An Efficient Parallel-Pipelined Algorithm and Architecture for Computation of 2-D DCT on Two Successive Processors

Tze-Yun Sung   Yi-Hsun Sung   Chih-Sin Chen
Electrical Engineering Department
Chung Hua University
Hisn-chu, Taiwan 300-12
Tel : 886-3-579-0902          Fax: 886-3-578-1150
E-mail : bobsung@btcx.com.tw, tom@btcx.com.tw, sin@btcx.com.tw

Abstract

2-D Discrete Cosine Transform (DCT) applies on image compression and saves more memories. In this paper, we use row-column decomposition method, and propose a parallel-pipelined architecture to implement a $8 \times 8$ DCT processor. This processor involves two 8-point DCT processors, two SRAMs (64 words), twelve multiplexers, sixteen registers, two timing controllers and counters. The complex arithmetic unit (AU) of DCT processor is designed by CORDIC arithmetic. This processor saves hardware and achieves high performance. The chip of $8 \times 8$ DCT processor is fabricated in a 0.25 $\mu$m CMOS. All control signals are generated internally on-chip. The chip has been implemented and applied on image compression successfully.

Keywords:DCT, row-column decomposition method, parallel-pipelined, CORDIC arithmetic, image compression.

1. Introduction

For the application of image data compression, a transform encoding technique is more suitable than that of a linear prediction coding (LPC). It is clear that the Walsh-Hadamard transform is the simplest one for implementation. The kernel matrix of the Walsh-Hadamard transform involves only additions and subtractions [1]. However, the Walsh-Hadamard transform is definitely not the best transform encoder due to the absence of symmetry property of the transform kernel. Currently, the Discrete Cosine Transform (DCT) becomes the best transform encoder because of that the transform kernel of the DCT involves both the periodicity and symmetry properties, and therefore the DCT can perform very close to the theoretically optimal Karhunen-Loeve transform [2]. Conventionally, a $2N$-point DCT or the double size Fast Fourier Transform (FFT) computation algorithm was employed to the implementation of DCT involving complex arithmetic [1], where the processor units as well as the computation time were $2N$ and $3N$, respectively required for the implementation of DFT algorithm, and were $2\log 2N$ and $O(\log 2N + 1)$, respectively required for the implementation of FFT algorithm. Some fast computation algorithms were proposed and discussed [3]-[11], and the VLSI chips implementations of some DCT concurrent architectures were accomplished for real-time applications [12]-[14].

In this paper, a parallel-pipelined computation
approach for the DCT implementation will be investigated. In section II, basic considerations for 2-D DCT algorithm will be described. In section III, both the parallel-pipelined 1-D and 2-D DCT architectures are presented. In section IV, the 8×8 2-D DCT architectures are developed, the chip of 8×8 2-D DCT processor is implemented, and comparisons of different architectures and chips will also be discussed. Finally, conclusions will be given.

2. 2-D Discrete Cosine Transform (DCT) Algorithm

The N-point 1-D DCT is defined as

\[
Y(m) = \sum_{n=0}^{N-1} \sqrt{2} K_m \cos \left( \frac{(2n+1)m\pi}{2N} \right) x(n),
\]

where \( m = 0, \ldots, N-1 \),

\[
K_m = \begin{cases} 1 & \text{if } m = 0, \\ \frac{1}{\sqrt{2}} & \text{if } m > 0. \end{cases}
\]

The \( M \times N \)-point DCT is defined as

\[
Z(u,v) = \frac{2}{\sqrt{M \cdot N}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) \cos \left( \frac{(2m+1)u\pi}{2M} \right) \cos \left( \frac{(2n+1)v\pi}{2N} \right),
\]

where \( u = 0, \ldots, M-1, v = 0, \ldots, N-1 \),

\[
c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0, \\ 1 & \text{if } k > 0. \end{cases}
\]

According to eq. (2), we have

\[
Z(u,v) = \frac{2}{\sqrt{M}} \sum_{m=0}^{M-1} \sqrt{2} c(u) \cos \left( \frac{(2m+1)u\pi}{2M} \right) \sum_{n=0}^{N-1} \sqrt{2} c(v) \cos \left( \frac{(2n+1)v\pi}{2N} \right) x(m,n),
\]

(3)

Where \( u = 0, \ldots, M-1, v = 0, \ldots, N-1 \),

\[
c(k) = \frac{1}{\sqrt{2}}, \text{ where } k = 0,
\]

\[
e 1, \text{ where } k > 0.
\]

We use row-column decomposition method, and then 8-point 1-D DCT can be represented as

\[
\begin{bmatrix}
Y(0) \\
Y(1) \\
Y(2) \\
Y(3) \\
Y(4) \\
Y(5) \\
Y(6) \\
Y(7)
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 \\
-1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1
\end{bmatrix} \begin{bmatrix}
x(0) \\
x(1) \\
x(2) \\
x(3) \\
x(4) \\
x(5) \\
x(6) \\
x(7)
\end{bmatrix}.
\]

(4)

Let

\[
Y = [Y(0) \ Y(1) \ Y(2) \ Y(3) \ Y(4) \ Y(5) \ Y(6) \ Y(7)]
\]

and

\[
X = [x(0) \ x(1) \ x(2) \ x(3) \ x(4) \ x(5) \ x(6) \ x(7)],
\]

and eq. (4) can be represented as

\[
Y = \frac{1}{\sqrt{8}} CX' = TX', \text{ where } \frac{1}{\sqrt{8}} C = T,
\]

where

\[
a = \sqrt{2} \cos \left( \frac{\pi}{16} \right), \quad b = \sqrt{2} \cos \left( \frac{\pi}{8} \right), \quad c = \sqrt{2} \cos \left( \frac{3\pi}{16} \right),
\]

\[
d = \sqrt{2} \cos \left( \frac{5\pi}{16} \right), \quad e = \sqrt{2} \cos \left( \frac{3\pi}{8} \right), \quad f = \sqrt{2} \cos \left( \frac{7\pi}{16} \right).
\]

According to eq. (3), we have

\[
Z = TXT' = T(X' T)' = TY'.
\]

Hence,

\[
2-D \ DCT(X) = 1-D \ DCT((1-D \ DCT(X))').
\]

3. A Parallel-Pipelined 2-D DCT
Architecture

According to eq. (4), the \( N \)-point 1-D DCT needs \( N \)-time unit, and we have
\[
y_n(m) = y_{n-1}(m) + \mu_m \cdot x(n-1)
\]
where \( \mu \) are coefficients in transform matrix \( (T) \), \( n \) is an \( n \)-th time clock, and \( y_0(m) = 0 \).

We use eq. (5) and propose the parallel-pipelined \( N \times N \) 2-D DCT architecture which is shown in Fig. 1. This architecture possesses two-transposition memory, two-1-D \( N \)-point DCT processor, de-multiplexer, multiplexer, system controller and interface. \( N \)-point 1-D DCT input-processor writes results to transposition memory-1 and transposition memory-2 alternately. \( N \)-point 1-D DCT output-processor reads data from transposition memory-1 and transposition memory-2 alternately, and output the results. The time/space due to parallel-pipelined \( N \times N \) 2-D DCT architecture is shown in Fig. 2.

4. Implementation of 2-D Parallel-Pipelined DCT Processor

- 1-D DCT Processor Implementation

At first, the 8-point 1-D DCT processor would be introduced; the 1-D DCT processor has complex arithmetic units (AUs), which can be accomplished by CORDIC arithmetic [15], [16]. The 1-D 8-point DCT processor is shown in Fig. 3.

The \( T \)-transform matrix of 16-point DCT is represented as
\[
\begin{vmatrix}
g & h & i & j & k & l & m & n \\
g & a & c & d & f & j & k & l \\
h & a & c & d & f & j & k & l \\
i & b & e & -b & e & b & e & -b \\
j & c & -c & c & f & a & d & -d \\
k & d & -d & a & d & a & f & c \\
l & e & -b & e & h & b & e & -b \\
m & j & -h & m & k & l & g & l \end{vmatrix}
\]

where
\[
g = \sqrt{2} \cos \left( \frac{\pi}{32} \right), h = \sqrt{2} \cos \left( \frac{3\pi}{32} \right), i = \sqrt{2} \cos \left( \frac{5\pi}{32} \right),
\]
\[
j = \sqrt{2} \cos \left( \frac{7\pi}{32} \right), k = \sqrt{2} \cos \left( \frac{9\pi}{32} \right), l = \sqrt{2} \cos \left( \frac{11\pi}{32} \right),
\]
\[
m = \sqrt{2} \cos \left( \frac{13\pi}{32} \right), n = \sqrt{2} \cos \left( \frac{15\pi}{32} \right).
\]

The 16-point DCT processor is also exploited by cascading an expanding module, which is shown in Fig. 4. This architecture has high-regularity and scalability.

- Chip Implementation of 8×8 2-D DCT Processor

8×8 2-D DCT processor involves two 1-D DCT processors, timing controller/counter, multiplexers and two-SRAM (64 words). The architecture of 8×8 2-D DCT processor is shown in Fig. 4. The chip layout of 8×8 2-D DCT processor is shown in Fig. 5. The chip is fabricated in a 0.25 \( \mu \) m CMOS, double-metal technology. All control signals are generated internally on-chip [17].

- Comparisons of Different 8×8 2-D DCT Architectures and Chips
Some high-performance implementations employ highly parallel approaches with high arithmetic cost. In this work, we focus on parallel-pipelined dedicated 2-D DCT architectures with a throughput rate of at least one output sample per cycle, targeted for video compression. In Table I, we select some typical proposed architecture and make the comparison [8]-[10]. In Table II, we list features of other typical DCT/IDCT chips. They still incur large gate count requirement [11]-[14].

5. Conclusions

Based on the symmetry property considerations and row-column decomposition method, a 1-D DCT computation algorithm has been proposed implemented into VLSI parallel-pipelined architecture. In this 1-D DCT processor, the CORDIC arithmetic were used to for the implementation of the DCT AU so as to perform complex multiply function, and there are $N$-AU required for computation of a frame of $N$-point data. Further, a 2-D DCT processor has been proposed in accordance with the 2-D DCT computation decomposed into two successive 1-D DCT kernels. These architectures avoid the complex shuffle implementations, are of trade-off designs and controlled by MCU, hence they are more scalable, flexible and efficient, and therefore very suitable for VLSI implementation.

![Fig.1 Parallel-pipelined $N \times N$ 2-D DCT architecture](image1)

<table>
<thead>
<tr>
<th>Time Unit ($t$)</th>
<th>1-D DCT-1(P1)</th>
<th>TM-1</th>
<th>TM-2</th>
<th>1-D DCT-2(P2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t=1\sim N$</td>
<td>active</td>
<td>P1 writes</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$t=N+1\sim 2N$</td>
<td>active</td>
<td>P2 reads</td>
<td>P1 writes</td>
<td>active</td>
</tr>
<tr>
<td>$t=2N+1\sim 3N$</td>
<td>active</td>
<td>P1 writes</td>
<td>P2 reads</td>
<td>active</td>
</tr>
<tr>
<td>$t=3N+1\sim 4N$</td>
<td>active</td>
<td>P2 reads</td>
<td>P1 writes</td>
<td>active</td>
</tr>
<tr>
<td>$t=4N+1\sim 5N$</td>
<td>active</td>
<td>P1 writes</td>
<td>P2 reads</td>
<td>active</td>
</tr>
</tbody>
</table>

![Fig. 2 Time/space due to parallel-pipelined $N \times N$ 2-D DCT architecture](image2)
(a) Complex arithmetic unit (AU) of DCT processor (X: complex number)

(b) 8-point 1-D DCT processor (X: complex number)

<table>
<thead>
<tr>
<th>$t$</th>
<th>CS$_2$</th>
<th>CS$_1$</th>
<th>CS$_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 3 8-point 1-D DCT Processor and sequence of CS$_n$
Fig. 4 The $8 \times 8$ 2-D DCT processor (SRAM: 64 words)

Fig. 5 The chip layout of $8 \times 8$ 2-D DCT processor
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-multipliers</td>
<td>28</td>
<td>64</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Real-adders</td>
<td>134</td>
<td>88</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>Complex-multipliers</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>Complex-adders</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>Memory (Words)</td>
<td>~384</td>
<td>~200</td>
<td>~370</td>
<td>~168</td>
</tr>
<tr>
<td>Hardware complexity (AUs)</td>
<td>O(N log N)</td>
<td>O(N²)</td>
<td>O(log N)</td>
<td>O(2^N)</td>
</tr>
<tr>
<td>Throughput (outputs/cycle)</td>
<td>16</td>
<td>8</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Scalability</td>
<td>poor</td>
<td>good</td>
<td>good</td>
<td>better</td>
</tr>
</tbody>
</table>

Table I Comparison of different 8×8 2-D DCT architectures

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td>Row-column method</td>
<td>Row-column method</td>
<td>Row-column method</td>
<td>Direct method</td>
<td>Row-column decomposition method</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Time-recursive</td>
<td>Variable threshold voltage scheme</td>
<td>14-multiplier Gate-array</td>
<td>Folding strategy</td>
<td>Folding-frame strategy</td>
</tr>
<tr>
<td>Throughput</td>
<td>1 pixel/cycle</td>
<td>1 pixel/cycle</td>
<td>1 pixel/cycle</td>
<td>1 pixel/cycle</td>
<td>1 pixel/cycle</td>
</tr>
<tr>
<td>Latency</td>
<td>72 clocks</td>
<td>112 clocks</td>
<td>80 clocks</td>
<td>86 clocks</td>
<td>64 clocks</td>
</tr>
</tbody>
</table>

Table II Comparison of different 8×8 2-D DCT chips

References


