# Unified Fully-Pipelined VLSI Implementations of Two-Dimensional Discrete Trigonometric Transforms *

Mon-Chau Shie[1], Wen-Hsien Fang[2]†, Ming-Lu Wu[2] and Feipei Lai[1]
[1]Dept. of Electrical Engineering & Dept. of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
[2] Dept. of Electronic Engineering, National Taiwan University of Science and Technology
Taipei, Taiwan
† Tel:886-2-27376412 Fax:886-2-27376424 Email:whf@et.ntust.edu.tw

## Abstract

This paper describes a unified VLSI architecture which can efficiently realize some popular two-dimensional discrete trigonometric transforms (2-D DTT). The computation of the 2-D DTT is based on the row-column decomposition approach. However, in contrast to previous schemes, efficient and unrestricted Clenshaw's recurrence formula along with the inherent symmetry of the trigonometric functions are adequately employed to render efficient recurrences for computing both of the row and column transforms. As such, the resulting VLSI architecture not only provides substantial hardware savings as compared with previous works, but it can also be applied to the 2-D DTT of *arbitrary* size. An input buffer along with a bidirectional circular shift matrix are addressed as well to enable the architecture to operate in a fully-pipelined manner.

## 1. Introduction

The 2-D discrete trigonometric transforms including the 2-D discrete Fourier transform (DFT), 2-D discrete Hartley transform (DHT), 2-D discrete sine transform (DST), and 2-D discrete cosine transform (DCT) have found applications in various facet of signal processing [1]. To facilitate real-time implementations, the development of efficient algorithms has been of great interest over the past few decades. Various fast algorithms have been advocated to alleviate the computational load involved. Despite their efficiency, most of these fast algorithms either place severe limitations on the input data size or require complex routing schemes [1, 2] which do not allow for efficient VLSI implementations.

*This work was supported by National Science Council of R.O.C. under contracts NSC 87-2213-E-011-014 and 87-2218-E-002-020.

In this paper, we address a new architecture which is free of the aforementioned drawbacks. The proposed approach is based on the row-column decomposition method. However, in contrast to previous schemes, efficient Clenshaw's recurrence formula [3] as well as the inherent symmetry of the trigonometric functions [4] have been adequately employed to yield succinct and highly structured recurrences for computing both of the row and column transforms of the 2-D DTT. These recurrences can, in particular, be efficiently realized by array structures as those of [4]. Consequently, the resulting VLSI architecture provides substantial hardware savings as compared with previous works. Since the developed recurrences are unrestricted, the corresponding structure is thus applicable to the 2-D DTT of *arbitrary* size.

Furthermore, a bidirectional circular shift matrix and an input buffer, which are utilized to carry out the transposition of the intermediate results and to rectify, if necessary, the input data in a correct order, respectively, are also introduced to maintain the pipelining of the whole architecture. In addition to the aforementioned advantageous features, the resulting VLSI architecture is highly regular, modular, and locally connected, thus lending it to efficient hardware implementations.

## 2. Recurrences for the Computation of 2-D DTT

In this section, we develop the required recurrences for computing the 2-D DTT. For a set of $N \times N$ data, $\{x(m,n); 0 \le m \le N-1, 0 \le n \le N-1\}$, the 2-D DTT are defined as

$$X_{DTT}(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m,n) ker_1(m,k) ker_2(n,l) \quad (1)$$

where $ker_1(m,k)ker_2(n,l)$ is the kernel function with $ker_1(m,k) = ker_2(m,k) = ker(m,k)$, and $ker(m,k)$ is equal to $e^{-j\frac{2\pi mk}{N}}$, $cas(\frac{2\pi mk}{N})$ ($cas(\cdot) \equiv \cos(\cdot) + \sin(\cdot)$), $\sin(\frac{(2m+1)k\pi}{2N})$, and $\cos(\frac{(2m+1)k\pi}{2N})$ for the 2-D DFT, 2-D DHT, 2-D DST, and 2-D DCT, respectively.

Before carrying out the row-column decomposition, some input data reordering schemes can be employed, which enable the derived recurrences to have a similar form for the aforementioned 2-D DTT so that a unified architecture is feasible. In addition, the symmetry of the trigonometric functions can be adequately exploited without inducing extra computations. More specifically, for the 2-D DST, after some manipulations the transform can be reexpressed as

$$X_S(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)\sin\frac{\pi(2m+1)k}{2N}\sin\frac{\pi(2n+1)l}{2N}$$

$$= \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \tilde{x}(m,n)\sin\frac{\pi(4m+1)k}{2N}\sin\frac{\pi(4n+1)l}{2N} \quad (2)$$

where

$$\tilde{x}(m,n) = \begin{cases} x(2m,2n) & (m,n)\in\mathcal{R}_1 \\ -x(2m,2N-2n-1) & (m,n)\in\mathcal{R}_2 \\ -x(2N-2m-1,2n) & (m,n)\in\mathcal{R}_3 \\ x(2N-2m-1,2N-2n-1) & (m,n)\in\mathcal{R}_4 \end{cases} \quad (3)$$

with $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$ corresponding to the indices of $\{0\le m\le\frac{N}{2}-1, 0\le n\le\frac{N}{2}-1\}$, $\{0\le m\le\frac{N}{2}-1, \frac{N}{2}\le n\le N-1\}$, $\{\frac{N}{2}\le m\le N-1, 0\le n\le\frac{N}{2}-1\}$, and $\{\frac{N}{2}\le m\le N-1, \frac{N}{2}\le n\le N-1\}$, respectively.

Similarly, the 2-D DCT can also be reexpressed as

$$X_C(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)\cos\frac{\pi(2m+1)k}{2N}\cos\frac{\pi(2n+1)l}{2N}$$

$$= \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \check{x}(m,n)\cos\frac{\pi(4m+1)k}{2N}\cos\frac{\pi(4n+1)l}{2N} \quad (4)$$

where

$$\check{x}(m,n) = \begin{cases} \tilde{x}(m,n) & (m,n)\in\mathcal{R}_1,\ \mathcal{R}_4 \\ -\tilde{x}(m,n) & (m,n)\in\mathcal{R}_2,\ \mathcal{R}_3 \end{cases} \quad (5)$$

Notice that the above data reordering scheme has also been employed in [5] for the computation of the 2D DCT. The approach of [5], however, is via the 2-D fast Fourier transform and thus involves complex arithmetic operations and complicated routings.

As such, the 2-D DTT can then be rewritten as

$$X_{DTT}(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} y(m,n)ker'(m,k)ker'(n,l)$$

$$= \sum_{n=0}^{N-1}\Big[\sum_{m=0}^{N-1} y(m,n)ker'(m,k)\Big]ker'(n,l) \quad (6)$$

where $y(m,n) = x(m,n)$ and $ker'(m,k) = ker(m,k)$ for both of the 2-D DFT and 2-D DHT, and $y(m,n) = \tilde{x}(m,n)$, $ker'(m,k) = \sin\frac{\pi(4m+1)k}{2N}$ and $y(m,n) = \check{x}(m,n)$, $ker'(m,k) = \cos\frac{\pi(4m+1)k}{2N}$ for the 2-D DST and 2-D DCT, respectively. Using Clenshaw's recurrence formula together with the symmetry of the trigonometric functions [4], the computation of both the row and column transform such as $X^{1D}_{DTT}(k) = \sum_{n=0}^{N-1} y(n)ker'(n,k)$, where $ker'(n,k)$ is the same as that of (6), the procedure can be summarized as follows:

$$X^{1D}_{DTT}(k) = \alpha\varphi_{N-1} + \beta\varphi_{N-2} \quad (7)$$

$$\text{and } X^{1D}_{DTT}(N-k) = \gamma\varphi_{N-1} + \lambda\varphi_{N-2} \quad (8)$$

where $\alpha = e^{j\omega_k}$, $\beta = \lambda = -1$, and $\gamma = e^{-j\omega_k}$ for the DFT, $\alpha = cas(-\omega_k)$, $\beta = \lambda = -1$, and $\gamma = cas(\omega_k)$ for the DHT, $\alpha = -\sin(\frac{3}{4}\omega_k)$, $\beta = -\sin(\frac{1}{4}\omega_k)$, $\gamma = \cos(\frac{3}{4}\omega_k)$ and $\lambda = -\cos(\frac{1}{4}\omega_k)$ for the DST, and $\alpha = \cos(\frac{3}{4}\omega_k)$, $\beta = -\cos(\frac{1}{4}\omega_k)$, $\gamma = -\sin(\frac{3}{4}\omega_k)$ and $\lambda = -\sin(\frac{1}{4}\omega_k)$ for the DCT, with $\omega_k = \frac{2\pi k}{N}$. The auxiliary variable $\varphi_n$ involved in the above recurrences satisfies the following recurrence

$$\varphi_n = 2\cos(\omega_k)\varphi_{n-1} - \varphi_{n-2} + y(n),\ n = 0,\cdots,N-1 \quad (9)$$

## 3. A Unified VLSI Architecture to Implement the 2-D DTT

This section addresses a unified VLSI architecture to implement the recurrences of the 2-D DTT developed above. The overall architecture, as depicted in Fig. 1, contains an input buffer (only required for the 2-D DCT/DST), two array processors for the row and column transforms, and a bidirectional circular shift matrix.

From above, we can find that both the row and column transforms of these 2-D DTT can be ingenuously realized by the same processor as addressed in [4], which, as shown in Fig. 2, consists of a cascade of $\frac{N}{2}+1$ preprocessing elements (PREs) and a postprocessing element (POE). The PRE is to compute the recurrence of (9) and is the same for all types of 2-D DTT, whereas the POE is to carry out the transformations of (7) and (8), and is different for each specific 2-D DTT. Consequently, the same architecture

(except some minor modifications of POEs) is applicable to all 2-D DTT and thus the overall design cost can be reduced.

In addition, a bidirectional circular shift matrix and an input buffer are also required to maintain the pipelining of the whole architecture. First, note that a transposition operation is required for the intermediate results between the row and column transforms. The traditional transposition circuit as that of [6] can not finish it in real time, thus destroying the overall pipelining. To overcome this, we address a new bidirectional circular shift matrix, as shown in Fig. 3 with $N = 8$, which switches the data flow direction between the horizontal and vertical directions every $N^2$ time steps. This implies that the intermediate results stored in the matrix rowwise (columnwise) will pump out columnwise (rowwise) and thus eliminate any extra delay between the two transforms.

Also, we can note that the input data needs to be shuffled before performing the 2-D DST and 2-D DCT. For this, we address an input buffer which, as shown in Fig. 4 with $N = 8$, is composed of a circular shift matrix and three arrays of shift registers. It performs the data permutation schemes of (3) and (5) by splitting the 2-D data into even-indexed and odd-indexed components in both the row and column directions. The circular shift matrix consists of $\frac{N^2}{2}$ shift registers. In the initial stage, after $\frac{N^2}{2}$ intermediate results feed in, the multiplexer followed sequentially selects from the $I_1^{\frac{N}{2}}$ to the $I_1^1$ arrays, and then it is back from the $I_1^1$ to the $I_1^{\frac{N}{2}}$ arrays in the reverse order, with $N$ consecutive time steps for each selected row. This order repeats every $N^2$ time steps so that the 2-D intermediate data can be split into even-indexed and odd-indexed rows. The row selected by the multiplexer will then feed the data sequentially to $I_2^E$ or $I_2^O$ while at the same time $I_1^1$ is ready to receive the next input data without any delay. The data fed into $I_2^E$ or $I_2^O$ (each of which consists of $\frac{N}{2}$ shift registers) alternates as depicted in [4] to split each coming row in into even-indexed and odd-indexed column components. After $N$ time steps, the contents in $I_2^E$ and $I_2^O$ are then transferred simultaneously to the adjacent array of shift registers $I_3$ and then pumped out sequentially. The parameters $m_1$ and $m_2$ are chosen as $m_1 = 1$ and $m_2 = -1$ in the first $\frac{N^2}{2}$ time steps and then $m_1 = -1$ and $m_2 = 1$ in the next $\frac{N^2}{2}$ time steps for the 2-D DST, whereas $m_1 = 1$ and $m_2 = 1$ for the 2-D DCT at any time steps. As such, the data reordering schemes of (5) and (3) can be done in real-time and thus allows for continuous movement of data.

## 4. Comparison and Discussion

In this section, we compare the proposed architecture with some previous works for computing the 2-D DCT. The comparison with other 2-D DTT yields similar results.

As we can observe from Table 1, the proposed architecture requires the minimal number of multipliers. On the other hand, the implementations of [2], despite offering higher throughput, call for lots of multipliers, more complex control schemes, and global routings. The I/O style involved is single-in-single-out (SISO), as contrasted to the serial-in-parallel-out (SIPO) of [7] and parallel-in-parallel-out (PIPO) of [2]. The data transmission is simple and does not require any global communication and broadcasting. After an initial delay, the transformed results will come out every $N^2$ time steps thereafter, i.e. throughput=$\frac{1}{N^2}$, due to the pipelining of the whole architecture. As a whole, the proposed architecture is modular, versatile, locally-connected, and fully-pipelined, thus offering a viable alternative to other schemes for efficient VLSI implementations.

# References


[1] K.R. Rao and R. Yip, *Discrete Cosine Transform,* Academic Press, San Diego, CA, 1990.

[2] N. I. Cho and S. U. Lee, "Fast algorithms and implementations of 2-D discrete cosine transform," *IEEE Trans. Circuits and Sys.*, vol. 38, pp. 297-305, 1991.

[3] W. Press. S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing,* Cambridge, UK: Cambridge University Press, 1992, 2nd ed.

[4] W.-H. Fang and M.-L. Wu, "An efficient unified systolic architecture for the computation of discrete trigonometric transforms," *in Proc. Int'l Symp. Circuits and Sys.*, pp. 1141-1144, 1997.

[5] J. Markhoul, "A fast cosine transform in both one and two dimensions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 29, pp. 27-34, 1980.

[6] M.-T. Sun *et al.*, "VLSI implementation of a 16 ×16 discrete cosine transform," *IEEE Trans. Circuits and Sys.*, vol. 36, pp. 610-616, 1989.

[7] K. J. R. Liu *et al.* "Optimal unified architectures for the real-time computation of time-recursive discrete sinusoidal transforms," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 4, pp. 168-180, April. 1994.

[8] U. Totzek and F. Matthiesen, "Two-dimensional discrete cosine transform with linear arrays," in *Proc. Int. Conf. Systolic Arrays*, J. McCanny *et al.* ed., Hertfordshire: Prentice-Hall, pp. 388-397, 1989.

| | [2] | [7] | proposed |
|---|---|---|---|
| Number of multipliers | $\frac{N^2}{2}log_2N$ | 8N | N+6 |
| Throughput | 1 | $\frac{1}{N^2}$ | $\frac{1}{N^2}$ |
| Interconnection | global | local | local |
| Broadcasting of I/O | yes | yes | no |
| I/O style | PIPO | SIPO | SISO |
| Continuous data flow | no | yes | yes |

Table 1: Comparison of various architectures for computing the 2-D DCT of size $N \times N$.

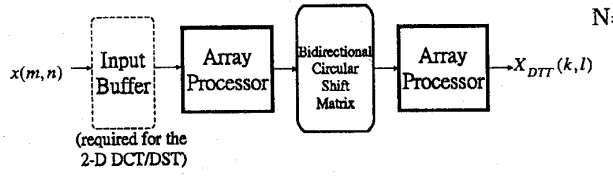


Figure 1: The overall architecture.

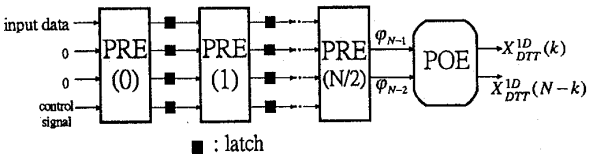


Figure 2: The structure of the array processor.

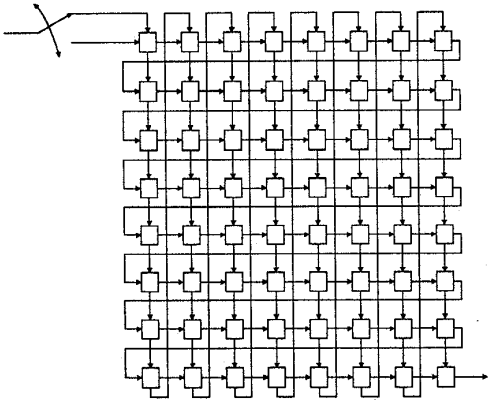

Figure 3: The bidirectional circular shift matrix with N=8.

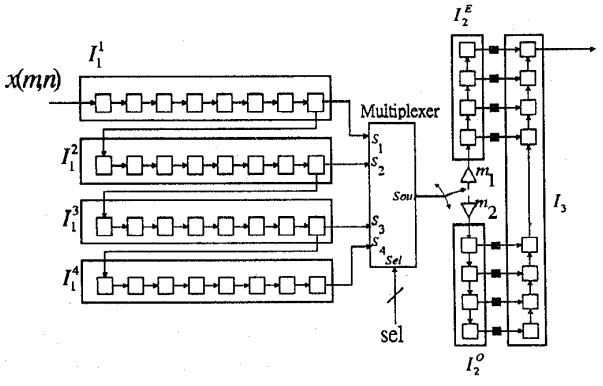


Figure 4: The input buffer with N=8.