	3	3	5	5	3	4	4	5	3.07e-7

⌖ – non-optimized configuration, * – optimized configuration

By comparing the relevant rows, in general, the fast DCT based on a reversible inverse performed better than the version based the traditional inverse. The foregoing can be observed by comparing the sample results shown in rows (OF1 OI1) and (OF1 OR1).

The results in rows (OF2 OI2) and (OF2 OR2) indicate that the delayed normalization versions of traditional inverse based fast DCT and those of the reversible inverse based fast DCT performed exactly the same. Further, similar observation can be made from the results in rows (OF3 OI3) and (OF3 OR3) for the post normalization versions of traditional inverse based fast DCT and those of the reversible inverse based fast DCT. The above results can be explained by the fact that the above mentioned DCT transforms differ only in the normalization steps. For the delayed normalization versions, both transforms would perform exactly the same kernel operations. However, their delayed combined normalizations would then involve essentially the same multiplication of a factor equal to 1/8. This explains why they had exactly same performance. For the post normalization versions, the combined normalizations are performed after the forward and inverse transforms. Other than the normalization matrices, the traditional inverse and reversible inverse transforms have exactly the same kernels. Further, the combined normalizations for the two versions both involve the multiplication of a factor equal to 1/64. Thus, both the traditional inverse and reversible inverse transforms have the same performance.

As shown in rows (LF1 LI1) and (UF1 UI1), the lifting structure version of LLM's fast DCT with a non-uniform normalization outperformed the fast DCT with uniform normalization. The relatively low performance of the uniform normalization version could be caused by the introduction of four additional scaling constants while converting the non-uniform normalization into a uniform one. However, only the fast DCT with uniform normalization can be converted into a version with delayed normalization or a version with post normalization. So, uniform normalization can be considered a desirable feature. The results in rows (LF1 LI1), (UF1 UI1), and (UF2 UI2) show that the delayed normalization version performed substantially better than both the non-uniform normalization and uniform normalization versions. Further, a post normalization version of the fast DCT outperformed a delayed normalization version of the transform, as shown by the sample results in rows (UF2 UI2) and (UF3 UI3).

Table 2 also shows that the fast DCTs based on the optimized signed digits assignment configurations generally performed better than the corresponding versions based on the non-optimized configurations.

Table 2: PSNRs of the approximated LLM's fast DCT algorithms with different complexities

LLM' fast algo.	*Forward*	*Inverse*	*Number of Additions*			
Configurations	*transform*	*transform*	*⌖42*	**42*	*⌖66*	**66*
Approx. norm.	OF1	OI1	16.90	21.72	28.90	34.85
Delayed norm.	OF2	OI2	42.79	43.43	46.69	46.70
Post norm.	OF3	OI3	43.03	43.91	49.78	49.82
Approx. norm.	OF1	OR1	16.93	21.81	29.84	36.19
Delayed norm.	OF2	OR2	42.79	43.43	46.69	46.70
Post norm.	OF3	OR3	43.03	43.91	49.78	49.82
Approx. norm.	LF1	LI1	17.66	33.32	37.82	39.19
Approx. norm.	LF1	LR1	17.65	33.94	40.73	45.69
Approx. norm.	UF1	UI1	17.61	34.42	29.30	36.48
Delayed norm.	UF2	UI2	46.90	47.09	48.54	48.54
Post norm.	UF3	UI3	47.64	48.48	50.34	50.34
Approx. norm.	UF1	UR1	17.63	23.30	29.96	43.81
Delayed norm.	UF2	UR2	45.99	46.84	47.52	47.52
Post norm.	UF3	UR3	47.38	49.22	50.28	50.28

Keys: OF – original forward, OI – original traditional inverse, OR – original reversible inverse; LF – lifting forward, LI – lifting traditional inverse, LR – lifting reversible inverse, UF – uniform forward, UI – uniform traditional inverse, UR – uniform reversible inverse.
⌖ – non-optimized configuration, * – optimized configuration

6. CONCLUDING REMARKS

In this paper, we investigated a number of issues having an impact on the performance of an approximated multiplierless DCT. The findings from the work provide useful insights into taking the appropriate design decisions when developing approximate multiplierless DCT algorithms. Such multiplierless fast DCT algorithms are normally obtained from a given fast DCT algorithm via a conversion method. The issues examined include: (1) structures of DCT algorithms; (2) types of inverse transforms; (3) types of normalizations in a transform; and (4) using an appropriate number of signed digits to approximate each constant in the transform.

In order to assess the impact of the above issues on the approximate multiplierless DCT, we implemented different versions of multiplierless DCT converted from LLM's fast DCT. By examining the results obtained from the different versions of multiplierless transforms, we have the following findings. (i) A fast DCT with lifting structures outperformed the version with non-lifting structures. Thus, when developing multiplierless DCT, we should convert any available planar rotation or butterfly structures into lifting structures. (ii) An approximate fast DCT based on a reversible inverse performed better than the version based on the traditional inverse since a reversible inverse could introduce a smaller propagation error. (iii) Uniform normalization is a good feature since it can be exploited to implement an approximated fast DCT with delayed normalization or post normalization; such delayed or post normalization versions could achieve substantially improved performance. Thus, we should try to convert a non-uniform normalization into uniform one. (iv) The approximate DCTs based on an optimized configuration of signed digits for approximating the transform matrix constants have been shown to perform better than those versions based on a non-optimized configuration. So, the algorithm developed in this paper could be exploited to find an optimized configuration for approximating the transform matrix constants.

REFERENCES

[1] K.R. Rao, and P. Yip, *Discrete Cosine Transform*, Academic Press, pp.82, 1990.

[2] W.Chen, C.H. Smith, and S.C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, Vol 25, pp. 1004-1009, Sept. 1977.

[3] C. Loeffler, A. Lightenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," *Proc. IEEE ICASSP*, Vol. 2, pp.988-001, Feb 1989.

[4] Y. Arai, T. Agui, and M. Nakajima, "A Fast DCT-SQ scheme for images," *Trans. IEICE.*, Vol. E-71, No. 11, pp.1095-1097, Nov. 1988.

[5] Y.J. Chen, S. Oraintara, T. Nguyen, "Integer Discrete Cosine Transform (IntDCT)," Submitted to *IEEE Trans. Signal Processing*, Feb. 23, 2000.

[6] B.G. Lee, "A new algorithm to computer the discrete cosine transform," *IEEE Transactions on Acoustic, Speech, and Signal Processing*, Vol. 32, pp. 1243-1245, Dec. 1984.

[7] H.S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoustic., Speech, Signal Processing*, Vol. 35, pp. 1455-1461, Oct. 1987.

[8] S.C. Chan, P.M. Yiu, "A multiplier-less 1-D and 2-D fast Fourier transform-like transform using sum-of-power-of-two (SOPOT) coefficients," *IEEE International Symposium on Circuits and System 2002*, Vol. 4, pp. IV-755-IV-758, 2002.

[9] J. Liang, T.D. Tran, "Fast Multiplierless approximations of the DCT with lifting scheme," *IEEE Transactions on Signal Processing*, Vol. 49, No. 12, pp. 3032-3044, Dec. 2001.

[10] Y.J. Chen, S. Oraintara, T.D. Tran, K. Amaratunga, T.Q. Nguyen, "Multiplierless approximation transforms using lifting scheme and coordinate descent with adder constraint," *Proc. IEEE Int. Conf. on ASSP*, Vol. 3 pp. 3136-3139, 2002.

[11] S. Chokchaitam, M. Iwahashi, N. Kambayashi, "Optimal word length allocation of integer DCT and its error analysis," *Signal Processing: Image Communication*, Vol. 19, Issue 6, pp. 465-478, July 2004.