HARDWARE EFFICIENT DCT IMPLEMENTATION FOR PORTABLE MULTIMEDIA TERMINALS USING SUBEXPRESSION SHARING

A.P.Vinod and E.M-K.Lai

School of Computer Engineering, Nanyang Technological University
Nanyang Avenue, Singapore 639798
Email: {asvinod, asmklai}@ntu.edu.sg

ABSTRACT

Low-power and high-speed implementation of discrete cosine transform (DCT) for mobile multimedia terminals presents a hardware design challenge. The cost of DCT implementation is dominated by the complexity of the multiplier. The systolic array and the memory-based designs do not consider the optimization of the multiplications in transform coding. A method to minimize the complexity of multiplication in DCT by efficient sharing of the common subexpressions that occur in the canonic signed digit (CSD) representation of the elements of the DCT matrix is presented here. Design example of a 8×8 *DCT using 16 bits shows that our method offers hardware reduction of 38% over conventional method.*

1. INTRODUCTION

The discrete cosine transform (DCT) has been widely recognized as the most effective technique among various transform-coding methods for image and video signal compression standards such as JPEG, MPEG, H.261 and H.263 [1]. As these standards find applications on the battery-operated systems such as portable computers, personal digital assistants (PDA) and portable communication equipments, it becomes imperative to design a low-power and high-speed DCT chip. The multiplier is the most power-consuming element in a DCT chip employed for portable multimedia applications. In CMOS technology, there are three sources of power dissipation arising from: switching (dynamic) currents, short-circuit currents, and leakage currents. Among these parameters, the switching component, which is a function of the effective capacitance, plays the most significant role [2]. It is possible to reduce power consumption by employing transformations such as reductions in critical path, number of operations, and average transition activity. These transformations result in architectures that minimize the effective capacitance of the circuit [2]. The critical path and transition activity can be minimized by employing efficient multiplication structures, as shown in the later part of this paper. Once these two parameters are optimized, the most obvious approach for capacitance reduction is to minimize the number of operations (and hence the number of switching events) in the data control flow graph. In a DCT chip, the number of operations is dominated by the amount of multiplication. Several techniques for efficient multiplication of the input data with the DCT coefficients have been proposed [3-7]. To meet the real-time video processing requirement, DCT implementations often use efficient dedicated hardware units that lead to high-speed but high hardware cost [3]. The multipliers used in the systolic array based designs [4], [5] consume a large silicon area, and hence these designs are not power efficient. The ROM based designs [6], [7] proposed to overcome the drawback of the systolic array designs reduce the complexity of multiplication by employing efficient ROM access operations. However, these designs [3-7] do not consider the optimization of the computationally intensive multiplication of the input data (image) with the DCT matrix, which is crucial in low-power implementations. In this paper, the numerical property of the elements of the DCT matrix is exploited to reduce the hardware cost. A method to implement low-complexity DCT using common subexpression sharing technique [8] is presented here.

The paper is organized as follows. In section 2, we analyze the complexity of multiplication in the DCT. The common subexpression sharing method is discussed in section 3. In section 4, we illustrate the implementation of DCT with design examples. Section 5 provides our conclusions.

2. MULTIPLIER COMPLEXITY IN DCT

The $N \times N$ DCT matrix $C = c(k,n)$ is defined as [1]:

$$c(k,n) = \begin{cases} \dfrac{1}{\sqrt{N}} & \text{for } k = 0 \text{ and } 0 \le n \le N-1 \\ \sqrt{\dfrac{2}{N}} \cos\left(\dfrac{\pi(2n+1)k}{2N}\right) & \text{for } 1 \le k \le N-1 \text{ and } \\ & 0 \le n \le N-1 \end{cases} \tag{1}$$

Using matrix notation, the DCT coefficients (Y) is obtained from $Y = CU$:

$$Y = \begin{bmatrix} c_{11} & c_{12} & c_{13} \cdot \cdot \cdot & c_{1N} \\ c_{21} & c_{22} & c_{23} \cdot \cdot \cdot & c_{2N} \\ . & . & . & . \\ . & . & . & . \\ c_{N1} & c_{N2} & \cdot \cdot \cdot & c_{NN} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ u_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{bmatrix} \qquad (2)$$

where u_i is the input data matrix and c_{ij} represent the elements of the DCT matrix. In digital systems, multiplication of a variable (image data, u_i) with a constant (DCT matrix elements, c_{ij}) is implemented using *shifts* and *adds* by representing the constant in CSD.

Definition 1 (Canonic Signed Digit, CSD): The number $b_{N-1}b_{N-2}...b_0$ is said to be in CSD representation if no two nonzero digits are consecutive and the number of nonzero digits is minimal, where $b_i \in \{0, +1, -1\}$.

In the case of a DCT chip for a specific application, the transform length (*N*) is fixed and hence the elements of the DCT matrix (*C*) are constants. Therefore, the *shifts* (decided by the CSD representations of the constants) can be hardwired, which is less expensive and hence the adder cost determines the hardware complexity. There are two classes of adders in a DCT structure, which are defined as follows.

Definition 2 (Inter-structure Adders): The adders used for computing the sum of the products, $\sum c_{ij}.u_i$, to obtain the output matrix (Y) are called inter-structure adders.

It requires N^2 multiplications and $N(N-1)$ additions to compute *Y*. For example, the first element of *Y* is computed using

$$y_1 = c_{11}.u_1 + c_{12}.u_2 + ... + c_{1N}.u_N \qquad (3)$$

The adders used to compute (3) are called inter-structural adders. If *N* is 8, it requires 7 (i.e., *N-1*) adders to obtain y_1.

Definition 3 (Intra-structure Adders): The adders used for computing the products, $c_{ij}.u_i$, are called intra-structure adders. In conventional implementation of multiplication using shifts and adds, if the number of nonzero bits in the CSD representation of c_{ij} is N_b, the number of intra-structure adders, (N_a), required to compute $c_{ij}.u_i$ is

$$N_a = N_b - 1 \qquad (4)$$

For example, consider the example of the multiplication to obtain the first term of (3), $y1(1) = c_{11}.u_1$. Assume $c_{11} = 0.6458 = 0.101001010101 = 2^{-1} + 2^{-3} + 2^{-6} + 2^{-8} + 2^{-10} + 2^{-12}$. The product $y1(1)$ can be expressed as

$$y1(1) = u_1 >> 1 + u_1 >> 3 + u_1 >> 6 + u_1 >> 8 + u_1 >> 10$$
$$+ u_1 >> 12 \qquad (5)$$

where u_1 is the data (image) and '>>' represents shift right operation. The adders used to compute (5) are called intra-structure adders. In this case, N_b is 6 and five intra-structure adders are required to obtain $c_{11}.u_1$ as in (5).

The objective of the design methods [3-7] is to reduce the number of inter-structure adders in DCT implementation by exploiting the redundancies in the values of c_{ij}. However, the actual cost of implementation is dominated by the cost of multipliers required to compute the products, $c_{ij}.u_i$, i.e., intra-structure adders. These works [3-7] do not address the hardware cost of each of these multiplications. We present a comparison of the hardware cost in terms of inter-structure and intra-structure adders required to obtain (3) for $N = 8$. For convenience, we assume that an identical number of adders are required to obtain each term $(c_{ij}.u_i)$ in (3). (The actual adder requirement depends on the number of nonzero bits in the CSD representations of the DCT matrix elements, $C_{12}, C_{13}, ...C_{18}$). Thus a total of forty intra-structure adders (considering 8 multiplications, requiring 5 adders per multiplication) are required to compute the products, $c_{11}.u_1, c_{12}u_2, ...c_{18}u_8$, of (3). Note that this adder requirement is substantially higher than the number of inter-structure adders (i.e., 7 adders) of (3). Therefore, a more apparent goal of reducing the number of operations (for reducing power) is to minimize the number of intra-structure adders. In next section, we show that the constant property of DCT matrix can be exploited to reduce the number of operations (intra-structure additions) for a low-power implementation.

3. COMMON SUBEXPRESSION SHARING

The goal of Hartley's common subexpression elimination (CSE) [8], which was originally proposed for digital filters, is to identify multiple occurrences of identical bit patterns that are present within each filter coefficient. Since the computation of multiple identical expressions needs to be implemented only once, the resources necessary for these operations can be shared. We reformulate the CSE technique in the context of DCT here. The pattern [1 0 1] in the above-mentioned example of c_{11} is present thrice, which can be expressed as a common subexpression (CS),

$$u_2 = u_1 + u_1 >> 2 \qquad (6)$$

Using the CS, the expression (5) can be written as

$$y1(1) = u_2 >> 1 + u_2 >> 6 + u_2 >> 10 \qquad (7)$$

Hence the multiplication structure optimized using the CSE given by (6) and (7) requires two adders less than the original structure of (5). Thus using CSE, the number of adders required to implement the multiplication is minimized. However, unlike the filter type operations, the direct application of CSE to DCT will not offer significant hardware reduction. In the case of a filter, the CSE can be applied to exploit the redundancies in multiplication of one variable (signal, x) with multiple constants (coefficients, $h(n)$). On the other hand, such multiplications (multiplication of a data element with multiple DCT matrix elements) do not occur in the direct implementation of DCT. Note that each data (u_i) is multiplied with distinct element c_{ij} in (3). Therefore, we need to reformulate the DCT matrix for efficient substitution of CSE.

3.1. Reformulation of DCT for CSE

Let us consider the expressions for the output Y in (2):

$$\begin{aligned} y_1 &= c_{11}.u_1 + c_{12}.u_2 + ... + c_{1N}.u_N \\ y_2 &= c_{21}.u_1 + c_{22}.u_2 + ... + c_{2N}.u_N \\ &\cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \\ y_N &= c_{N1}.u_1 + c_{N2}.u_2 + ... + c_{NN}.u_N \end{aligned} \qquad (8)$$

The objective of our method is to minimize the intra-structure adders required for computing the products, $c_{ij}.u_i$. It can be seen from (8) that each data needs to be multiplied with several constants. For example, the data u_1 is multiplied with $c_{11}, c_{21}, ...c_{N1}$, which can be expressed as

$$u_1 \cdot [c_{11} \ c_{21}.......c_{N1}]^T \qquad (9)$$

By sharing the CS that exist in the matrix elements c_{ij} in (9), the number of intra-structure adders can be reduced. Thus, in order to obtain the products $c_{ij}.u_i$, the DCT computation can be reformulated as

$$\begin{aligned} &u_1 \cdot [c_{11} \ c_{21}.......c_{N1}]^T \\ &u_2 \cdot [c_{12} \ c_{22}.......c_{N2}]^T \\ &\cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \\ &u_N \cdot [c_{1N} \ c_{2N}.......c_{NN}]^T \end{aligned} \qquad (10)$$

The CSE technique can be applied for efficient computation of (10).

4. DESIGN EXAMPLE

We present the design of a 8×8 DCT using the CSE method reformulated in previous section. The DCT matrix is obtained using (1). Due to space constraints, we illustrate the implementation of (9) and the same method can be extended to obtain (10). The DCT matrix elements, c_{ij}, of (9) are listed in Table I (Note that N is taken as 8).

Table I DCT matrix elements of (9)

c_{11}	0.3536	c_{51}	0.3536
c_{21}	0.4904	c_{61}	0.2778
c_{31}	0.4619	c_{71}	0.1913
c_{41}	0.4157	c_{81}	0.0975

The CS present in the 12-bit CSD representation of these elements are shown in Fig. 1. The numbers in the first row of Fig. 1 represent the number of bitwise right shifts. The number of intra-structure adders, (N_a), required to compute $c_{ij}.u_i$ in conventional method can be determined using (4). For example, in the case of c_{11}, $N_b = 6$, and hence five adders are required to obtain $c_{11}.u_1$. Similarly, it can be determined from Fig. 1 that a total of thirty-one intra-structure adders are required to obtain (9) in conventional method. Using CS, the bit patterns [1 0 –1] (shown inside the solid rectangle), and [1 0 1] (dotted rectangle) in Fig. 1 can be expressed as

$$u_2 = u_1 - u_1 >> 2 \text{ and } u_3 = u_1 + u_1 >> 2 \qquad (11)$$

Using (11), we can express $c_{11}.u_1$ as

$$c_{11}u_1 = u_2 >> 2 - u_2 >> 6 + u_3 >> 10 \qquad (12)$$

Note that only two adders are needed to compute (12). Table II lists the expressions for computing the products, $c_{ij}.u_i$ of (9). It requires only sixteen adders (fourteen adders in Table I and two adders for obtaining the CS (11)) to implement (9). This is a reduction of 48% over conventional method. Using CSE, the total number of intra-structure adders required to obtain all the products of the 8×8 DCT (i.e., when $N = 8$ in (10)) is 130, whereas the adder requirement is 210 in conventional implementation. Thus, the adder reduction achieved using the CSE is 38%. The critical path length and the transition activity also need to be minimized apart from reducing the number of additions. We employ the tree-structure shown in Fig. 2, which performs parallel addition to implement the multiplication. The input operands shown in Fig. 2 represent the data u_i shifted corresponding to the bit positions in the CSD representation of c_{ij}.

Definition 4 (Adder-step): One addition stage in a maximal path of decomposed multiplications is termed the adder-step (AS).

Definition 5 (Critical path): The number of adder-steps in a multiplication structure is called the critical path. The number of adder-steps, (A_n), required to compute the sum of n operands is given by $2^{A_n} \geq n.$ From this, we obtain

$$A_n = \left\lceil \frac{\log_{10}(n)}{\log_{10}(2)} \right\rceil \qquad (13)$$

The A_n obtained (13) is the lowest number of adder-steps (lower bound) possible to achieve in an addition structure since the tree structure considered in our method performs parallel addition. Therefore, our method always results in a minimum adder-step implementation and hence has the lowest critical path. Moreover, when compared with a chain (serial) implementation, the signal paths are more balanced in a tree implementation and hence the amount of extra transitions is reduced.

	1	2	3	4	5	6	7	8	9	10	11	12
c_{11}	0	1	0	-1	0	-1	0	1	0	1	0	1
c_{21}	0	1	0	0	0	1	0	-1	0	-1	0	-1
c_{31}	0	1	0	0	0	-1	0	-1	0	1	0	-1
c_{41}	0	1	0	-1	0	1	0	1	0	-1	0	0
c_{51}	0	1	0	-1	0	-1	0	1	0	1	0	0
c_{61}	0	0	1	0	1	0	1	0	-1	0	0	-1
c_{71}	0	0	1	0	-1	0	0	1	0	-1	0	0
c_{81}	0	0	0	1	0	-1	0	0	0	1	0	-1

Fig. 1.CS in the CSD form of c_{ij} in (9).

Table II Expressions for computing the products, $c_{ij}.u_i$ of (9)

$c_{11}.u_1$	$u_2 >> 2 - u_2 >> 6 + u_3 >> 10$
$c_{21}.u_1$	$u_1 >> 2 + u_2 >> 6 - u_3 >> 10$
$c_{31}.u_1$	$u_1 >> 2 - u_3 >> 6 + u_2 >> 10$
$c_{41}.u_1$	$u_2 >> 2 + u_3 >> 6 - u_1 >> 10$
$c_{51}.u_1$	$u_2 >> 2 - u_2 >> 6 + u_1 >> 10$
$c_{61}.u_1$	$u_3 >> 3 + u_2 >> 7 - u_1 >> 12$
$c_{71}.u_1$	$u_2 >> 3 + u_2 >> 8$
$c_{81}.u_1$	$u_2 >> 4 + u_2 >> 10$

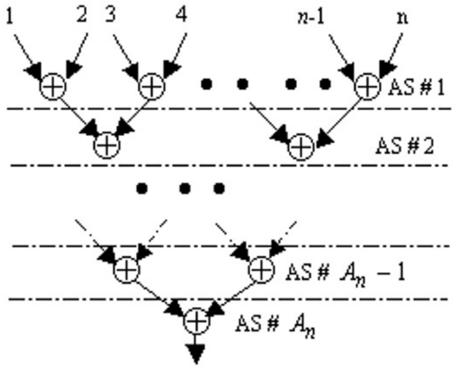

Fig. 2. Tree structure used for addition.

5. CONCLUSIONS

We have proposed an efficient method based on CSE for low-complexity implementation of a DCT chip. The complexity of multiplication in DCT is analyzed. While the conventional low-power implementation methods focus on reducing the number of inter-structure adders, our method is based on minimizing the number of intra-structure adders, which is the most power-consuming component in a DCT chip. Our method results in a multiplication structure that has minimum number of operations, critical path and transition, and hence offers a power efficient solution.

6. REFERENCES

[1] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video and Audio Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Trans. On CAD,* vol. 14, no. 1, pp. 12-31, Jan. 1995.

[3] P. Pirsch, N. Ranganathan, "VLSI architectures for video compression-A survey," *Proc. IEEE,* vol. 83, pp. 220-246, Feb. 1995.

[4] L. W. Chang and M. C. Wu, "A unified systolic array for discrete cosine and sine transforms," *IEEE Trans. Signal Processing,* vol. 39, pp. 192-194, Jan. 1991.

[5] J. I. Guo, C. M. Liu, and C. W. Jen, "A new array architecture for prime length discrete cosine transform," *IEEE Trans. Signal Processing,* vol. 41, no. 1, pp. 436-442, 1993.

[6] D. Slawecki and W. Li, "DCT/IDCT processor design for high data rate image coding," *IEEE Trans. Circuits Syst. Video Technology,* vol. 2, pp. 135-146, June 1992.

[7] J. I. Guo, C. M. Liu, and C. W. Jen, "The efficient memory-based VLSI arrays for DFT and DCT," *IEEE Trans.Circuits Syst. II,* vol. 39, no. 10, pp. 723-733, 1992.

[8] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II,* vol. 43, pp. 677-688, Oct. 1996.