# Discrete Cosine Transformer with Variable-Length Basis Vector for MPEG-4 Video Codec

Ryo KURODA† Gen FUJITA† Takao ONOYE†† Isao SHIRAKAWA†

†Department of Information Systems Engineering,
Faculty of Engineering, Osaka University,
Yamada-Oka, Suita, Osaka, 565-0871 Japan
Tel.: +81(6)6879-7808,
FAX: +81(6)6875-5902
E-mail: {kuroda,fujita, sirakawa}@ise.eng.osaka-u.ac.jp

††Department of Communications and Computer Engineering,
Kyoto University,
Sakyo, Kyoto, 606-8501 Japan
Tel.: +81(75)753-4803
FAX: +81(75)753-4804
E-mail: onoye@kuee.kyoto-u.ac.jp

## Abstract

In this paper a VLSI architecture of the Shape-Adaptive Discrete Cosine Transform (SA-DCT) is described, which can be employed dedicatedly for MPEG-4 video codec. Adopting a fast DCT algorithm, the number of multipliers can be reduced by half in comparison with a conventional algorithm. This SA-DCT core with a small additional amount of hardware can perform the SA-Inverse DCT (SA-IDCT) by sharing multipliers and a transportation memory. The proposed SA-DCT core is integrated with 40,000 gates by using $0.35\mu m$ triple-metal CMOS technology, which operates at 20 MHz, and hence enables the realtime codec of CIF (352 x 288 pixels) pictures.

## 1 Introduction

In recent years, the low bit rate communication has attracted a great amount of interest especially in terms of the mobile computing, for which MPEG-4 is the latest standardization algorithm.

The main feature of MPEG-4[1] consists in the possibility of object-based image access to encoded video data. Fig. 1 illustrates MPEG-4 object-based video coding and decoding. The foreground objects (a man and a car) can be segmented from the background. A change of background image and high rate compression in transmitting an arbitrary object can be achieved by means of the object-based coding.

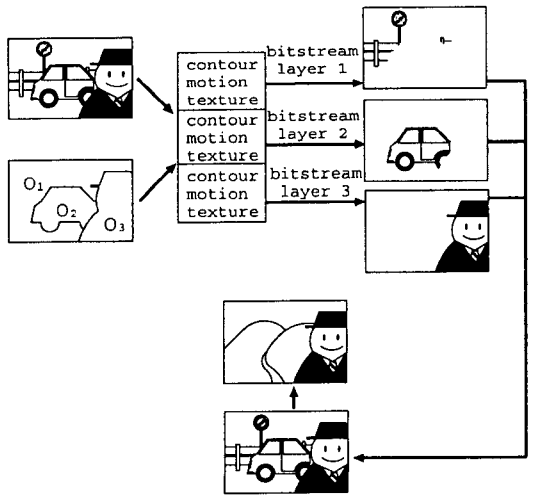


Fig. 1 Object-based coding.

The main encoding/decoding process of MPEG-4 is executed by the so-called MC-DCT coding in the same manner as H.263 and MPEG-2.

The conventional DCT, which performs very effectively spatial redundancy reduction, transforms pixel values of each block (8 x 8 pixels) to the frequency region. In this DCT process, consider a set of pixels in a block which constitute a boundary of an object, and then the pixels inside the object can be separated by this set from those outside the object. Now, supposing that a column/row contains a pixel on the boundary, we can see that there arise noises in the DCT coefficients of this column/row. This means that in this block there occurs an image distortion, which degrades the image quality.

To cope with the defect, an SA-DCT algorithm[2] has been devised to be adopted in MPEG-4, which achieves a variety of sophisticated schemes, but necessitates a considerable amount of additional hardware.

The SA-DCT calculations can be executed by using multimedia enhanced DSPs. However, for the mobile and portable use, the reduction of power consumption should be explored much further, and hence there still remains much room for devising VLSI architectures of SA-DCT/IDCT.

In this paper, a novel architecture is proposed for the SA-DCT core, which also can perform the SA-IDCT process. A fast SA-DCT algorithm is adopted in order to reduce the computation labor by half in comparison with the conventional architecture[3]. The SA-DCT core has been integrated with 160,000 transistors by a $0.35\mu m$ triple-metal CMOS technology.

## 2 SA-DCT Algorithm

Fig. 2 illustrates an outline of the SA-DCT process. Black and white regions indicate foreground objects and the background, respectively.

In this SA-DCT, the following three phases are executed.

[Phase 1]: An object in a block is specified.
[Phase 2]: Apply the following procedure to each column $j$:

1. Let column $j$ contain $N_j$ pixels of the object. Traverse pixels from top to bottom to seek a maximal sequence of those which represent pixel values of the object [remark that the value of a pixel

on the boundary is set to a specific value], and every time such a sequence is sought, the pixel values are shifted to the uppermost possible locations, one after another.

2. Column vector $x_j$ are transformed in the vertical direction by using a $N_j$-point one-dimensional DCT. Consequently, vector $a_j$, which consist of $N_j$ DCT coefficients, are calculated.

[Phase 3]: Apply the following procedure to each row $i$:

1. Let row $i$ contain $M_i$ coefficients. Traverse the coefficient from left to right to seek a maximal sequence of pixel values of the object, and every time such a sequence is sought, the pixel values are shifted to the leftmost possible locations, one after another.
2. Row vector $b_i$ is transformed with the horizontal $M_i$-point 1D DCT.

As a result, row vector $c_i$, or the entire SA-DCT coefficients are acquired.

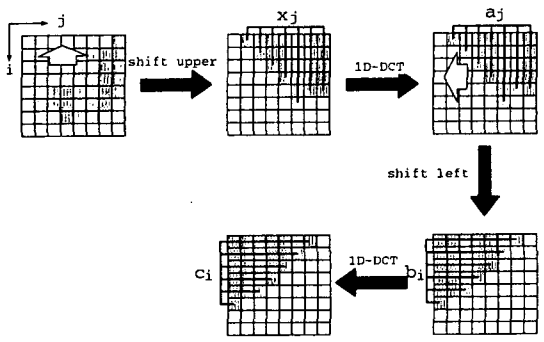


Fig. 2 SA-DCT

An N-point 1D DCT/IDCT is specified, as follows,

DCT:

$$C(u) = \sqrt{\frac{2}{N}}\alpha(u)\sum_{n=0}^{N-1} f(n)\cos\frac{\pi u(2n+1)}{2N}, \quad \text{for } u = 0,1,\ldots,N-1 \tag{1}$$

IDCT:

$$f(n) = \sqrt{\frac{2}{N}}\sum_{u=0}^{N-1} \alpha(u)C(u)\cos\frac{u\pi(2n+1)}{2N}, \quad \text{for } n = 0,1,\ldots,N-1 \tag{2}$$

(provided, $\alpha(0) = 1/\sqrt{2}$, $\alpha(u) = 1, u \neq 0$)

SA-IDCT can be calculated pixel values from SA-DCT coefficients by using the binary shape information which represents whether each pixels belong to the object, or not.

Owing to different calculations with the use of the shape information, there is a clear advantage for shape-adaptive DCT in the case of high image quality.

# 3 Multi-point DCT/IDCT

Multi-point 1D-DCT/IDCT is the most computationally intensive process in the whole SA-DCT /IDCT. Especially, the SA-DCT/IDCT needs multipliers much more than a conventional DCT/IDCT, since the N-point 1D-DCT/IDCT should be applied to N=1, 2, ..., 8.

To reduce the number of multipliers, a fast algorithm is required, which can share hardwares as much as possible in each value of $N$.

Recently, a variety of approaches have been proposed for fast multi-point DCT/IDCT algorithms. Table 1 shows the number of multiplications of each algorithm. However, the data flows of these algorithms are too complex to share multipliers. Specifically, they have to use 4 types of multipliers for $N = 2^n, 3^n, 5$, and 7 and hence these algorithms are difficult to reduce the number or multipliers further more.

Table 1 Fast multi-point DCT/IDCT algorithm.

| N | Chen [4] | Wang [6] | Lee [5] | Suehiro [7] | Hou [8] |
|---|---|---|---|---|---|
| 4 | 6 | 5 | 4 | 4 | 4 |
| 8 | 16 | 13 | 12 | 12 | 12 |
| 16 | 44 | 35 | 32 | 32 | 32 |
| 32 | 116 | 91 | 80 | 80 | 80 |
| 64 | 292 | 227 | 192 | 192 | 192 |

Thus we propose a new multi-point 1D-DCT matrix calculator, which is programmable so as to set matrix coefficients to be used in multipliers. In the proposed 1D-DCT calculator, the same multipliers can be shared in all cases of $N = 1, 2, ..., 8$.

# 4 VLSI Architecture

## 4.1 Organization

The overall organization of our SA-DCT/IDCT core is shown in Fig. 3, which consists mainly of Shift Block and 1D-DCT/IDCT Block.

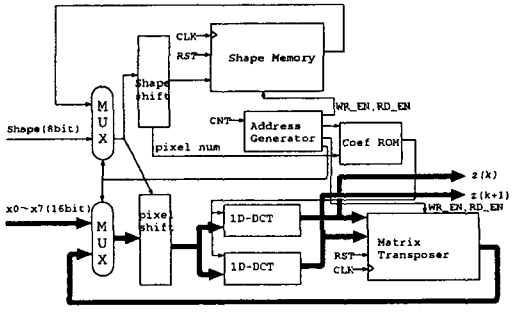


Fig. 3 Organization of SA-DCT.

## 4.2 Shift Block

Shift Block is to execute Phases 2-1 and 3-1 explained above. Shift Block is to shift pixel values and shape information separately. Fig. 4 shows examples of shifting pixel values.

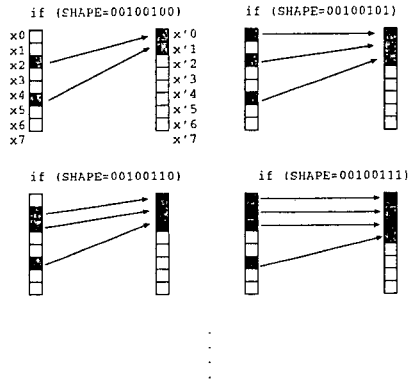


Fig. 4 Example of shifting pixel values.

Specifically, Shift Block has the functional of
(1) counting the number of pixel values of each maximal sequence and specifying the initial location of such a sequence, and
(2) shifting the pixel values of the object to the uppermost/leftmost locations, in each column/row.

It should be noticed here that, even if a block has a hole in a column/row, that is, a set of those pixel values of an object, which are located in a column/row of a block, constitute two or more maximal sequences, they can be packed compactly to the uppermost/leftmost of the column/row.

The shape information shift can be done by counting the number of object pixels. Fig. 5 shows an outline of the shifting shape information.

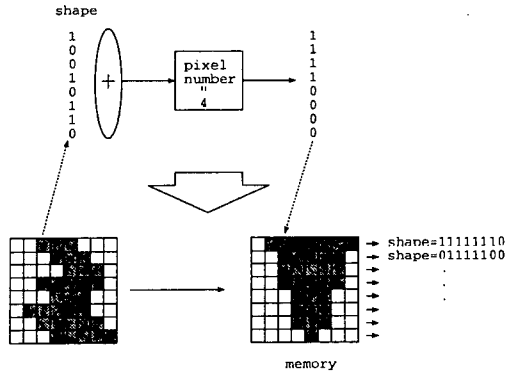


Fig. 5 Outline of shifting shape information.

### 4.3 1D-DCT/IDCT Block

The 1D-DCT/IDCT Block employs a 1×4 matrix calculator as shown in Fig. 6.

Since coeff0～coeff3 are programmable, all cases of $N = 1, 2, ..., 8$ matrix calculations can be executed only in one clock cycle.

1D-DCT/IDCT Block is to calculate DCT coefficients for each column/row of a block. Since the number of those pixel values of an object which are located in a column/row varies from 0 to 8, the N-point 1D-DCT/IDCT should be applied to N=1, 2, ..., 8.

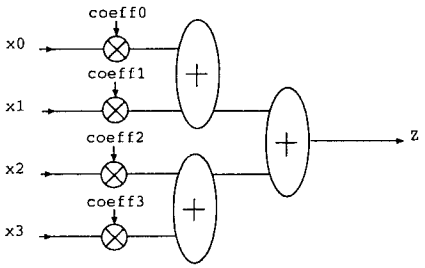


Fig. 6 1 × 4 matrix calculator

According to the Chen's algorithm, the 8-point 1D-DCT can be performed by means of two 4x4 matrices, in which the whole of 32 entries can be expressed with the use of only 8 parameters[4]. Each of the N-point 1D-DCT (N=1, 2, ..., 8) can also be executed by means of the 4x4 matrices in which the entries are specified in each case of N=1, 2, ..., 8. Furthermore, we employ the fast SA-DCT algorithm which can reduce the number of multipliers by making the best use of the symmetry of the 4x4 matrices; for example, Equations (3)-(4) and (5)-(6) show the cases of $N = 8$ and $N = 7$, respectively.

DCT:

$$\begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} = \begin{bmatrix} A & A & A & A \\ B & C & -C & -B \\ A & -A & -A & A \\ C & -B & B & -C \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_3 \\ z_5 \\ z_7 \end{bmatrix} = \begin{bmatrix} D & E & F & G \\ E & G & -D & -F \\ F & -D & -G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} \quad (3)$$

IDCT:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} A & B & A & C \\ A & C & -A & -B \\ A & -C & -A & B \\ A & -B & A & -C \end{bmatrix} \begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} + \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \\ z_5 \\ z_7 \end{bmatrix}$$

$$\begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \end{bmatrix} = \begin{bmatrix} A & B & A & C \\ A & C & -A & -B \\ A & -C & -A & B \\ A & -B & A & -C \end{bmatrix} \begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} - \begin{bmatrix} D & E & F & G \\ E & -G & -D & -F \\ F & -D & G & E \\ G & -F & E & -D \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \\ z_5 \\ z_7 \end{bmatrix} \quad (4)$$

DCT:

$$\begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} = \begin{bmatrix} A & A & A & A \\ B & C & D & E \\ F & G & H & I \\ J & K & L & M \end{bmatrix} \begin{bmatrix} x_0 + x_6 \\ x_1 + x_5 \\ x_2 + x_4 \\ x_3 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_3 \\ z_5 \end{bmatrix} = \begin{bmatrix} N & O & P & Q \\ R & S & T & U \\ V & W & X & Y \end{bmatrix} \begin{bmatrix} x_0 - x_6 \\ x_1 - x_5 \\ x_2 - x_4 \\ x_3 \end{bmatrix} \quad (5)$$

IDCT:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} A & B & F & J \\ A & C & G & K \\ A & D & H & L \\ A & E & I & M \end{bmatrix} \begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} + \begin{bmatrix} N & R & V \\ O & S & W \\ P & T & X \\ Q & U & Y \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \\ z_5 \end{bmatrix}$$

$$\begin{bmatrix} x_6 \\ x_5 \\ x_4 \end{bmatrix} = \begin{bmatrix} A & B & F & J \\ A & C & G & K \\ A & D & H & L \end{bmatrix} \begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} - \begin{bmatrix} N & R & V \\ O & S & W \\ P & T & X \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \\ z_5 \end{bmatrix} \quad (6)$$

A matrix multiplier in Fig. 6 can be improved to be adapted well to this fast algorithm. Fig. 7 shows the improved matrix multiplier.

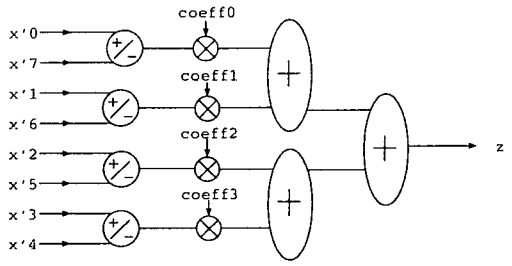


Fig. 7 Improved 1 × 4 matrix calculator

### 4.4 Transportation/Shape Memory

A time-shared implementation of one 1-D DCT in the proposed SA-DCT can attain 2-D SA-DCT.

Due to the difference of directions between the first and second SA-DCT, Transportation Memory and Shape Memory are provided. Specifically, DCT coefficients are stored in Transportation Memory and binary shape information is stored in Shape Memory. Fig. 8 exemplifies data reading/writing for 1D-DCT/IDCT.

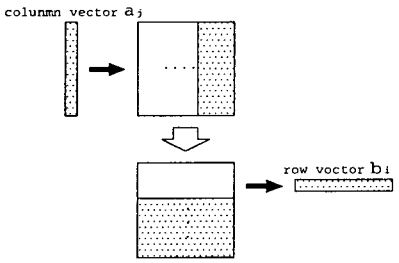


Fig. 8 Transportation/Shape Memory

## 5 Implementation Results

The proposed architecture has been synthesized through the use of Synopsys Design Compiler, in which the maximum clock frequency is 20 MHz. In this architecture, the 2D-DCT for a block can be performed in 64 cycles for N=8. Consequently, the proposed SA-DCT can code 1 MB (a macroblock is a unit of 16 × 16 pixels) in 384 cycles, and hence 52,000 MB per second. Thus our architecture satisfies Main@L2 (23,760MB/s), and 4 CIF format (704 x 576 pixels) at 30 frames/s.

## 6 Conclusion

This paper has discribed a VLSI architecture of the SA-DCT/IDCT. The distinctive feature of this architecture is that the number of multiplier has been reduced by the fast DCT algorithm. Since SA-IDCT shares multipliers and a transportation memory, this SA-DCT core can perform SA-IDCT with a small amount of additional hardware. In fact, our SA-DCT architecture has reduced the computation complexity by half in comparison with the conventional architecture. Development is continuing on an integrated set of architectures for VLSI implementation of the MPEG-4 codec.

Table 2 Implementation result.

| | |
|---|---|
| Technology | 0.35 COM triple-level Al |
| Number of Trs. | 160,000 |
| Clock frequency | 20MHz |

## References


[1] ISO/IEC 14496-2: "Information Technology - Generic coding of audio-visual objects", *International Standard*, (1999).

[2] P. Kauff, B. Makai, S. Rauthenberg, U. Golz, J. L. P. D. Lameillieure, T. Sikora: "Functional coding of video using a Shape-Adaptive DCT algorithm and an object-based motion, prediction toolbox," *IEEE Trans. Circuits and System for Video Technology*, vol. 7, no. 1, pp. 181–195 (Feb. 1997).

[3] H. Ohashi, T. Nakamura, S. Kurohmaru, H. Fujimoto, and J. Michiyama: "A study of variable length base 1-dimension DCT/IDCT circuit", *Technical Report of IEICE*, ICD98-299, pp. 23–28, (Mar. 1999).

[4] W. H. Chen, C. H. Smith, and S. C. Fralick: "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Communications*, vol. 9, no. 9, pp. 1004–1009 (Sept. 1997).

[5] B. G. Lee: "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing.*, vol. ASSP-32, pp. 1243–1245 (Dec. 1984).

[6] Z. Wang: "Fast algorithm for the discrete *W* transform and for the discrete fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 803–816 (Aug. 1984).

[7] N. Suehiro and M. Hatori: "Fast algorithm for the DFT and ohter sinusoidal transforms," *IEEE Trans. Acoust., Speech, Signal Processing.*, vol. ASSP-34, pp. 642–644 (June 1986).

[8] Hsieh S. Hou: "A fast recursive algarithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, No. 10, pp. 1455–1461 (Oct.1987).