A Recursive Computation of the 2-D DCT: Algorithm, Architectures and FPGA Implementation

Shaofeng An

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

February, 2008

Your file Votre référence
ISBN: 978-0-494-40877-3
Our file Notre référence
ISBN: 978-0-494-40877-3

ABSTRACT

A Recursive Computation of the 2-D DCT: Algorithm, Architectures and FPGA Implementation

Shaofeng An

The discrete cosine transform (DCT) is widely used in the area of signal and image processing. The 2-D DCT has been used in image compression and become part of image and video standards. The 2-D DCT computation involves a large amount of data. Many applications require the systems to be in small volume and operate in real-time. Designing such a system for 2-D DCT is a challenging task.

In this thesis, a new recursive algorithm and two types of circuit architectures are presented for the computation of the 2-D DCT. The new algorithm permits to compute the 2-D DCT by a simple procedure of the 1-D recursive calculations involving only *cosine* coefficients. A recursive kernel for the proposed algorithm contains a small number of operations. Also, it requires a smaller number of pre-computed data compared to many of existing algorithms in the same category. The kernel can be easily implemented in a simple circuit block with a short critical delay path. In order to evaluate the performance improvement resulting from the new algorithm, an architecture for the 2-D DCT designed by direct mapping from the computation structure of the proposed algorithm has been implemented on an FPGA board. The results show that the reduction

of the hardware consumption can easily reach 25% and the clock frequency can increase 17% compared to a system implementing a recently reported 2-D DCT recursive algorithm. For a further reduction of the hardware, another architecture has been proposed for the same 2-D DCT computation. Using one recursive computation block to perform different functions in each clock cycle, this architecture needs only approximately one half of the hardware that is required in the first architecture, which has been confirmed by an FPGA implementation.

Acknowledgements

I wish to express my sincere appreciation to my supervisor Dr. Chunyan Wang for her help and support during my study at Concordia. Her insight, expertise and wealth of knowledge are invaluable to me. Without her help, I could not finish this study.

I would like to thank my friends of Minhong Zhou, Li Zhu, Chengwei Gu, Jing Li, Bo Li and Yang Tang for their reviews of this thesis and helps to my life.

Finally, I would like to thank my family, especially my wife Xujun Li, for all the love and support from them.

CONTENTS

List of Acronyms and Abbreviations

ASIC	Application specific integrated circuit
DCT	Discrete cosine transform
1-D DCT	One dimension discrete cosine transform
2-D DCT	Two dimension discrete cosine transform
DST	Discrete sine transform
DFT	Discrete Fourier transform
DHT	Discrete Hadamard transform
DIT	Decimation-in-time
DIF	Decimation-in-frequency
FPGA	Field programmable gate array
IDCT	Inverse discrete cosine transform
JPEG	Joint Photographic experts group
KLT	Karhunen-Loeve transform
MPEG	Moving pictures expert group
Mod	The module operators
RTL	Register transfer level
VLSI	Very large scale integration

List of Primary Symbols

A	Matrix with cosine basis function of 1-D DCT
I	$N \times N$ identity matrix
M	Size of the image block of the condensed 1-D DCT
N	Size of the image block of the 2-D DCT
n_1	First dimensional index Variable of the input of the 2-D DCT
n_2	Second dimensional index Variable of the input of the 2-D DCT
k_1	First dimensional index Variable of the output of the 2-D DCT
k_2	Second dimensional index Variable of the output of the 2-D DCT
$u(k)$	Constant factor of the DCT
$x_a(m_1)$	First term of the pre-addition
$x_s(m_2)$	Second term of the pre-addition
X_{ac1}	First output of the kernel
X_{ac2}	Second output of the kernel
$X(k)$	Output of the 1-D DCT
$x(n)$	Input of the 1-D DCT
$x(n_1, n_2)$	Input of the 2-D DCT
$Y(\omega_1, \omega_2)$	Output of the kernel
$X(k_1, k_2)$	Output of the 2-D DCT

List of Figures

List of Tables

Chapter 1

Introduction

Image and video processing is gaining more importance in our daily lives because of wide applications of such signals in the communication and security systems. The signal of an image contains a large number of data. Image signal transmission, processing and storage require a wide brand and large memory space. Thus, image compression is often necessary to apply in the procedure in order to facilitate image transmission.

The objective of an image compression is to reduce the redundancy of image data. There are two kinds of image compression, lossy and lossless ones. The lossless compression is usually applied to images requiring high quality preservation, while a lossy compression is widely used for natural images where minor loss of fidelity is acceptable to achieve a substantial compression rate.

One of the most commonly-used lossy compressions is by means of the transform coding. There are several transforms, such as the discrete Fourier transform (DFT), the discrete cosine transform (DCT), the discrete Hadamard transform (DHT), and the Karhunen-Loeve (KLT), used in the coding process. The DCT may be the most commonly used transform for this purpose.

In this chapter, the important role of the DCT in the image compression will be described. Then the motivation and objective of the work of this thesis will be presented. The scope of the work and the organization of the thesis will also be presented.

1.1 DCT in Image Compression

As mentioned previously, transform coding is one of the most commonly-used methods of lossy compression. Among the varieties of transforms, the discrete cosine transform employs cosine function instead of complex exponential functions employed in DFT. Therefore, the DCT has a computational complexity lower than that of the DFT by almost a factor of 2. Besides, the energy of the signal information in most cases tends to be concentrated on its low-frequency components, which matches the characteristic of "low-pass" of the DCT. Hence, the 2-D DCT-based image coding has been adopted as a standard in many applications, such as JPEG, MPEG-2, MPEG-4, and CCITT Recommendation H. 26x [1].

The procedure of the 2-D DCT-based image compression is shown in Fig. 1.1. In this procedure, the 2-D DCT provides the information of the signal in the spatial frequency domain, and thus by carrying out the quantization in this domain, the amount of information can be reduced, hence achieving a data compression. The quality of the 2-D DCT computation is very important in this compression of image signals. The 2-D DCT deals with a large amount of uncompressed image data and its operation is computationally intensive. Moreover, in many procedures where the image compression

is used a speed for real-time processing is needed. With the requirements of the speed and the large flow of data, it is very challenging to develop a system that is able to handle the computation task at a fast pace. Moreover, if there is a restriction of space to allocate the circuit of the 2-D DCT, the challenge will be even more critical.

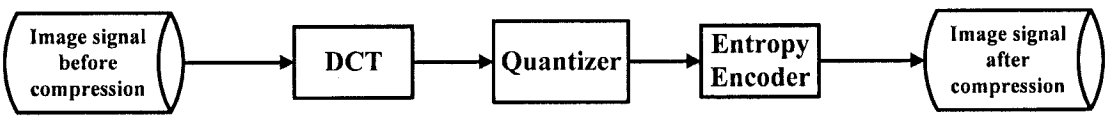

Fig. 1.1 Procedure of a DCT image compression [2].

1.2 Motivation and Objective of the Work

As mentioned in Section 1.1, the discrete cosine transform (DCT) is very important and widely used in the application of image compression. Although the computation of the 2-D DCT is well defined, its calculation procedure can be varied according to the way of decomposing the 2-D computations into 1-D ones. Thus, the development of a 2-D DCT algorithm becomes an important theme in this particular area of signal processing. Also, a 2-D DCT algorithm can be implemented in different architectures and results in different performance, which makes the research on the 2-D DCT algorithms and architectures significant.

The objectives of the work presented in this thesis have two aspects. The first is to develop an efficient algorithm for the 2-D DCT computation. The algorithm needs to be designed to have a small number of recursive cycles and simple calculations in each cycle.

In particular, each cycle should involve a reduced number of multiplications and additions. The other objective is to design architectures for the recursive algorithm. A circuit architecture needs to be built to implement the new algorithm to assess the performance improvement resulting from it. Also, aiming at improving the performance at the architecture level, new circuit structures are to be developed for recursive algorithm. These structures will be implemented on FPGA boards to validate the results of the research.

1.3 Scope and Organization of the Work

For this thesis work, the research on the development of a recursive 2-D DCT algorithm will be based on non-row-column decomposition method, in order to have a small number of recursive cycles. The design of the architectures has its emphasis on reducing the circuit complexity for the 2-D DCT computation.

The thesis is organized as follows. In Chapter 2, the background for developing the 2-D DCT algorithm is presented. Also, some of the existing algorithms and architectures relevant to this work are also presented. Chapter 3 is dedicated to the description of the proposed recursive algorithm for the 2-D DCT. In Chapter 4, the designs of the architectures implementing the proposed algorithm are presented. The FPGA implementations of these architectures and the results are also described in this chapter. Chapter 5 summarizes the work of this thesis and highlights its technical contributions to the field.

Chapter 2

Background

In the imaging and video signals, a high degree of correlation exists between the intensity values of adjacent pixels of an image. By removing such a large number of redundant information, image compression can be achieved in the process. The discrete cosine transform (DCT) is widely used in the applications for the compression of digital image and video signals. Through the performance of the DCT operation, most of the energy is found to be concentrated in the low frequency region. Thus, it is possible to make the compression for the image signal by neglecting the high frequency components of the DCT. At the receiving end, the image can be reconstructed by an inverse DCT operation.

Because of the importance of the two dimensional (2-D) DCT/IDCT in digital image processing, especially in video compression, different algorithms and architectures have been presented for the calculation and implementation. Usually, there are two categories for the 2-D DCT computation. The first category uses the techniques of matrix analysis and decomposition [3] [4]. The second category employs the method of polynomials and number theory [5] [6]. In these two categories, there are two groups to compute the 2-D

DCT. In the first group of algorithms, the row-column decomposition uses 1-D DCT computation to compute the 2-D DCT computation [3] [6]. In the second group, the 2-D DCT computation is carried out directly on the 2-D data [4] [5].

In this chapter, the background of the discrete cosine transform is presented. In the first section, the definitions of the 1-D DCT/IDCT and the 2-D DCT/IDCT and the row-column decomposition are introduced. Then in the second section the recursive algorithms of the 2-D DCT computation is given. Especially, the recursive algorithm for 1-D DCT and the development of the recursive algorithm for 2-D DCT are introduced in this section.

2.1 Discrete Cosine Transform

The computation of the discrete cosine transform is one of the processes of transforming a block of data from the spatial domain to the frequency domain [7]. The inverse process of restoring the spatial domain to the frequency domain is carried out through the inverse discrete cosine transform.

2.1.1 One dimension discrete cosine transform (1-D DCT)

The one-dimensional (1-D) DCT of a sequence of input data with N elements is defined as

$$X(k) = \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{N-1} u(k) \cos\left(\frac{2n+1}{2N} k\pi\right) x(n) \tag{2.1}$$

where $k = 0, 1, \ldots, N\text{-}1$, and $u(0) = (1/2)^{1/2}$, and $u(k) = 1$ for $k \neq 0$.

The 1-D inverse DCT of a sequence of N points can be expressed as

$$x(n) = \sqrt{\frac{2}{N}} \cdot \sum_{k=0}^{N-1} u(n) \cos\left(\frac{2k+1}{2N} n\pi\right) X(k) \tag{2.2}$$

where $n = 0, 1, \ldots, N\text{-}1$.

As described before, 1-D DCT/IDCT is widely used in digital signal processing, especially for the imaging and video processing. In many cases, 1-D DCT/IDCT computation is the approach for computing the 2-D DCT/IDCT. As shown in (2.1) and (2.2), both 1-D DCT and 1-D IDCT require N^2 multiplications and $N(N\text{-}1)$ additions for the straightforward computation. Because there is a large number of a data computation, various approaches for the 1-D DCT/IDCT computation have been developed.

2.1.2 Two dimension discrete cosine transform (2-D DCT)

For a set of 2-D data $x(n_1, n_2)$ with $0 \leq n_1 \leq N - 1$ and $0 \leq n_2 \leq N - 1$, the 2-D DCT is defined as

$$X(k_1, k_2) = \frac{2}{N} u(k_1) u(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) \times \cos\frac{2\pi(2n_1 + 1)k_1}{4N} \cos\frac{2\pi(2n_2 + 1)k_2}{4N} \tag{2.3}$$

where $k_1, k_2 = 0, 1, \ldots, N\text{-}1$, $u(k) = 2^{(-1/2)}$ for $k = 0$, and $u(k) = 1$ for $k \neq 0$.

Then, 2-D inverse DCT can be expressed as

$$x(n_1, n_2) = \frac{2}{N} u(n_1) u(n_2) \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} X(k_1, k_2) \times \cos \frac{2\pi (2k_1 + 1) n_1}{4N} \cos \frac{2\pi (2k_2 + 1) n_2}{4N} \tag{2.4}$$

where $n_1, n_2 = 0, 1, 2, \ldots, N\text{-}1$.

As mentioned before, the 2-D DCT/IDCT computation is very important in the signal and video compression. The 2-D DCT/IDCT computation needs a large number of data computations and it requires N4 multiplications. Besides, the two dimension input data is also a difficulty for the computation. The development of efficient algorithms for computing the 2-D DCT is of great interest.

2.1.3 Row-column decomposition of the 2-D DCT

Because of the large number of 2-D data should be calculated in the 2-D DCT computation, the decomposition is very necessary for the computation and implementation of the 2-D DCT. A straightforward implementation of (2.3) requires N^4 multiplications for the computation of the 2-D DCT. There are many different methods to compute the two dimension DCT. These method can be divided into two categories, fast algorithms which compute the 2-D DCT by means of the given 2-D data, and the algorithms which use row-column decomposition approach in which the 2-D DCT is implemented by using two one-dimensional DCT and a matrix transpose operation.

The row-column decomposition, which has the advantage of regularity for VLSI implementation, is the most popular and effective approach of 2-D DCT computation in

image and video coding applications. The 2-D DCT is separated into a matrix form of two 1-D DCTs (the row-column decomposition) as follows.

$$X = A \cdot x \cdot A^T \tag{2.5}$$

where $x = [x(i,j), i,j = 0, 1, \ldots, N\text{-}1]$, $X = [X(i,j), i,j = 0, 1, \ldots, N\text{-}1]$ and A are $N \times N$ arrays, representing the spatial input data, the frequency domain output data, and the matrix with the cosine basis functions respectively. By using the virtue of the orthogonality of A, $A \cdot A^T = I_N$, where I is an $N \times N$ identity matrix. For example, for $N = 8$, we have

$$A = \begin{bmatrix} \alpha & \alpha & \alpha & \alpha & \alpha & \alpha & \alpha & \alpha \\ \beta & \gamma & \Delta & \varepsilon & -\varepsilon & -\Delta & -\gamma & -\beta \\ \Gamma & \delta & -\delta & -\Gamma & -\Gamma & -\delta & \delta & \Gamma \\ \gamma & -\varepsilon & -\beta & -\Delta & \Delta & \beta & \varepsilon & -\gamma \\ \alpha & -\alpha & \alpha & \alpha & \alpha & -\alpha & -\alpha & \alpha \\ \Delta & -\beta & \varepsilon & \gamma & -\gamma & -\varepsilon & \beta & -\Delta \\ \delta & -\Gamma & \Gamma & -\delta & -\delta & \Gamma & -\Gamma & \delta \\ \varepsilon & -\Delta & \gamma & -\beta & \beta & -\gamma & \Delta & -\varepsilon \end{bmatrix} \tag{2.6}$$

where

$$\begin{bmatrix} \alpha \\ \beta \\ \Gamma \\ \gamma \\ \Delta \\ \delta \\ \varepsilon \end{bmatrix} = \sqrt{\frac{2}{N}} \begin{bmatrix} \cos\frac{\pi}{4} \\ \cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} \\ \cos\frac{5\pi}{16} \\ \cos\frac{3\pi}{8} \\ \cos\frac{7\pi}{16} \end{bmatrix} \tag{2.7}$$

Using the row-column decomposition, X can be computed using two 1-D DCT transforms given by

$$Y = Ax^T \tag{2.8a}$$

$$X = AY^T \tag{2.8b}$$

where Y is an intermediate product matrix. The above decomposition to a matrix product results in a reduction in computational complexity to $2N_3$ multiplications and has been used in the implementations of most the image compression algorithm.

The standard block diagram of computing the 2-D DCT using row-column decomposition is shown in Fig. 2.1.

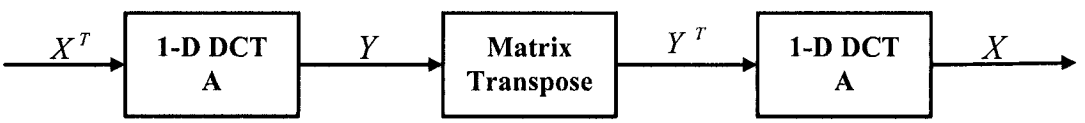

Fig. 2.1 2-D DCT computation using row-column decomposition.

By using this row-column decomposition, the 2-D DCT can be computed by a simple and regular structure. Because the 2-D DCT is decomposed into two 1-D DCTs, many 1-D DCT computation algorithms can be used for this computation, such as the recursive algorithm.

2.2 Recursive Algorithm for the DCT Computation

A recursive algorithm is a mathematic method to compute the output of a discrete

system by means of the returned value and the smaller index inputs. If a system is defined recursively, a recursive algorithm to compute its members or values reflects the definition. In this way, the computation for the system can be much simpler.

Considering the general form of an input-output relationship is

$$y(n) = -\sum_{k=1}^{n} a(k)y(n-k) + \sum_{k=0}^{M} b(k)x(n-k) \qquad (2.9)$$

Let us consider an example of (2.9) as $y(n) = y(n - 1) - 0.25y(n - 2) + x(n)$, where the input sequence $x(n) = \delta(n)$, and the two initial conditions are $y(-1) = 1.0$, and $y(-2) = 0.4$.

Then y(n) can be computed in a recursive manner as follows: $y(0) = y(-1) - 0.25y(-2) + x(0)$. Since x(n) $=\delta(n)$, $x(0) = 1$ can be substituted and get $y(0) = 1.0 - 0.25(0.4) + 1 = 1.9$. Next, $y(1) = y(0) - 0.25y(-1) + x(1)$. It is known that $y(0) = 1.9$ from the step shown above, and also that $x(1) = 0$. So it can be obtained that $y(1) = 1.9 - 0.25(1.0) +0 = 1.65$. Next, for $n = 2$, by substituting the known values from above, we can get $y(2) = 1.65 - 0.25(1.9) + 0 = 1.175$. We can continue to calculate the values of he output $y(n)$ for n = 3, 4, 5, 6, ...

The calculation method mentioned above is known as the recursive algorithm. The system with the function of (2.9) is defined recursively. The output of the system can be calculated by the returned values and the smaller index inputs. In this way, the computation procedure can be simple and for many cases it can provide a natural way to carry out the computation system.

2.2.1 Recursive algorithm for the 1-D DCT

Because of the importance of the DCT computation in the digital processing, particularly in video compression, there are different methods to compute the 1-D DCT. In many of them, 1-D DCT computation can be obtained by direct factorization of the DCT coefficient matrices. Normally, the factorization schemes fall into decimation-in-time (DIT) or the decimation-in-frequency (DIF) [7]. When the components of this factorization are sparse, the decomposition represents a fast algorithm. However, the complicated index mapping of global interconnection from the input and to the output data makes the hardware implementation difficult.

In addition to the algorithms mentioned above, there exist many other approaches to compute the 1-D DCT. Using the recursive algorithms is one of them to reach good goal of computation. Goertzel initially employs the periodicity of the finite trigonometric sequence to reduce the computation of the discrete Fourier transform [8]. His recursive structure not only simplifies the computation but also reduces the realization complexity. A great amount of 1-D DCT algorithms featuring highly regular calculations can be implemented in a small number of circuit blocks for it to be performed in a recursive manner [9]-[13], which may help to achieve a low hardware cost. Usually, these recursive algorithms can be grouped into two types: Chebyshev Polynomial recurrence and Clenshaw's recurrence formula.

One recursive algorithm for the DCT computation on the Chebyshev polynomial

factorization is presented in [12]. The following trigonometric identity can be used for the Chebyshev polynomial.

$$\cos(\gamma\alpha) = 2\cos\alpha\cos\left[(\gamma-1)\alpha\right] - \cos\left[(\gamma-2)\alpha\right] \tag{2.10}$$

If the scale factors of $(2/N)^{1/2}u(k)$ is not considered during the computation, the equation (2.1) can be reformulated as

$$X(k) = \cdot\sum_{n=0}^{N-1}\cos\left(\frac{2n+1}{2N}k\pi\right)x(n) \tag{2.11}$$

Assuming $P(k,n) = \cos\frac{(2n+1)k\pi}{2N} = \cos\frac{(2n+1)\alpha}{2}$, $\alpha = k\pi/N$ and

$$A(k,n) = \sum_{n=0}^{N-1}x(n)\cos\frac{(2n-1)k\pi}{2N} = \sum_{n=0}^{N-1}x(n)P(k,n) \tag{2.12}$$

the 1-D DCT can be computed by using the following Chebyshev polynomial recurrence:

$P(k, -1) = P(k, 0) = cos(\alpha/2)$ and

$$P(k, n+1) = 2cos(\alpha)P(k, n) - P(k, n-1) \tag{2.13}$$

$A(k, -1) = 0$ and

$$A(k, n) = A(k, n-1) + x(n)P(k, n) \tag{2.14}$$

where $X(k) = A(k, N-1)$, $k = 0, 1, \ldots, N-1$. In this way, $X(k)$ can be calculated in N recursive steps from the input sequence $x(n)$ by using (2.13) and (2.14). For an N-point input of $x(n)$, this recursive algorithm requires $2N(N-1)$ multiplications and additions.

Besides, the Clenshaw's recurrence is another type of the recursive algorithm to compute the 1-D DCT [13]. It needs a linear combination of the form

$$f(x) = \sum_{n=0}^{N} c(n)F(x,n) \tag{2.15}$$

in which $F(x, n)$ obeys a recurrence relation

$$F(x,n+1) = \alpha(x,n)F(x,n) + \beta(x,n)F(x,n-1) \tag{2.16}$$

By using the functions $\alpha(x, n)$ and $\beta(x, n)$, the sum $f(x)$ can be computed as

$$f(x) = c(N)F(x,N) - \beta(x,n)F(x,N-1)\psi(N-1) - F(x,N)\psi(N-2) \tag{2.17}$$

where $\psi(n)$ can be obtained by the following relationship:

$\psi(-2) = \psi(-1) = 0$ and

$$\psi(n) = \left[\psi(n-2) - \alpha(x,n)\psi(n-1) - c(n)\right] / \beta(x,n+1),\ n = 1, 2, \ldots, N\text{-}1 \tag{2.18}$$

If we define

$$\lambda_k = k\pi / N \quad \text{and}$$

$$F(\lambda_k, n) = \cos\left[(n+1/2)(k\pi / N\right] = \cos\left[(n+1/2)\lambda_k\right] \tag{2.19}$$

the 1-D DCT can be expressed as

$$f(\lambda_k) = X(k) = \sum_{n=0}^{N-1} x(n)F(\lambda_k, n), \qquad k = 1, 2, \ldots, N\text{-}1 \tag{2.20}$$

Because of the identity

$$\cos\left[(n+3/2)\lambda_k\right)] = 2\cos(\lambda_k)\cos\left[(n+1/2)\lambda_k\right)] - \cos\left[(n-1/2)\lambda_k\right)] \tag{2.21}$$

the calculation of $F(\lambda_k, n)$ can be made recursively as following:

$$F(\lambda_k, n+1) = 2\cos(\lambda_k)F(\lambda_k, n) - F(\lambda_k, n-1) \tag{2.22}$$

Comparing the equations (2.22) and (2.16), we can obtain that $\alpha(x, n) = 2cos(\lambda_k)$ and $\beta(x, n) = -1$. Substitute (2.17) into (2.20), we can find

$$
\begin{aligned}
X(k) &= x(N-1)F(\lambda_k, n-1) - F(\lambda_k, n-2)\psi(N-2) - F(\lambda_k, N-1)\psi(N-3) \\
&= (-1)^k \left[x(n-1)\cos(\lambda_k/2) + \cos(3\lambda_k/2)\psi(N-2) - \cos(\lambda_k/2)\psi(N-3)\right] \\
&= (-1)^k \cos(\lambda_k/2)\left[\psi(N-1) - \psi(N-2)\right]
\end{aligned} \tag{2.23}
$$

where $\psi(n)$ can be obtained from (2.18) as

$\psi(-2) = \psi(-1) = 0$ and

$$\psi(n) = 2\cos(\lambda_k)\psi(n-1) - \psi(n-2) + x(n),\ n = 1, 2, \ldots, N\text{-}1 \tag{2.24}$$

In this way, $\psi(n)$ can be generated from the input $x(n)$ recursively. At the N^{th} step, $X(k)$ can be generated by the equation (2.22) for k = 1, 2, ..., N-1. For an N-point sequence $x(n)$, this recursive algorithm requires about N^2 multiplications and additions.

The recursive algorithm makes the 1-D DCT computation easy to be implemented with relatively simple processing elements and simple interconnections among the processing elements. Identical or similar processing elements in a hardware implementation can greatly reduce the cost of the design and layout process and thus the recursive algorithm for the 1-D computation is well suited for VLSI implementation.

2.2.2 Recursive Algorithm and implementation for the 2-D DCT

It is very difficult to compute the 2-D DCT due to the large number of the two dimension data in the computation process. For an N×N point input sequence, the 2-D DCT requires $O(N^4)$ multiplications and corresponding additions. Because of this, the decomposition for the 2-D data is very necessary and various fast computational algorithms and corresponding architectures have been proposed so as to improve the efficiency of the 2-D DCT computation. As mentioned before, the algorithms of the 2-D

DCT computation can be classified into two groups: one is called row-column decomposition [14]-[16] which is described in Section 2.1.3, another is carried out directly on the 2-D data [17] - [20].

The same as the two groups of the methods mentioned before, the recursive algorithm can also be applied for the 2-D DCT computation in two categories. In the first category, through the row-column decomposition, the 2-D DCT computation can be decomposed into two 1-D DCTs by means of row-column-wise or column-row-wise form. That is, it begins by processing the row (or column) elements of the input data block as 1-D DCT and stores the results in an intermediate memory; it then computes the transposed column (or row) elements of the intermediate results to further yield the 2-D DCT results. Because the row-column decomposition can reduce the 2-D DCT computation into two separate 1-D DCTs, the recursive algorithms for the 1-D DCT mentioned in Section 2.2.1 can be applied to these 1-D DCTs [9]-[11]. In this way, the 2-D DCT computation can be calculated by the combination of row-column decomposition and recursive algorithm.

The block diagram of the recursive algorithm for the row-column approach for the 2-D DCT is shown in Fig. 2.2. The row-column decomposition can be implemented in a simple and regular structure.

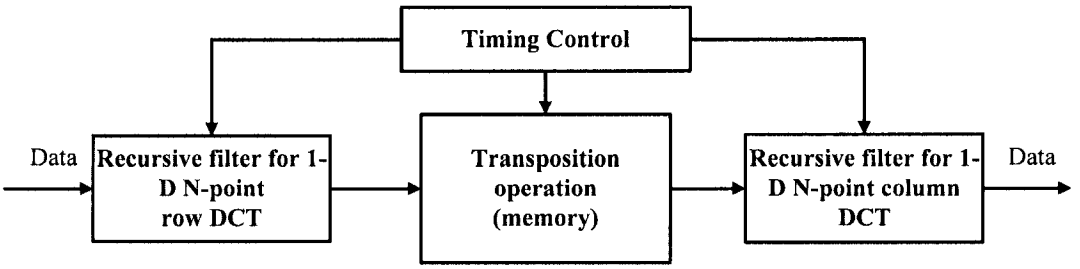

Fig. 2.2 Block diagram of the recursive algorithm for the row-column approach for the 2-D DCT.

However, in the applications of the row-column decomposition for the 2-D DCT, because the temporal results of the 1-D DCTs are required to stored during the computation, large number of transposition memory is required to store those temporal results. Meanwhile, a great number of recursive cycles are required to complete the 2-D transformation by using 1-D recursive structures. To make a structure of low cost and small circuit volume and smaller number of recursive number, many approaches are proposed, which uses direct 2-D data and recursive algorithm for the 2-D DCT. In these cases, the 2-D DCT computation is reformulated, by means of, e.g., conversions of variables, into terms of 1-D DCTs and/or DSTs. It is important that the reformulated computation keeps its regularity and modularity to facilitate the implementation.

In this category, through the division of the index variables, the 2-D DCT can be grouped into several cases. Then in each case, the 2-D variables can be combined and transformed into some 1-D variables. In this way, the 2-D DCT in each case can be converted into several 1-D DCTs/DSTs. Then, these 1-D DCTs/DSTs can be calculated by the recursive algorithm. The block diagram of the recursive algorithm for the direct

2-D data for 2-D DCT is shown as in Fig. 2.3.

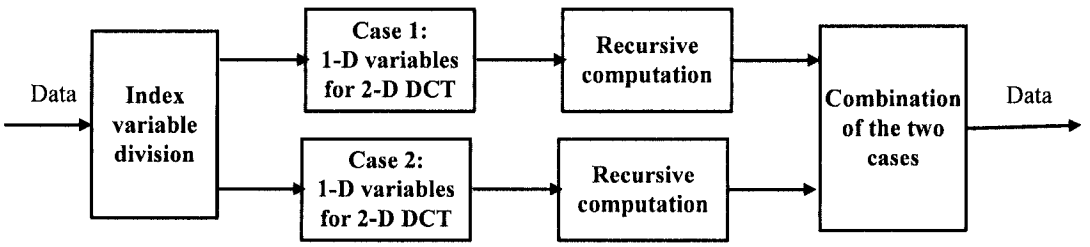

Fig. 2.3 Block diagram of the recursive algorithm for the 2-D data for 2-D DCT.

There are many different methods to implement the 2-D DCT computation. In the two categories of the approaches for the 2-D DCT mentioned above, most of them are implemented through directly mapping the computation algorithms. Two examples will be introduced in these two categories as follows.

The architecture reported in [12] presented a 2-D systolic array of N×N basic cells for computing the 2-D DCT, based on the use of the Chebyshev polynomial to generate the transform kernel values recursively as mentioned in Section 2.2.2. The 1-D DCT array is constructed by using the Chebyshev polynomial. The 2-D DCT array is based on the row-column decomposition. The architecture is shown in Fig. 2.4. It can give a simple and regular communication, and control structures. Thus it is well suited to VLSI implementation. However the architecture in [12] makes an increase of time complexity and chip area.

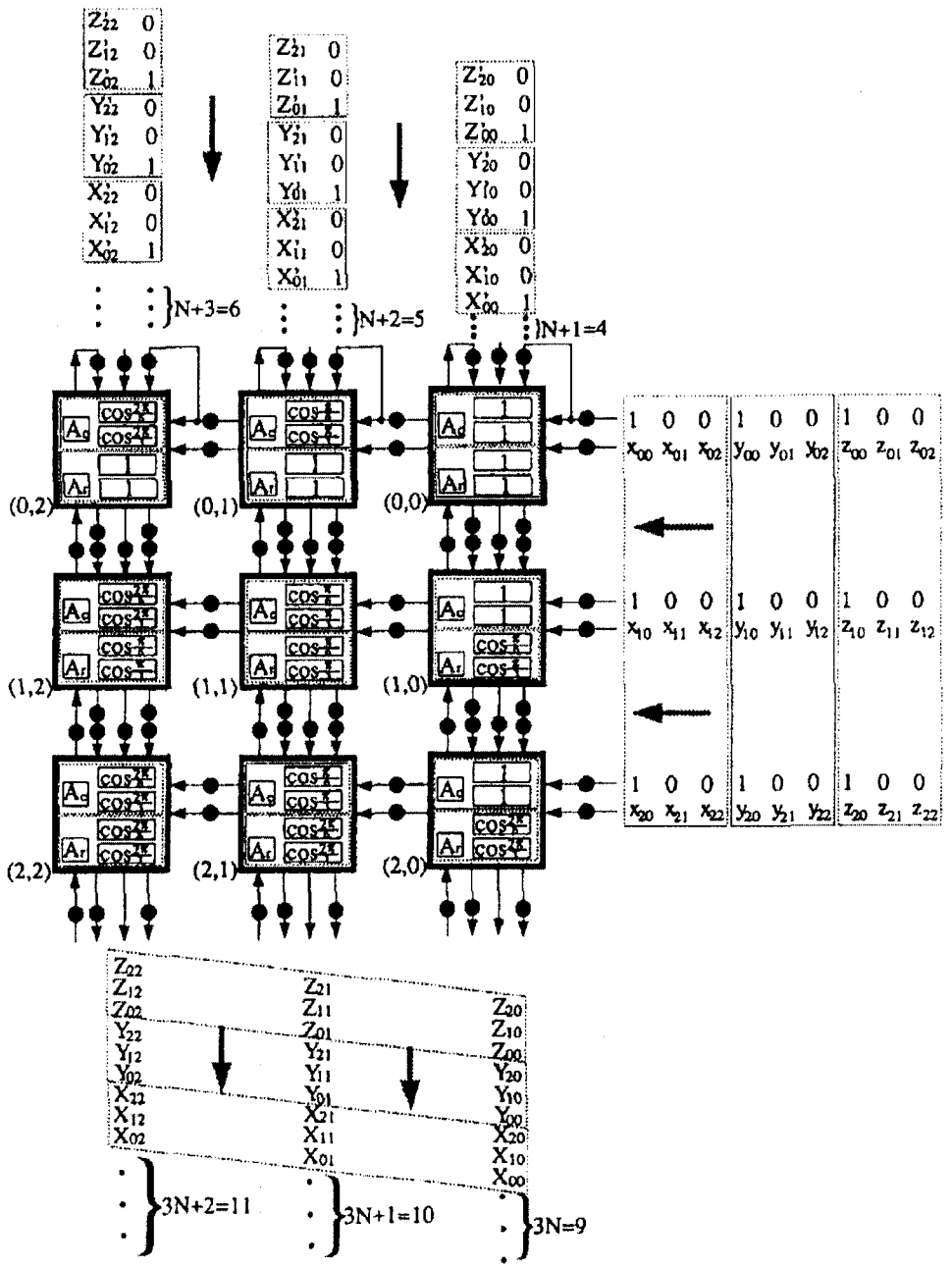

Fig. 2.4 Implementation architecture of the 2-D systolic array for the 2-D DCT in [12].

In the second category of the 2-D DCT computation, the latest computation structure using the recursive algorithm is presented in [21]. In the recursive algorithm of [21], by means of the prime-and-no-prime nature of the index variables, the 2-D DCT can be

grouped into two cases. In these two cases, the terms with 2-D index variables can be reformulated and simplified into terms with 1-D variables. In another words, it uses the theory of pre-adding the data with the same transform base and characterizing the periodicity of transform bases by means of the index separation method in [21]. In this way, the 2-D DCT computation can be complete by terms of 1-D DCTs and 1-D DSTs, and these 1-D DCTs/DSTs can be calculated by the recursive kernel. The implementation structure is presented in Fig. 2.5.

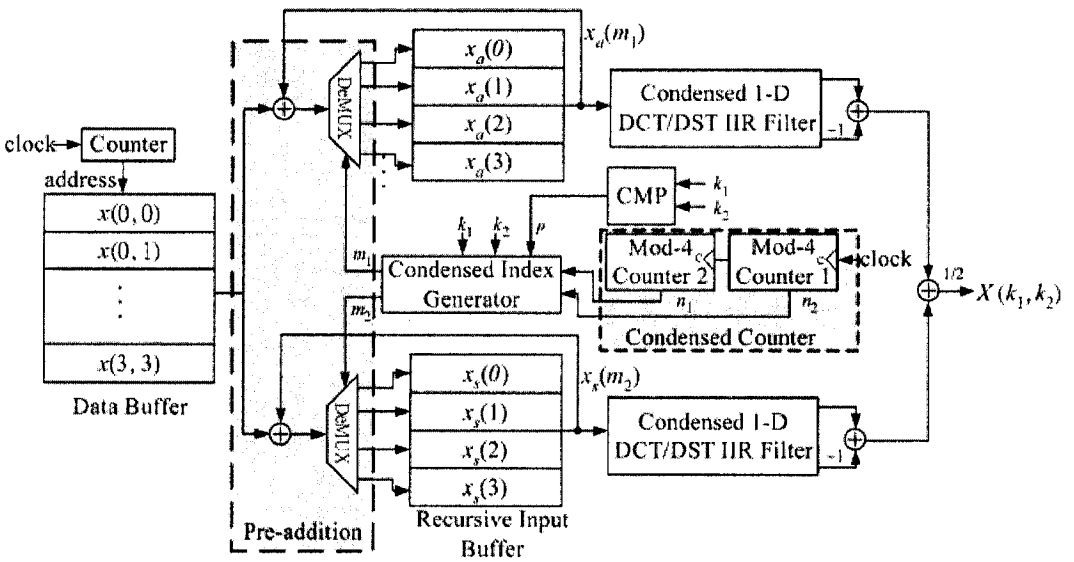

Fig. 2.5 Implementation structure of the recursive 4×4 2-D DCT in [21].

As shown in Fig. 2.5, because of the pre-addition procedure of the 2-D input signals, the number of the recursive cycle can be reduced greatly and the circuit structure is very simple and modular. Meanwhile, because there is no the temporal results required to stored, no large number of transposition memories are needed in the architecture of [21].

Meanwhile, because of the pre-addition procedure, the number of recursive cycle can be reduced greatly.

2.3 Summary

In this chapter, some background material on the mathematical description of the DCT algorithms, especially the recursive algorithm, has been proposed. In Section 2.1, the definition of the discrete cosine transform and the representation of its inverse have been presented. The row-column decomposition approach for the 2-D DCT computation has been given. Section 2.2 describes the recursive algorithm for the DCT computation. In this section, the definition of the recursive algorithm has been presented firstly. Then the recursive algorithm for the 1-D DCT computation has been described and two types of the recursive algorithms, which are Chebyshev Polynomial recurrence and Clenshaw's recurrence formula, have been given. Finally this section also presents the approaches of computing the 2-D DCT by using the recursive algorithm. The two types of the approaches of the recursive algorithm have been briefly described.

As mentioned in Section 2.2.3, the recursive algorithms can be used for computing the 2-D DCT by means of the row-column decomposition or direct 2-D data decomposition. The row-column decomposition requires a large number of transposition memories to store the temporal results and the number of recursive cycle is great in this row-column approach. In the direct 2-D data category, the recursive algorithm has been proposed in [21] which groups the 2-D data into two cases by mean of

prime-and-no-prime nature of the index variables. By means of the pre-addition procedure of the data with the same transform base, the 2-D DCT can be converted into two 1-D DCTs and two 1-D DSTs. In this way, the number of recursive kernel can be reduced. However, all these implementations are direct mapping of the existing algorithms of the 2-D DCT. In the next chapter, these problems will be solved and an implementation which is not direct mapping will be proposed as a part of the investigation undertaken in this thesis.

Chapter 3

Proposed Recursive Algorithm for the 2-D DCT Computation

In the preceding chapter, the development of the algorithms for the 2-D DCT computation relevant to the thesis work has been summarized. Among the approaches used in these algorithms, the recursive one can result in a small number of circuit blocks in the implementation of the 2-D DCT calculation to achieve low hardware consumption. Compared to a parallel processing, a recursive one may require more time, because the computation task is completed by means of cycle-by-cycle calculations. However, the penalty of such calculation cycles on the processing speed may not be sever if i) the number of calculation cycles is minimized, and ii) an optimized recursive kernel is designed to minimize the time required for each cycle. A recursive algorithm can be based on a row-column decomposition [22]-[25]. In such an algorithm, the 2-D DCT computation is decomposed strait-forwardly to those of rows and columns. It can be performed in a simple repetitive procedure. However, some recursive algorithms are designed with a decomposition method that is less strait-forward than the row-column ones, aiming at reducing the number of recursive cycles to meet the requirement of fast

processing. The number of recursive cycles in the algorithm reported in [21] may be the smallest among those in literature.

In this chapter, a recursive algorithm for the 2-D DCT computation is proposed based on the algorithm in [21]. The proposed algorithm has a simpler computation procedure, leading to a reduced number of multiplication and addition operations. Thus, the complexity of the recursive kernel is expected to be simpler for an easy implementation.

In Section 3.1, the background of the proposed algorithm is presented. Section 3.2, the recursive computation of the algorithm is described in details and the new recursive kernel is presented. Section 3.3 summarizes the work of the new algorithm.

3.1 Background of the Proposed Algorithm

Our algorithm is developed based on the condensed 1-D transform method reported in [21]. With this method, the 2-D DCT computation is decomposed into 1-D DCTs and 1-D DSTs. The purpose of the decomposition, in general, is to separate a term of calculations with 2-D index variables into two or more terms, each of which has only an 1-D index variable, and thus the 2-D computation can be performed as 1-D ones. Normally, a non-row-column decomposition method is to group the terms with 2-D index variables according to some characters other than the natural row or column decompositions. In [21], they are regrouped into two cases, one for the index variables being prime numbers and the other for non-prime numbers. Because of the periodical

nature of the trigonometric calculations, the computation with a non-prime index variable can be easily reformulated and simplified for the purpose of separating one term involving two index variables into two or more 1-D index variable terms. Moreover, the reformulated terms can be calculated, in terms of 1-D DCT or DST, in the main recursive procedure and in the blocks of so-called pre-additions [21]. In this way, the 2-D DCT can be converted to 1-D DCTs and DSTs calculated with the results of the pre-additions. The computation procedure of the condensed 1-D transform method is described as follows.

As mentioned in the section 2.1.2, for a set of 2-D data $x(n_1, n_2)$ with $0 \le n_1 \le N - 1$ and $0 \le n_2 \le N - 1$, the 2-D DCT is defined as

$$X(k_1, k_2) = \frac{2}{N} u(k_1) u(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) \times \cos \frac{2\pi (2n_1 + 1) k_1}{4N} \cos \frac{2\pi (2n_2 + 1) k_2}{4N} \tag{3.1}$$

where $k_1, k_2 = 0, 1, \ldots, N-1$, $u(k) = 2^{(-1/2)}$ for $k = 0$, and $u(k) = 1$ for $k \neq 0$. By using trigonometric identities, (3.1) can be developed and expressed as

$$X(k_1, k_2) = \frac{1}{2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2) \times \left\{ \begin{array}{l} \cos \dfrac{2\pi \left[(2n_1 + 1) k_1 + (2n_2 + 1) k_2 \right]}{4N} \\ + \cos \dfrac{2\pi \left[(2n_1 + 1) k_1 - (2n_2 + 1) k_2 \right]}{4N} \end{array} \right\}$$

$$X(k_1,k_2) = \frac{1}{2}\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} x(n_1,n_2) \times \left\{ \begin{array}{l} \cos\dfrac{\pi(n_1k_1+n_2k_2)}{N}\cos\dfrac{\pi(k_1+k_2)}{2N} \\ -\sin\dfrac{\pi(n_1k_1+n_2k_2)}{N}\sin\dfrac{\pi(k_1+k_2)}{2N} \\ +\cos\dfrac{\pi(n_1k_1-n_2k_2)}{N}\cos\dfrac{\pi(k_1-k_2)}{2N} \\ -\sin\dfrac{\pi(n_1k_1-n_2k_2)}{N}\sin\dfrac{\pi(k_1-k_2)}{2N} \end{array} \right\} \tag{3.2}$$

$$X(k_1,k_2) = \frac{1}{2}\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} x(n_1,n_2) \times \left\{ \begin{array}{l} (-1)^{s_1}\cos\dfrac{\pi\left[(n_1k_1+n_2k_2)\bmod N\right]}{N} \\ \times\cos\dfrac{\pi(k_1+k_2)}{2N} - (-1)^{s_1} \\ \times\sin\dfrac{\pi\left[(n_1k_1+n_2k_2)\bmod N\right]}{N} \\ \times\sin\dfrac{\pi(k_1+k_2)}{2N} + (-1)^{s_2} \\ \times\cos\dfrac{\pi\left[(n_1k_1-n_2k_2+2N^2)\bmod N\right]}{N} \\ \times\cos\dfrac{\pi(k_1-k_2)}{2N} - (-1)^{s_2} \\ \times\sin\dfrac{\pi\left[(n_1k_1-n_2k_2+2N^2)\bmod N\right]}{N} \\ \times\sin\dfrac{\pi(k_1-k_2)}{2N} \end{array} \right\} \tag{3.3}$$

where $s_1 = [(n_1k_1 + n_2k_2)/N]$ and $s_2 = [(n_1k_1 - n_2k_2 + 2N^2)/N]$. To compute $X(k_1, k_2)$, the 2-D DCT is divided into two cases according to the relationship of k_1, k_2 and N, and k_1 and k_2 are expressed as $k_1 = d_1r^{p1}$ and $k_2 = d_2r^{p2}$.

Case 1: k_1 or k_2 is prime to N. Let $p = min\ \{p_1, p_2\}$, $\omega_1 = d_1$, $\omega_2 = d_2$ and $M = N$.

Case 2: $k_1 \neq 0$ and $k_2 \neq 0$, k_1, k_2 and N have a common divisor r^p where $p = min\ \{p_1, p_2\}$, $\omega_1 = d_1r^{-p1}$, $\omega_2 = d_2r^{-p2}$ and $M = Nr^{-p}$.

In these two cases, two variables m_1 and m_2 are introduced and they are defined as $m_1 = \omega_1n_1 + \omega_2n_2\ mod\ M$, and $m_2 = \omega_1n_1 - \omega_2n_2 + 2M^2\ mod\ M$. In each case, because of the separation of the variable index, the data with the same transform based can be computed by a pre-addition process. The pre-addition operation with $x(n_1, n_2)$ is then performed to generate $x_a(m_1)$ *and* $x_s(m_2)$ defined as follows.

In Case 1:

$$x_a(m_1) = \sum_{n=0}^{M-1} (-1)^{s_1} x(n_1, n_2) \tag{3.4}$$

$$x_s(m_2) = \sum_{n=0}^{M-1} (-1)^{s_2} x(n_1, n_2) \tag{3.5}$$

where $s_1 = [(\omega_1n_1 + \omega_2n_2)/M]$, $s_2 = [(\omega_1n_1 - \omega_2n_2 + 2M^2)/M]$, $n = n_2$ for $p_1 = 0$ and $n = n_1$ for $p_2 = 0$.

In Case 2:

$$x_a(m_1)=\sum_{n=0}^{M-1}\sum_{i_1=0}^{r^p-1}\sum_{i_2=0}^{r^p-1}(-1)^{s_a}\times x(n_1+i_1M,n_2+i_2M) \tag{3.6}$$

$$x_s(m_2)=\sum_{n=0}^{M-1}\sum_{i_1=0}^{r^p-1}\sum_{i_2=0}^{r^p-1}(-1)^{s_s}\times x(n_1+i_1M,n_2+i_2M) \tag{3.7}$$

where $s_a = [(\omega_1 r^p n_1 + \omega_2 r^p n_2)/M] + i_1\omega_1 + i_2\omega_2$, $s_s = [(\omega_1 r^p n_1 - \omega_2 r^p n_2 + 2M^2)/M] + i_1\omega_1 + i_2\omega_2$, $n = n_2$ for $p_1 = p$, and $n = n_1$ for $p_2 = p$.

With (3.4) ~ (3.7), the computation of the 2-D DCT, without including the normalization factor $2u(k_1)u(k_2)/N$, can be written as the summation of four condensed 1-D DCTs and 1-D DSTs multiplied by multiplication factors $cos(\omega_1 \pm \omega_2)/2M$ and $sin(\omega_1 \pm \omega_2)/2M$, which is shown as follows.

$$X(\omega_1 r^p,\omega_2 r^p)=\frac{1}{2}\left\{\begin{array}{l}\displaystyle\sum_{m_1=0}^{M-1}x_a(m_1)\begin{bmatrix}\cos(m_1\pi/M)\cos((\omega_1+\omega_2)\pi/2M)\\-\sin(m_1\pi/M)\sin((\omega_1+\omega_2)\pi/2M)\end{bmatrix}\\+\displaystyle\sum_{m_2=0}^{M-1}x_s(m_2)\begin{bmatrix}\cos(m_2\pi/M)\cos((\omega_1-\omega_2)\pi/2M)\\-\sin(m_2\pi/M)\sin((\omega_1-\omega_2)\pi/2M)\end{bmatrix}\end{array}\right\} \tag{3.8}$$

In the approach described above, the computation of the 2-D DCT can be reformulated into terms comprising 1-D DCTs and 1-D DSTs. The recursive kernel for the computation is shown in Fig. 3.1. It can also be presented using a more compact format as illustrated in Fig. 3.2. It shows that each computation cycle requires six multiplications with the DST and DCT coefficients. The diagram of the structure of the recursive algorithm in [21] is shown in Fig. 3.3. The 2-D DCT is carried out in parallel by

two channels, each consisting of a pre-addition block and a 1-D recursive kernel shown in Fig. 3.1. It would be desirable to further simplify the computation so as to reduce the number of multiplications in the recursive kernel while removing the DST calculation.

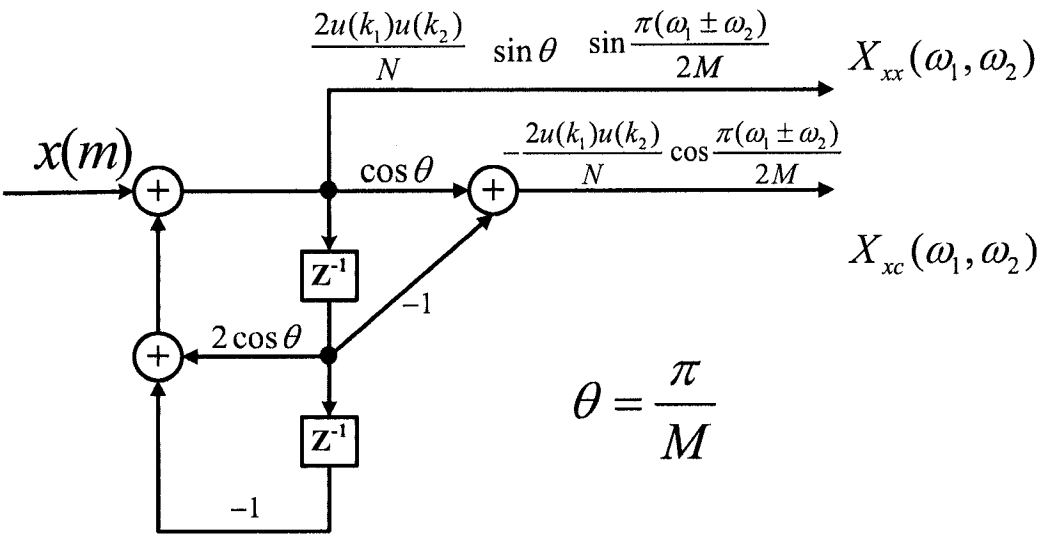

Fig.3.1. Recursive kernel for 1-D DCT/DST [21].

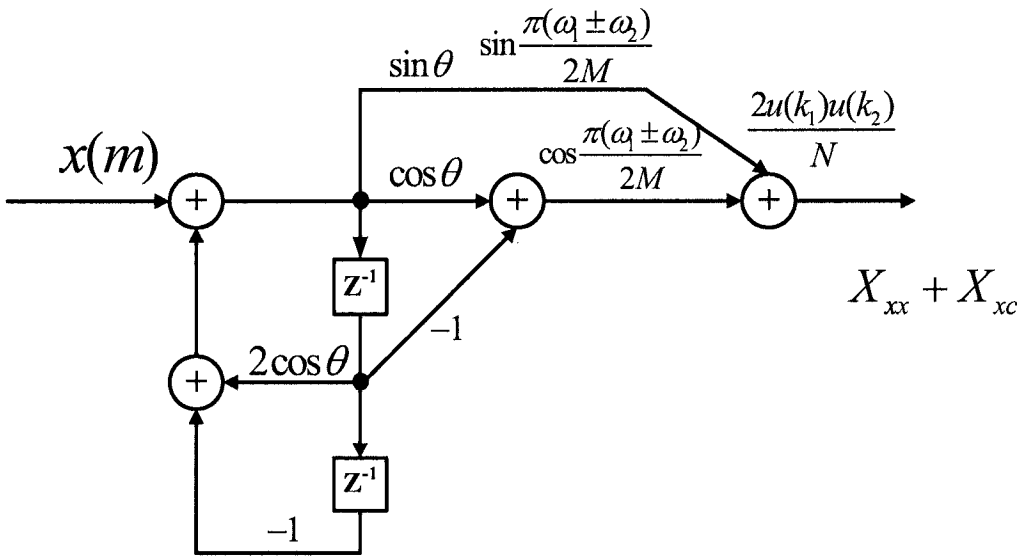

Fig. 3.2. Recursive kernel for the computation of the 2-D DCT defined in [21].

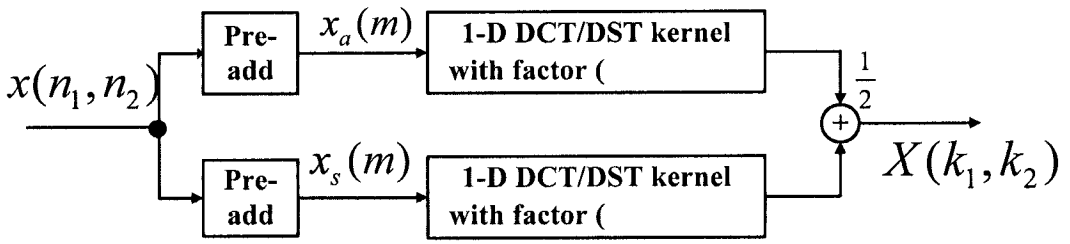

Fig. 3.3. Structure of the 2-D DCT computation according to the algorithm of [21]. The detail of the 1-D recursive kernel is shown in Fig. 3.2.

3.2 Proposed Algorithm for the 2-D DCT Computation

As mentioned in the preceding section, the 2-D DCT computation can be reformulated to the pre-addition and 1-D transform. Through this reformulation, it can be calculated by means of 1-D DCTs and DSTs. In this way, the number of the recursive cycles will be reduced greatly. However, in the recursive kernel, both DCTs and DSTs are needed in the computation and six multiplications are required in each recursive cycle. In this section, further work on mathematic reformulation of the 2-D DCT computation is presented. The objective of the work is to remove the DSTs in each recursive calculation cycle and to simplify the computation procedure. A new recursive kernel is also proposed to improve the hardware structure and eventually the operation speed.

3.2.1 Computation procedure for each recursive cycle

Based on the method described in Section 3.1, our work aims at developing a computation algorithm for the 2-D DCT using 1-D DCT modules with a reduced number

of multiplications. The equation (3.8) for the 2-D DCT computation involves multiple *sine* and *cosine* terms. Totally six multiplications are required in each cycle. A further simplification of the computation is to reduce the number of multiplications, and to make the computation to contain only *cosine* (or *sine*) terms.

Observing the equation (3.8), it is easy to see that, using trigonometry identity $cos(u + v) = cosu\ cosv - sinu\ sinv$, the 2-D DCT computation shown in (3.8) can be expressed as [26]

$$X(\omega_1 r^p, \omega_2 r^p) = \frac{1}{2}\left\{\begin{array}{l}\sum_{m_1=0}^{M-1} x_a(m_1)\cos\left((m_1 + \frac{\omega_1+\omega_2}{2})\pi/M\right) \\ +\sum_{m_2=0}^{M-1} x_s(m_2)\cos\left((m_2 + \frac{\omega_1-\omega_2}{2})\pi/M\right)\end{array}\right\} \tag{3.9}$$

Comparing (3.9) and (3.8), one can see that, the equation containing four *sine* and four *cosine* variables is reformulated to one of only two *cosine* ones by the sum trigonometry identity. By this reformulation, the number of multiplications used in each cycle of the recursive computation in (3.9) is evidently smaller than that in (3.8). Thus, the recursive kernel of (3.9) can be made much simpler than that of (3.8). Moreover, as (3.9) involves only 1-D DCTs terms, the generation of 1-D DSTs is not needed, which permits a significant simplification of the pre-computation of the 2-D DCT.

Equation (3.9) can be written as the sum of X_{ac1} and X_{ac2} defined as

$$X_{ac_1}(\omega_1,\omega_2)=\sum_{m_1=0}^{M-1}x_a(m_1)\cos\left((m_1+\frac{\omega_1+\omega_2}{2})\pi/M\right) \tag{3.10}$$

$$X_{ac_2}(\omega_1,\omega_2)=\sum_{m_2=0}^{M-1}x_s(m_2)\cos\left((m_2+\frac{\omega_1-\omega_2}{2})\pi/M\right) \tag{3.11}$$

Observing (3.10) and (3.11), one can find that they differ form each other only in the angles $(\omega_1 + \omega_2)/2$ and $(\omega_1 - \omega_2)/2$. It is possible to use the same structure to implement the computation of X_{ac1} or X_{ac2}. Also, the computation procedure should be designed in such a way that this structure involves a minimum number of multipliers and requires shortest delay for the computation.

3.2.2 Recursive kernel and computation structure for the proposed algorithm

Assuming that $\omega = (\omega_1 \pm \omega_2)/2$, $m' = M-1 - m$, and $x_a(m)$ or $x_s(m)$ is generalized as $x(m)$, (3.10) or (3.11) can be expressed as

$$\begin{aligned} X_{ac}(\omega_1,\omega_2) &= \sum_{m=0}^{M-1}x(m)\cos\left(\frac{m+\omega}{M}\pi\right) \\ &= \sum_{m'=0}^{M-1}x(M-1-m')\cos\left(\pi-\frac{1+m'-\omega}{M}\pi\right) \\ &= -\sum_{m'=0}^{M-1}x(M-1-m')\cos\left(\frac{1+m'-\omega}{M}\pi\right) \end{aligned} \tag{3.12}$$

Making $Y(\omega_1, \omega_2) = - X_{ac}(\omega_1, \omega_2)$, $j = M-1$ and $\theta = \pi/M$, we have

$$Y(\omega_1,\omega_2) = -X_{ac}(\omega_1,\omega_2) = \sum_{m'=0}^{M-1} x(M-1-m')\cos\left(\frac{1+m'-\omega}{M}\pi\right)$$
$$= \sum_{m'=0}^{j} x(j-m')\cos(1+m'-\omega)\theta \tag{3.13}$$

As $cos(m' + 1)\theta = 2cos\theta cosm'\theta - cos(m' - 1)\theta$, (3.13) can be rewritten as

$$\begin{aligned}
Y(\omega_1,\omega_2) &= \sum_{m'=0}^{j} x(j-m')\cos(1+m'-\omega)\theta \\
&= \sum_{m'=0}^{j} x(j-m')\begin{bmatrix} 2\cos\theta\cos(m'-\omega)\theta \\ -\cos(m'-\omega-1)\theta \end{bmatrix} \\
&= \left\{\begin{bmatrix} 2\cos\theta\left[\cos(\omega\theta)x(j)+Y_{j-1}(\omega_1,\omega_2)\right] \\ -\left[\begin{matrix}\cos(\omega+1)\theta x(j)+ \\ \cos(\omega\theta)x(j-1)+Y_{j-2}(\omega_1,\omega_2)\end{matrix}\right] \end{bmatrix}\right\} \\
&= \left\{\begin{bmatrix} \left[2\cos\theta\cos\omega\theta-\cos(\omega+1)\theta\right]x(j) \\ -\cos\omega\theta x(j-1) \\ +2\cos\theta Y_{j-1}(\omega_1,\omega_2)-Y_{j-2}(\omega_1,\omega_2) \end{bmatrix}\right\} \\
&= \begin{bmatrix} \cos(\omega-1)\theta x(j)-\cos\omega\theta x(j-1) \\ +2\cos\theta Y_{j-1}(\omega_1,\omega_2)-Y_{j-2}(\omega_1,\omega_2) \end{bmatrix}
\end{aligned} \tag{3.14}$$

The transform function of the system for (3.14) is given as

$$\frac{Y\left((\omega_1,\omega_2),z\right)}{X(z)} = \frac{\cos(\omega-1)\theta-\cos\omega\theta Z^{-1}}{1-2\cos\theta Z^{-1}+Z^{-2}} \tag{3.15}$$

It should be noted that (3.15) is applicable for both $X_{ac1}(\omega_1, \omega_2)$ and $X_{ac2}(\omega_1, \omega_2)$ as $\omega = (\omega_1 \pm \omega_2)/2$.

Based on (3.15), we propose a recursive computation kernel as illustrated in Fig. 3.4. The structure shown in Fig. 3.5 includes the proposed kernel and is for the same 2-D DCT computation as that of Fig. 3.3. Comparing the kernel shown in Fig. 3.4 with that in Fig. 3.2, one can see that the former needs only *cosine* coefficients, and involves four multiplications and three additions, whereas the latter requires both *cosine* and *sine* coefficients and employs six multiplications and four additions. It can be expected that the proposed algorithm can be implemented in a simpler circuit structure with potentially shorter clock cycle-time.

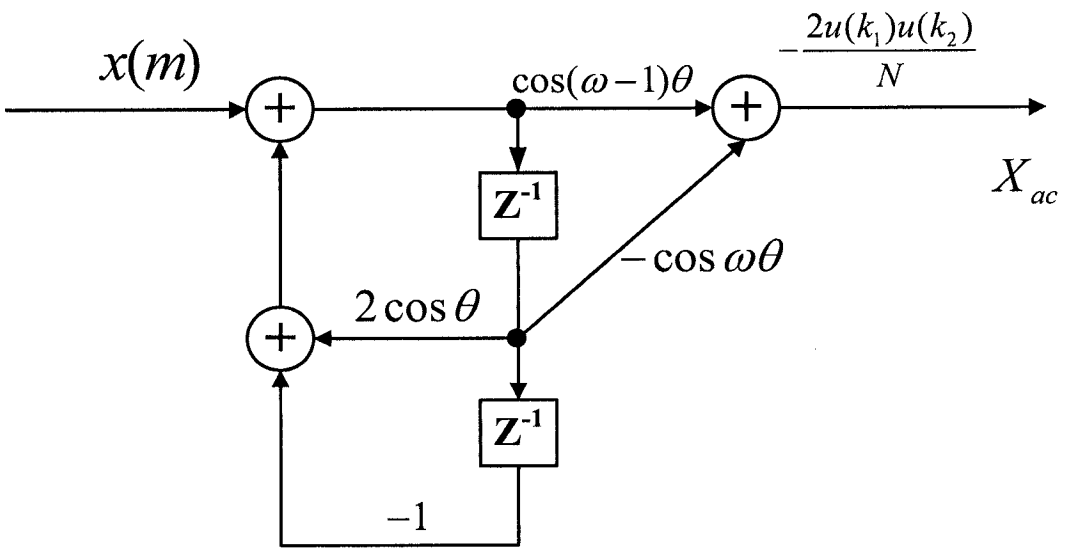

Fig. 3.4. Proposed recursive kernel.

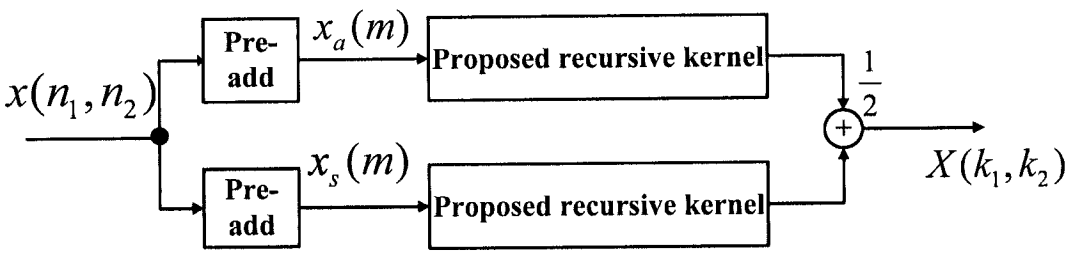

Fig. 3.5. Computation structure of the proposed algorithm for the 2-D DCT using the proposed recursive kernels

To verify the equivalence of the function of the recursive kernel illustrated in Fig. 3.2 and that in Fig. 3.4, simulations using Simulink have been done. The results, shown in Figs. 3.6 and 3.7, are identical, conforming that the kernels can replace each other for the same computation. The option of computing the 2-D DCT by the proposed recursive operation illustrated in Fig.3.4 can lead to an effective reduction of the number of multiplications and that of additions.

It should be mentioned that, similar to the recursive kernel reported in [21], the proposed one can not only be used for the DCT, bet also in the computations of the IDCT, the DST and the IDST, by means of different inputs and pre-addition procedures.

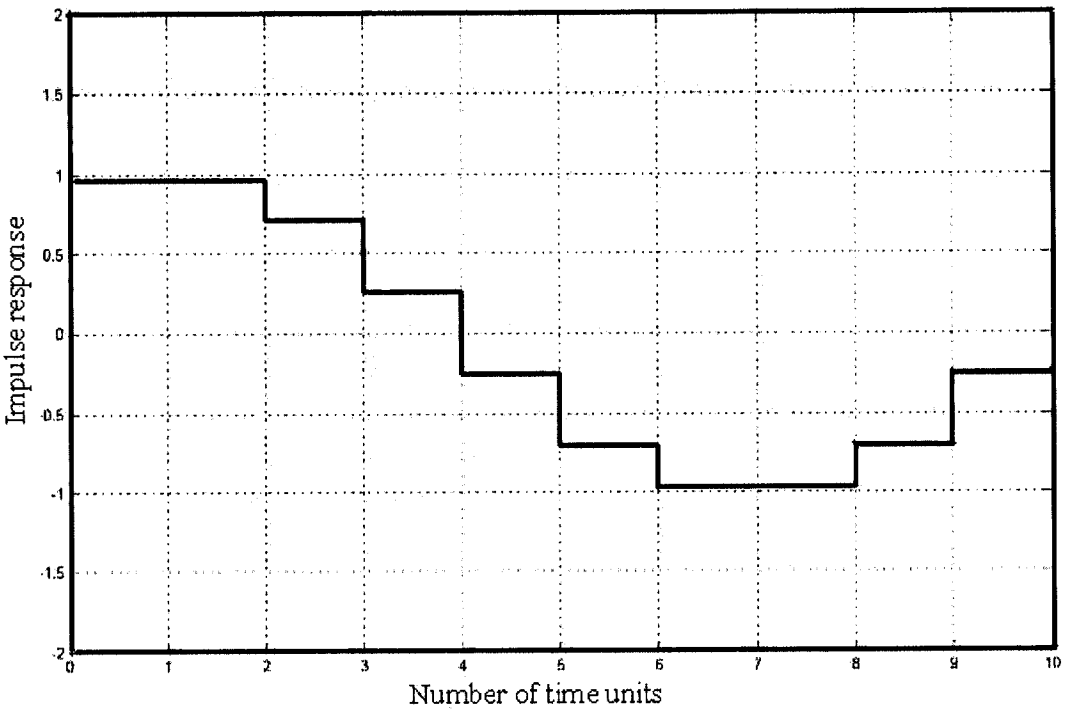

Fig. 3.6. Simulation result of the recursive kernel shown in Fig. 3.4.

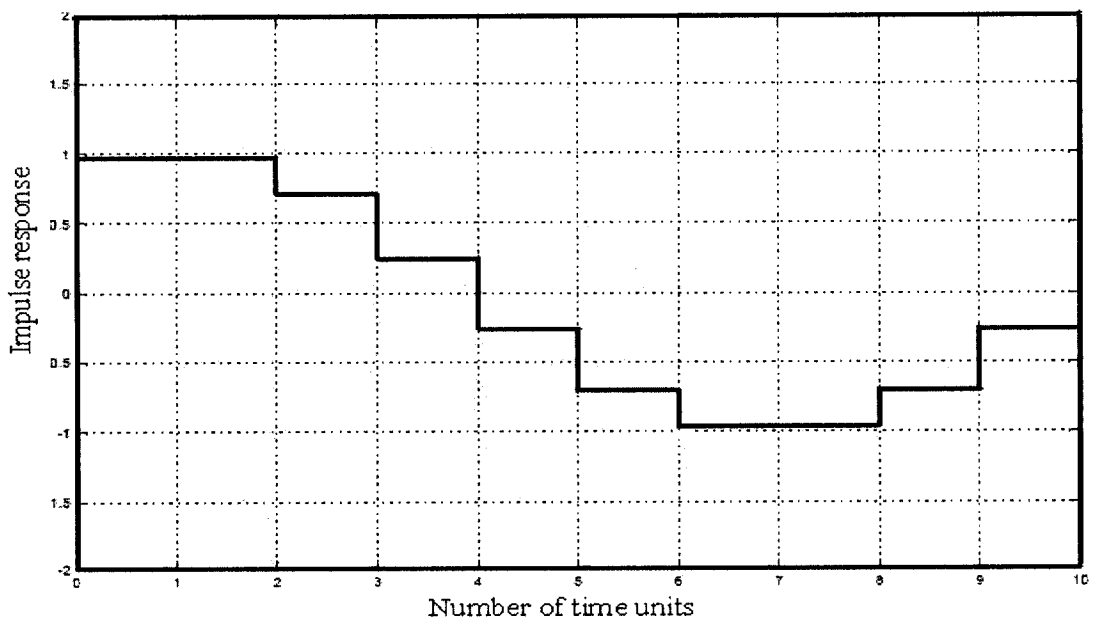

Fig. 3.7. Simulation result of the recursive kernel shown in Fig. 3.2.

3.3 Summary

In this chapter, a new recursive algorithm and the computation structure for the 2-D DCT computation has been proposed. This algorithm is developed on the basis of the condensed 1-D transforms for the 2-D DCT computation reported in [21], and the 2-D computation is decomposed according to the prime-and-non-prime nature of the index variables, instead of a straightforward row-column decomposition, to minimize the number of the recursive cycles. The new algorithm aims at reducing the number of multiplications and to remove the computation of *sine* terms to simplify the procedure in each cycle. By using the trigonometry identity, the computation for the 2-D DCT is reformulated. It is much simplified and can be calculated by means of only 1-D DCTs instead of the computation involving both 1-D DCTs and 1-D DSTs in [21]. Also, the simplification helps to reduce the number of input coefficients and the amount of calculations.

Based on the new computation procedure of the 2-D DCT, a recursive kernel has been presented. The proposed kernel involves only four multiplications and three additions, whereas that in [21] needs six multiplications and four additions. The simulation results using Simulink have confirmed that these two recursive kernels can perform the same function and thus equivalent in the 2-D DCT computation. Therefore, the proposed kernel can replace the one in [21] and the computation structure using the proposed kernel can be made simpler than that of [21]. Further more, like the kernel reported in [21], the

proposed one can also be used in the computation of 2-D IDCT, DST and IDST calculations.

The next chapter is dedicated to the design of two architectures implementing the proposed algorithm to demonstrate the effectiveness of the improvement in performance resulting from the new algorithm. FPGA implementations for the architectures and that of the algorithm in [21] are presented and the results are compared.

Chapter 4

Proposed Architectures and the FPGA Implementations

In chapter 3, a new 2-D DCT algorithm has been presented. The recursive kernel in this algorithm requires only *cosine* computation terms and a smaller number of calculations compared to that in [21]. Thus, the proposed algorithm can lead to a reduction on the number of multiplications and input coefficients. To demonstrate the effectiveness of the new algorithm, the 1-D DCT circuit block is designed by directly mapping the proposed recursive kernel presented in Section 3.2.2, and by using this 1-D DCT circuit block, two architectures for the 2-D DCT computation are developed. The first architecture is designed in parallel by using two 1-D DCT circuit blocks. The other one uses only one circuit block, which operates by time division, to compute the 2-D DCT calculation. The FPGA simulations of these two architectures are done and the simulation results will be compared.

In Section 4.1, the proposed 1-D DCT circuit block is presented. Section 4.2, the first architecture is presented in details. Then the architectures using only one DCT circuit block are described in Section 4.3. Section 4.4 summarizes the work of the hardware

designs and implementations.

4.1 Circuit Block of the Recursive Kernel

The computation of the proposed recursive kernel, shown in Fig. 3.4, can be easily mapped into a simple circuit block as shown in Fig. 4.1. This circuit does not need sine coefficients and it is, in fact, a 1-D DCT block. Using the same mapping, another block for the recursive kernel [21] is also built, illustrated in Fig. 4.2, for the comparison purpose.

Comparing the structures of the two blocks shown in Figs. 4.1 and 4.2, one can have the following observations:

- It is confirmed that the block shown in Fig. 4.1 requires four multipliers and three adders, instead of six multipliers and four adders in that in Fig. 4.2. Thus, a significant reduction of circuit complexity should be expected in the hardware implementation.

- Besides the inputs of x_a and x_s generated by the pre-computation modules, both blocks receive other pre-computed inputs. The block of Fig. 4.2 needs six such inputs, namely *a, b, c, d, e* and *f*, whereas in that of Fig. 4.1 only four are needed. Therefore, the pre-computation operations required in the system using the block of Fig. 4.1 can be made much simpler than that of Fig. 4.2.

- The length of the most critical delay path in a block determines the required duration of the clock cycle. One can easily see that there are three multipliers in the

critical path of the block of Fig. 4.2, and only two in that of Fig. 4.1. The delay in the latter is obviously much shorter than that in the former. Also, taking the number of adders in the critical paths of the two blocks into consideration, one can expect that the delay of the circuit block of the proposed recursive kernel is at least 33% shorter than that of Fig. 4.2. The proposed algorithm can, therefore, lead to a significant increase of the clock frequency.

Having a smaller number of operators such as multipliers and adders and fewer input coefficients, the circuit block for the proposed recursive kernels can perform a smaller number of operations for the same computation as that shown in Fig. 4.2. It can thus be implemented with a smaller number of basic calculation units and shorter delay path to improve both hardware efficiency and processing speed. The VLSI architectures for the 2-D DCT with the proposed block shown in Fig. 4.1 are presented in the following sub-sections.

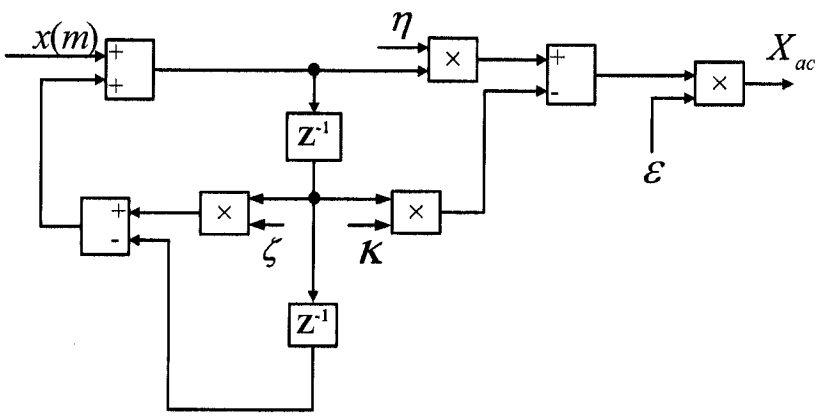

Fig. 4.1. Structure of the circuit block for the proposed recursive kernel. In this structure, the coefficient inputs are mapped from the kernel shown in Fig. 4, i.e. $\eta = cos(\omega-1)\theta$, $\kappa = cos\omega\theta$, $\varepsilon = -u(k_1)u(k_2)/N$, and $\zeta = 2cos\theta$.

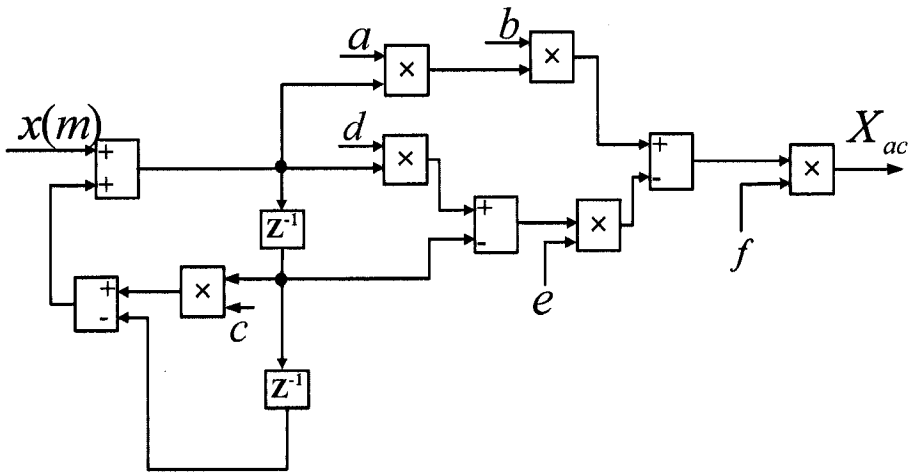

Fig. 4.2. Structure of the circuit block for the computation kernel of [21]. In this structure, the inputs, $a = sin\theta$, $b = sin((\omega_1 \pm \omega_2)\pi/2M)$, $c = 2cos\theta$, $d = cos\theta$, $e = cos((\omega_1 \pm \omega_2)\pi/2M$, and $f = -u(k_1)u(k_2)/N$.

4.2 Architecture-1 for the Proposed Algorithm

As mentioned in the previous sections, the 2-D DCT computation can be implemented by using the proposed circuit block of the new recursive kernel shown in Fig. 4.1. With this block, the computation structure shown in Fig. 3.5 can be easily mapped into an architecture illustrated in Fig. 4.3. In this architecture, two identical circuit blocks operate in parallel. However, it should be noted that one block receives the inputs $\alpha = cos((\omega_1+\omega_2)/2-1)\pi/M$ and $\gamma = cos((\omega_1+\omega_2)\pi/2M)$, while the inputs $\beta = cos((\omega_1-\omega_2)/2-1)\pi/M$ and $\delta = cos((\omega_1-\omega_2)\pi/2M)$ are applied to the other block. Hence, the former produces X_{ac1} and the latter X_{ac2}. The final output signal $X(k_1, k_2)$ is generated by an addition of the two outputs of the blocks.

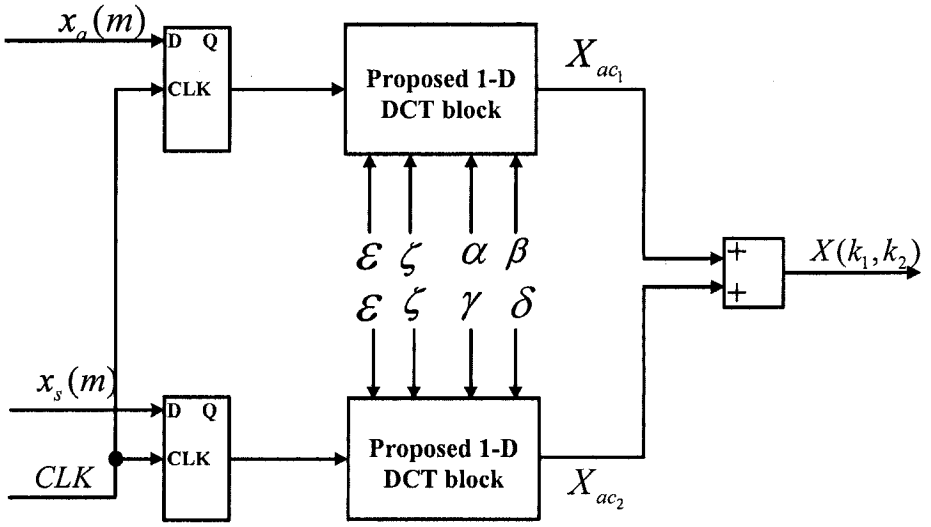

Fig. 4.3. Proposed architecture for the new 2-D DCT computation. The structure of the proposed 1-D DCT block is shown in Fig. 4.1. The inputs $x_a(m)$ and $x_s(m)$ are generated by a pre-addition block.

Using a similar direct mapping, a circuit architecture implementing the algorithm of [21] is obtained, as illustrated in Fig. 4.4. It includes two identical circuit blocks shown in Fig. 4.2.

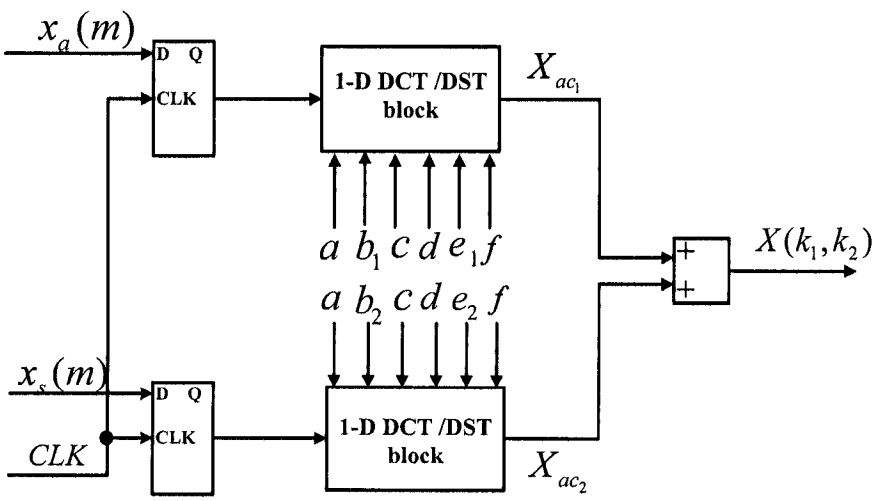

Fig. 4.4. Architecture for the 2-D DCT algorithm of [21]. The structure of the 1-D DCT/DST block is shown in Fig. 4.2.

The architectures shown in Figs. 4.3 and 4.4 can be easily implemented with FPGA technology. In this implementation, Virtex-II Pro Platform FPGA boards of xc2vp7 are used. For the two architectures, the inputs and outputs are of 12-bit float-point signals. The results are presented in Table I. One can note the following points.

• The architecture of the proposed algorithm requires only 75% of hardware consumption compared to that of [21]. This results from the smaller number of multiplications and additions in the proposed algorithm that involves only 1-D DCT computation.

• Both architectures can be easily implemented in the FPGA board of xc2vp7. Obviously, the hardware utilization of the architecture shown in Fig. 4.3 is much less than that of Fig. 4.4, which means that the architecture of Fig. 4.3 permits the integration of more logic functions in the same board. Furthermore, this architecture can also be implemented in a small and low-cost board such as xc2vp4 that may not be suitable to implement the one of Fig. 4.4.

• Because of the smaller number of operations in the critical path of the computation, the required clock cycle duration of the architecture shown in Fig. 4.3 is only 83% of that of [21], which implies a speed improvement resulting from the proposed algorithm. The

reduction of the cycle time can be more significant if the algorithm is implemented in a custom-designed integrated circuit.

• The power dissipation of the circuit architecture for the proposed algorithm is almost the same with that of [21], proving that the proposed algorithm can save hardware consumption and processing time without sacrificing the power efficiency.

• The throughput of the architecture shown in Fig. 4.3 is the same as that in Fig. 4.4, which is one input sample per clock cycle. Both of them need N clock cycles to produce an output sample if the dimension of the 2-D signal is $N \times N$. The number of the cycles may be further reduced in case of the architecture shown in Fig 4.3 by means of the input-folding method [21], and this reduction is done at the expense of increasing the hardware consumption.

It should be noted that when considering the pre-addition procedure in the hardware implementation, because most are the simple adders in the pre-addition, the hardware requirement of the pre-addition should be very small. The hardware consumption of the proposed algorithm should be around 78% of that of [21].

TABLE I

FPGA IMPLEMENTATION RESULTS OF THE ARCHITECTURES SHOWN IN FIGS. 4.3 AND 4.4

		Architecture shown in Fig. 4.4	Architecture shown in Fig. 4.3
Slice Registers	Number	272	217
	Utilization	2%	2%
Occupied Slices	Number	4,008	2,997
	Utilization	81%	60%
4 Input LUTs	Number	7,710	5,777
	Utilization	78%	58%
Minimum Clock Cycle (ns)		9.103	7.611

In conclusion, the FPGA results agree with the expected performance of the circuit architecture for the proposed algorithm. The improvement in terms of hardware consumption and operation speed has been achieved at no expense of power efficiency.

4.3 Architecture-2 for the Proposed Algorithm

The architecture shown in Fig. 4.3 employs two circuit blocks to generate X_{ac1} and X_{ac2}, respectively, for the 2-D DCT computation. The two blocks execute, in fact, the same operation with partially different inputs. It is thus possible to use only one circuit block for both two functions of X_{ac1} and X_{ac2}, and the matter is to select the right inputs for the two functions. By examining the architecture shown in Fig. 4.3, it is easy to see

that the calculation of X_{ac1} requires $\alpha = cos((\omega_1+\omega_2)/2\text{-}1)\pi/M$ and $\gamma = cos((\omega_1+\omega_2)\pi/2M)$, while X_{ac2} needs $\beta = cos((\omega_1-\omega_2)/2\text{-}1)\pi/M$, and $\delta = cos((\omega_1-\omega_2)\pi/2M)$. One can use simple multiplexers to select the input signals (α & γ) or (β & δ) in order that the circuit block produces the right output. In this way, the proposed algorithm can be implemented in the circuit using only one circuit block for all the recursive operations, as illustrated in Fig. 4.5, which can result in a significant reduction of the hardware consumption and enable an even-lower-cost circuit implementation.

In the architecture shown in Fig. 4.5, the signal inputs via the multiplexers are synchronized with the clock signal, of which each cycle consists of two phases. In the first phase, i.e. CLK = '1', x_a, α and γ are applied to the circuit block and during the second phase, x_s, β and δ are selected to be applied. The two outputs of the circuit block generated during the two phases are summed up to generate the final output signal $X(k_1, k_2)$. In this way, the computation of the 2-D DCT can be realized by only one 1-D DCT block, instead of the two in Fig. 4.3.

The architecture in Fig. 4.5 can also be used for other computation tasks if the main circuit block is replaced by another module. In many cases of signal processing, the computation can be decomposed into two parts, one by processing cores and the other by pre-computation blocks. It is possible to use only one processing core for different functions during different phases, while applying different pre-computed inputs. In this

manner, the required hardware can be considerably reduced, which leads to a reduction of the circuit cost in a great scale.

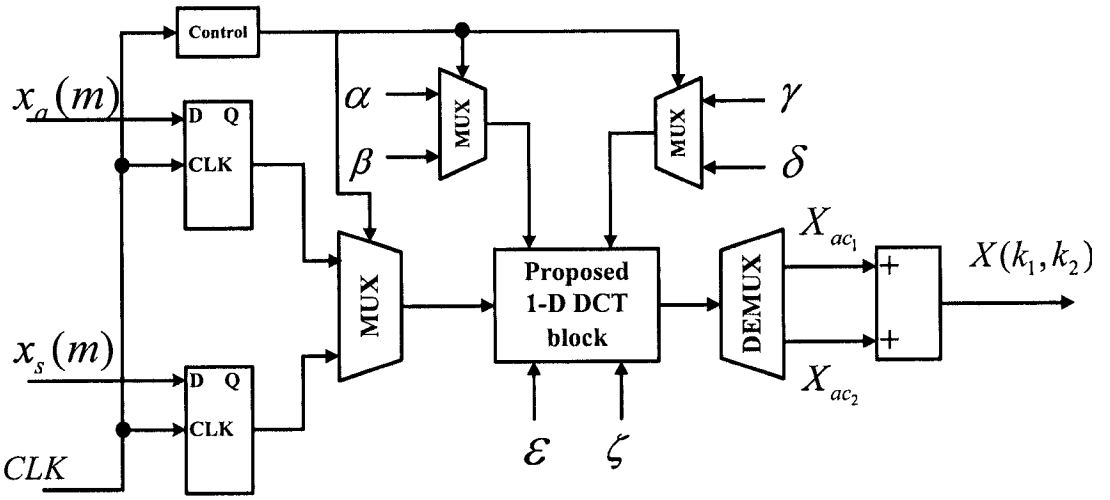

Fig. 4.5. Architecture for the 2-D DCT proposed for further improvement of hardware consumption. In this structure, $\alpha=cos((\omega_1+\omega_2)/2-1)\pi/M$, $\beta=cos((\omega_1-\omega_2)/2-1)\pi/M$, $\gamma=cos((\omega_1+\omega_2)\pi/2M)$, $\delta=cos((\omega_1-\omega_2)\pi/2M)$, $\varepsilon=-u(k_1)u(k_2)/N$, and $\zeta=2cos\pi/M$.

The same method of using a single processing core can be easily applied to simplify the structure implementing the algorithm of [21]. In the architecture shown in Fig. 4.4, among the inputs *a, b, c, d, e* and *f* of each circuit block, *b* and *e* are not common for the two blocks. By means of multiplexers, the structure in Fig. 4.4 is converted to that illustrated in Fig. 4.6. It is evident that the conversion reduces the hardware to one half of that used for Fig. 4.4, as only one processing block, instead of two, is included in the circuit.

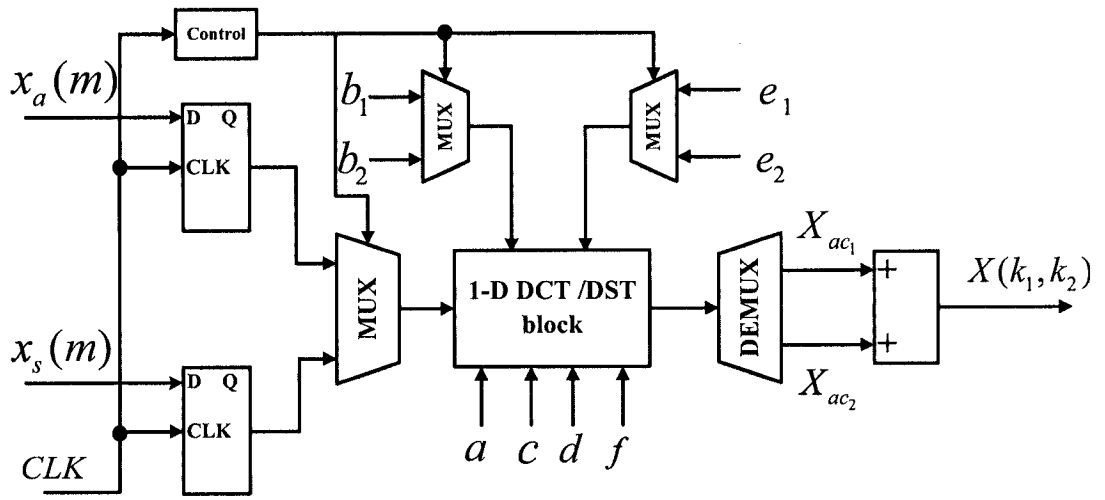

Fig. 4.6. Architecture for the implementation of the 2-D DCT algorithm [21]. This structure includes only one 1-D DCT/DST block. Its pre-computed inputs are $a = \sin\theta$, $b_1 = \sin((\omega_1+\omega_2)\pi/2M)$, $b_2 = \sin((\omega_1-\omega_2)\pi/2M)$, $c = 2\cos\theta$, $d = \cos\theta$, $e_1 = \cos((\omega_1+\omega_2)\pi/2M)$. $e_2 = \cos((\omega_1-\omega_2)\pi/2M)$, and $f = -u(k_1)u(k_2)/N$.

The circuit architectures shown in Figs. 4.5 and 4.6 have been implemented in the same kind of FPGA boards as that presented in Section III-B, i.e. Virtex-II of xc2vp7, under the same conditions of 12 bits float-point signals. The FPGA results of the architecture shown in Fig. 4.5, with the comparison to those of Fig. 4.3, are presented in Table II to illustrate the difference in hardware consumption between the two architectures performing the same computation. Table III shows the results of those of Figs. 4.5 and 4.6.

From the FPGA results shown in Table II and Table III, the following points can be noticed.

•The proposed method of using a single processing core helps to reduce significantly the hardware consumption in all the aspects, including logic gates and memories. In the case of implementing the two different 2-D DCT algorithms, the reduction of the hardware is consistently at a rate of 43%.

• As the result of the reduced hardware requirement, the circuit architectures designed using the proposed method can be easily integrated in a wide range of FPGA boards.

• By using the proposed method, the recursive block used in each of the circuit architectures shown in Figs. 4.5 and 4.6 operates to compute X_{ac1} and X_{ac2} successively, not simultaneously, in each clock cycle. Thus the duration of the cycle is expected to be doubled, compared to that of the architectures shown in Figs. 4.3 and 4.4. However, Table II or III shows that the required clock cycle of the architecture illustrated in Fig. 4.5 or 4.6 is about three times of that in Fig. 4.3 or 11, which may be due to some redundant structure of the FPGA board.

• It should be mentioned that the throughput of the architectures shown in Figs.4.5 and Fig. 4.6 is the same as that of Figs. 4.3 and 4.4 and the number of the recursive cycles is also the same, which is N.

TABLE II

FPGA RESULTS OF THE ARCHITECTURES SHOWN IN FIGS. 4.5 AND 4.3

Structures for the proposed algorithm		Architecture shown in Fig. 4.3	Architecture shown in Fig. 4.5
Slice Registers	Number	217	148
	Utilization	2%	1%
Occupied Slices	Number	2,997	1,709
	Utilization	60%	34%
4 input LUTs	Number	5,777	3,241
	Utilization	58%	32%
Minimum Clock Cycle (ns):		7.611	20.916

TABLE III

FPGA RESULTS OF THE ARCHITECTURES SHOWN IN FIGS. 4.6 AND 4.4

Structures for the algorithm in [21]		Architecture shown in Fig. 4.4	Architecture shown in Fig. 4.6
Slice Registers	Number	272	172
	Utilization	2%	1%
Occupied Slices	Number	4,008	2,207
	Utilization	81%	44%
4 input LUTs	Number	7,710	4,203
	Utilization	78%	42%
Minimum Clock Cycle(ns):		9.103	25.592

The FPGA results provide a good confirmation of the significant reduction of hardware consumption by applying the one-processing-core method. This reduction comes with some increase of processing time and is good to use if the speed is not a critical issue. In any case, it provides a good trade-off of hardware and processing time.

4.4 Summary

In this chapter, the hardware implementation of the proposed algorithm has been presented. By easily mapping the proposed recursive kernel, a 1-D DCT circuit block is obtained to perform the recursive computation. The new block requires a smaller number of multipliers and adders, i.e., four multipliers and three adders instead of six and four in that of [21]. Also, it needs fewer pre-computed inputs. Besides, because of the simpler structure of the circuit block, the length of its critical path is shorter than that of [21]. Applying this circuit block, two architectures for the 2-D DCT computation have been designed and the FPGA implementation of these two architectures has also been carried out. The architecture-1, designed by a direct mapping of the 2-D DCT computation scheme, employs two 1-D DCT circuit blocks. Its implementation aims at a demonstration of the effectiveness of the new algorithm. The FPGA results of the new algorithm have resulted in a reduction of the hardware of 25% and that of the clock duration of 17%. The architecture-2 has been proposed to provide a simple structure of the 2-D DCT computation by using only one 1-D DCT circuit block. The computation of

the 2-D DCT is carried by time division. In this way the hardware reduction is very significant, while the computation time is extended. The FPGA results of the architecture-2 have shown that by means of this one DCT block method, the hardware consumption can be reduced consistently at a rate of 43%. Because of the time division, the clock cycle is longer than that of the first design, which makes a trade-off with the hardware efficiency.

Chapter 5

Conclusion

5.1 Concluding Remarks

The work presented in this thesis is in the topic area of algorithms and architectures for the 2-D discrete cosine transform (DCT). It also involves the hardware implementation of the computation.

There are two objectives of the work. One is to develop a new recursive algorithm for the 2-D DCT. The emphasis is on reducing the number of calculations, in particular, that in each cycle. The other objective is to design architectures for the 2-D DCT recursive algorithms, aiming at reducing the circuit complexity for low-cost implementation.

To develop a new recursive 2-D DCT algorithm, a study of existing ones has first been carried out. In order to have the minimum number of recursive cycles in the iterative computation for the 2-D DCT, the study was focused on the non-row-column decomposition methods. The new algorithm has been proposed based on the method of using the condensed 1-D transform [21]. By means of mathematics reformulation, the calculation procedure is simplified. The new algorithm requires, in its recursive kernel, a

smaller number of calculations, i.e., only 2/3 of the multiplications and 3/4 of the additions needed in [21]. Moreover, it does not include DST calculation and has fewer input coefficients. Thus, it is expected that the new algorithm can lead to an improvement of performance of the system where it is implemented. It should also be noted that the recursive kernel of the proposed algorithm can be used in the computation of 2-D IDCT, DST or IDST by means of different pre-addition procedures.

A circuit architecture for the 2-D DCT has been built by direct-mapping of the recursive kernel of the proposed algorithm. It is to evaluate the performance improvement resulting from the new algorithm. The reduction of the calculation in the algorithm has made the circuit simpler, also the critical delay path shorter. The architecture has been implemented by using the Xilinx Virtex-II Pro Platform FPGA boards of xc2vp7. The FPGA results have shown that the new algorithm has contributed a hardware reduction of 25% and shortened the clock duration by 17%. It is evident that, by using the new algorithm, both space and circuit delay can be reduced significantly without sacrificing other specifications.

A new architecture scheme to implement a recursive 2-D DCT algorithm has also be proposed. It makes it possible to reduce very significantly the hardware resources for a given 2-D DCT computation, while increasing the clock duration. Two 2-D DCT algorithms have been implemented using the new scheme in FPGA boards. It has been proved, by the FPGA implementation, that the new scheme results in a hardware reduction of 43% in the cases of the two algorithms. The clock cycle in the circuit is

expected to be doubled. However, by using the new algorithm, the time in each clock cycle can be reduced to make the time loss in the new scheme less critical. The scheme provides a good trade-off of the hardware-speed, enabling low cost applications.

5.2 Suggestions for Future Investigation

In the area of the 2-D DCT computation and its implementation, the way of decomposing the computation determines the complexity of the implementation. Thus, the methods of decomposition are worth research efforts. More studies will be conducted to investigate the existing methods and to develop new non-row-column decomposition algorithms. Also, research work is needed to find an optimal way of designing computation process of the pre-additions.

In the work of this thesis, the circuit design has been focused on the architectural level. The block for recursive kernel has been built by direct mapping. One may reduce the length of the critical delay path by reorganizing the calculations in each recursive cycle. Thus the operation speed can be further improved.

The 2-D DCT algorithm and architecture described in this thesis are for general-purpose, i.e. without specifying a particular application. We can design a 2-D DCT algorithm that is specifically tailored to a given system performing a particular function. Research on the design of special-purpose 2-D DCT architecture is needed.

Reference

[1] "IEEE Standard Specifications for the Implementation of 8×8 Inverse Discrete Cosine Transform", *IEEE Standard 1180-1990*, March, 1991.

[2] Pennebaker, W. B. and Mitchell, J. L. *JPEG - Still Image Data Compression Standards*, Van Nostrand Reinhold, 1993.

[3] Weiping Li, "A New Algorithm to Compute the DCT and its Inverse," *IEEE Trans. on Signal Processing*, Vol. 39, No. 6, pp. 1305-1313, June 1991.

[4] Yung-Pin Lee, T. Chen, L. Chen, M. Chen, C. Ku, "A Cost-Effective Architecture for 8×8 DCT/IDCT Using Direct Method," *IEEE Trans. Circuits Syst. Video Tech.*, Vol. 7, No. 3, June 1997.

[5] A. Madisetti, A. N. Willson, "A 100 MHz 2-D 8×8 DCT/IDCT Processor for HDTV Applications," *IEEE Trans. Circuits Syst. Video Tech.*, Vol. 5, No.2, pp. 158-164, April 1995.

[6] L. Montalvo, A. Guyot, "Combinational Digital-set converters for Hybrid Radix-4 Arithmetic," in *Proc. of IEEE International Conference on Computer Design*

ICCD'94, April 1994, pp. 498-164.

[7] K. R. Rao and P. Yip, *Discrete Cosine Transform Algorithms, Advantages Applications*. New York: Academic, 1990.

[8] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *Amer. Math. Month.*, vol. 65, pp. 34–35, 1958.

[9] Y. H. Chan, L. P. Chau, and W. C. Siu, "Efficient implementation of discrete cosine transform using recursive filter structure," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 550–552, Dec. 1994.

[10] J. F. Yang and C. P. Fan, "Compact recursive structures for discrete cosine transform," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 314–321, Apr. 2000.

[11] J. F. Yang and C. P. Fan, "Recursive implementation of discrete cosine transforms: with selectable fixed coefficient filters," *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 211–216, Feb. 1999.

[12] C. L. Wang and C. Y. Chen, "High-Throughput VLSI Architectures for the 1-D and 2-D Discrete Cosine Transforms", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 31–40, Feb. 1995.

[13] M. F. Aburdene, J. Zheng, and R. J. Kozick, "Computation of discrete cosine transform using Clenshaw's recurrence formula," *IEEE Signal Processing Lett.*, vol. 2, pp. 155–156, Aug. 1995.

[14] P. A. Ruetz, P. Tong, D. Bailey, P. A. Luthi, and P. H. Ang, "A high-performance full-motion video compression chip set," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 111–122, June 1992.

[15] Kim and J. S. Koh, "An area efficient DCT architecture for MPEG-2 video encoder," *IEEE Trans. Consumer Electron.*, vol. 45, pp. 62–67, Feb. 1999.

[16] D. Gong, Y. He, and Z. Cao, "New Cost-Effective VLSI Implementation of a 2-D Discrete Cosine Transform and Its Inverse," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, pp. 405-415, Apr. 2004.

[17] M. A. Haque, "A two-dimensional fast cosine transforms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 1532–1539, June 1985.

[18] H. S. Hou, "A fast recursive algorithms for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1455–1461, Oct. 1987.

[19] J. F. Yang and C. P. Fan, "Fast structural two dimensional discrete cosine transform algorithms," *IEICE Trans. Fund. Electron. Commun. Comput. Sci.*, vol. E81-A, pp. 1210–1215, 1998.

[20] N. I. Cho, I. D. Yun, and S. U. Lee, "On the regular structure for the fast 2-D DCT algorithm," *IEEE Trans. Circuit Syst. II*, vol. 40, pp. 259–266, Apr. 1993.

[21] C. H. Chen, B. D. Liu, and J. F. Yang, "Direct recursive structures for computing radix-r two-dimensional DCT/IDCT/DST/IDST," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, pp. 2017–2030, Oct. 2004.

[22] Z. Wang, G. A. Jullien, and W. C. Miller, "Recursive algorithms for the forward and inverse discrete cosine transform with arbitrary length," *IEEE Signal Processing Lett.*, vol. 1, pp. 101–102, July 1994.

[23] M. F. Aburdene, J. Zheng, and R. J. Kozick, "Computation of discrete cosine transform using Clenshaw's recurrence formula," *IEEE Signal Processing Lett.*, vol. 2, pp. 155–156, Aug. 1995.

[24] J. F. Yang and C. P. Fan, "Compact recursive structures for discrete cosine transform," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 314–321, Apr. 2000.

[25] J. L. Wang, C. B. Wu, B. D. Liu, and J. F. Yang, "Implementation of the discrete cosine transform and its inverse by recursive structures," *in Proc. IEEE Workshop Signal Processing Systems*, Oct. 1999, pp. 120–130.

[26] S. An and C. Wang, "A recursive algorithm for 2-D", *Proc. URSI International Symposium on Signals, Systems and Electronics,* Montreal, Canada, July 2007, pp.335 – 338.