New Hardware-Efficient Algorithm and Architecture for the Computation of 2-D DCT on a Linear Systolic Array

Shen-Fu Hsiao   Wei-Ren Shiue
Institute of Computer and Information Engineering
National Sun Yat-Sen University
Taiwan

Abstract

A new recursive algorithm for fast computation of two-dimensional discrete cosine transforms (2-D DCT) is derived by converting the 2-D data matrices into 1-D vectors and then using different partition methods for the time and frequency indices. The algorithm first computes the 2-D complex DCT (2-D CCT) and then produces two 2-D DCT outputs simultaneously through a post-addition step. The decomposed form of the 2-D recursive algorithm looks very like a radix-4 FFT algorithm and is in particular suitable for VLSI implementation since the common entries in each row of the butterfly-like matrix are factored out in order to reduce the number of multipliers. A new linear systolic architecture is presented which leads to a hardware-efficient architectural design requiring only *logN* multipliers plus *3logN* adders/subtractors for the computation of two $N \times N$ DCTs.

1. Introduction

Two-dimensional discrete cosine transform (2-D DCT), in particular the 8×8 DCT, is one of the core operations in current standard image and video coding. There are a lot of papers discussing the fast implementation of the DCT/IDCT. One category of papers decomposes the original 8×8 DCT into two 4×4 matrix-vector multiplications through input data reordering. The overall 2-D DCT architectures of these papers consist of 1-D DCT processors with transpose memory to perform operations on the row and column of the input data. Due to the structural regularity, many 2-D DCT chips are fabricated using this method [4-6]. However, the computation complexity of this method is usually high.

Another group of papers has lower computation complexity but leads to irregular structures [1-3]. These papers present fast DCT algorithms and derive the corresponding signal flow graphs (SFGs) that look like those for FFT. However, the SFG-like architectures require a lot of hardware area and the input/output burden is heavy. The irregular global interconnection wiring in the SFG-like architecture and the high hardware cost make these architectures not very suitable for VLSI implementation. Furthermore, as required in the first group of papers, these 2-D DCT approaches usually calls for transpose memory for data exchange of the intermediate results.

In this paper, we will present a new recursive algorithm and the corresponding linear systolic architecture with very low hardware complexity for fast computation of the 2-D DCT. The new 2-D DCT architecture requires only 3 complex multipliers and 9 complex adders for the computation of two concurrent 2-D 8×8 DCTs. Furthermore, the developed linear array architecture is fully pipelinable and is easily scalable to process 2-D DCT of general size N (with N power of 2).

2. Recursive Algorithm for 2-D DCT

The 2-D $N \times N$ DCT $y(k_1, k_2)$ of the input signal $x'(n_1, n_2)$ is defined as

$$y(k1,k2) = \frac{2}{N} b(k1) b(k2) \sum_{n1=0}^{N-1} \sum_{n2=0}^{N-1} x'(n1,n2) \cos\left(\frac{\pi(n1+0.5)k1}{N}\right) \cos\left(\frac{\pi(n2+0.5)k2}{N}\right) \tag{1}$$

$$b(k) = \begin{cases} 1/\sqrt{2}, & k = 0 \\ 1, & k \neq 0 \end{cases}$$

where the transform size N will be assumed to be power of 2 in the following. Reordering the time indices n_1 and n_2 as follows

$$\begin{aligned} & x(i1,i2) = x'(2i1,2i2), \\ & x(i1,N-i2-1) = x'(2i1,2i2+1), \\ & x(N-i1-1,i2) = x'2i1+1,2i2, \\ & x(N-i1-1,N-i2-1) = x'(2i1+1,2i2+1) \end{aligned} \qquad i_1,i_2 = [0,N/2-1]\,, \tag{2}$$

we obtain from the original input signal $x'(n1,n2)$ the permuted input signals $x(n1,n2)$. Using the reordered input signal $x(n1,n2)$ and neglecting the post-scaling factors, Eqn. (1) of the 2-D DCT becomes

$$z(k1,k2) = \sum_{n1=0}^{N-1} \sum_{n2=0}^{N-1} x_{n1,n2} \cos\left(\theta_{k1} + \frac{2\pi k1n1}{N}\right) \cos\left(\theta_{k1} + \frac{2\pi k1n1}{N}\right) \tag{3}$$

$$k1,k2,n1,n2 = [0,N-1], \quad \theta_k = \frac{\pi k}{2N}.$$

From now on, we focus on the fast computation of $z(n1,n2)$ with the permuted input $x(n1,n2)$. First, we define an $N \times N$ 2-D complex DCT (2-D CCT) as

$$w(k1,k2) = \sum_{n1=0}^{N-1} \sum_{n2=0}^{N-1} x(n1,n2) W_N^{\frac{k1}{4}+k1n1} W_N^{\frac{k2}{4}+k2n2}, \tag{4}$$

$$k1,k2,n1,n2 = [0,N-1], \quad W_N = \exp(-j\frac{2\pi}{N}).$$

The relationship between the 2-D DCT of Eqn. (3) and the 2-D CCT of Eqn. (4) is

$$z(k1,k2) = \frac{1}{2}\left(\mathrm{Re}(w(k1,k2)) - \mathrm{Im}(w(N-k1,k2))\right)$$

We consider two permuted 2-D signals $x1(n1,n2)$, $x2(n1,n2)$ and combine them into another 2-D complex signal $x(n1,n2) = x1(n1,n2) + jx2(n1,n2)$. The 2-D DCTs $z1(k1,k2)$ and $z2(k1,k2)$ for $x1(n1,n2)$ and $x2(n1,n2)$ respectively can be computed from the 2-D CCT $w(k_1,k_2)$ of the complex 2-D signal as follows:

$$\begin{aligned} z1(k1,k2) &= \frac{1}{4}\begin{pmatrix} \mathrm{Re}[w(k1,k2)] - \mathrm{Re}[w(N-k1,N-k2)] \\ -\mathrm{Im}[w(N-k1,k2)] - \mathrm{Im}[w(k1,N-k2)] \end{pmatrix} \\ z2(k1,k2) &= \frac{1}{4}\begin{pmatrix} -\mathrm{Im}[w(k1,k2)] + \mathrm{Im}[w(N-k1,N-k2)] \\ +\mathrm{Re}[w(N-k1,k2)] + \mathrm{Re}[w(k1,N-k2)] \end{pmatrix} \end{aligned} . \tag{5}$$

Thus we can compute the 2-D CCT of the complex composite signal $x3(n1,n2) = x1(n1,n2) + jx2(n1,n2)$ and generate two 2-D DCTs $z1(k1,k2)$ and $z2(k1,k2)$ simultaneously using the post-processing operation characterized by Eqn. (5). In the following, we will focus on the fast computation of the 2-D CCT in Eqn. (4).

The 2-D CCT kernel operation could be split into sixteen parts, as shown in Tab. 1 where (•) denotes $(N/2 \times N/2)$ for clarity reason. Each of the two time indices n_1 and n_2 is partitioned into the first and the second half parts, and each of the two frequency indices k_1 and k_2 is partitioned into the even and the odd parts. The relationship of the sixteen parts is illustrated in Tab. 1 where $CCT(\bullet)$ denotes the $N/2 \times N/2$ 2-D CCT with input time indices $n_1 = [0, N/2-1]$, $n_2 = [0, N/2-1]$ and the frequency indices $k_1 = 2i_1$, $k_2 = 2i_2$, $i_1, i_2 = [0, N/2-1]$.

	$k_1 = 2i_1$ $k_2 = 2i_2$	$k_1 = 2i_1$ $k_2 = 2i_2 + 1$	$k_1 = 2i_1 + 1$ $k_2 = 2i_2$	$k_1 = 2i_1 + 1$ $k_2 = 2i_2 + 1$
$(n_1, n_2) = (n_1^{'}, n_2^{'})$	CCT $(\bullet)$	CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n2'}{2}}$	CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n1'}{2}}$	CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{4}+\frac{n1'}{2}+\frac{n2'}{2}}$
$(n_1, n_2) = (n_1^{'}, n_2^{''})$	CCT $(\bullet)$	-CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n2'}{2}}$	CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n1'}{2}}$	-CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{4}+\frac{n1'}{2}+\frac{n2'}{2}}$
$(n_1, n_2) = (n_1^{''}, n_2^{'})$	CCT $(\bullet)$	CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n2'}{2}}$	-CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n1'}{2}}$	-CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{4}+\frac{n1'}{2}+\frac{n2'}{2}}$
$(n_1, n_2) = (n_1^{''}, n_2^{''})$	CCT $(\bullet)$	-CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n2'}{2}}$	-CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{8}+\frac{n1'}{2}}$	CCT $(\bullet)$ $\times W_{N/2}^{\frac{1}{4}+\frac{n1'}{2}+\frac{n2'}{2}}$

Tab. 1: The relationship of the DCT operations for the sixteen partitioned regions where $(\bullet) \equiv (N/2 \times N/2)$.

In order to represent the 2-D CCT kernel into a matrix form, we encode the input data samples from a 2-D matrix to a 1-D vector using the row-major order, as shown in Eqn. (6).

$$X^{(0)} = \begin{array}{c|c} & n2 = 0 \cdots N-1 \\ \begin{array}{c} n1 \\ \Downarrow \\ 0 \\ \vdots \\ N-1 \end{array} & \begin{bmatrix} & & \\ & x(n1,n2) & \\ & & \end{bmatrix} \end{array} \rightarrow \underline{X}^{(0)} = \begin{array}{c} n1,n2 \\ \Downarrow \\ 0,0 \\ 0,1 \\ 0,2 \\ \vdots \\ N-1,N-1 \end{array} \begin{bmatrix} \\ x(n1,n2) \\ \\ \end{bmatrix} \qquad (6)$$

where $X^{(0)}$ denotes the original 2-D input data matrix and $\underline{X}^{(0)}$ represents the corresponding 1-D representation. Dividing the 2-D matrix $X^{(0)}$ into four parts according to the partitioning method in Tab. 1, we obtain another 2-D matrix $X^{(1)}$

$$X^{(1)} = \left[\begin{array}{c|c} Xp,p & Xp,r \\ \hline Xr,p & Xr,r \end{array}\right], \quad \begin{array}{l} Xp,p = x(n1:[0,N/2-1], n2:[0,N/2-1]) \\ Xp,r = x(n1:[0,N/2-1], n2:[N/2,N]) \\ Xr,p = x(n1:[N/2,N], n2:[0,N/2-1]) \\ Xr,r = x(n1:[N/2,N], n2:[N/2,N]) \end{array}$$

where Xp,p, Xp,r, Xr,p, and Xr,r are the partitioned $N/2 \times N/2$ sub-matrixes. A totally different 1-D vector representation of $\underline{X}^{(1)}$ is derived by transforming the four 2-D sub-matrixes Xp,p, Xp,r, Xr,p, and Xr,r into their respective 1-D vectors $\underline{Xp,p}$, $\underline{Xp,r}$, $\underline{Xr,p}$, and $\underline{Xr,r}$ using the similar 2-D-to-1-D representation as in Eqn. (6), and cascading them together into a 1-D vector.

By repeating $s = \log_2 N$ times the above partitioning for the progressively smaller 2-D submatrices, and converting them into the corresponding 1-D vector representations, we can generate the transformed 1-D vector $\underline{X}^{(s)} \equiv \tilde{X}$ either from the original 1-D vector $\underline{X}^{(0)}$ or from $\underline{X}^{(1)}$ as follows:

$$\tilde{X} = \underline{X}^{(s)} = PP(N \times N)\underline{X}^{(0)} = \begin{bmatrix} PP(\bullet) & & & \\ & PP(\bullet) & & \\ & & PP(\bullet) & \\ & & & PP(\bullet) \end{bmatrix} \underbrace{\begin{bmatrix} \underline{Xp,p} \\ \underline{Xp,r} \\ \underline{Xr,p} \\ \underline{Xr,r} \end{bmatrix}}_{\underline{X}^{(1)}} \qquad (7)$$

where the $N^2 \times N^2$ permutation matrix $PP(N \times N)$ denotes the operation of the overall transformation starting from the original 1-D vector $\underline{X}^{(0)}$ of size N^2, while the $N^2/4 \times N^2/4$ matrix $PP(N/2 \times N/2)$ represents the 2-D to 1-D transformation for the 1-D vector of size $N^2/4$, i.e., $\underline{X}_{p,p}, \underline{X}_{p,r}, \underline{X}_{r,p}$, or $\underline{X}_{r,r}$.

Similarly, the recursive even-odd partitioning for the frequency index and the corresponding 2-D to 1-D conversion can be represented as

$$\hat{Y} = \underline{Y}^{(s)} = RR(N \times N)\underline{X}^{(0)} = \begin{bmatrix} RR(\bullet) & & & \\ & RR(\bullet) & & \\ & & RR(\bullet) & \\ & & & RR(\bullet) \end{bmatrix} \underbrace{\begin{bmatrix} \underline{Ye,e} \\ \underline{Ye,o} \\ \underline{Yo,e} \\ \underline{Yo,o} \end{bmatrix}}_{\underline{Y}^{(1)}} \qquad (8)$$

where the $N^2 \times N^2$ permutation matrix $RR(N \times N)$ denotes the "*bit-reversed*" operation for the 1-D representation $\underline{Y}^{(0)}$ of the 2-D output data $Y^{(0)}$ using the even-odd partitioning illustrated in Tab. 1.

With the above definition of the 1-D vector representation, the relationship of the partitioned 2-D CCT operations in Tab. 1 can be expressed as:

$$\underbrace{\begin{bmatrix} \underline{Ye,e} \\ \underline{Ye,o} \\ \underline{Yo,e} \\ \underline{Yo,o} \end{bmatrix}}_{\underline{Y}^{(1)}} = \left[\begin{array}{cccc} T(\bullet) & T(\bullet) & T(\bullet) & T(\bullet) \\ \hdashline T(\bullet)C1(\bullet) & -T\bullet)C1(\bullet) & T(\bullet)C1(\bullet) & -T(\bullet)C1(\bullet) \\ \hdashline T(\bullet)C2(\bullet) & T(\bullet)C2(\bullet) & -T(\bullet)C2(\bullet) & -T(\bullet)C2(\bullet) \\ \hdashline T(\bullet)C3(\bullet) & -T(\bullet)C3(\bullet) & -T(\bullet)C3(\bullet) & T(\bullet)C3(\bullet) \end{array}\right] \underbrace{\begin{bmatrix} \underline{Xp,p} \\ \underline{Xp,r} \\ \underline{Xr,p} \\ \underline{Xr,r} \end{bmatrix}}_{\underline{X}^{(1)}} \qquad (9)$$

where $(\bullet)$ is the abbreviation for $N/2 \times N/2$. The three $N^2/4 \times N^2/4$ diagonal matrices are defined as

$$\begin{array}{l} C1(N/2 \times N/2) = diag(W_{N/2}^{n2^{'}/2}), \quad C2(N/2 \times N/2) = diag(W_{N/2}^{n1^{'}/2}) \\ C3(N/2 \times N/2) = diag(W_{N/2}^{(n1^{'}+n2^{'})/2}) \end{array} \qquad (10)$$

The $N^2/4 \times N^2/4$ coefficient matrix $T(N/2 \times N/2)$ represents the 2-D CCT of size $N/2 \times N/2$. Combining Eqns. (7), (8), (9), and observing that the inverse matrix $PP^{-1}(N \times N) = PP^T(N \times N)$, we can derive the recursive formula for the $N \times N$ 2-D CCT:

$$\begin{aligned} \hat{Y} &= \begin{bmatrix} RR(\bullet) & & & \\ & RR(\bullet) & & \\ & & RR(\bullet) & \\ & & & RR(\bullet) \end{bmatrix} \left[\begin{array}{cccc} T(\bullet) & T(\bullet) & T(\bullet) & T(\bullet) \\ \hdashline T(\bullet)C1(\bullet) & -T\bullet)C1(\bullet) & T(\bullet)C1(\bullet) & -T(\bullet)C1(\bullet) \\ \hdashline T(\bullet)C2(\bullet) & T(\bullet)C2(\bullet) & -T(\bullet)C2(\bullet) & -T(\bullet)C2(\bullet) \\ \hdashline T(\bullet)C3(\bullet) & -T(\bullet)C3(\bullet) & -T(\bullet)C3(\bullet) & T(\bullet)C3(\bullet) \end{array}\right] \\ &\qquad \begin{bmatrix} PP^T(\bullet) & & & \\ & PP^T(\bullet) & & \\ & & PP^T(\bullet) & \\ & & & PP^T(\bullet) \end{bmatrix} \tilde{X} \qquad (11) \\ &= \left[\begin{array}{cccc} \hat{T}(\bullet) & \hat{T}(\bullet) & \hat{T}(\bullet) & \hat{T}(\bullet) \\ \hdashline \hat{T}(\bullet)\hat{C1}(\bullet) & -\hat{T}(\bullet)\hat{C1}(\bullet) & \hat{T}(\bullet)\hat{C1}(\bullet) & -\hat{T}(\bullet)\hat{C1}(\bullet) \\ \hdashline \hat{T}(\bullet)\hat{C2}(\bullet) & \hat{T}(\bullet)\hat{C2}(\bullet) & -\hat{T}(\bullet)\hat{C2}(\bullet) & -\hat{T}(\bullet)\hat{C2}(\bullet) \\ \hdashline \hat{T}(\bullet)\hat{C3}(\bullet) & -\hat{T}(\bullet)\hat{C3}(\bullet) & -\hat{T}(\bullet)\hat{C3}(\bullet) & \hat{T}(\bullet)\hat{C3}(\bullet) \end{array}\right] \tilde{X} \\ &= \begin{bmatrix} \hat{T}(\bullet) & & & \\ & \hat{T}(\bullet) & & \\ & & \hat{T}(\bullet) & \\ & & & \hat{T}(\bullet) \end{bmatrix} \begin{bmatrix} I(\bullet) & & & \\ & \hat{C1}(\bullet) & & \\ & & \hat{C2}(\bullet) & \\ & & & \hat{C3}(\bullet) \end{bmatrix} \begin{bmatrix} I(\bullet) & I(\bullet) & I(\bullet) & I(\bullet) \\ I(\bullet) & -I(\bullet) & I(\bullet) & -I(\bullet) \\ I(\bullet) & I(\bullet) & -I(\bullet) & -I(\bullet) \\ I(\bullet) & -I(\bullet) & -I(\bullet) & I(\bullet) \end{bmatrix} \tilde{X}. \end{aligned}$$

where again $(\bullet)$ is the abbreviation for $N/2 \times N/2$. The $N^2/4 \times N^2/4$ coefficient matrix $\hat{T}(N/2 \times N/2)$ is defined as

$$\hat{T}(N/2\times N/2) \equiv RR(N/2\times N/2)T(N/2\times N/2)PP^{T}(N/2\times N/2)$$

which represents the 2-D CCT with input and output indices reordered. The three permuted diagonal matrices are defined as

$$\hat{C}_j(N/2\times N/2) = PP(N/2\times N/2)C_j(N/2\times N/2)PP^{T}(N/2\times N/2)$$

where $j = 1,2,3$. Based on Eqn. (11), we may express the recursive formula for the 2-D $(N\times N)$ CCT as

$$\hat{Y} = \hat{T}(N\times N)\,\tilde{X} = \begin{bmatrix} \hat{T}(\bullet) & & & \\ & \hat{T}(\bullet) & & \\ & & \hat{T}(\bullet) & \\ & & & \hat{T}(\bullet) \end{bmatrix} B(N\times N)\,\tilde{X} \tag{12}$$

where

$$B(N\times N) = \underbrace{\begin{bmatrix} I(\bullet) & & & \\ & \hat{C}_1(\bullet) & & \\ & & \hat{C}_2(\bullet) & \\ & & & \hat{C}_3(\bullet) \end{bmatrix}}_{BD(N\times N)} \underbrace{\begin{bmatrix} I(\bullet) & I(\bullet) & I(\bullet) & I(\bullet) \\ I(\bullet) & -I(\bullet) & I(\bullet) & -I(\bullet) \\ I(\bullet) & I(\bullet) & -I(\bullet) & -I(\bullet) \\ I(\bullet) & -I(\bullet) & -I(\bullet) & I(\bullet) \end{bmatrix}}_{BB(N\times N)}$$

The $N^2\times N^2$ matrix $B(N\times N) = BD(N\times N)BB(N\times N)$ looks like a radix-4 butterfly operation where $BD(N\times N)$ is a diagonal matrix and $BB(N\times N)$ is a band matrix with seven nonzero bands. In the next section, we will show that the operation of $BD(N\times N)$ can be mapped to a single multiplier while the operation of $BB(N\times N)$ can be realized by three adders only.

3. Systolic Architecture for 2-D CCT

We will use the example of the 4 x 4 2-D CCT to illustrate the mapping from the recursive algorithm of Eqn. (12) for the $N\times N$ 2-D CCT to the linear systolic architecture consisting of log_2N stages. By applying the recursion of Eq. (12) two times, we could express the 2-D CCT of size 4 x 4 as:

$$\hat{Y} = \underbrace{\begin{bmatrix} B(2\times2) & & & \\ & B(2\times2) & & \\ & & B(2\times2) & \\ & & & B(2\times2) \end{bmatrix}}_{Stage\ 2} \underbrace{B(4\times4)}_{Stage\ 1}\,\tilde{X} \tag{13}$$

where the two butterfly operations are

$$B(4\times4) = \underbrace{\begin{bmatrix} I(2\times2) & & & \\ & \hat{C}_1(2\times2) & & \\ & & \hat{C}_2(2\times2) & \\ & & & \hat{C}_3(2\times2) \end{bmatrix}}_{BD(4\times4)} \underbrace{\begin{bmatrix} I(2\times2) & I(2\times2) & I(2\times2) & I(2\times2) \\ I(2\times2) & -I(2\times2) & I(2\times2) & -I(2\times2) \\ I(2\times2) & I(2\times2) & -I(2\times2) & -I(2\times2) \\ I(2\times2) & -I(2\times2) & -I(2\times2) & I(2\times2) \end{bmatrix}}_{BB(4\times4)} \tag{14}$$

$$B(2\times2) = \underbrace{\begin{bmatrix} I(1\times1) & & & \\ & \hat{C}_1(1\times1) & & \\ & & \hat{C}_2((1\times1) & \\ & & & \hat{C}_3((1\times1) \end{bmatrix}}_{BD(2\times2)} \underbrace{\begin{bmatrix} I(1\times1) & I(1\times1) & I(1\times1) & I(1\times1) \\ I(1\times1) & -I(1\times1) & I(1\times1) & -I(1\times1) \\ I(1\times1) & I(1\times1) & -I(1\times1) & -I(1\times1) \\ I(1\times1) & -I(1\times1) & -I(1\times1) & I(1\times1) \end{bmatrix}}_{BB(2\times2)}$$

The 7-band matrices of $BB(4\times4)$ and $BB(2\times2)$ contains elements of only 1, -1 or 0. Thus, no multiplier is required in the band-matrix-vector multiplication.

Let us consider the realization of the DG for the first stage of the 2-D 4x4 CCT. i.e the multiplication of the 16 x 16 band matrix $BB(4\times4)$. As shown in Fig. 1, each of the three PEs, PE(1,1), PE(1,2), and PE(1,3), requires an additional multiplexer to select the direct input or the delayed input. The detailed operations for the control signals are included in Fig. 1. Due to the regularity (with period of 16 cycles), the control signal can be generated from the third and fourth bits of a counter.

The PE(1,0) needs the bit-inversion and the increment (the addition of the carry-in bit of one) when the control signal C(1,0) is -1. We could postpone the increment to the next PE, i.e., PE(1,1), if PE(1,1) has control signal of 1, as marked by 1'. However, when the control signals of both PE(1,0) and PE(1,1) are -1, the increment of PE(1,0) must be postponed further to PE(1,2) which is marked by C(1,2)=1'''. The control signal is also used to select either the left or the right input of the multiplexers. Since the control signal of 1 or -1 in each PE might imply different operations, we label them as 1, 1', 1'', 1''', or -1, -1'. The detailed operations of all the different control signals in each PE are included in Fig. 1.

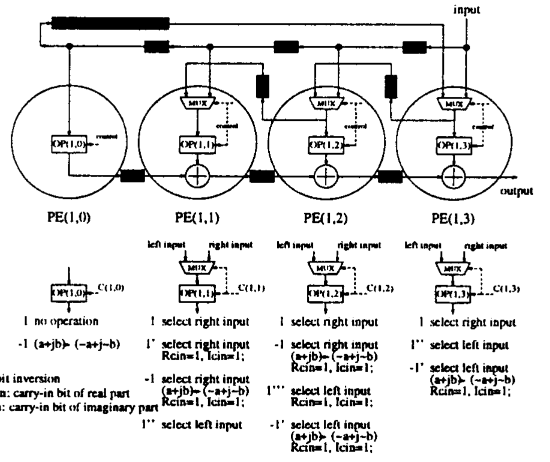
input
OP(1,0)
control
MUX
OP(1,1)
OP(1,2)
OP(1,3)
output
PE(1,0)
PE(1,1)
PE(1,2)
PE(1,3)
left input
right input
C(1,0)
C(1,1)
C(1,2)
C(1,3)
1 no operation
-1 (a+jb)⇒(~a+j~b)
~: bit inversion
Rcin: carry-in bit of real part
Icin: carry-in bit of imaginary part
1 select right input
1' select right input Rcin=1, Icin=1;
-1 select right input (a+jb)⇒(~a+j~b) Rcin=1, Icin=1;
1'' select left input
1 select right input
-1 select right input (a+jb)⇒(~a+j~b) Rcin=1, Icin=1;
1''' select left input Rcin=1, Icin=1;
-1' select left input (a+jb)⇒(~a+j~b) Rcin=1, Icin=1;
1 select right input
1'' select left input
-1' select left input (a+jb)⇒(~a+j~b) Rcin=1, Icin=1;

Fig. 1: The architecture and the controlling operations of all PEs for stage 1 of the 2-D 4x4 CCT.

The architecture for the second stage of the 4x4 2-D CCT is shown in Fig. 2 which includes the detailed operations for the control signals. Again the different control signals of 1, 1', 1'', 1''', and -1, -1' mark the difference of carry-in bit and the selection of the multiplexer input. The control signals repeat every four cycles and thus can be generated from the two least significant bits of a counter.

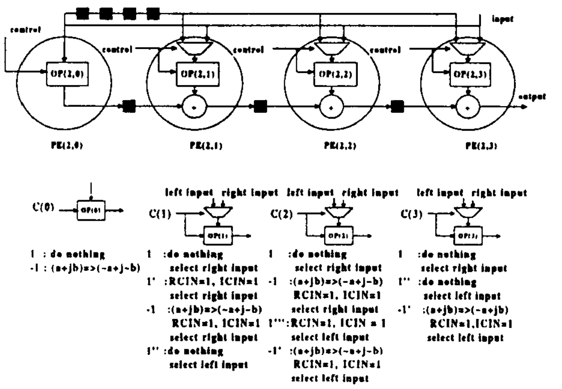
control
input
output
OP(2,0)
OP(2,1)
OP(2,2)
OP(2,3)
PE(2,0)
PE(2,1)
PE(2,2)
PE(2,3)
left input right input
C(0)
C(1)
C(2)
C(3)
OP(0)
OP(1)
OP(2)
OP(3)
1 : do nothing
-1 : (a+jb)=>(~a+j~b)
1 :do nothing select right input
1' :RCIN=1, ICIN=1 select right input
-1 :(a+jb)=>(~a+j~b) RCIN=1, ICIN=1 select right input
1'' :do nothing select left input
1 :do nothing select right input
-1 :(a+jb)=>(~a+j~b) RCIN=1, ICIN=1 select right input
1''':RCIN=1, ICIN = 1 select left input
-1' :(a+jb)=>(~a+j~b) RCIN=1, ICIN=1 select left input
1 :do nothing select right input
1'' :do nothing select left input
-1' :(a+jb)=>(~a+jb) RCIN=1,ICIN=1 select left input

Fig. 2: The architecture for stage 2 of the 2-D 4x4 CCT with the detailed operations for the control signals in each PE.

An overall architecture for the computation of the 4x4 2-D CCT is shown in Fig. 3 which includes two multipliers implementing the multiplication of $BD(4\times4)$ and $BD(2\times2)$ in Eqn. (14). In fact, we can save a lot of power by adding some control to the multipliers to bypass the input while the multiplied coefficients are 1, as can be observed in $BD(4\times4)$ and $BD(2\times2)$. The linear systolic architecture is easily scaleable for computation of general $N \times N$ 2-D CCT. Fig. 4 shows the systolic architecture for the stage i ($i \neq \log_2 N$) except for the last stage which is the same as Fig. 3. In general only $3\log_2 N$ complex adders and $\log_2 N$ complex multipliers are required for the computation of an $N \times N$ 2-D CCT.

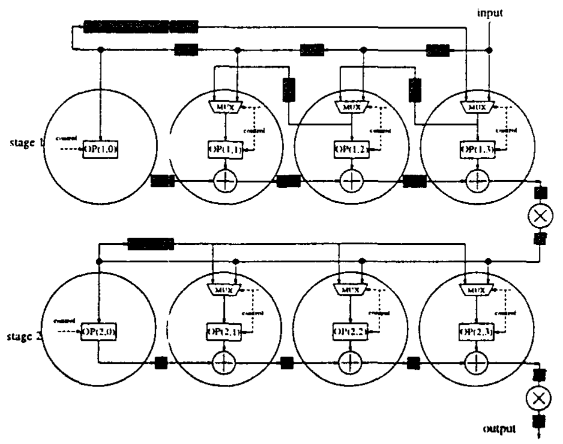

Fig. 3: The overall systolic architecture for computation of the 4x4 2-D CCT.

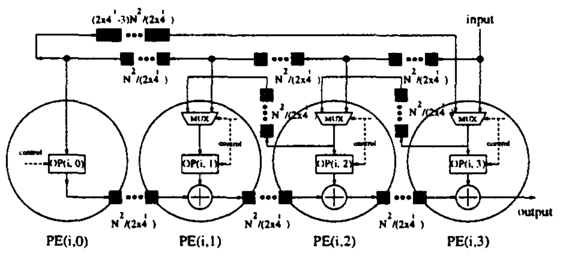

Fig. 4: A general architecture for the i-th ($i \neq \log_2 N$) stage of an $N \times N$ 2-D CCT

We have demonstrated a novel architecture for computing the 2-D $N \times N$ CCT using only *3* $log_2 N$ complex adders and $log_2 N$ complex multipliers. As mentioned in Sec. 2, two 2-D $N \times N$ DCTs can be generated simultaneously from the computed 2-D $N \times N$ CCT by appending a post-processing unit performing the simple post-addition in Eqn. (5). The computation complexity of our proposed 2-D DCT algorithm is low because we represent the 2-D DCT of size $N \times N$ into a 1-D DCT with size of N^2, and exploit fully the fast recursive algorithm for the 1-D DCT problem.

The time index transformation in Eqn. (7), characterized by the permutation matrix $PP(N \times N)$, can be combined with the input reordering of Eqn. (2), and be implemented by an input addressing ROM for fetching input data signals of correct order. Similarly, we can merge the frequency index transform characterized by the permutation matrix $RR(N \times N)$ in Eqn. (8) with the output reordering in Eqn. (5), and furthermore with the zigzag operation when applied to the standard video compression. Thus, all the output reordering operation can be realized by an output addressing ROM for generating computation results of desired order.

Tab. 2 compares our 8x8 DCT architecture with a recently propose one in [3]. The architecture in [3] fetches 16 input signals at a time, and generates 16 outputs simultaneous. Thus it requires a lot of input/output pins, making it difficult for VLSI implementation. Furthermore, the architecture contains a lot of global interconnection routing and several transpose memories for exchange of intermediate computing results. In contrast, our proposed method has very regular kernel architecture with local connection. It takes as input one complex word (two real words) and generates one complex word using arithmetic units of only 3 complex multipliers and 9 complex adders. The regularity and the reduced number of arithmetic operators in our architecture make it favorable in terms of hardware and power cost. Besides, due to the systolic structure, our architecture is easily pipelined and scalable to the 2-D DCT computation of any size without adding too much hardware cost.

8x8 DCT	Direct method [3]	Proposed method
multipliers	28 real multipliers	2 complex multipliers
adders	134 real adders	9 complex adders
transpose memory	required	not required
# of pins for inputs	16 words	2 words
# of pins for outputs	16 words	2 words
throughput	16 cycles/clock	2 cycles /clock
structure regularity	not very regular	very regular
wiring	global	local
scalability	difficult	easy

Tab. 2: Comparison of two 8x8 2-D DCT architectures.

4. Conclusion

This paper presents a new algorithm and the corresponding novel architecture for the computation of 2-D DCT. The algorithm converts the 2-D transform problem into the 1-D case, and utilizes the different time and frequency index partition methods to reduce the computation complexity. Furthermore, the common factors in the decomposed matrix during the algorithm derivation are factored out in order to save the number of multipliers. The derived linear systolic architecture is very regular and easily scalable to any order.

References:

[1] N. I. Cho, I. D. Yun, and S. U. Lee, "On Regular Structure for the Fast 2-D DCT Algorithm", IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 40, NO. 4. pp. 259-266, Apr. 1993.

[2] H. S. Hou, *"A fast recursive algorithm for computing the discrete cosine transform,"* IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, no. 10, pp. 1455-1461, Oct. 1987.

[3] Y. P. Lee, T. H. Chen, L. G. Chen, M. J. Chen, and C. W. Ku, *"A cost-effective architecture for 8x8 two-dimensional DCT/IDCT using direct method,"* IEEE Trans. Circuits and Systems for Video Technology, vol. 7, no. 3, pp. 459-467, June 1997.

[4] A. Madisetti and A. N. Willson, "A 100 MHz 2-D 8 x 8 DCT/IDCT Processor for HDTV Applications", IEEE Trans. Circuits and Systems for Video Technology, Vol. 5, No. 4, pp. 158-165, Apr. 1995.

[5] D. Stawecki and W. Li, *"DCT/IDCT Processor Design for High Data Rate Image Coding"*, IEEE Trans. Circuits and Systems for Video Technology, Vol. 2, No. 6, pp. 135-146, June 1992.

[6] S. Uramoto, *et al.*, "A 100-MHz 2-D Discrete Cosine Transform Core Processor", IEEE Journal of Solid-State Circuits, Vol. 27, No. 4, pp. 492-498, Apr. 1992.