D4.2

PRACTICAL FAST 1-D DCT ALGORITHMS WITH 11 MULTIPLICATIONS

Christoph Loeffler*, Adriaan Ligtenberg, and George S. Moschytz***

* Institute for Signal and Information Processing, ETH Zürich, CH-8092 Zürich, Switzerland.
** AT&T Bell Laboratories, Crawfords Corner Rd., Holmdel N.J.07733

ABSTRACT

A new class of practical fast algorithms is introduced for the *Discrete Cosine Transform (DCT)*, an important transform that is of particular interest in image compression. For an 8-point DCT only *11 multiplications and 29 additions* are required.

A systematic approach is presented to generate the different members in this class all having the same minimum arithmetic complexity. The structure of many of the published algorithms can be found in members of this class.

An extension of the algorithm for longer transformations is presented. As a result, the 16-point DCT requires only 31 multiplications and 81 additions, which is, to our knowledge, less than the currently published algorithms.

1 INTRODUCTION

The Discrete Cosine Transform (DCT) was first introduced in 1974 by Ahmed et al [1]. Primarily applied to real data values, this transform has found wide applications in image processing, data compression, filtering, and other fields.

A number of fast algorithms for the one-dimensional DCT (1-D DCT) have been published in the last decade. A two-dimensional DCT can be obtained by applying first a 1-D DCT over the rows followed by a 1-D DCT to the columns of the input-data matrix. Single-chip implementations have been reported for the 8-bit DCT ([2]) and for the two-dimensional 8 × 8-bit DCT (e.g. [3]).

The DCT is currently in the process of being standardized in different national and international committees because of its importance in still picture and video coding.

The N-point DCT is defined as follows: A given data sequence $\{x_n, n = 0, 1, 2, ..., N-1\}$ is transformed into the output sequence $\{y_n, n = 0, 1, 2, ..., N-1\}$ by the function given in equation (1).

$$y(k) = C \cdot a_k \cdot \sum_{n=0}^{N-1} x(n) \cdot \cos\left(\frac{2\pi \cdot (2n+1) \cdot k}{4N}\right) \qquad (1)$$

$$k = 0, ..., N-1$$

$$where\ a_0 = \cos\left(\tfrac{\pi}{4}\right);\ a_k = 1 \quad k = 1, ..., N-1$$

In the following sections, several known algorithms for the Discrete Cosine Transform will first be discussed and their respective complexity for length $N = 8$ will be compared. Then a class of new DCT algorithms requiring no more than 11 multiplications and 29 additions will be proposed. Furthermore, variations of these algorithms will be discussed, including a solution where all multiplications are executed in parallel.

Finally a 16-point algorithm based on the same structure as our 8-bit algorithm will be presented.

2 KNOWN ALGORITHMS

Many different algorithms to compute the discrete Cosine Transform have been proposed in recent years ([4], [5], [6], [7], [8], [9], [10]). All of the most recent proposals need 12 multiplications and 29 additions to complete an 8-point DCT (see table 1).

author ref.	Chen [4]	Wang [5]	Lee [6]	Vetterli [7]	Suehiro [8]	Hou [9]	our Alg.
mult.	16 (13)	13	12	12	12	12	11
add.	26 (29)	29	29	29	29	29	29

Table 1: Number of operations for an 8-point DCT

Chen's fast DCT algorithm [4], the first one published, exhibits a very regular structure. The published number of multiplications and additions can easily be changed to the numbers shown in parenthesis in table 1 by using the same method to calculate a rotation as is used in all later publications (3 multiplications and 3 additions per rotation).

Wang [5] has a method to easily obtain algorithms for the Discrete Sine Transform (DST), the Discrete W-Transform (DWT) and the Discrete Fourier Transform (DFT) from his DCT algorithm.

Lee's algorithm [6] has very regular first stages, but has irregular data flow in the last stage and needs the inverse of cosine values as coefficients. This can lead to numerical overflow problems.

Vetterli [7] uses a recursive formula for his algorithm; however, additional operations required to connect the recursively calculated blocks lead to an increased complexity in the communication-structure of his algorithm.

Suehiro [8] needs fewer multiplications than Wang, but his solution still allows to apply Wang's method to obtain algorithms for DST, DWT and DFT from the DCT-algorithm.

Hou [9] proposes a recursive algorithm, basing each DCT of length N on two DCTs of length $N/2$. The algorithm is regular, with the exception of the last stage, where some irregularities are introduced for larger lengths.

Duhamel [10] shows that the theoretical lower bound for an 8-point DCT is *11 multiplications*. This result is obtained by looking at the DCT as an algorithm based on a cyclic convolution, and applying methods of Winograd [11]. Heidemann [12] came to the same result.

However, there are some special solutions, which require less multiplications for the actual DCT, but move the complexity to another part of the calculation. Examples for these include a solution based on number theoretical transforms requiring only 8 multiplications for an 8 bit DCT explained in Duhamel's paper ([10]) and the Arithmetic Fourier Transform [13]. The first example causes additional costs for signal transformation and increased word-length, the second one requires unequally spaced sampling instants for the signal. Therefore both examples do not lead to overall less complex solutions than the algorithms mentioned before.

It follows that most of the published algorithms for an 8-point Discrete Cosine Transform use only one multiplication more than the minimal number required.

3 11-MULTIPLICATION 8-POINT DCT ALGORITHMS

We have found a class of 8-point DCT algorithms requiring **only 11 multiplications and 29 additions.**

CH2673-2/89/0000-0988 $1.00 © 1989 IEEE

In fact, the same method can be applied to many of the algorithms described before to reduce the number of multiplications by one. As will be shown later, some of the algorithms published are special members of the class of algorithms presented here.

The number of multiplications has been reduced to the theoretical lower bound without an increase in the number of additions as compared with the published algorithms. An algorithm of this class is shown in figure 1. The stages of the Algorithm, numbered

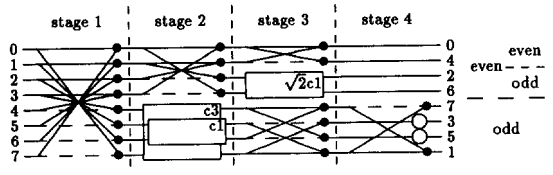

Figure 1: 8-point DCT algorithm with 11 multiplications. For Symbols see figure 2

1 to 4, are parts that have to be executed in series and can not be evaluated in parallel because of data dependencies. However calculations inside one stage can be parallelized. At the right-hand side of figure 1 the structure of the algorithm is shown. In stage 2 the algorithm separates in two parts, one for the even coefficients, one for the odd coefficients. The even part is nothing else than a 4-point DCT, again separating in even and odd part in stage 3.

Figure 2 explains the building blocks of the algorithm.

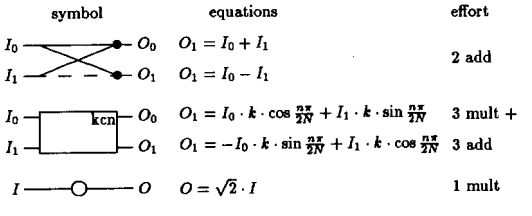

Figure 2: symbols used to display an algorithm structure

The second building block, the rotation, can be calculated using only 3 multiplications and 3 additions instead of 4 multiplications and 2 additions using the equivalence shown in equation (2).

$$\begin{aligned} y_0 &= a \cdot x_0 + b \cdot x_1 = (b-a) \cdot x_1 + a \cdot (x_0 + x_1) \\ y_1 &= -b \cdot x_0 + a \cdot x_1 = -(a+b) \cdot x_0 + a \cdot (x_0 + x_1) \end{aligned} \quad (2)$$

The constant C in equation (1) was chosen to be $C = \sqrt{2}$, which results in $C \cdot a_k = 1$ for $k = 0$, allowing the output y_0 to be evaluated without any multiplication.

For the DCT_N and inverse DCT ($IDCT_N$) these constants must satisfy equation (3) in order to obtain the original signal unscaled after the forward and inverse transformation.

$$C_{DCT} \cdot C_{IDCT} = \frac{4}{N^2} \quad (3)$$

We include the factor $\sqrt{2}$ in both the DCT and the IDCT. The remaining factor $\frac{2}{N^2}$ can easily be implemented by right-shifting the original and/or the transformed signal. Note that the inverse DCT uses exactly the same algorithmic structure as the DCT itself, but in reverse order. Outputs thus become inputs and vice versa.

The proposed algorithm separates clearly into even and odd parts after the first stage; the even part is further separated following the second stage. The maximum path length is *2 multiplications and 4 additions* (inside the rotator we count 1 multiplication and 2 additions in cascade). This algorithm was generated from the full matrix equation in a systematic way using graph transformations and equivalence relations (see [14]). The result mentioned in [14] actually has 31 additions; however, the 8 additions in the final stage of the odd part can easily be condensed into 6 additions which leads to the structure shown in figure 1.

4 VARIATIONS OF OUR ALGORITHM

As mentioned above there is an entire class of DCT algorithms which have the same number of multiplications and additions. Other solutions can be derived from the one shown in figure 1 in different ways. In the following, we distinguish between variations of the first or final stages of the structure in figure 1.

4.1 Variations of the Final Stages

In the calculation of the *even part* of the algorithm (figure 1, top 4 rows) only the stages 2 and 3 can be exchanged (figure 3). In the *odd part*, several different variations are possible: *Four*

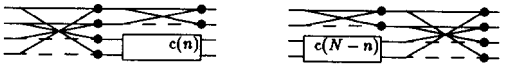

Figure 3: inversion of stages 2 and 3 of the even part

different angles for each of the two rotations in stage 2 can be selected. From the 16 combinations of these angles, 8 lead to a minimum complexity solution (see figure 4).

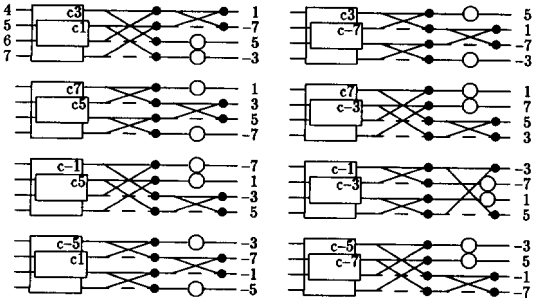

Figure 4: variations of stages 2, 3 and 4 of the odd part

As can be seen, the order of the odd outputs as well as their signs can be changed by varying stages 2, 3 and 4 of the algorithm.

In each of these variations the block formed by stages 2, 3 and 4 can be reversed as well (analogous to the even part, see figure 3).

The algorithm of Suehiro, patented by Toshiba in [15], is a variation of our basic algorithm containing reversed stages 2 and 3 of the even part as well as inverted stages 2, 3 and 4 of the odd part.

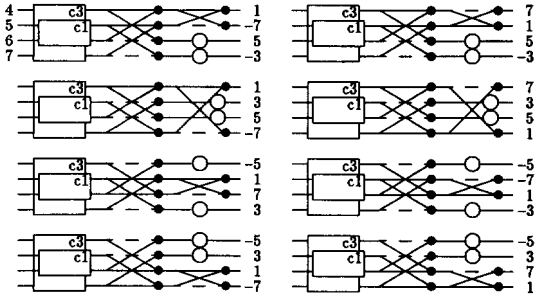

Figure 5: variations by inverting add/subtract modules

The signs of the output signals and their order can further be changed by inverting a cross-add module or a combination of them, i.e. by exchanging the adder and the subtractor. By doing so in the middle stage of figure 4, the last stage of the algorithm changes. This is demonstrated for the first alternative shown in figure 4 in figure 5.

The same transformations can be applied to all 8 variations shown in figure 4. As can be seen, all possible orders of the output signals can be obtained by choosing the appropriate variation of our algorithm.

4.2 Variations of the First Stage

Besides the completely symmetrical stage 1 shown in our basic algorithm (figure 1) we found *four other first stages* that lead to solutions of the same low complexity. They are shown in figure 6.

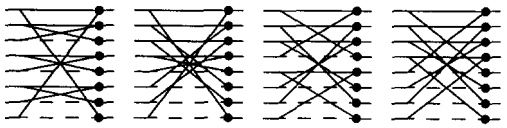

Figure 6: variations of stage 1 of the algorithm

As an example, figure 7 shows a variation of our algorithm starting with the type of stage 1 shown in the leftmost part of figure 6.

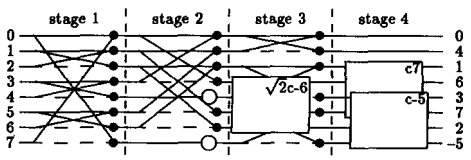

Figure 7: example algorithm with different first stage

This algorithm is a simplified version of the one described in [2] which has been implemented in silicon. Its maximal path length is again 2 multiplications and 4 additions, but the separation between even and odd path does not occur until after stage 2. This is true for all variations of the algorithm which start with one of the first stages shown in figure 6. In all of these solutions, further variations can be found by exchanging stages 3 and 4 of the odd part or by inverting modules as explained in the previous subsection.

First stages having less than two *symmetrical* cross add/subtracts (symmetrical means combining rows n and $N-1-n$) lead to solutions having more than 29 additions.

4.3 Solutions having parallel multiplications

Whereas in the even part of our algorithm the three multiplications needed to calculate the rotation can be executed in parallel, the odd part always includes signal paths having 2 multiplications in cascade. Finding a solution with X *parallel* multiplications is equivalent to expanding the matrix of the odd part, given in equation 4 to a diagonal matrix D of size X.

	i_0	i_1	i_2	i_3
y_1	c_7	c_5	c_3	c_1
y_3	$-c_5$	$-c_1$	$-c_7$	c_3
y_5	c_3	c_7	$-c_1$	c_5
y_7	$-c_1$	c_3	$-c_5$	c_7

$$c_k = \cos(\tfrac{k \cdot \pi}{16}) \qquad (4)$$

In (4) i_0, i_1, i_2 and i_3 are the inputs to the odd part (after the first stage), y_1, y_3, y_5, and y_7 the output DCT coefficients and c_1, c_3, c_5 and c_7 the matrix coefficients defined in (1).

The expansion must be performed in a way that the columns of D are linear combinations of the input signals, and its rows can be linearly combined to get the odd output coefficients y_1, y_3, y_5, and y_7 of the DCT. This diagonal matrix can be obtained in two steps: first the matrix (4) is expanded by adding new columns representing multiplications with sum-terms of the input-signals. The goal is to introduce so many zeroes, that in none of the columns more than one type of non-zero-values appears. The resulting matrix after this first step is shown in equation (5). This matrix-equation requires not more than 9 multiplications. The matrix can be expanded to a diagonal matrix as shown in equation (6).

This matrix has dimension 9. Calculating the odd part using this matrix (see figure 8), thus leads to an algorithm having a total of 12 multiplications and 32 additions (3 multiplications and 9 additions are used for evaluating the even part, 8 additions are needed in the first stage, see figure 1).

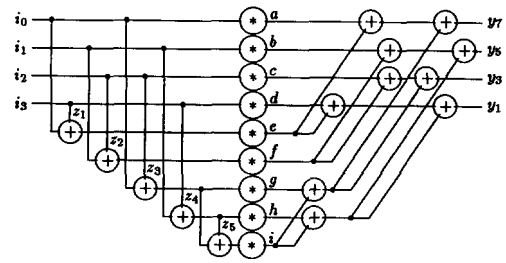

Figure 8: calculation of the odd part with parallel multiplications

The price for having at most one multiplication per path is, therefore, *one additional multiplication and 3 additions.*

	i_0	i_1	i_2	i_3	i_0+i_3	i_1+i_2	i_0+i_2	i_1+i_3	$i_0+i_1+i_2+i_3$
y_1	0	0	0	c_1+c_3 $-c_5-c_7$	$-c_3+c_7$	0	0	$-c_3+c_5$	c_3
y_3	0	0	c_1+c_3 $+c_5-c_7$	0	0	$-c_1-c_3-c_5$	0		c_3
y_5	0	c_1+c_3 $-c_5+c_7$	0	0	0	$-c_1-c_3$	0	$-c_3+c_5$	c_3
y_7	$-c_1+c_3$ $+c_5-c_7$	0	0	0	$-c_3+c_7$	0	$-c_3-c_5$	0	c_3

(5)

	i_0	i_1	i_2	i_3	z_1	z_2	z_3	z_4	z_5
a	$-c_1+c_3$ $+c_5-c_7$	0	0	0	0	0	0	0	0
b	0	c_1+c_3 $-c_5+c_7$	0	0	0	0	0	0	0
c	0	0	c_1+c_3 $+c_5-c_7$	0	0	0	0	0	0
d	0	0	0	c_1+c_3 $-c_5-c_7$	0	0	0	0	0
e	0	0	0	0	$-c_3+c_7$	0	0	0	0
f	0	0	0	0	0	$-c_1-c_3$	0	0	0
g	0	0	0	0	0	0	$-c_3-c_5$	0	0
h	0	0	0	0	0	0	0	$-c_3+c_5$	0
i	0	0	0	0	0	0	0	0	c_3

where $z_1 = i_0 + i_3$ $\quad z_2 = i_1 + i_2$ $\quad z_3 = i_0 + i_2$ $\quad z_4 = i_1 + i_3$ $\quad z_5 = i_0 + i_1 + i_2 + i_3$

and $y_1 = d + e + h + i$ $\quad y_3 = c + f + g + i$ $\quad y_5 = b + f + h + i$ $\quad y_7 = a + e + g + i$

(6)

5 16-POINT DISCRETE COSINE TRANSFORM

Although most current implementations use either 8-bit DCTs or 8 × 8-bit DCTs, algorithms for extended lengths are interesting and will become increasingly important. The number of operations needed for several published 16-bit DCT algorithms is shown in table 2.

author ref.	Chen [4]	Wang [5]	Lee [6]	Vetterli [7]	Suehiro [8]	Hou [9]	our Alg.
mult.	44 (35)	35	32	32	32	32	31
add.	74 (83)	83	81	81	81	81	81

Table 2: number of operations for a 16-point DCT

As Duhamel states in [10], The lower bound for the number of multiplications is only 26. We extended our 8-bit algorithm in a recursive way to the 16-bit algorithm shown in figure 9. It needs

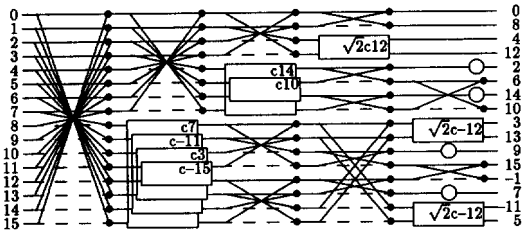

Figure 9: 16-point DCT algorithm

31 multiplications and 81 additions, the minimum path length is 2 multiplications and 6 additions. Although this solution requires 5 multiplications more than the theoretical lower bound, it can still be calculated using the same number of additions, and fewer multiplications than the known algorithms.

6 CONCLUSION

We have presented a new class of practical fast 8-point DCT algorithms. These algorithms have only 11 multiplications and 29 additions. It has been shown here, how these algorithms can be varied in order to obtain different communication structures, different output orders, as well as how to change the signs of the outputs. These variations do not impose a higher number of operations for performing the DCT and can be used to optimize hardware implementation.

An extension for a 16-point DCT results in an algorithm requiring 31 multiplications and 81 additions. This algorithm does not reach the theoretical minimum number of multiplications, but nevertheless it has one multiplication less than, and the same number of additions as the best currently known algorithms. More complicated graph transformation are involved to achieve the minimum theoretical bound. It has further been shown, that an 8-point DCT can be calculated with 12 multiplications *in parallel*, i.e. with no signal path having more than one multiplication in cascade.

Acknowledgements

The authors like to thank Martin Vetterli, Hemant Bheda and Jay O'Neill for numerous helpful comments.

REFERENCES

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," IEEE Transactions on Computer, Vol. C-23, Jan. 1974, pp. 90-93.

[2] M. Vetterli, A. Ligtenberg, "A Discrete Fourier-Cosine Transform Chip," IEEE Journal on Selected Areas of Communications, Vol. SAC-4, No. 1, Jan. 1986, pp. 49-61.

[3] A. Ligtenberg, J. H. O'Neill, "A Single Chip Solution for an 8 by 8 two Dimensional DCT," Proceedings IEEE International Symposium on Circuits and Systems, ISCAS-87, Philadelphia, May 1987, pp. 1128-1131.

[4] W. A. Chen, C. Harrison, and S. C. Fralick, "A Fast computational Algorithm for the Discrete Cosine Transform," IEEE Transactions on Communications, Vol. COM-25, No. 9, Sept. 1977, pp. 1004-1011.

[5] Z. Wang, "Fast Algorithms for the Discrete W-Transform and for the Discrete Fourier Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-32, No. 4, Aug. 1984, pp. 803-816.

[6] Byeong Lee, "A New Algorithm to Compute the Discrete Cosine Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-32, No. 6, Dec. 1984, pp. 1243-1245.

[7] M. Vetterli, H. Nussbaumer, "Simple FFT and DCT Algorithms with Reduced Number of Operations," Signal Processing (North Holland), Vol. 6, No. 4, Aug. 1984, pp. 267-278.

[8] N. Suehiro and M. Hatori, "Fast Algorithms for the DFT and other Sinusoidal Transforms," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-34, No. 3, June 1986, pp. 642-644.

[9] H. S. Hou, "A Fast Recursive Algorithm for Computing the Discrete Cosine Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-35, No. 10, Oct. 1987, pp. 1455-1461.

[10] P. Duhamel and H. H'Mida, "New 2^n DCT Algorithms suitable for VLSI Implementation," Proceedings IEEE International conference on Acoustics, Speech and Signal Processing, ICASSP-87, Dallas, April 1987, pp. 1805-1808.

[11] S. Winograd, "Some bilinear Forms whose multiplicative Complexity depends on the Field of Constants," Math. System Theory, 1977, Vol. 10, pp. 169-180.

[12] M. T. Heidemann and C. S. Burrus, "Multiply/Add Tradeoffs in Length 2^n FFT Algorithms," Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-85, Tampa, March 1985, pp. 780-783.

[13] G. Fischer, D. W. Tufts and G. Sadavis, "VLSI-Implementation of the Arithmetic Fourier-Transform (AFT), A New Approach to High-Speed Computation for Signal-Processing," VLSI Signal Processing III, IEEE-Press, New York, Nov. 1988, pp. 264-275.

[14] C. Loeffler, A. Ligtenberg, and G.S. Moschytz, "Algorithm-Architecture Mapping for Custom DSP Chips," Proceedings IEEE International Symposium on Circuits and Systems, ISCAS-88, Helsinki, May 1988, pp. 1953-1956.

[15] N. Suehiro, Japanese patent kokai 62-61159, Toshiba Research & Development Center, Kawasaki, Japan.