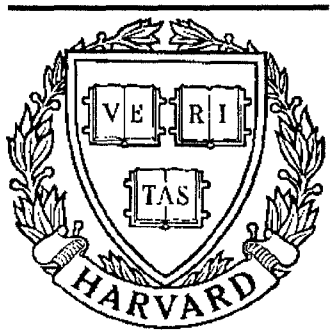
THESIS REPORT
Ph.D.

S Y S T E M S
R E S E A R C H
C E N T E R



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

VLSI Algorithms and Architectures for Time-Recursive Discrete Sinuoidal Transforms with Applications to Real-Time Video Communications

by C-T. Chiu
Advisor: K.J.R. Liu

Ph.D. 92-17

Abstract

Title of Dissertation: VLSI Algorithms and Architectures for Time-Recursive Discrete Sinuoidal Transforms with Applications to Real-Time Video Communications

Ching-Te Chiu, Doctor of Philosophy, 1992

Dissertation directed by: Assistant Professor K. J. Ray Liu
Department of Electrical Engineering

In this dissertation, we address the problem of developing efficient VLSI algorithms and architectures for discrete sinusoidal transforms in real-time applications for video communication systems. The major difficulty of this problem is that the resulting architectures should compute a huge amount of data at very high speed for real-time video applications and match the requirement of VLSI architectures, regularity, modularity and locality. In traditional FFT based algorithms, the serial data is buffered and then transformed using the FFT scheme.

We propose a "time-recursive" approach to perform transforms that merge the buffering and transform operations into a single unit. The transformed data are updated according to a recursive formula, whenever a new datum arrives. Therefore the waiting time is completely eliminated. The unified lattice and IIR

architectures for time-recursive transforms are proposed. The resulting architectures are regular, modular, and have only local interconnections and are better suited for VLSI implementations. There is no limitation on the transform size N and the number of multipliers required for computing the DCT by lattice and IIR structures are $6N-8$ and $2N-2$ respectively. In the case of dual generation of the DCT and DST by IIR structure, only $1.5N$ multipliers are required for each transform on average. The throughput of this scheme is one input sample per clock cycle.

We also apply the time-recursive approach to multidimensional separable transforms. The resulting d-dimensional structures are fully-pipelined and consist of only d 1-D transform arrays and shift registers for computing a d-D DXT. The delay time due to transpositions of the conventional d-D transforms is eliminated in our approach. It is shown that the architecture is optimal in the sense that the number of the multipliers used is minimum and both speed and area are asymptotically optimal.

The VLSI implementation of the lattice module based on the distributed arithmetic is also described. The chip can dually generate the DCT and DST simultaneously. It has been fabricated under $2\mu m$ double-metal CMOS technology and tested to be fully functional with a throughput rate 14.5-MHz and a data processing rate of 116Mb/s.

VLSI Algorithms and Architectures for Time-Recursive Discrete Sinuoidal Transforms with Applications to Real-Time Video Communications

by

Ching-Te Chiu

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1992

Advisory Committee:

Assistant Professor K. J. Ray Liu, Chairman/Advisor
Professor Joseph F. JáJá
Professor Ramalingam Chellappa
Professor Robert Newcomb
Associate Professor Clyde P. Kruskal

Dedication

To MY DEAR HUSBAND, CHIEN-JEN(SIMON) CHEN

for his support, consideration and infinite love.

To MY LOVELY SON, MARK(SHIN-HANN) CHEN

for being the enjoyable source of my life.

Acknowledgements

First and foremost, I would like to express my gratitude to my advisor, K. J. Ray Liu for his guidance, sustained encouragement and support throughout my graduate study. His brilliant problem-shooting skills and dedication to work will always be an inspiration to me.

I would like to thank Prof. Joseph Ja'Ja' for his valuable discussions and comments on my thesis. I would also like to thank the other members of my thesis committee: Profs. Rama Chellappa, Robert Newcomb, and Clyde Kruskal for their insightful comments and suggestions and thank Profs. Yung Ju Chen and Neil Goldsman for helping me in various ways during my study in Maryland.

I would like to thank the Systems Research Center, Electrical Engineering Department, and the National Science Fundation under the grant ECD-8803012 for giving me the financial support and providing excellent resources and study environment.

I am grateful to Ravi Kolagotla for collaborating the DCT/DST projects and also to many friends here for their help in using computer facilities: Ying-Min Huang, Jenn-Sen Leu, Hung-WeinChiou, Nol Ranaand, Shu-Suu Yu, Po-Yang F. Lin, Nam Phamdo, Vishnuuss Srinivasan, Xiaonong Ran, Guu-Chang Yang, and An-Yeu Wu.

I would like to thank my family: my husband, my son, my parents, and my parents-in-law, for their spiritual support and encouragement throughout my study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The field of signal processing has developed dramatically over the last several decades owing to applications in such diverse fileds as speech, image, and video communication, biomedical engineering, acoustics, sonar, radar, seismology, consumer electronics, and many others. Discrete sinusoidal transforms such as discrete cosine transform (DCT), discrete sine transform (DST), and discrete Hartley transform (DHT), discrete Fourier transform (DFT), Lopped Orthogonal Transform (LOT), and Complex Lapped Transform (CLT) are powerful tools in many applications of signal processing. Due to the advances in ISDN network and high definition television (HDTV) technology, high speed transmission and processing of speech, image and video signals become very desirable. Therefore, many computational tasks involved with digital signal transmission and processing require real-time operations. Real-time operation means the speed of the computational tasks can match the signal sampling or transmission rate. Clearly the high computational rates required in HDTV systems cannot be achieved by general-purpose parallel computers because of severe system overheads. The only way to meet the high computational rates of real-time signal processing is

by developing special-purpose architectures which exploit the regularity, recursiveness, and locality of the signal processing algorithms. In this dissertation, we focus on developing real time VLSI algorithms and architectures for discrete sinusoidal transforms for video applications.

The organization of the rest of this Chapter is as follows. The applications and techniques involved in video communication are described in Section 1.1. We describe the significances and applications of various transform coding schemes in Section 1.2 and review the evolution of the algorithms and architectures of these transforms in Section 1.3. The motivation for using time-recursive approach for achieving real time computation of these discrete transforms is given in Section 1.4. In Section 1.5, we give an overview of the results that we use from the time-recursive concept to handle the following signal processing tasks: unified one-dimensional lattice structures for discrete sinusoidal transform, two-dimensional DCT lattice structures with application to HDTV systems, optimal unified architectures for discrete sinusoidal transform, and VLSI implementation of the dual generated DCT and DST lattice structure. We conclude with the dissertation organization in Section 1.6.

1.1 Standards on the video coding systems

With the advances in technologies such as video compression, telecommunication, consumer electronics, the era of digital video has arrived. This new technology accelerates the availability of video applications such as digital laser-disc, electronic camera, videophone, videoconferencing, image and interactive video tools on computers, HDTV, and multimedia systems.

Unlike the digital audio technology of the past few decades, the data involved with still or motion pictures are so huge that data compression is inevitable. Compression methods are based on the nonlinearity of human vision which is more sensitive to energy with lower spatial frequency. Hence pictures can be lossly encoded with much less data than the original image without significantly decreasing the quality of the reconstructed image. For still images, data compression exploits correlation in space and for video signals, in both space and time. It is hard to distinguish a reconstructed image that was encoded with a 20:1 compression ratio from the original. Video data, even after compression at ratios at 100:1, can be decompressed with close to analog videotape quality. Among many transforms, the DCT is most widely used in speech and image processing for data compression. This is due to its better energy compaction property and its near optimal performance which is closest to that of the Karhunen-Loeve Transform (KLT) among many discrete transforms for highly correlated signals, especially for the first order Markov process [22, 1, 4].

When we develop high data compression schemes to reduce transmission/storage capacity, we also require sophisticated picture coding technology to integrate the whole system performance. In order to make the signals in different systems be compatible, standards for picture coding are strongly required. Three digital video standards that have been proposed are the Joint Photographics Experts Group (JPEG) standard for still picture compression; the Consultative Committee on International Telephony and Telegraphy (CCITT) Recommendation H.261 for video teleconferencing; and the Moving Pictures Experts Group (MPEG) for full-motion compression on digital storage media [31, 94, 93, 91].

The JPEG baseline algorithm is based on the transform coding approach.

The source image is divided into non-overlapping blocks of 8×8 pixels which is then transformed using the 8×8 two-dimensional DCT. The resulting 2-D DCT coefficients represent the frequency content of the given block where most of the energy concentrate near the zero-frequency or direct current term. Next, the DCT coefficients are quantized. Following quantization, the coefficients are zigzag scanned to arrange in the order of ascending frequency. Then, the dc and low frequency coefficients are encoded by using Huffman-style coding schemes. There is another DCT-based JPEG algorithm called the extended system which provides higher compression performance through arithmetic coding. The third mode of JPEG coding is the independent function which untilizes a 2-D Differential Pulse Code Modulation technique. The DCT-based algorithms can achieve higher compression ratio but are lossy. The spatial prediction algorithm has lower compression performance compared with the DCT-based algorithm[31].

The CCITT Recommendation H. 261 specifies a method of communication for videoconferencing and videophone [31] It is also known as the p*64 standard because the data rate on the communication channel is p times 64-kb/s, where p is a positive integer in the range 1 to 30. For p=1, then low-quality video signal for use in picture phones can be transmitted over a 64-kb/s line. If p=30, a high quality video signal for teleconferencing can be transmitted over a 2-Mb/s line. The CCITT H.261 encoder is a hybrid coder which combines motion compensated interframe prediction with the DCT.

The MPEG standard is designed for motion picture coding for digital storage media whose data throughput rate is 1.5Mb/s. A hybrid coding scheme known as, the motion compensated interframe prediction and DCT, is also used in MPEG. The prediction scheme not only predicts from the past but also from

the future. There are three function associated with prediction: forward motion compensation, backward motion compensation, and interpolative motion compensation.

From the discussion of these standards mentioned above, we observe that the DCT is a very important technique in video signal processing. Due to the high data rate in the video communication systems, special-purpose chip sets are required to perform real-time computation and match the computation speed. For example, the HDTV system proposed by General Instrument Corporation require a video data rate at 14.38Mb/s[30]. In this dissertation, we propose a promising DCT architecture which can achieve the high speed requirement of the HDTV systems.

1.2 Significances and applications of various discrete sinusoidal transforms

The DCT is the most popular transform coding used in data compression. Other discrete sinusoidal transforms, like the DST, DHT, DFT, LOT, and CLT are also very effective in many signal processing applications.

Jain in [5] show that the performance of the DST approaches that of the KLT for a first-order Markov sequence with given boundary conditions, especially for signal with low correlation coefficients [5, 6]. Rose, Heiman, and Dinstein proposed a new image coding method for low bit rates which is based on alternate use of the DCT and DST on image blocks. This procedure achieves the removal of redundancies in the correlation between neighboring blocks, as well as the preservation of continuity across the block boundaries.

In 1983, Bracewell introduced the DHT [2] which uses a transform kernel similar to that of the discrete Fourier transform (DFT), except that it is a real-valued transform. Therefore, it is simpler than the DFT with respect to the computational complexity [7]. Like the DCT and DST, the DHT has found many applications in signal and image processing [2, 3, 53, 57].

The lapped orthogonal transforms introduced by Cassereau, Staelin, and Jager with the basis functions upon which the signal is projected are overlapped for adjacent blocks. The LOT can reduce the artifacts near block boundaries which are generated by traditional block transform for low bit rate coding [76]. The CLT introduced by Young and have good performance in the application of the motion estimation[62].

1.3 Evolution of the algorithms and architectures

Since the introduction of DCT, many algorithms have been proposed to improve the computation speed and to reduce the hardware complexity. These algorithms can be classified into the following categories: (1) indirect computation, (2) matrix factorization, (3) recursive computation, and (4) systolic structure implementation. The indirect computation [8, 9, 10, 11, 12] applies the existing fast algorithms in the DFT or the Walsh-Hadamard transform to the DCT. It is not particularly efficient because the inherent properties of the DCT are not exploited. The matrix factorization [13, 15, 85, 55] decomposes the DCT into multiplications of many sparse matrices, therefore the numbers of multiplications and additions can be substantially reduced. The recursive computations

[45, 7] calculate higher-order DCT coefficients from lower-order ones, but their signal flow architectures need global communication which is not suitable for VLSI implementation. By using the recursive properties effectively, this kind of DCT algorithms has fewer multipliers and adders, while additional multiplexers are required. As for the systolic structure implementation [46, 69, 56], it uses existing systolic architectures for the DFT or other transforms to implement the DCT in a systolic manner. But some of the methods require that the number of samples of the signal must be decomposed into mutually prime numbers. Like the DCT, many fast algorithms have been proposed to improve the performance of the DST, DFT, and DHT [3, 48, 49, 5, 6]. Basically, they can be classified into the same ways as those of the DCT and similar advantages and disadvantages can also been found.

1.4 Motivation

In this section we first briefly describe the requirements of special-purpose architectures for real-time signal processing [77, 78].

1. **Simplicity, modularity, and regularity of design:** This is an important factor in VLSI design because this will greatly reduce the design time and cost.
2. **Parallel and pipelined processing:** The degrees of parallelism and pipelined structures determine the concurrency and throughput of the system.
3. **Communication:** Local and regular communication for data flow determine the cost and efficiency of VLSI implementation.

Fast and efficient algorithms to implement transform coding schemes have been of interest for the past decade. Most of the algorithms proposed are focused on reducing the computational time and hardware complexity by assuming that all the input signals are available at the same time. However, in the high speed image system such as HDTV, digitized images are available in a sequential or stream fashion. Waiting for data to become ready will slow down these algorithms. Moreover, the architectures of these algorithms require global communication, that is, they need more wire connections, which increase the complexity of the circuitry and reduce the system performance.

In real-time signal processing applications, especially in speech/image communications and radar/sonar signal processing, input data arrive serially. In traditional fast algorithms (such as the FFT), the serial data is buffered and then transformed using the FFT scheme of complexity $O(N \log N)$[60]. Buffering the serial data requires $O(N)$ time. The goal of this dissertation is to study a novel architecture that merges the buffering and transform operations into a single unit of total hardware complexity $O(N)$. Unlike the FFT, this architecture has only local interconnections and is better suited for VLSI implementations. It is important to note that the proposed architectures generate time-recursive transforms, not just block transforms, *i.e.,* the transform of the N points $[x(t+1), x(t+2), \ldots, x(t+N)]$ is generated one clock cycle after the transform of $[x(t), x(t+1), \ldots, x(t+N-1)]$ is generated. To generate time-recursive transforms, the traditional fast algorithms based architectures require $O(logN)$ time using $O(N \log N)$ hardware, while the architectures we propose require only a constant time with $O(N)$ hardware. Time-recursive transforms are currently gaining widespread use in motion estimation, video signal processing,

and in reducing blocking effects in data compression.

1.5 Main contribution

We propose a "time-recursive" approach to perform transform coding on a real time basis. The transformed data are updated according to a recursive formula, whenever new data arrive. Therefore the waiting time required for other algorithms is completely eliminated. Based on this new idea, several significant results are developed and are summarized as follows.

This is the first unified algorithm proposed that can be used to compute all the discrete sinusoidal transforms. We also discover the fundmental dual generation properties between these transforms.

A new unified parallel lattice architecture for the DCT/DST/DHT/DFT/LOT/CLT that are useful in image processing is derived. Here "unified" architecture means that different transforms can be computed using the same structure. We reduce the number of multipliers from $\frac{N}{2} * \ln N$ to $6N - 8$ for 1-D DCT. Moreover, the resulting architectures are regular, modular, locally-connected and suitable for VLSI implementation. From the speed point of view, this unified architecture can obtain the the transform results immediately whenever a new datum arrives. Therefore, the system throughput rate is highly increased and is better than others for matching the high speed requirement of video communication systems. The other unique contribution of this architecture is that there is no constraint on the size of the image.

Data processed in image and video signal processing are two-dimensional(2-D) information. The drawback of the conventional 2-D transforms is the delay

time due to a operation called "transposition". We derive a new time-recursive parallel 2-D DCT structure which can eliminate the transposition time. The system is fully-pipelined with throughput rate N clock cycles for $N \times N$ successive input data frame. The conventional 2-D DCT systolic array's throughput rate is $2N + 1$ clock cycles. The basic building block of this architecture is the unified architecture just described, therefore it presevers all the advantages mentioned above.

Although the number of multipliers of our lattice structure is a linear function of N, while that of the others is $O(N \ln(N))$. The number of multiplier of the lattice structure is larger than others when N is small. We further derive a direct IIR structure that can reduce the number of multipliers of 1-D DCT from $6N-8$ to $2N-2$. In the time-recursive lattice architecture, two transforms called the dual generated pairs, are obtained simultaneously. The unified direct IIR structure is more suitable for applications where only one transform is required. We also provide a theoretical justification for the fact that any discrete transform whose basis functions satisfy the *Fundamental Recurrence Formula* has a second-order autoregressive structure in its filter realization. We also demonstrate that dual generation transform pairs share the same autoregressive structure. We extend these time-recursive concepts to multi-dimensional transforms. The resulting d-dimensional structures are fully-pipelined and consist of only d 1-D transform arrays and shift registers.

In addition to theoretical derivations, the implementation of these algorithms into workable VLSI chips is also an important part of this thesis. The VLSI implementation of the lattice module based on the distributed arithmetic is also described. This is the first chip that can dually generate the DCT and

DST simultaneously. It has been fabricated using $2\mu m$ double-metal CMOS technology and has been tested to be fully functional with a throughput rate 14.5-MHz and a data processing rate of 116Mb/s.

1.6 Organization

In this thesis we present unified parallel algorithms and architectures for real time computation of the time-recursive discrete sinusoidal transforms.

In Chapter 2 we develop unified parallel lattice structures that can dually generate the DCT and DST simultaneously as well as the DHT are developed. This structure can compute the DCT and DST simultaneously and immediately whenever the input data arrives. The architecture is regular, modular, and without global communication. Besides, there is no limitation on N and the total number of multipliers is a linear function of N. This makes it very suitable for VLSI implementation and real time video signal processing.

In Chapter 3 we develop a new real-time parallel 2-D DCT lattice structure which is fully-pipelined with throughput rate N clock cycles for $N \times N$ successive input data frame. The 2-D DCT architecture is module, regular, and requires only two 1-D DCT blocks which can be extended directly from the 1-D DCT and and no transposition is required.

In Chapter 4 an optimal unified architecture that can efficiently compute the DCT, DST, Hartley, Fourier, Lapped Orthogonal, and Complex Lapped transforms for a continuous input data stream is proposed. This structure uses only half as many multipliers as the lattice structure.

In Chapter 5 the VLSI implementation of the lattice module is described. The

chip has been fabricated using MOSIS facilities under $2\mu m$ CMOS technology and the chip can be operated at 14.5MHz.

In Chapter 6 we summarize the results obtained in this dissertation and suggest some directions for future research in this area.

Chapter 2

Parallel Lattice Structures

The problems of unified efficient computations of the DCT, DST, DHT, and their inverse transforms are considered. In particular, a new scheme employing the time-recursive approach to compute these transforms is presented. Using such an approach, unified parallel lattice structures that can dually generate the DCT and DST simultaneously as well as the DHT are developed. It is also shown that the DCT, DST, DHT and their inverse transforms share an almost identical lattice structure. The lattice structures can also be formulated into pre-lattice and post-lattice realizations. Two methods, the $SISO$ and double-lattice approaches, are developed to reduce the number of multipliers in the parallel lattice structure by $2N$ and N respectively. The trade-off between time and area for the block data processing is also considered. The concept of filter bank interpretation of the time-recursive sinusoidal transforms is also discussed.

2.1 Introduction

Transform coding has found lots of applications in image, speech, and digital signal transmission and processing. Due to the advances in ISDN network and HDTV technology, high speed transmission of digital video signal becomes very desirable. Among many transforms, the discrete cosine transform (DCT), discrete sine transform (DST), and discrete Hartley transform (DHT) are very effective in transform coding applications to digital signals such as speech and image signals.

In this Chapter, we propose unified time-recursive lattice structures that can be used for the discrete orthogonal transforms mentioned above, *i.e.*, the DCT, DST, and DHT. We consider the orthogonal transforms from a time-recursive point of view instead of the whole block of data. We do so because in digital signal transmission, data arrive serially. Also, many operations such as filtering and coding are done in a time-recursive way. Based on this approach, the resulting architectures are almost identical for the DCT, DST, and DHT, and their inverses. Our structures decouple the transformed data components, hence, there is no global communication needed. Besides, the number of multipliers in these structures is a linear function of N, so they require fewer multipliers than most other algorithms when N is large. Therefore, our architectures are very suitable for VLSI implementation. One of the important characteristics of these structures is that the transform size N can be any integer, which is not the case for most of the fast algorithms for discrete transforms which do have certain constraints on N. Another important result is that based on the time-recursive approach, the dual generation properties of the DCT, DST, and DHT, as well as some related inverse transforms, can be obtained.

The rest of the Chapter is organized as follows. In Section 2.2, the dual generation of lattice structures for the DCT and DST with the time-recursive approach is considered. The inverse discrete cosine transform (IDCT) and inverse discrete sine transform (IDST) based on the lattice structures are discussed in Section 2.3. In Section 2.4, the time-recursive lattice structure for the DHT is presented. All the above time-recursive properties are derived by updating the time index by one. With block data processing, the time index is updated by more than one. The detailed effects and results of block data processing are discussed in Section 2.5. Denormalized methods to reduce the number of multipliers in those lattice structures are considered in Section 2.6. Then we compare these kinds of lattice structures with other architectures in terms of the number of multipliers and adders in Section 2.7. The synthesis bank structures based on the time-recursive concept is discussed in Section 2.8. Finally, we give the conclusion in Section 2.9.

2.2 Dual Generation of DCT and DST

We will show an efficient implementation of the DCT from the time-recursive point of view as an alternative to find fast algorithms through matrix factorizations or convert the DCT to DFT, which can be implemented on various existing architectures. Focusing on the sequence instead of the block of input data, we can obtain not only the time-recursive relation between the DCT of two successive data sequences, but also a fundamental relation between DCT and DST. In the following, the time-recursive relation for the DCT will be considered first.

2.2.1 Time-Recursive Discrete Cosine Transform

The one-dimensional (1-D) DCT of a sequential input data starting from $x(t)$ and ending with $x(t+N-1)$ is defined as

$$X_c(k,t) = \frac{2C(k)}{N} \sum_{n=t}^{t+N-1} x(n) \cos\left[\frac{\pi[2(n-t)+1]k}{2N}\right]$$
$$k = 0, 1, ..., N-1, \tag{2.1}$$

where

$$C(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0 \text{ or } k = N, \\ 1 & \text{otherwise.} \end{cases}$$

Here the time index t in $X_c(k,t)$ denotes that the transform starts from $x(t)$. Since the function $C(k)$ has a different value only when $k = 0$, we can consider those cases that $C(k)$'s equal one (*i.e.* $k = 1, 2, ..., N-1$.) first and re-examine the case for $k = 0$ later on. In transmission systems data arrive seriesly, therefore we are interested in the the 1-D DCT of the next input data vector $[x(t+1), x(t+2), ..., x(t+N)]$. From the definition, it is given by

$$X_c(k,t+1) = \frac{2}{N} \sum_{n=t+1}^{t+N} x(n) \cos\left[\frac{\pi[2(n-t-1)+1]k}{2N}\right]. \tag{2.2}$$

This can be rewritten as

$$X_c(k,t+1) = \overline{X}_c(k,t+1) \cos\left(\frac{\pi k}{N}\right) + \overline{X}_s(k,t+1) \sin\left(\frac{\pi k}{N}\right), \tag{2.3}$$

where

$$\overline{X}_c(k,t+1) = \frac{2}{N} \sum_{n=t+1}^{t+N} x(n) \cos\left[\frac{\pi[2(n-t)+1]k}{2N}\right], \tag{2.4}$$

and

$$\overline{X}_s(k,t+1) = \frac{2}{N} \sum_{n=t+1}^{t+N} x(n) \sin\left[\frac{\pi[2(n-t)+1]k}{2N}\right]. \tag{2.5}$$

As we can see, a DST-like term $\overline{X}_s(k,t+1)$ appears in (2.5). This motivates us to investigate the time-recursive DST.

2.2.2 Time-Recursive Discrete Sine Transform

There are several definitions for the DST. Here we prefer the definition proposed by Wang in [49]. The 1-D DST of a data vector $[x(t), x(t+1), ..., x(t+N-1)]$ is defined as

$$X_s(k,t) = \frac{2C(k)}{N} \sum_{n=t}^{t+N-1} x(n) \sin\left[\frac{\pi[2(n-t)+1]k}{2N}\right], \quad k = 1, ..., N. \quad (2.6)$$

Note that the range of k is from 1 to N. Again, we consider those cases that $D(k)$'s equal one first, *i.e.*

$$X_s(k,t) = \frac{2}{N} \sum_{n=t}^{t+N-1} x(n) \sin\left[\frac{\pi[2(n-t)+1]k}{2N}\right]. \quad (2.7)$$

The DST of the time update sequence $[x(t+1), x(t+2), ..., x(t+N)]$ is given by

$$\begin{aligned} X_s(k,t+1) &= \frac{2}{N} \sum_{n=t+1}^{t+N} x(n) \sin\left[\frac{\pi[2(n-t-1)+1]k}{2N}\right] \\ &= \overline{X}_s(k,t+1) \cos\left(\frac{\pi k}{N}\right) - \overline{X}_c(k,t+1) \sin\left(\frac{\pi k}{N}\right). \end{aligned} \quad (2.8)$$

Here the terms $\overline{X}_s(k,t+1)$ and $\overline{X}_c(k,t+1)$ that are used in (2.3) to generate $X_c(k,t+1)$ appear in the equation of the new DST transform $X_s(k,t+1)$ again. This suggests that the DCT and DST can be dually generated from each other.

2.2.3 The Lattice Structures

From (2.3) and (2.8), it is noted that the new DCT and DST transforms $X_c(k,t+1)$ and $X_s(k,t+1)$, can be obtained from $\overline{X}_c(k,t+1)$ and $\overline{X}_s(k,t+1)$ in the lattice form as shown in Fig. 2.1. The next step is to update $\overline{X}_c(k,t+1)$ and $\overline{X}_s(k,t+1)$ from the previous transforms $X_c(k,t)$ and $X_s(k,t)$. We notice that $X_c(k,t)$ and $\overline{X}_c(k,t+1)$ have similar terms except the old datum $x(t)$ and the

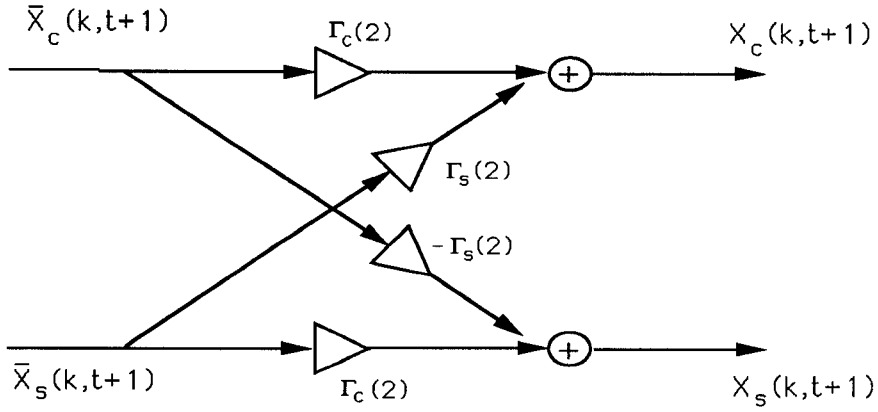


Figure 2.1: The lattice module.

incoming new datum $x(t+N)$. Therefore $\overline{X}_c(k,t+1)$ and $\overline{X}_s(k,t+1)$ can be obtained by deleting the term associated with the old datum $x(t)$ and updating the new datum $x(t+N)$ as

$$\begin{aligned}\overline{X}_c(k,t+1) &= X_c(k,t) - x(t)\left(\frac{2}{N}\right)\cos\left(\frac{\pi k}{2N}\right) \\ &\quad + x(t+N)\left(\frac{2}{N}\right)\cos\left[\frac{\pi(2N+1)k}{2N}\right] \\ &= X_c(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\cos\left(\frac{\pi k}{2N}\right), \end{aligned} \tag{2.9}$$

and

$$\begin{aligned}\overline{X}_s(k,t+1) &= X_s(k,t) - x(t)\left(\frac{2}{N}\right)\sin\left(\frac{\pi k}{2N}\right) \\ &\quad + x(t+N)\left(\frac{2}{N}\right)\sin\left[\frac{\pi(2N+1)k}{2N}\right] \\ &= X_s(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\sin\left(\frac{\pi k}{2N}\right). \end{aligned} \tag{2.10}$$

From (2.3), (2.8), (2.9), and (2.10), the new transforms $X_c(k,t+1)$ and $X_s(k,t+1)$ can be calculated from the previous transforms $X_c(k,t)$ and $X_s(k,t)$ by adding the effect of input signal samples $x(t)$ and $x(t+N)$. This demonstrates that the DCT and DST can be dually generated from each other in a recursive way. The time-recursive relations for the new transforms $X_c(k,t+1)$ and $X_s(k,t+1)$ as well as the previous transforms $X_c(k,t)$ and $X_s(k,t)$ are given by

$$X_c(k,t+1) = \left\{X_c(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\cos\left(\frac{\pi k}{2N}\right)\right\}\cos(\frac{\pi k}{N}) + \left\{X_s(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\sin\left(\frac{\pi k}{2N}\right)\right\}\sin(\frac{\pi k}{N}) \quad (2.11)$$

and

$$X_s(k,t+1) = \left\{X_s(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\sin\left(\frac{\pi k}{2N}\right)\right\}\cos(\frac{\pi k}{N}) - \left\{X_c(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\cos\left(\frac{\pi k}{2N}\right)\right\}\sin(\frac{\pi k}{N}) \quad (2.12)$$

Now, let us consider the cases for $k = 0$ in the DCT and $k = N$ in the DST respectively. According to (2.1), the 1-D DCT of the time-update input vector $[x(t+1), x(t+2), ..x(t+N)]$ for $k = 0$ is

$$X_c(0,t+1) = \frac{2}{N\sqrt{2}} \sum_{n=t+1}^{t+N} x(n). \quad (2.13)$$

The relation of $X_c(0,t+1)$ with the old transformed datum $X_c(0,t)$ is

$$X_c(0,t+1) = X_c(0,t) + \frac{2}{N\sqrt{2}}\left[-x(t) + x(t+N)\right]. \quad (2.14)$$

And, the time-recursive relation between the new transforms $X_s(N,t+1)$ and the previous transforms $X_s(N,t)$ is

$$\begin{aligned} X_s(N,t+1) &= \frac{2}{N\sqrt{2}} \sum_{n=t+1}^{t+N} x(n)(-1)^{n-t-1} \\ &= X_s(N,t) + \frac{2}{N\sqrt{2}}\left[-x(t) + (-1)^{N-1}x(t+N)\right]. \end{aligned} \quad (2.15)$$

The complete time-recursive lattice modules for ($k = 0, 1, 2, .., N-1.$) are shown in Fig. 2.2. It consists of a $N+1$ shift register and a normalized digital filter performing the plane rotation. The multiplications in the plane rotation can be reduced to addition and substration for $k = 0$ in the DCT and $k = N$ in the DST respectively. The following illustrates how this dually generated DCT and DST lattice structure works to obtain the DCT and DST with length N of a series of input data $[x(t), x(t+1), .., x(t+N-1), x(t+N), ...]$ for a specific k. The initial values of the transformed signals $X_c(k, t-1)$ and $X_s(k, t-1)$ are set to zero; so are the initial values in the shift register in the front of the lattice module. The input sequence $[x(t), x(t+1), ...]$ shifts sequentially into the shift register as shown in Fig. 2.2. Then the output signals $X_c(k, t)$ and $X_s(k, t)$, $k = 0, 1, ..., N-1, N$, are updated recursively according to (2.11), (2.12), (2.14) and (2.15). After the input datum $x(t+N-1)$ shifts into the shift register, the DCT and DST of the input data vector $[x(t), x(t+1), ..., x(t+N-1)]$ are obtained at the output for this index k. It takes N clock cycles to get the $X_c(k, t)$ and $X_s(k, t)$ of the input vector $[x(t), x(t+1), ..., x(t+N-1)]$. Since there are N different values for k, the total computational time to obtain all the transformed data is N^2 clock cycles, if only one lattice module is used. In this case, the delay time and throughput are the same N^2 clock cycles.

A parallel lattice array consists of N lattice modules can be used for parallel computations and it improves the computational speed drastically as shown in Fig. 2.3. Here we have seen that the transform domain data $X(k, t)$ have been decomposed into N disjoint components that have the same lattice modules with different multiplier coefficients in them. In this case the total computational delay time decreases to N clock cycle. It is important to notice that when the

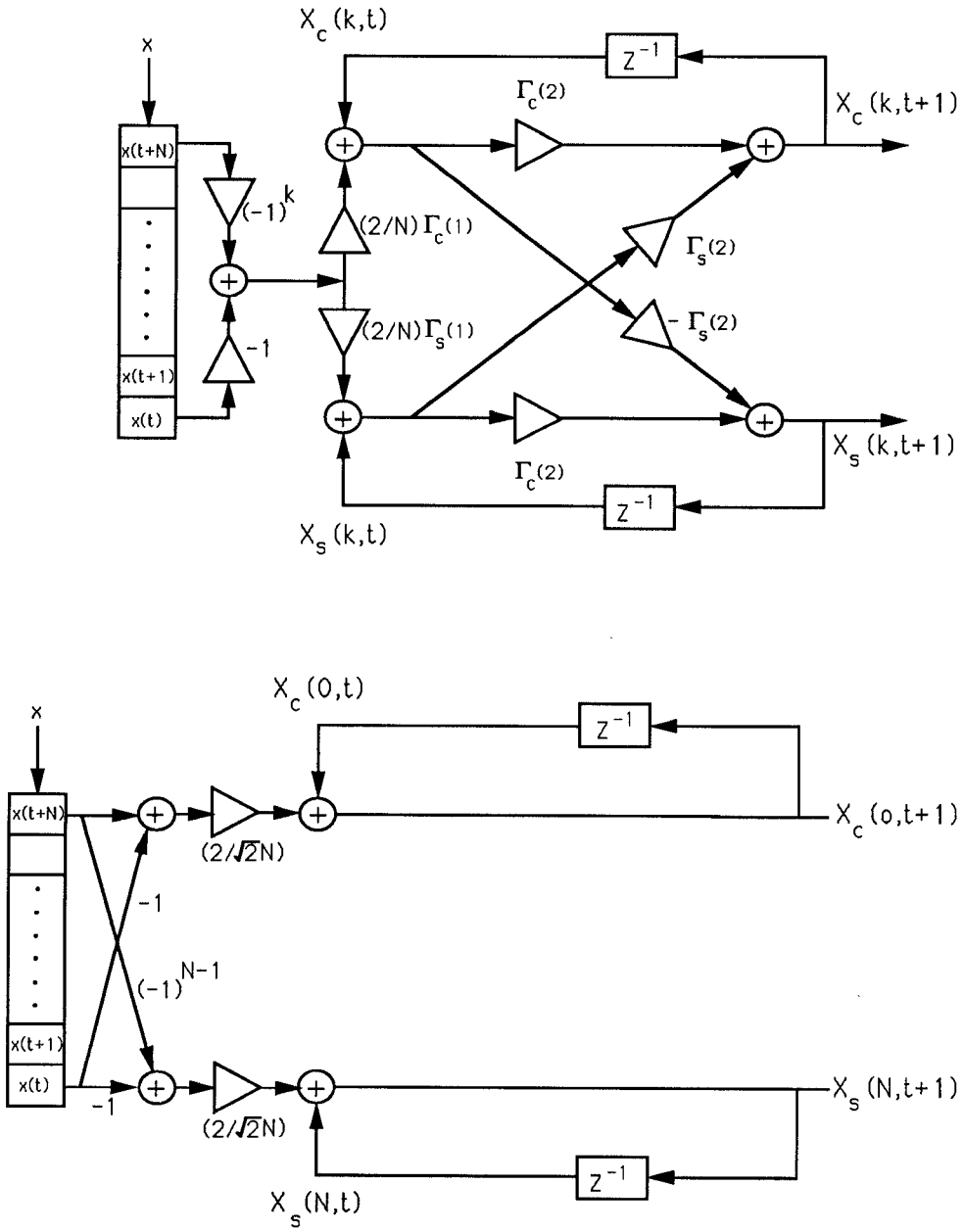


Figure 2.2: The lattice structure for the DCT and DST with coefficients $C(k)$'s and $D(k)$'s, $k = 0, 1, 2, ..., N-1, N$.

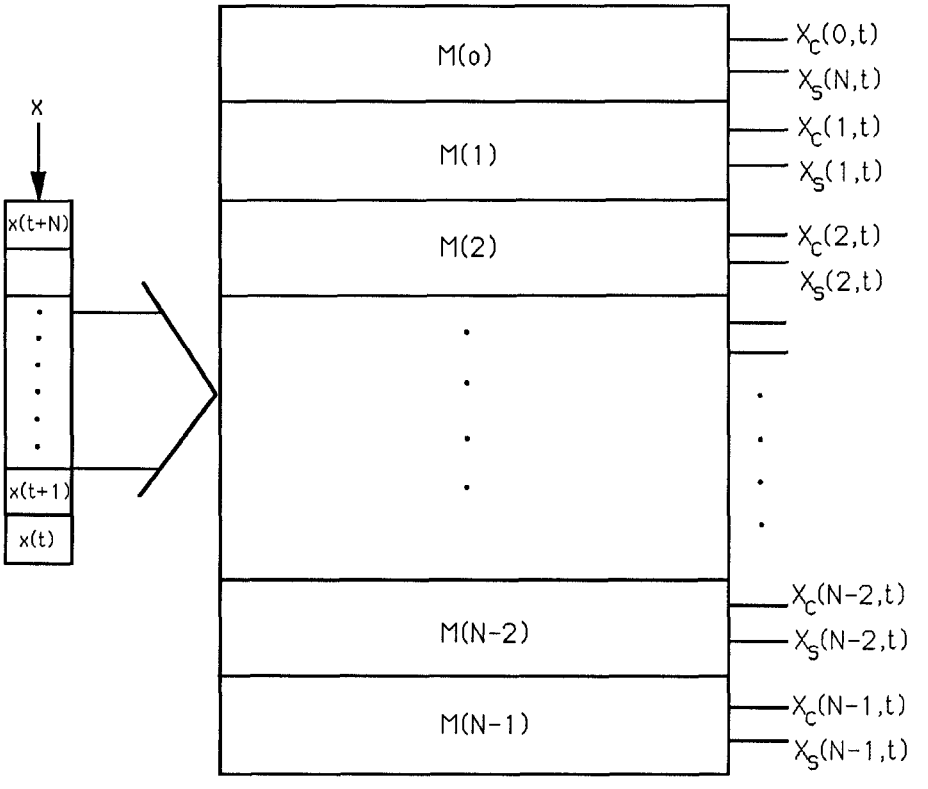


Figure 2.3: The parallel lattice structure for the DCT and DST.

next input datum $x(t+N)$ arrives, the transformed data of the input data vector $[x(t+1), x(t+2), ..., x(t+N)]$ can be obtained immediately. Likewise, it takes only one clock cycle to generate the transformed data of subsequent inputs. That is, the latency and throughput of this parallel system are N and 1 respectively.

It is obvious that this lattice structure is quite different from the signal flow graph realization obtained from the fast DCT algorithms [13, 15]. Since there is no global communication and the structure is modular and regular, it is suitable for practical VLSI implementation. The most interesting result is that this architecture can be applied to any value of N. From this point of view, it is more attractive than existing algorithms. In fact, most algorithms [73, 69] are limited to the sequence length N which either must be power of 2 or must be decomposable into mutually prime numbers. In addition, this lattice structure reveals some interesting properties of the DCT and DST, *i.e.*, the DCT and DST can be generated simultaneously. The DCT is near optimal to the KLT transform in highly correlated signals, while the DST approaches the KLT in signals with low correlation coefficient. As we are able to obtain the DCT and DST at the same time, this lattice structure is very useful especially when we do not know the statistics of the incoming signal. Furthermore, we can use a single lattice module with only 6 multipliers and 5 adders to recursively compute any N-point DCT and DST simultaneously. To obtain the transformed data in parallel, we need N lattice modules. As mentioned before, it is suitable for VLSI implementation since all the modules have the same structure except the 0'th module which can be simplified as shown in Fig. 2.2. This parallel lattice structure requires $6N-4$ multipliers and $5N-1$ adders.

2.3 Inverse Transforms

2.3.1 Time-Recursive IDCT

According to the definition of the DCT in (2.1), the IDCT for the transform domain sequence $[X(t), X(t+1), ..., X(t+N-1)]$ is

$$x_c(n,t) = \sum_{k=t}^{t+N-1} C(k-t)X(k)\cos\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1. \tag{2.16}$$

The coefficients $C(k)$'s are given in (2.1). From the time-recursive point of view, the IDCT of the new sequence $[X(t+1), X(t+2), ..., X(t+N)]$ can be expressed as

$$x_c(n,t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k)\cos\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right]. \tag{2.17}$$

Similar to the previous sections, we can decompose (2.17) into

$$x_c(n,t+1) = \overline{x}_c(n,t+1)\cos\left[\frac{\pi(2n+1)}{2N}\right] + \overline{x}_{as}(n,t+1)\sin\left[\frac{\pi(2n+1)}{2N}\right], \tag{2.18}$$

where

$$\overline{x}_c(n,t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k)\cos\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \tag{2.19}$$

and

$$\overline{x}_{as}(n,t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k)\sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right]. \tag{2.20}$$

In order to be a dually generated pair of the IDCT given in (2.16), we define the auxiliary inverse discrete sine transform (AIDST) as

$$x_{as}(n,t) = \sum_{k=t}^{t+N-1} C(k-t)X(k)\sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1. \tag{2.21}$$

Although this definition utilizes the same sine functions as the transform kernel, it is not the inverse transform of the DST. To differentiate it from the IDST, we call this the AIDST. Comparing to the IDST defined in (2.26) , we observe that the AIDST has the special coefficients $C(0) = \frac{1}{\sqrt{2}}$ associated with the first term, while the IDST with the last term. The AIDST for the data sequence $[X(t+1), X(t+2), ..., X(t+N)]$ can be written as

$$x_{as}(n, t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k) \sin\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right]. \quad (2.22)$$

By using the trigonometric function expansions, $x_{as}(n, t+1)$ becomes

$$x_{as}(n, t+1) = \overline{x}_{as}(n, t+1) \cos\left[\frac{\pi(2n+1)}{2N}\right] - \overline{x}_c(n, t+1) \sin\left[\frac{\pi(2n+1)}{2N}\right]. \quad (2.23)$$

Lattice Structure for IDCT

Combining (2.18) and (2.23), we observe that the IDCT and the AIDST can be generated in exactly the same way as the dual generation of the DCT and DST. Therefore, the lattice structure in Fig.2.1 can be applied here except that the coefficients must be modified. Since the coefficients $C(k)$'s are inside the expression in the inverse transform, the relation between $x_c(n, t)$ and $\overline{x}_c(n, t+1)$ will be different from what we have in the DCT. Equations (2.16) and (2.19) as well as (2.20) and (2.21) have the same terms for $k \in \{t+2, t+3, ..., t+N-1\}$. After adding the effects of the terms for $k = t$ and $k = t+1$, we obtain

$$\overline{x}_c(n, t+1) = x_c(n, t) - \frac{1}{\sqrt{2}}X(t) + \left(\frac{1}{\sqrt{2}} - 1\right) \cos\left[\frac{\pi(2n+1)}{2N}\right] X(t+1), \quad (2.24)$$

and

$$\overline{x}_{as}(n, t+1) = x_{as}(n, t) + (-1)^n X(t+N) + \left(\frac{1}{\sqrt{2}} - 1\right) \sin\left[\frac{\pi(2n+1)}{2N}\right] X(t+1). \quad (2.25)$$

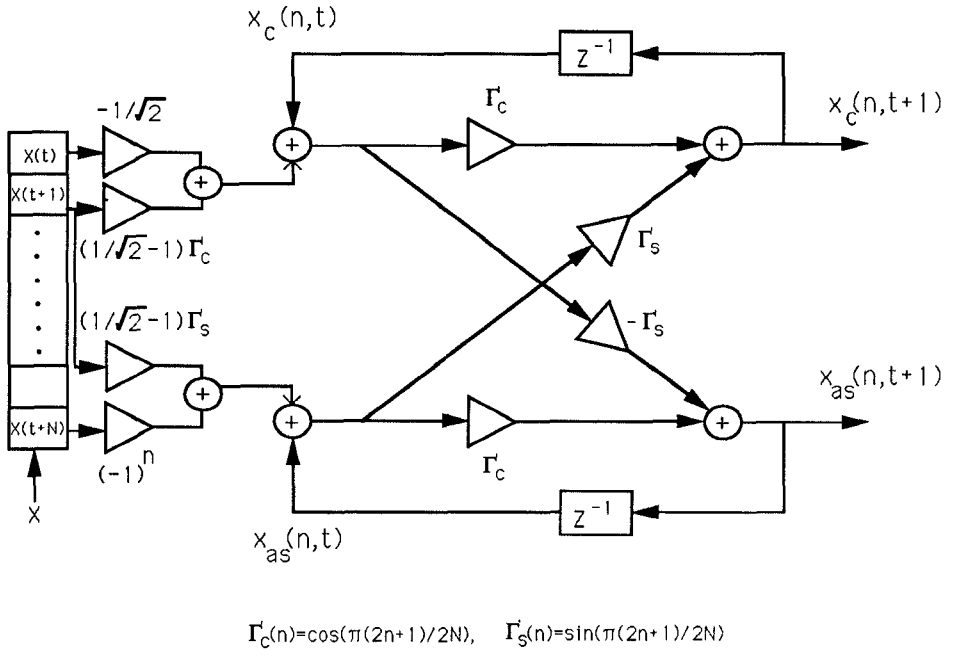


Figure 2.4: The lattice structure for the IDCT and AIDST.

The complete lattice module for the IDCT and AIDST is shown in Fig.2.4. This IDCT lattice structure has the same lattice module as that of the DCT except for the input stage where one more adder and one more multiplier are required. The procedure to calculate the inverse transformed data is the same. Therefore, this IDCT lattice structure has the same advantages as that of the DCT. To obtain the inverse transform in parallel, we need N such IDCT lattice modules where $7N$ multipliers and $6N$ adders are required. Again, we see that the numbers of adders and multipliers are linear functions of N. Here we should notice that to obtain the inverse transform of the original input data sequence, for example, $[x(0), x(1), x(2), ..., x(N-1)]$ and $[x(N), x(N+1), ..., x(2N-1)]$, it is sufficient only to send the transformed data corresponding to these two blocks, *i.e.*, $[X(0), X(1), ..., X(N-1)]$ and $[X(N), X(N+1), ..., X(2N-1)]$ respectively, although we have all the intermediate transformed data. Then by

applying the time-recursive algorithm mentioned above, we obtain the original data after $X(N-1)$ and $X(2N-1)$ arrive, the intermediate data obtained by the inverse transform are redundant.

2.3.2 Time-Recursive IDST

From the definition of the DST in (2.6), the IDST for the transform domain sequence $[X(t+1), X(t+2), ..., X(t+N)]$ is given by

$$x_s(n,t) = \sum_{k=t+1}^{t+N} D(k-t)X(k) \sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1. \tag{2.26}$$

The coefficients $D(k)$'s are given in (2.6). Analogous to Section 3.1, we define the auxiliary inverse discrete cosine transform (AIDCT)

$$x_{ac}(n,t) = \sum_{k=t+1}^{t+N} D(k-t)X(k) \cos\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1, \tag{2.27}$$

which is the dually generated counterpart of the IDST. The IDST and AIDCT of the new sequence of transformed data $[X(t+2), X(t+3), ..., X(t+N+1)]$ are given respectively by

$$x_s(n,t+1) = \sum_{k=t+2}^{t+N+1} D(k-t-1)X(k) \sin\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right], \tag{2.28}$$

and

$$x_{ac}(n,t+1) = \sum_{k=t+2}^{t+N+1} D(k-t-1)X(k) \cos\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right]. \tag{2.29}$$

Same as before, we can decompose (2.28) and (2.29) to

$$x_s(n,t+1) = \overline{x}_s(n,t+1) \cos\left[\frac{\pi(2n+1)}{2N}\right] - \overline{x}_c(n,t+1) \sin\left[\frac{\pi(2n+1)}{2N}\right], \tag{2.30}$$

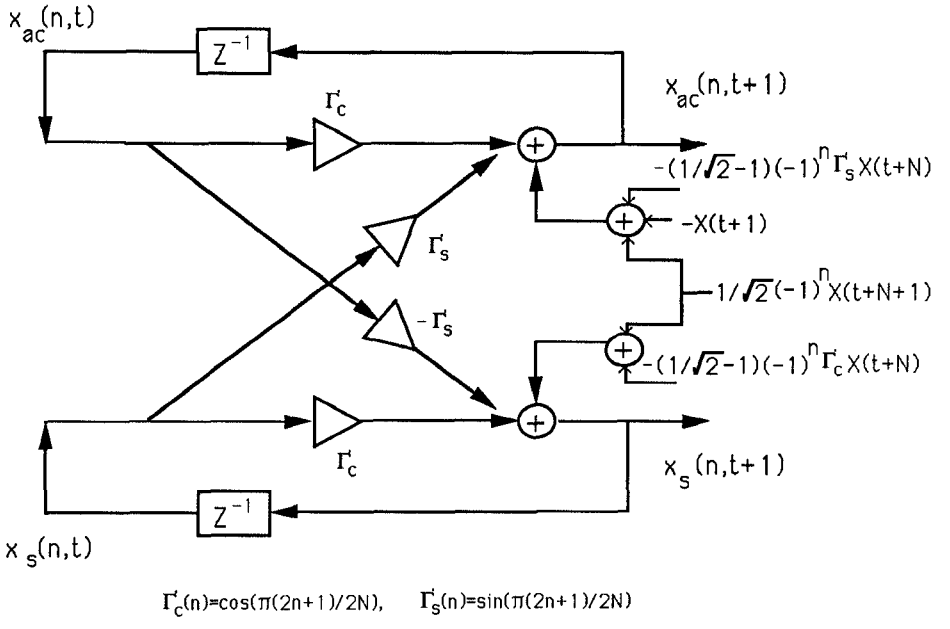


Figure 2.6: The post-lattice structure for the IDST and AIDCT.

signals are added at the end of the lattice. From now on, we call this lattice structure a post-lattice module and the previous ones as pre-lattice modules. This post-lattice module needs 7 multipliers and 7 adders, less than required for the corresponding pre-lattice module. A parallel post-lattice structure, which generates N transformed data simultaneously, requires $7N$ multipliers and $7N$ adders. All the forward and inverse transform pairs mentioned above have pre-lattice and post-lattice structures. Not all post-lattice structures are superior to their pre-lattice counterparts in the hardware complexity. For example, the IDCT and AIDST post-lattice form can be expressed as

$$\begin{aligned} x_{as}(n,t+1) &= \cos\left[\frac{\pi(2n+1)}{2N}\right] x_{as}(n,t) - \sin\left[\frac{\pi(2n+1)}{2N}\right] x_c(n,t) \\ &\quad -(\frac{1}{\sqrt{2}})\sin\left[\frac{\pi(2n+1)}{2N}\right] X(t) \\ &\quad +(-1)^n X(t+N)\cos\left[\frac{\pi(2n+1)}{2N}\right], \end{aligned} \tag{2.38}$$

and

$$\begin{aligned} x_c(n,t+1) &= \cos\left[\frac{\pi(2n+1)}{2N}\right] x_c(n,t) + \sin\left[\frac{\pi(2n+1)}{2N}\right] x_{as}(n,t) \\ &\quad -(\frac{1}{\sqrt{2}})\cos\left[\frac{\pi(2n+1)}{2N}\right] X(t) - X(t+1) + (\frac{1}{\sqrt{2}} - 1)X(t+1) \\ &\quad +(-1)^n \sin\left[\frac{\pi(2n+1)}{2N}\right] X(t+N). \end{aligned} \tag{2.39}$$

This post-lattice module has 9 multipliers and 7 adders which are more than its pre-lattice realization. As to the DCT and DST, the post-lattice form can be expressed as

$$\begin{aligned} X_c(k,t+1) &= \cos\left(\frac{\pi k}{N}\right) X_c(k,t) + \sin\left(\frac{\pi k}{N}\right) X_s(k,t) \\ &\quad +(\frac{2}{N})\cos\left(\frac{\pi k}{2N}\right)\left[-x(t) + (-1)^k x(t+N)\right], \end{aligned} \tag{2.40}$$

and

$$\begin{aligned} X_s(k,t+1) &= \cos\left(\frac{\pi k}{N}\right) X_s(k,t) - \sin\left(\frac{\pi k}{N}\right) X_s(k,t) \\ &\quad -(\frac{2}{N})\sin\left(\frac{\pi k}{2N}\right)\left[-x(t) + (-1)^k x(t+N)\right]. \end{aligned} \tag{2.41}$$

In this case, the pre-lattice and post-lattice modules have the same numbers of multipliers and adders.

2.4 Discrete Hartley Transform (DHT)

According to Bracewell's definition of the DHT in [2], the data sequence $x(n)$ and the DHT transformed data $X(k)$ have the following relation

$$\begin{aligned} X_h(k,t) &= \frac{1}{N}\sum_{n=t}^{t+N-1} x(n) cas\left(\frac{2\pi k(n-t)}{N}\right) \\ &= \frac{1}{N}\sum_{n=t}^{t+N-1} x(n)\left[\cos\left(\frac{2\pi k(n-t)}{N}\right) + \sin\left(\frac{2\pi k(n-t)}{N}\right)\right], \\ &\quad k = 0, 1, ..., N-1. \end{aligned} \tag{2.42}$$

The DHT uses real expressions $\cos(\frac{2\pi k(n-t)}{N}) + \sin(\frac{2\pi k(n-t)}{N})$ as the transform kernel, while discrete Fourier transform (DFT) uses the complex exponential expression $\exp(\frac{i2\pi k(n-t)}{N})$ as the transform kernel. Because the kernel of the DHT is a summation of cosine and sine terms, we can separate them into a combination of a DCT-like and a DST-like transforms as follows:

$$X_h(k,t) = \acute{X}_c(k,t) + \acute{X}_s(k,t), \tag{2.43}$$

where

$$\acute{X}_c(k,t) = \frac{1}{N}\sum_{n=t}^{t+N-1} x(n)\left[\cos\left(\frac{2\pi k(n-t)}{N}\right)\right], \tag{2.44}$$

and

$$\acute{X}_s(k,t) = \frac{1}{N}\sum_{n=t}^{t+N-1} x(n)\left[\sin\left(\frac{2\pi k(n-t)}{N}\right)\right]. \tag{2.45}$$

The $\acute{X}_c(k,t)$ is the so-called DCT-I and the $\acute{X}_s(k,t)$ is the DST-I that are defined by Yip and Rao in [51]. Since the DHT can be decomposed into the combination of the DCT-I and DST-I, the dual generation of both for the DHT is thus possible. The DCT-I and the DST-I of the data sequence $[x(t+1), x(t+2), ..., x(t+N)]$ are

$$\acute{X}_c(k,t+1) = \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\cos\left[\frac{2\pi k(n-t-1)}{N}\right], \tag{2.46}$$

and

$$\acute{X}_s(k,t+1) = \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\sin\left[\frac{2\pi k(n-t-1)}{N}\right]. \tag{2.47}$$

The new transforms $\acute{X}_c(k,t+1)$ and $\acute{X}_s(k,t+1)$ can be further expressed as

$$\acute{X}_c(k,t+1) = \overline{\acute{X}}_c(k,t+1)\cos\left(\frac{2\pi k}{N}\right) + \overline{\acute{X}}_s(k,t+1)\sin\left(\frac{2\pi k}{N}\right), \tag{2.48}$$

and

$$\acute{X}_s(k,t+1) = \overline{\acute{X}}_s(k,t+1)\cos\left(\frac{2\pi k}{N}\right) - \overline{\acute{X}}_c(k,t+1)\sin\left(\frac{2\pi k}{N}\right), \tag{2.49}$$

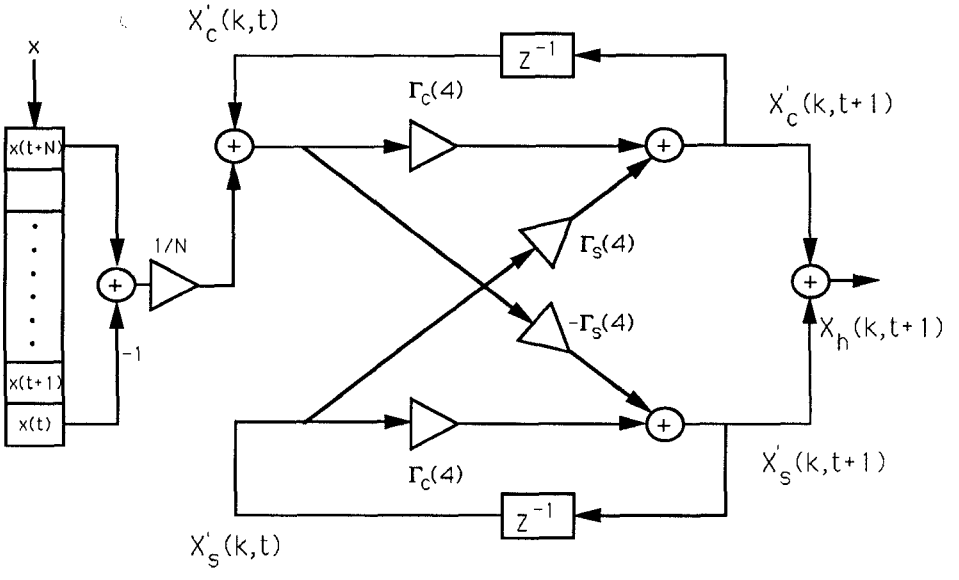


Figure 2.7: The lattice structure for the DHT for $k = 1, 2, ..., N-1$.

where

$$\begin{aligned} \overline{\acute{X}}_c(k,t+1) &= \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\cos\left(\frac{2\pi k(n-t)}{N}\right) \\ &= \acute{X}_c(k,t) + \frac{1}{N}\left[-x(t)+x(t+N)\right], \end{aligned} \tag{2.50}$$

and

$$\begin{aligned} \overline{\acute{X}}_s(k,t+1) &= \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\sin\left(\frac{2\pi k(n-t)}{N}\right) \\ &= \acute{X}_s(k,t). \end{aligned} \tag{2.51}$$

The lattice module for the DHT for $k = 1, ..., N-1$ is shown in Fig. 2.7 and Fig. 2.8. From Fig. 2.7, we can see that the numbers of multipliers and adders are less than those of the dual generation of the DCT and DST. The total numbers of multipliers and adders in the parallel DHT lattice architecture are $5N-4$ and $5N-3$ respectively.

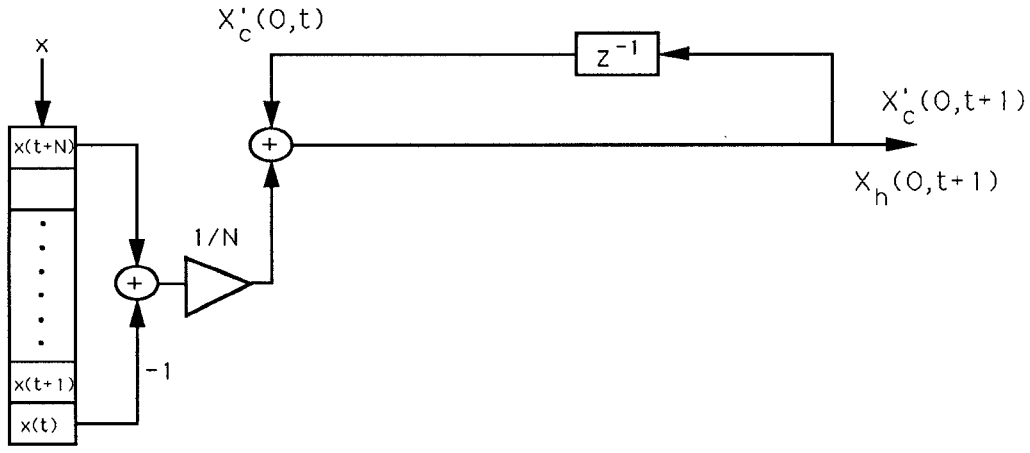


Figure 2.8: The lattice structure for the DHT for $k = 0$.

2.5 Block Processing

All the time-recursive discrete transforms derived above are based on the block-size-one update which means the time index is updated by one. That is, at each iteration only the effect of one old datum is removed and the information of one new datum is added. We are interested in the relation between the area-time complexity (AT) and block size. This motivates us to discuss the effect on the lattice structure when the block size is increased.

2.5.1 Block Processing of time-recursive DCT and DST

We begin the discussion of block processing with the block-size-two update. Here we assume the time index t in (2.1) is zero for simplicity, and we will use this in the following discussions. As before, the transformed data $X_c(k, 2)$ and $X_s(k, 2)$

are defined as the DCT and DST of the input vector $[x(2), x(3), ..., x(N), x(N+1)]$. That is,

$$X_c(k,2) = \sum_{n=2}^{N+1} x(n) \cos\left\{\frac{\pi[2(n-2)+1]k}{2N}\right\}, \tag{2.52}$$

and

$$X_s(k,2) = \sum_{n=2}^{N+1} x(n) \sin\left\{\frac{\pi[2(n-2)+1]k}{2N}\right\}. \tag{2.53}$$

To obtain $X_c(k,2)$ from $X_c(k,0)$ and $X_s(k,2)$ from $X_s(k,0)$ directly, we can rewrite $X_c(k,2)$ and $X_s(k,2)$ as

$$X_c(k,2) = \overline{X}_c(k,2) \cos\left(\frac{2\pi k}{N}\right) + \overline{X}_s(k,2) \sin\left(\frac{2\pi k}{N}\right), \tag{2.54}$$

and

$$X_s(k,2) = \overline{X}_s(k,2) \cos\left(\frac{2\pi k}{N}\right) - \overline{X}_c(k,2) \sin\left(\frac{2\pi k}{N}\right), \tag{2.55}$$

where

$$\begin{aligned} \overline{X}_c(k,2) &= \sum_{n=2}^{N+1} x(n) \cos\left[\frac{\pi(2n+1)k}{2N}\right] \\ &= X_c(k,0) + [-x(0) + (-1)^k x(N)] \cos\left(\frac{\pi k}{2N}\right) \\ &\quad + [-x(1) + (-1)^k x(N+1)] \cos\left(\frac{3\pi k}{2N}\right), \end{aligned} \tag{2.56}$$

and

$$\begin{aligned} \overline{X}_s(k,2) &= \sum_{n=2}^{N+1} x(n) \sin\left[\frac{\pi(2n+1)k}{2N}\right] \\ &= X_s(k,0) + [-x(0) + (-1)^k x(N)] \sin\left(\frac{\pi k}{2N}\right) \\ &\quad + [-x(1) + (-1)^k x(N+1)] \sin\left(\frac{3\pi k}{2N}\right). \end{aligned} \tag{2.57}$$

The lattice module for the block-size-two update is shown in Fig. 2.9. There are two more multipliers in the lattice, *i.e.*, the total number of multipliers is eight.

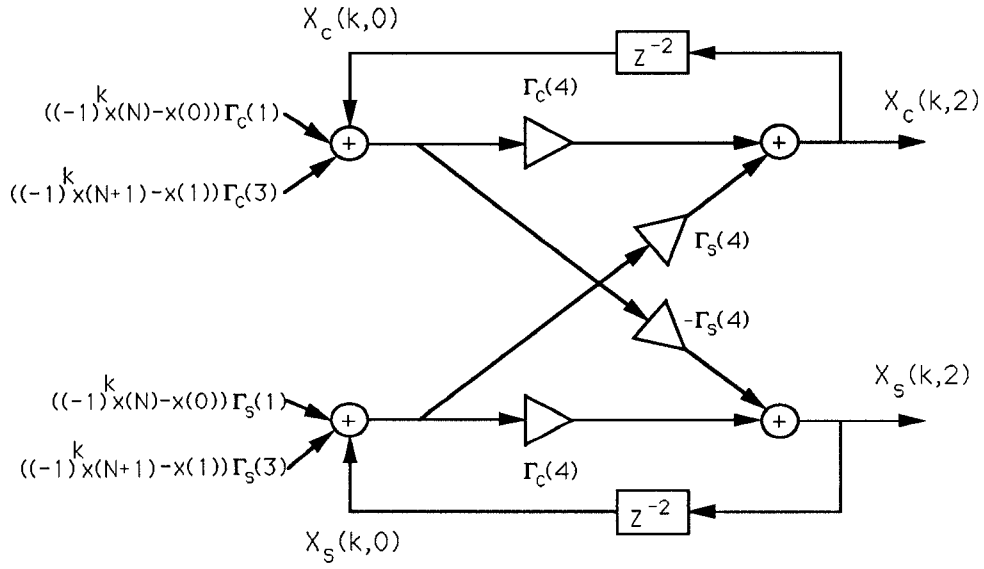


Figure 2.9: The lattice structure for block-size-two operation on the DCT and DST.

To obtain the transformed data in parallel, we need N such lattice modules. The latency for this kind of parallel structure is $N/2$ and the total number of multipliers is $8N$. Since there is no complex communication problem in the lattice structure, the area-time complexity (AT) can be approximated by the product of the number of multipliers and the time latency, plus the area-time complexity of the adders, which is $o(m\log(m))$ for adding m data. Next, let us consider the more general case for the block-size-m update, where m ranges from one to N. The 1-D DCT and DST of block-size-m update are to obtain the transform of $[x(m), x(m+1), ..., x(N+m-1)]$ directly from the transform of $[x(0), x(1), ..., x(N-1)]$. We have

$$X_c(k,m) = \sum_{n=m}^{N+m-1} x(n)\cos\left[\frac{\pi[2(n-m)+1]k}{2N}\right], \tag{2.58}$$

and

$$X_s(k,m) = \sum_{n=m}^{N+m-1} x(n) \sin\left[\frac{\pi[2(n-m)+1]k}{2N}\right]. \tag{2.59}$$

Applying the same procedure in the case of block-size-two update, we can write (2.58) and (2.59) as

$$X_c(k,m) = \overline{X}_c(k,m)\cos\left(\frac{m\pi k}{N}\right) + \overline{X}_s(k,m)\sin\left(\frac{m\pi k}{N}\right), \tag{2.60}$$

and

$$X_s(k,m) = \overline{X}_s(k,m)\cos\left(\frac{m\pi k}{N}\right) - \overline{X}_c(k,m)\sin\left(\frac{m\pi k}{N}\right). \tag{2.61}$$

where

$$\begin{aligned}
\overline{X}_c(k,m) &= \sum_{n=m}^{N+m-1} x(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right] \\
&= X_c(k,0) - \sum_{n=0}^{m-1} x(n)\cos\left(\frac{\pi(2n+1)k}{2N}\right) \\
&\quad + \sum_{n=N}^{N+m-1} x(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right] \\
&= X_c(k,0) \\
&\quad + \sum_{n=0}^{m-1}[-x(n)+(-1)^k x(N+n)]\cos\left[\frac{\pi(2n+1)k}{2N}\right],
\end{aligned} \tag{2.62}$$

and

$$\begin{aligned}
\overline{X}_s(k,m) &= \sum_{n=m}^{N+m-1} x(n)\sin\left[\frac{\pi(2n+1)k}{2N}\right] \\
&= X_s(k,0) - \sum_{n=0}^{m-1} x(n)\sin\left[\frac{\pi(2n+1)k}{2N}\right] \\
&\quad - \sum_{n=N}^{N+m-1} x(n)\sin\left[\frac{\pi(2n+1)k}{2N}\right] \\
&= X_c(k,0) \\
&\quad + \sum_{n=0}^{m-1}[-x(n)+(-1)^k x(N+n)]\sin\left[\frac{\pi(2n+1)k}{2N}\right].
\end{aligned} \tag{2.63}$$

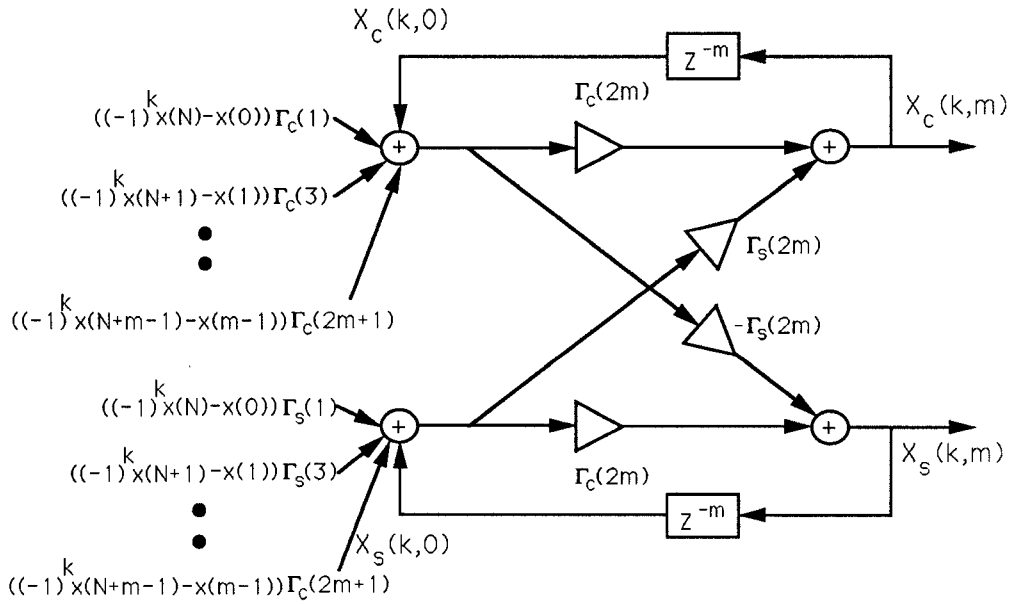


Figure 2.10: The lattice structure for block-size-m operation on the DCT and DST.

Combining those input terms with same cosine and sine multiplier coefficients together, we can obtain the lattice module for block size m as shown in Fig. 2.10. To obtain the transform data $X(k)$ in parallel, N lattice modules of Fig. 2.10 are required. The total number of multipliers of the parallel structure is $(4+2m)N$, the total number of adders is $(3m+2)N$, and the throughput is 1. The area-time complexity due to multipliers and adders are $(4+2m)N$ and $(3m+2)N\log((3m+2)N)$ respectively. Denote ATm as the area-time complexity of the block-size-m update, then $ATm = (4+2m)N+(3m+2)N\log((3m+2)N)$. For example, $AT1 = 6N + [5N\log(5N)]$ and $AT2 = 8N + [8N\log(8N)]$. In the limiting case of the block-size-N update, *i.e.*, we move a whole block of the input data sequence, $ATN \simeq (4+2N)N + 3N^2\log(3N^2)$. In general, the area-time product gets smaller as block size m decreases. We found that when $m = 1$, the

minimum AT complexity is achieved.

2.6 Multiplier-Reduction of the lattice structure

In the VLSI implementation, the number of multipliers is an important factor to the cost and complexity of the system. In this section, we develop two methods to reduce the number of multipliers in our parallel lattice structures. The first scheme makes use of a series input series output ($SISO$) approach and $2N$ multipliers can be saved; the trade-off is that the latency and throughput is increased. The second approach, which reconstructs the structure into a double-lattice realization, saves N multipliers and the latency remains intact.

2.6.1 $SISO$ Approach

Let us consider this problem through a general lattice structure as shown in Fig. 2.11. Denote the output and input data at time t as $(X_c(t), X_s(t))$ and (x_{ct}, x_{st}) respectively, where the input and output have the following relations

$$\begin{aligned} X_c(t) &= [X_c(t-1) + \Gamma_1 x_{ct}]\,\Gamma_2 + [X_s(t-1) + \Gamma_3 x_{st}]\,\Gamma_4, \\ X_s(t) &= [X_s(t-1) + \Gamma_3 x_{st}]\,\Gamma_2 - [X_c(t-1) + \Gamma_1 x_{ct}]\,\Gamma_4. \end{aligned} \tag{2.64}$$

By dividing both equations by Γ_4, we have

$$\begin{aligned} X_c(t)/\Gamma_4 &= [X_c(t-1) + \Gamma_1 x_{ct}]\,\Gamma_2/\Gamma_4 + [X_s(t-1) + \Gamma_3 x_{st}]\,, \\ X_s(t)/\Gamma_4 &= [X_s(t-1) + \Gamma_3 x_{st}]\,\Gamma_2/\Gamma_4 - [X_c(t-1) + \Gamma_1 x_{ct}]\,. \end{aligned} \tag{2.65}$$

The lattice structure manifesting the above relations is shown in Fig. 2.12. It is noted that only four multipliers exist in this structure and the outputs

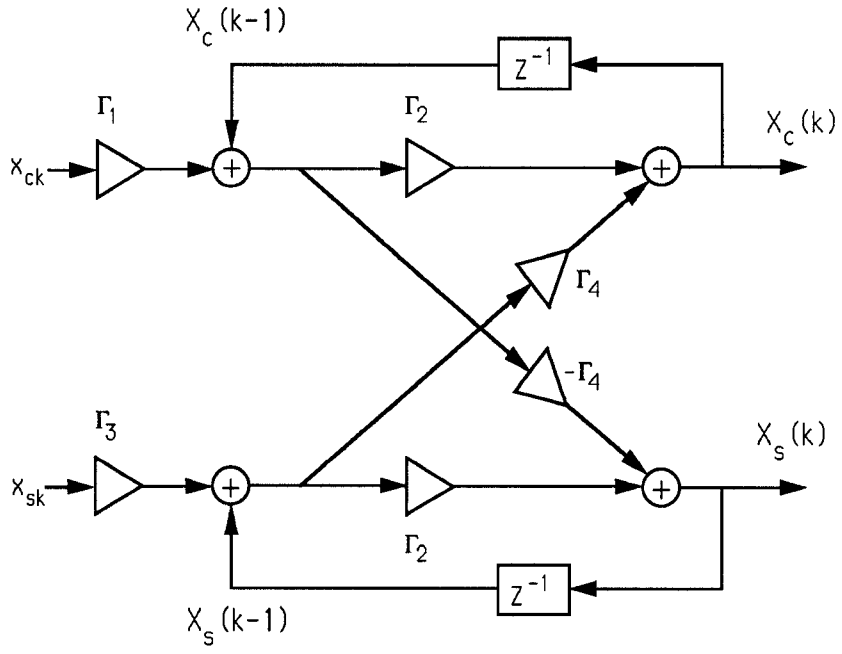


Figure 2.11: The general lattice module.

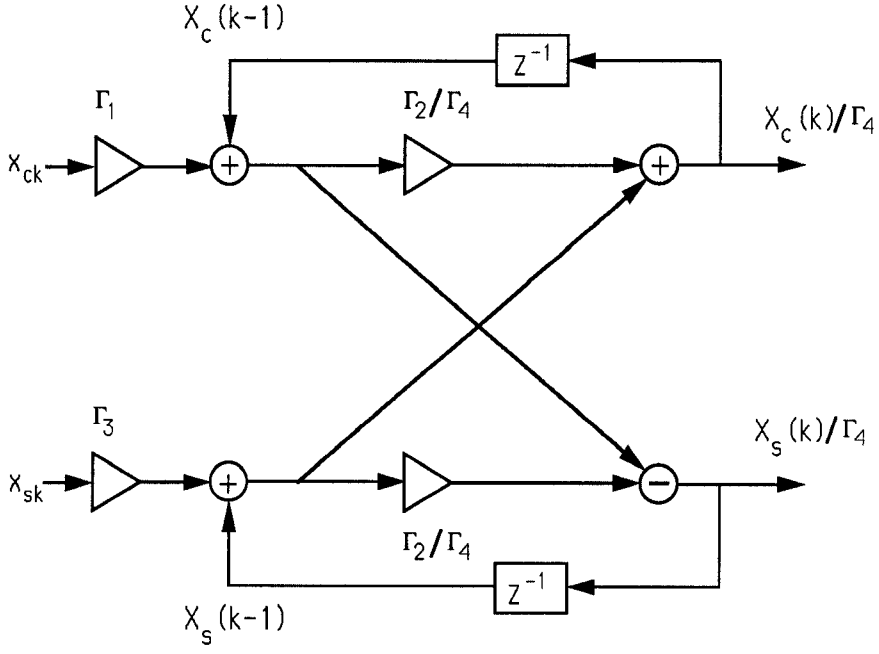


Figure 2.12: The model of multiplier-reduction.

obtained differ from the original one by a factor Γ_4. To examine the effect of this multiplier reduction on the recursive operation from $X_c(1)$ to $X_c(N)$, we start with the derivation from $t = 1$. That is

$$\begin{aligned} X_c(1)/\Gamma_4 &= [X_c(0) + \Gamma_1 x_{c1}]\,\Gamma_2/\Gamma_4 + [X_s(0) + \Gamma_3 x_{s1}]\,, \\ X_s(1)/\Gamma_4 &= [X_s(0) + \Gamma_3 x_{s1}]\,\Gamma_2/\Gamma_4 - [X_c(0) + \Gamma_1 x_{c1}]\,. \end{aligned} \tag{2.66}$$

For $t = 2$

$$\begin{aligned} X_c(2)/\Gamma_4 &= [X_c(1) + \Gamma_1 x_{c2}]\,\Gamma_2/\Gamma_4 + [X_s(1) + \Gamma_3 x_{s2}]\,, \\ X_s(2)/\Gamma_4 &= [X_s(1) + \Gamma_3 x_{s2}]\,\Gamma_2/\Gamma_4 - [X_c(1) + \Gamma_1 x_{c2}]\,. \end{aligned} \tag{2.67}$$

Because the outputs at time $t = 1$ are $X_c(1)/\Gamma_4$ and $X_s(1)/\Gamma_4$, $X_c(1)$ and $X_s(1)$ at (2.67) should be replaced by $X_c(1)/\Gamma_4$ and $X_s(1)/\Gamma_4$. To keep the above equations valid, we can multiply both equations by $1/\Gamma_4$ as shown

$$\begin{aligned} X_c(2)/\Gamma_4^2 &= [X_c(1)/\Gamma_4 + (\Gamma_1/\Gamma_4)x_{c2}]\,\Gamma_2/\Gamma_4 + [X_s(1)/\Gamma_4 + (\Gamma_3/\Gamma_4)x_{s2}]\,, \\ X_s(2)/\Gamma_4^2 &= [X_s(1)/\Gamma_4 + (\Gamma_3/\Gamma_4)x_{s2}]\,\Gamma_2/\Gamma_4 \\ &\quad - [X_s(1)/\Gamma_4 + (\Gamma_1/\Gamma_4)x_{c2}]\,. \end{aligned} \tag{2.68}$$

The coefficients of the input multipliers are Γ_1/Γ_4 and Γ_3/Γ_4, instead of Γ_1 and Γ_3 at time $t = 1$, and the output sare $X_c(2)/\Gamma_4^2$ and $X_s(2)/\Gamma_4^2$. For $t = N$, the recursive equations become

$$\begin{aligned} X_c(N)/\Gamma_4^N &= \left[X_c(N-1)/\Gamma_4^{N-1} + (\Gamma_1/\Gamma_4^{N-1})x_{cN}\right]\Gamma_2/\Gamma_4 \\ &\quad + \left[X_s(N-1)/\Gamma_4^{N-1} + (\Gamma_3/\Gamma_4^{N-1})x_{sN}\right], \\ X_s(N)/\Gamma_4^N &= \left[X_s(N-1)/\Gamma_4^{N-1} + (\Gamma_3/\Gamma_4^{N-1})x_{sN}\right]\Gamma_2/\Gamma_4 \\ &\quad - \left[X_c(N-1)/\Gamma_4^{N-1} + (\Gamma_1/\Gamma_4^{N-1})x_{cN}\right]. \end{aligned} \tag{2.69}$$

From the above derivations, we observe that the two multipliers can be removed by using variable multipliers in the input stage where the coefficients $(\Gamma_1, \Gamma_1/\Gamma_4, .., \Gamma_1/\Gamma_4^{N-1})$ and $(\Gamma_3, \Gamma_3/\Gamma_4, .., \Gamma_3/\Gamma_4^{N-1})$ are stored in shift registers. The structure is shown in Fig. 2.13. The output can be obtained by multiplying the factor Γ_4^N. This kind of rearrangement does not save multipliers. However, for N such lattice structures, the number of multipliers can be reduced by using variable multipliers at the output stage and the coefficients for each stage $\Gamma_4^N(i)$, $i = 0, 1, 2, .., N-1$, are stored in the shift registers. Fig. 2.14 shows the final structure where the total number of multipliers is $4N + 2$. This means that the number of multipliers for N parallel such lattice structures is reduced from $6N$ to $4N+2$. The tradeoff is that $2N+2$ shift registers are required and the latency becomes $2N$ instead of N. Also, this resulting structure is a $SISO$ system, while the original parallel structure is a $SIPO$ system.

For example, the variable-multiplier method derived above can be applied to the lattice structure of the DCT and DST. There are no multipliers needed for $t = 0$, therefore the module remains the same. For $t = 1, 2, .., N-1$, the multiplier-reduced lattice structure is shown in Fig. 2.14, where the coefficients are $\Gamma_1 = \cos(k\pi/2N)$, $\Gamma_2 = \cos(k\pi/N)$, $\Gamma_3 = \sin(k\pi/2N)$, and $\Gamma_4 = \sin(k\pi/N)$. The total number of multipliers is $4N-2$ and the latency for this $SISO$ structure is $2N$.

It is readily seen that the *SISO* approach for multiplier reduction is in fact a denormalization of the orthogonal rotation in the lattice. It is well-known that the orthogonal rotation is numerical stable so that the roundoff errors will not be accumulated. However, the denormalized lattice does not have such a nice numerical property in finite-precision implementation, *i.e.* the roundoff errors

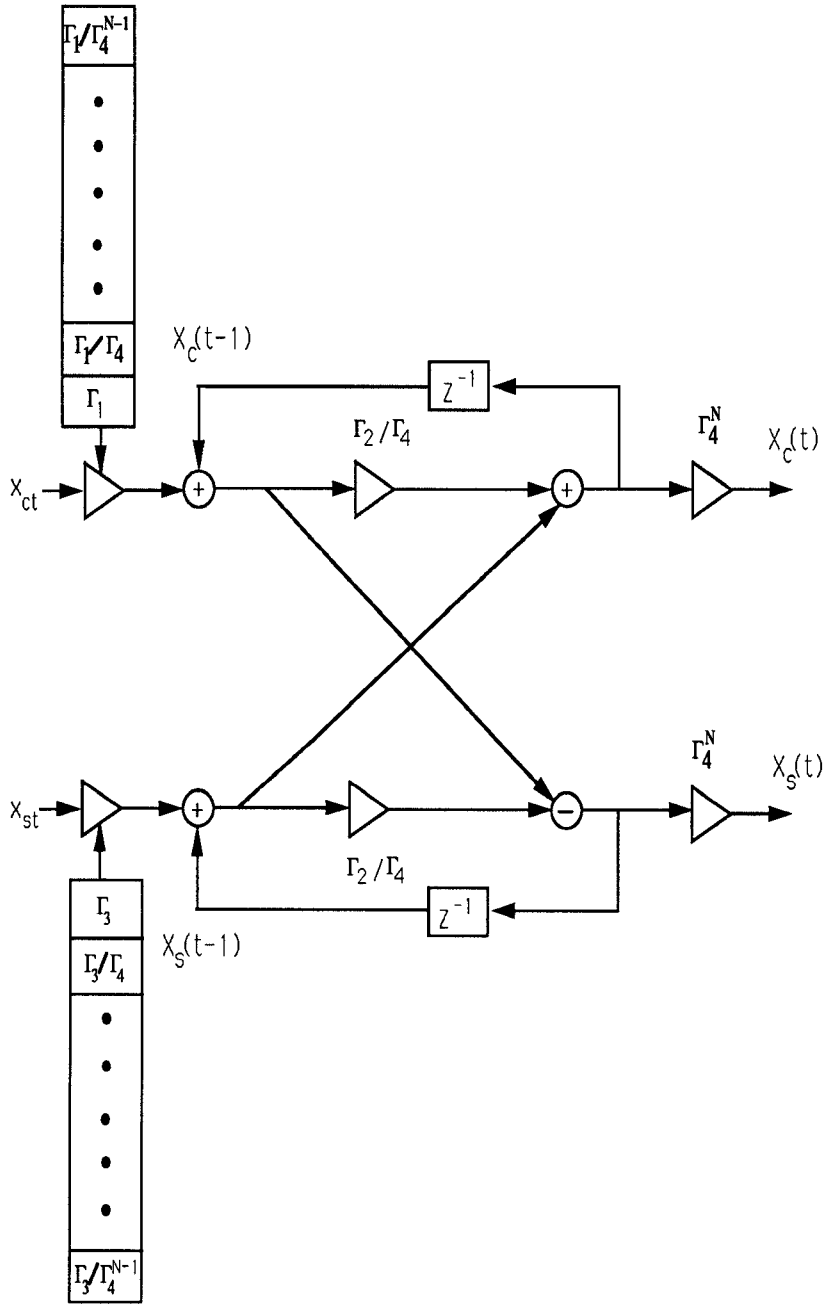


Figure 2.13: The multiplier-reduced lattice module.

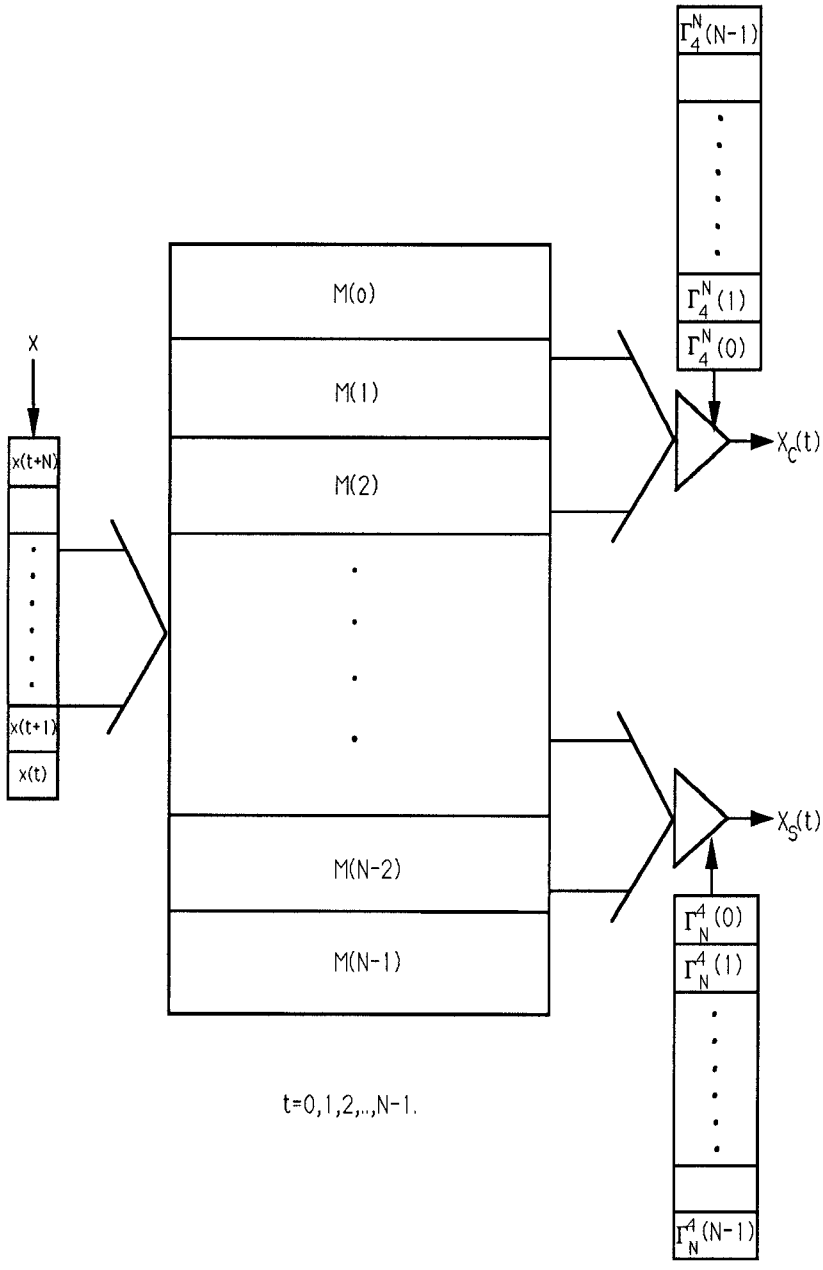


Figure 2.14: The complete parallel multiplier-reduced lattice structure.

may continue to accumulate and lower the signal-to-noise ratio. This effect can be minimized by giving enough register length such as double precision in the implementation. Also, we note that since $\Gamma_4 < 1$, Γ_4^N could be very small. Not enough precision may result in bad numerical accuracy when Γ_4^N is multiplied at the output stage. Thus, the registers that store Γ_4^N do need enough precision to avoid the accuracy problem. The problems addressed here are consequences of the tradeoff between complexity and performance.

2.6.2 Double-lattice Approach

Generally, a post-lattice structure has the following forms

$$\begin{aligned} X_c(k) &= \Gamma_2 X_c(k-1) + \Gamma_4 X_s(k-1) + \Gamma_1 x_{ck}, \\ X_s(k) &= \Gamma_2 X_s(k-1) - \Gamma_4 X_c(k-1) + \Gamma_3 x_{sk}. \end{aligned} \tag{2.70}$$

Based on the relationships shown below

$$\begin{aligned} &\Gamma_2 X_c(k-1) + \Gamma_4 X_s(k-1) = \\ &\quad \frac{1}{2}\left\{(\Gamma_2+\Gamma_4)[X_c(k-1)+X_s(k-1)] + (\Gamma_2-\Gamma_4)[X_c(k-1)-X_s(k-1)]\right\}, \end{aligned} \tag{2.71}$$

and

$$\begin{aligned} &\Gamma_2 X_s(k-1) - \Gamma_4 X_c(k-1) = \Gamma_2 X_s(k-1) + \Gamma_4 X_c(k-1) - 2\Gamma_4 X_c(k-1) \\ &\quad = \frac{1}{2}\{(\Gamma_2+\Gamma_4)[X_c(k-1)+X_s(k-1)] - (\Gamma_2-\Gamma_4)[X_c(k-1)-X_s(k-1)]\} \\ &\quad -2\Gamma_4 X_c(k-1), \end{aligned} \tag{2.72}$$

$X_c(k)$ and $X_s(k)$ can be rearranged in the following manner

$$X_c(k) = \frac{1}{2}(\Gamma_2+\Gamma_4)[X_c(k-1)+X_s(k-1)] \tag{2.73}$$

$$
\begin{aligned}
& + \frac{1}{2}(\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)] \\
& + \Gamma_1 x_{ck} \\
& = \frac{1}{2}\{t1 + t2\} + \Gamma_1 x_{ck}, \\
X_s(k) \;=\; & \frac{1}{2}(\Gamma_2 + \Gamma_4)[X_c(k-1) + X_s(k-1)] \qquad (2.74) \\
& - \frac{1}{2}(\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)] \\
& - 2\Gamma_4 X_c(k-1) + \Gamma_3 x_{ck} \\
& = \frac{1}{2}\{t1 - t2\} - 2\Gamma_4 X_c(k-1) + \Gamma_3 x_{ck}. \qquad (2.75)
\end{aligned}
$$

where

$$t1 = (\Gamma_2 + \Gamma_4)[X_c(k-1) + X_s(k-1)], \qquad (2.76)$$

and

$$t2 = (\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)]. \qquad (2.77)$$

The operational flow graph of (2.73) is illustrated in Fig. 2.15. Instead of calculating the outputs from (2.70) directly (that requires 6 multipliers and 4 adders), the first lattice adds and subtracts $X_c(k-1)$ and $X_s(k-1)$, then multiplies the results by $\Gamma_2 + \Gamma_4$ and $\Gamma_2 - \Gamma_4$ respectively. The results are called $t1$ and $t2$ as defined in (2.76) and (2.77). The second lattice adds and subtracts $t1$ and $t2$ again, then divides the results by 2, which can be achieved by right shifting. Finally, we complete the computations by adding the inputs $\Gamma_1 x_{ck}$ and $\Gamma_3 x_{ck} - 2\Gamma_4 X_c(k-1)$. This reconstruction can save one multiplier. A parallel post-lattice structure with N lattice modules requires $6N$ multipliers and $4N$ adders. As for this reconstructed parallel structure, only $5N$ multipliers and $7N$ adders are needed. This approach can be applied to all the parallel post-lattice structures of different orthogonal transforms. In general, this parallel

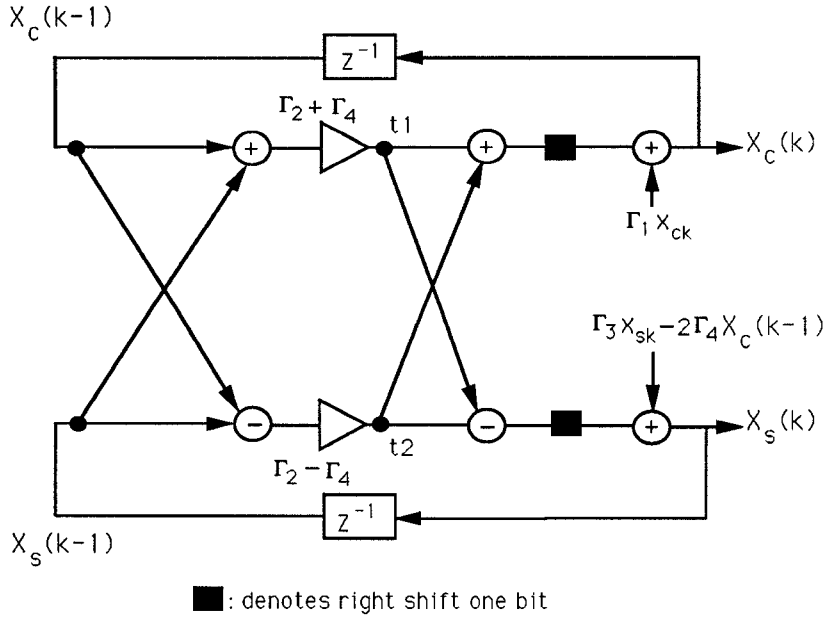


Figure 2.15: The double-lattice form of the post-lattice realization.

double-lattice structure can save N multipliers, but requires $3N$ more adders. The latency is N clock cycles and the system remains $SIPO$.

2.7 Comparisons of Architectures

From the previous discussions, we see that the proposed unified parallel lattice structures have many attractive features. There are no constraints on the transform size N. It dually generates the two discrete transforms DCT and DST simultaneously. Since it produces the transformed data of subsequent input vector every clock cycle, it is especially efficient for systems with series input data such as communication systems. Further, the structure is regular, modular, and without global communication. As a consequence, it is suitable for VLSI implementation.

	LiuChiu1	LiuChiu2	chen [14] *et. al.*	Lee[2]	Hou[5]
No. of Multipliers	$6N-4$	$4N$	$N\ln(N)$ $-3N/2+4$	$(N/2)\ln(N)$	$N-1$
latency	N	$2N$	$N/2$	$\frac{1}{2}[\ln(N)*$ $(\ln(N)-1)]$	$O(3N/2)$
limitation on N	no	no	powerof 2	power of 2	power of 2
commun.	local	local	global	global	global
I/O	*SIPO*	*SISO*	*PIPO*	*PIPO*	*SIPO*

Table 2.1: Comparison of different DCT algorithms

Here, we would like to compare our lattice structures of the DCT and DST with those proposed in [13, 15, 7]. The architecture in [13] uses the matrix factorization method which is a representative of fast algorithms. In [15], an improved fast structure with fewer multipliers is proposed. Hou's architecture in [7] uses recursive computations to generate the higher order DCT from the lower order one. The characteristics of these structures are discussed in the introduction. A comparison regarding their inherent properties is listed in Table 2.1. To be clear, the quantitative comparisons in terms of the parameters, which are the numbers of multipliers, adders, and the latency, are given in Table 2.2, Table 2.3, and Table 2.4.

The lattice architecture with six multipliers in the module as shown in Fig. 2.2 is called Liu-Chiu1 structure, the one in Fig. 2.14 is called Liu-Chiu2, and

NO	Liuchiu1	LiuChiu2	Chen	Lee	Hou
8	44/2	32/2	16	12	7
16	92/2	64/2	44	32	15
32	188/2	128/2	116	80	31
64	380/2	256/2	292	192	63

Table 2.2: Comparision of the number of multipliers

NO	Liu-Chiu1	Liu-Chiu2	Chen	Lee	Hou
8	39/2	39/2	26	29	18
16	79/2	79/2	74	81	41
32	159/2	159/2	194	209	88
64	319/2	319/2	482	513	183

Table 2.3: Comparision of the number of adders

NO	Liu-Chiu1	Liu-Chiu2	Chen	Lee	Hou
8	8	16	4	6	13
16	16	32	6	10	21
32	32	64	8	15	44
64	64	128	10	21	73

Table 2.4: Comparision of the latency

the parallel structure with the double-lattice modules as shown in Fig. 2.15 is called Liu-Chiu3. The structure in Liu-Chiu1 has $6N-4$ multipliers, $5N-1$ adders, and the latency is N. There are $4N$ multipliers, $5N-1$ adders, and the latency is $2N$ in the structure of Liu-Chiu2. The number of multipliers is reduced by the order $2N$ in the expense of doubling the latency and the data flow becoming $SISO$. The Liu-Chiu3 architecture has $5N$ multipliers and $7N$ adders and the latency is N clock cycles. From these Tables, it is noted that the number of multipliers in our architectures is higher than that of others when N is small. This is due to the dual generation of two transforms structure which is compatible with Lee's. Since the numbers of multipliers and adders of our structures are on the order N, our algorithms have fewer multipliers and adders than those proposed in [13, 15]. Although Hou's algorithm has the fewest multipliers, his architecture needs global communications and the design complexity is mu of other structures can not start until all of the data in the block arrive.

A comparison for our DHT structure based on the lattice module in Fig. 2.7 and different DHT algorithms [52, 69] is listed in Table 2.5. The architecture in [52], a representative fast algorithm, is developed base on the existing FFT method. Chaitali-JaJa's algorithm in [69] decomposes the transform size N into mutually prime numbers and implements them in a systolic manner. Their structure needs extra registers and the latency is higher than others. It is easy to see that our structure is better than others in terms of hardware complexity and speed.

	Liu-Chiu	Sorenson[23]	Chaitali-JaJa[18]
No. of Multipliers	$4N$	$N\ln(N)$ $-3N+4$	$N1+N2$
Adders	$5N-2$	$3N\ln(N)$ $-3N+4$	$4N$ $+\sqrt{N1}$
latency	N	$N\ln(N)$	$N1+N2$
limitation on N	no	power of 2	$N=N1*N2$, $N1, N2$ are m.p.
commun.	local	global	local
I/O	$SIPO$	$PIPO$	$SISO$

Table 2.5: Comparision of different DHT algorithms,*m.p. means "mutual prime".

2.8 Filter Bank Interpretation

Multirate digital filters and filter banks find applications in communications, speech processing, and image compression. There are two basic types of filter banks. An analysis bank is a set of analysis filters $H_k(z)$ and N-fold decimators which split a signal into N subbands. A synthesis filter bank (the right part of Fig. 2.16) consists of N synthesis filters $F_k(z)$ and N-fold interpolators, which combine N signals into a reconstructed signal $\hat{x}(n)$. As described in Section 2, the time-recursive approach decomposed the transformed domain data into N different components. If we are interested in the block-size-N transform and perform the N-fold decimation in the outputs of every lattice modules, the analysis bank is simply the series-input-parallel-output filter bank described in Fig. 2.16. Under this condition, the analysis bank is equivalent to perform a transformation and the synthesis bank to perform an inverse transformation on successive blocks of N data samples. In this section, we describe how to employ the time-recursive concept to generate the synthesis banks based on the DCT, DST and DHT.

2.8.1 Synthesis bank structure based on DCT

To perform the inverse transform in the synthesis bank, we feed the DCT transformed domain components $X_c(k)$ into the synthesis modules and combine all the outputs of every synthesis modules to produce the original input sequence $x_c(n)$. That is, the synthesis bank performs the following inverse DCT operations

$$x_c(n) = \sum_{k=0}^{N-1} C(k) X_c(k) \cos\left[\frac{\pi k(2n+1)}{2N}\right]. \tag{2.78}$$

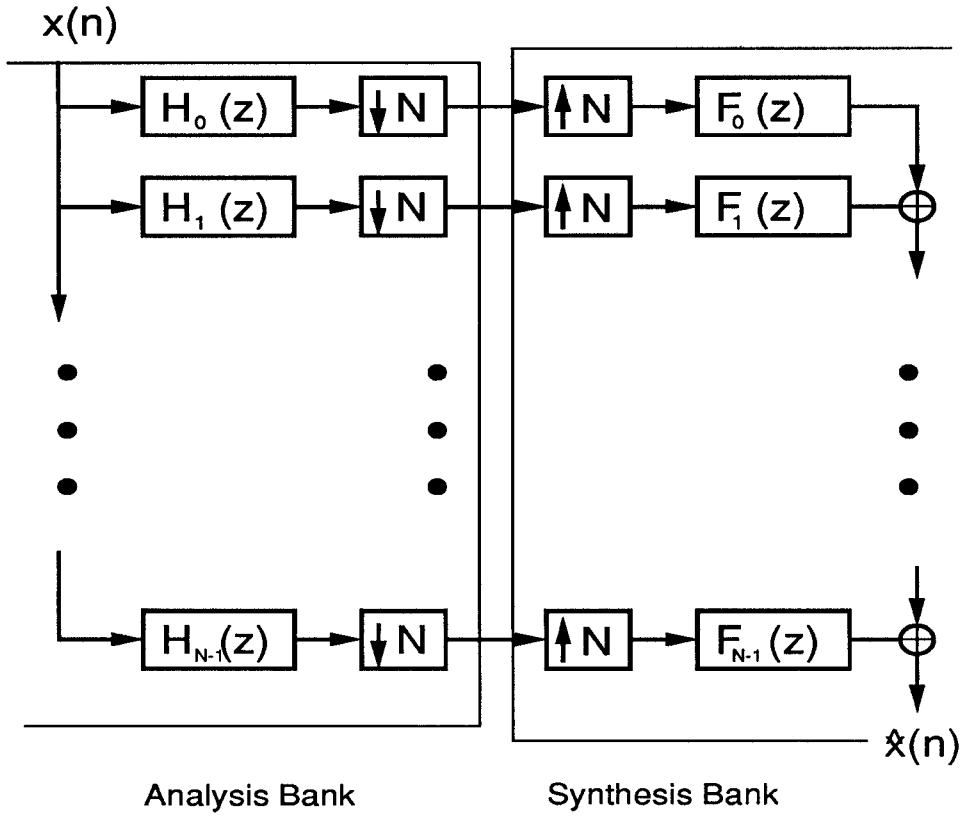


Figure 2.16: The filter bank structure.

Since in the synthesis bank different transform components are sent to independent synthesis modules, we therefore focus on a specific transform component. Denote $\overline{x}_c(n,k)$ as the output signal generated by a specific synthesis module

$$\overline{x}_c(n,k) = C(k)X_c(k)\cos\left[\frac{\pi k(2n+1)}{2N}\right]. \tag{2.79}$$

The time-recursive concept can be applied here to update $\overline{x}_c(n,k)$ recursively. Use the result in section 2.3.1 that IDCT and AIDST can be generated from each other recursively and denote $\overline{x}_{as}(n,k)$ as the auxiliary inverse sine transform generated by a specific synthesis filter. We can obtain the following recursive-generated relations for $\overline{x}_c(n,k)$ and $\overline{x}_{as}(n,k)$ as

$$\begin{aligned}\overline{x}_c(n+1,k) &= C(k)X_c(k)\cos\left[\frac{\pi k[2(n+1)+1]}{2N}\right] \\ &= \overline{x}_c(n,k)\cos\left(\frac{\pi k}{2N}\right) - \overline{x}_{as}(n,k)\sin\left(\frac{\pi k}{2N}\right),\end{aligned} \tag{2.80}$$

and

$$\begin{aligned}\overline{x}_{as}(n+1,k) &= C(k)X_c(k)\sin\left[\frac{\pi k[2(n+1)+1]}{2N}\right] \\ &\overline{x}_{as}(n,k)\cos\left(\frac{\pi k}{2N}\right) + \overline{x}_c(n,k)\sin\left(\frac{\pi k}{2N}\right).\end{aligned} \tag{2.81}$$

(2.80) and (2.81) suggest that the $\overline{x}_c(n+1,k)$ and $\overline{x}_{as}(n+1,k)$ can be dually generated from the previous values $\overline{x}_c(n,k)$ and $\overline{x}_{as}(n,k)$ in a lattice form as shown in Fig. 2.17. Because the initial values for $x_c(0,k)$ and $x_{as}(0,k)$ are

$$\overline{x}_c(0,k) = X_c(k)\cos\left[\frac{\pi k}{2N}\right], \tag{2.82}$$

and

$$\overline{x}_{as}(0,k) = X_{as}(k)\sin\left[\frac{\pi k}{2N}\right], \tag{2.83}$$

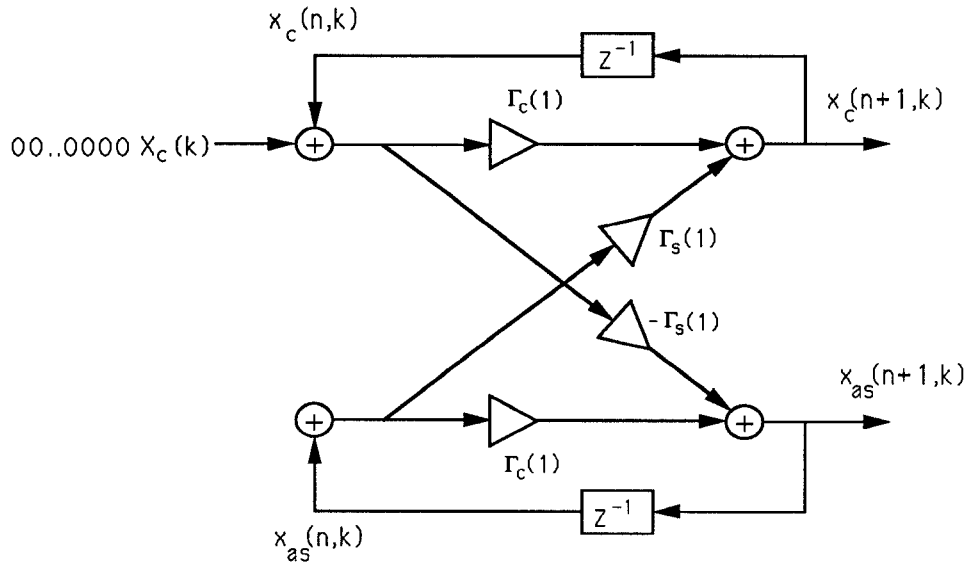


Figure 2.17: The synthsis bank structure of the DCT.

this means that the $x_c(n+1,k)$ and $x_{as}(n+1,k)$ can be generated by sending a sequence with $X_c(k)$ as the first element followed by $N-1$ zeros into the input of the synthesis module. This is exactly the up sampling procedure required in the synthesis bank structure. The $\overline{x}_{as}(n,k)$ output is reset every N clock cycles. The synthesis module diagram for the DCT is plotted in Fig. 2.17. The inverse transform is obtained by summing all the outputs of the synthesis modules.

2.8.2 Synthesis bank structure of the DST and DHT

In this section, we apply the same approach mentioned in the previous section to the DST and DHT. The results are summaried as below. By using the dual generation concept, the operation of the synthesis module for the DST is

$$\overline{x}_s(n+1,k) = D(k)X_s(k)\sin\left[\frac{\pi k[2(n+1)+1]}{2N}\right]$$

$$= \overline{x}_s(n,k)\cos\left(\frac{\pi k}{2N}\right) + \overline{x}_{ac}(n,k)\sin\left(\frac{\pi k}{2N}\right), \tag{2.84}$$

and

$$\begin{aligned}\overline{x}_{ac}(n+1,k) &= D(k)X_s(k)\cos\left[\frac{\pi k[2(n+1)+1]}{2N}\right] \\ &= \overline{x}_{ac}(n,k)\cos\left(\frac{\pi k}{2N}\right) - \overline{x}_s(n,k)\sin\left(\frac{\pi k}{2N}\right).\end{aligned} \tag{2.85}$$

Because $D(k)$ and $C(k)$ have the same values for $k = 1, 2, ..., N-1$ and $D(N) = C(0)$. Therefore, the structure of the synthesis modules for the DST are the same as that for the DCT except for $k = N$.

As for the DHT, the IDHT is defined as

$$\begin{aligned}x_h(n) &= \sum_{k=0}^{N-1} X_h(k)cas\left(\frac{2\pi kn}{N}\right) \\ &= \sum_{k=0}^{N-1} X_h(k)\left[\cos\left(\frac{2\pi kn}{N}\right) + \sin\left(\frac{2\pi kn}{N}\right)\right], \\ &\quad n = 0, 1, ..., N-1.\end{aligned} \tag{2.86}$$

Again, we separate them into a combination of a DCT-like and a DST-like transforms as follows:

$$x_h(n) = \acute{x_c}(n) + \acute{x_s}(n). \tag{2.87}$$

The operation of the synthesis module for the DHT is generated from $\acute{x_c}(n)$ and the $\acute{x_s}(n)$ by the following relation

$$\acute{x_c}(n+1,k) = \acute{x_c}(n,k)\cos\left(\frac{2\pi k}{N}\right) - \acute{x_s}(n,k)\sin\left(\frac{2\pi k}{N}\right), \tag{2.88}$$

and

$$\acute{x_s}(n+1,k) = \acute{x_s}(n,k)\cos\left(\frac{2\pi k}{N}\right) - \acute{x_c}(n,k)\sin\left(\frac{2\pi k}{N}\right). \tag{2.89}$$

To obtain the IDHT $x_h(n)$, we must sum up both of the outputs of the synthesis modules. It is noted that the multiplier coefficients in the synthesis module for the DHT is different from that of the DCT and DST.

2.9 Summary

In this Chapter, unified time-recursive algorithms and lattice structures that can be applied to the DCT, DST, DHT and their inverse transforms, are considered. In fact, there are various forms of sin and cosine transform pairs, (the DCTI/DSTI, DCTII/DSTII, DCTIII/DSTIII, DCTIV/DSTIV, and Complex Lapped Transform(CLT)) as mentioned in [51, 62]. They also have their time-recursive lattice realizations. The procedures to attain the lattice structures of different transforms are similar and the resulting $SIPO$ lattice structures differ only in the multiplying coefficients and the input stage. All the transform pairs have their pre- and post-lattice realizations that differ in that the input signals are added in the front and the end of the lattice respectively. The hardware complexity of the pre-lattice realizations and their post-lattice counterparts depends on the definitions of the transforms and it cannot be readily determined which one is better. The number of multipliers in all the parallel lattice structures is a linear function of the transform size N and the latency is N clock cycles. Two methods, the $SISO$ and double-lattice approaches, are developed to reduce the number of multipliers for the parallel lattice structures. The $SISO$ approach can reduce $2N$ multipliers and the latency becomes $2N$. The double-lattice approach can reduce N multipliers and the latency remains intact. From the discussion of the block processing, it is noted that the area-time complexity is efficient when the block size m is small, especially when $m = 1$. All the resulting parallel structures are module, regular, and only locally connected. Further, there is no constraint on the transform size N. It is obvious that the design complexity of these structures is relatively low compared with other algorithms. The characteristics of these algorithms are suitable for processing series input

data since the transformed data for sequential input can be obtained every clock cycle. Therefore, it is very attractive to VLSI implementations and high speed applications such as HDTV signal coding and transmission.

Since the orthogonal rotation is the major operation in the lattice, it is noted that such rotation can be easily implemented using CORDIC (COordinate Rotation DIgital Computer) [58, 59] which is known as an efficient method for the computation of orthogonal rotations and trigonometric functions.

Chapter 3

Two-Dimensional DCT Lattice Structures

The two-dimensional discrete cosine transform (2-D DCT) has been widely recognized as the most effective technique in image data compression, especially for high-speed video processing applications, such as "HDTV." In this Chapter, we propose a new fully-pipelined architecture to compute the 2-D DCT from a frame-recursive point of view. Based on this approach, two real-time parallel lattice structures for successive frame and block 2-D DCT are developed. These structures are fully-pipelined with throughput rate N clock cycles for a $N \times N$ successive input data frame. These are the fastest pipelined structures known so far. Moreover, the resulting 2-D DCT architectures are modular, regular, and locally-connected and require only two 1-D DCT blocks which are extended directly from the 1-D DCT structure without transposition. It is therefore very suitable for VLSI implementation for high speed HDTV systems. We also propose a parallel 2-D DCT architecture and a new scanning pattern for HDTV systems to achieve higher performance. The VLSI implementation of the 2-D DCT using distributed arithmetic to increase computational efficiency and re-

duce round off error is also discussed.

3.1 Introduction

It is well known that block coding based on the two dimensional (2-D) DCT produces highly compact 2-D transformed coefficients on the spatial domain [44]. By applying appropriate bit allocations and entropy coding schemes, *i.e.*, variable length coding and run length coding [41, 36], the bit rate of the HDTV systems can be greatly reduced [29, 36]. Because of the hardware limitations in practical applications, only small transform block size (typically 8 by 8 or 16 by 16) is used.

Many 2-D DCT algorithms have been proposed to reduce the computational complexity and to increase the operational speed. These algorithms can be divided into two groups, the row-column method and the direct 2-D method. The row-column method computes the 2-D DCT by applying the one-dimensional (1-D) DCT on the rows (or columns) of the input data frames, storing the transformed results in an intermediate matrix, transposing the matrix, and performing the 1-D DCT again on the columns (or rows) of the transposed matrix. The 2-D DCT, then, is decomposed into two 1-D DCTs. Since there exist many 1-D DCT algorithms, there are also many realizations for the row-column methods. The performance of a specific row-column method depends on the realization of the 1-D DCT algorithm. The classification of the 1-D DCT algorithms and their comparisons are given in [17]. The systolic array approaches, which are assumed to be fully-pipelined, also employ the row-column method in the 2-D DCT realization [15, 18]. Therefore, the fully-pipelined operation is impaired

since the second DCT can not start until the transposition of the first transform coefficients finishes.

The direct 2-D method, in contrast to the row-column decomposition method, is a complete 2-D approach. Duhamel and Guillemot [80] proposed two direct 2-D methods, the indirect and direct polynomial transforms (PT) for the 2-D DCT. Vetterli [79] used the indirect PT approach for 2-D DCT to map the N by N DCT into a real N by N DFT followed by a number of rotations, and the real N by N DFT can be realized by using PT. The direct PT for the 2-D DCT is shown to be more effective than the indirect approach [20], but the procedure for a general implementation is complicated. It has also been shown [79] that these kinds of direct 2-D methods are more efficient than the row-column methods. However, due to their simplicity in hardware realization, row-column decomposition methods are still widely adopted in implementing the 2-D DCT chips [43, 29].

All the approaches mentioned above do have a common requirement: the availability of all the data in the processing 2-D block. This may not be true for the real-time data transmission systems such as HDTV systems. To eliminate the waiting time for the data to arrive, a time-recursive processing concept can be exploited, *i.e.*, the result is adaptively updated when a new datum arrives. Once all the data arrive, the result is completely available. One of the most important issues here is to design a real-time VLSI system that is compatible with the data transmission speed. In this paper, we show that frame-recursive architecture is a feasible solution. Liu and Chiu [17] proposed new unified parallel lattice structures for time-recursive 1-D orthogonal sinusoidal transforms. These transforms are decoupled into N independent lattice modules, hence, there are

no global communications in the structure. Moreover, every lattice module is regular and modular and has the same architecture. The number of multipliers of the lattice structure for the best case in [17] is $4N$. Therefore, it is very suitable for VLSI implementation.

In this Chapter, we propose a new architecture for the 2-D DCT by employing the frame-recursive concept on the successive input frames. It is a direct 2-D method and the transposition in the row-column method is eliminated. Because the system requires only two 1-D lattice DCT module arrays, the hardware complexity of this system equals that of the row-column method for series rows (or columns) input frame. All the components needed in the architecture are independent lattice modules and shift registers. The total number of multipliers required for an N by N 2-D DCT is $8N$. There is no limitations on the transform size N and the structure can be easily extended to any number N, including the prime numbers. This is a rather promising architectures either from speed or hardware point of view. Since the system is modular, regular, and fully-pipelined, it is very suitable for high speed video signal transmission. We show that by employing distributed arithmetic technique in hardware implementation, the system performance is further improved. For the HDTV applications, very high operating speed is achieved by using parallel processings and appropriate scanning arrangements.

We organize the rest of the Chapter as follows. In Section 3.2, the algorithm to achieve the 2-D DCT using the frame-recursive manner is proposed. It is shown that we can dually generate a 2-D discrete sine-cosine transform (DSCT) simultaneously. The architectures for calculating moving frame 2-D DCT and block 2-D DCT are discussed in Section 3.3. Comparisons of different 2-D DCT

algorithms are given in Section 3.4. In Section 3.5, we consider the application of our 2-D DCT block architecture to the high speed HDTV system. Finally, the conclusion is given in Section 3.6.

3.2 Dual Generation of 2-D DCT and DSCT

In this section, we describe a new architecture for 2-D DCT that requires only two 1-D DCT arrays. Focusing directly on the 2-D transformed signal and applying the frame recursive approach, we can derive not only the frame recursive relation of two successive frames of the 2-D DCT, but also the dual generation properties between the 2-D DCT and 2-D discrete sine-cosine transform (DSCT). Here the DSCT serves as an auxiliary transform which supports the time-recursive computations of the 2-D DCT.

3.2.1 Frame-Recursive 2-D Discrete Cosine Transform

The $N \times N$ 2-D DCT $\{X_c(k,l,t) : k,l = 0,1,...,N-1.\}$ of an $N \times N$ 2-D data sequence $\{x(m,n) : m = t, t+1, ..., t+N-1; n = 0,1,...,N-1.\}$ is defined as

$$X_c(k,l,t) = \frac{4}{N^2} C(k)C(l) \sum_{m=t}^{t+N-1} \sum_{n=0}^{N-1} x(m,n) \cos\left[\frac{\pi[2(m-t)+1]k}{2N}\right] \cdot \cos\left[\frac{\pi(2n+1)l}{2N}\right]. \quad (3.1)$$

Here the time index t in $X_c(k,l,t)$ denotes that the transform starting from the t'th row of the 2-D data sequence $\{x(m,n) : m = 0,1,2,....; n = 0,1,...,N-1.\}$ as shown in Fig. 3.1. In the following, we call $X_c(k,l,t)$ the 2-D DCT of the t'th frame of the 2-D data sequence $x(m,n)$. To derive the time-recursive relations between the successive data frames, let us start by considering the 2-D DCT of

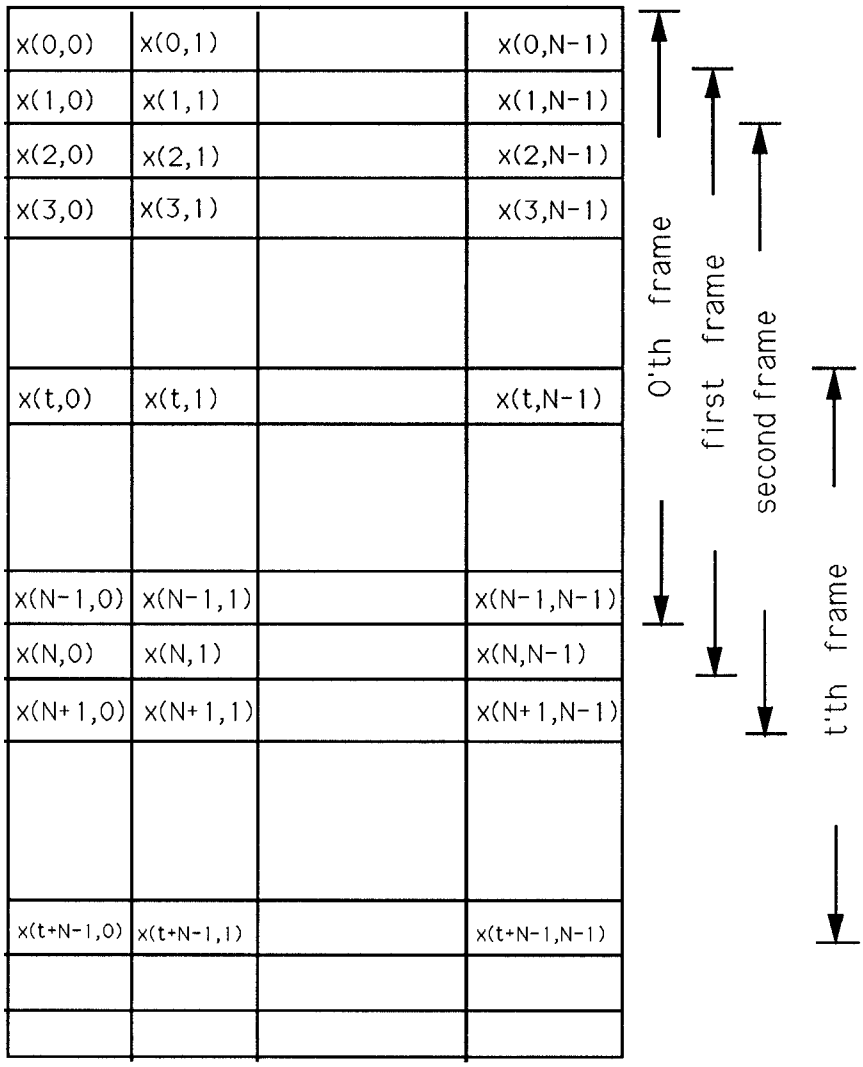


Figure 3.1: The 2-D successive data frame.

the t'th frame data sequence,

$$X_c(k,l,t) = \frac{4}{N^2}C(k)C(l)\sum_{m=t}^{t+N-1}\sum_{n=0}^{N-1}x(m,n)\cos\left[\frac{\pi(2(m-t)+1)k}{2N}\right] \cdot\cos\left[\frac{\pi(2n+1)l}{2N}\right]. \quad (3.2)$$

Instead of focusing on $X_c(k,l,t)$ and utilizing various techniques to reduce the computational complexity. We will consider the 2-D DCT sequence of the $(t+1)$'th frame, which is

$$X_c(k,l,t+1) = \frac{4}{N^2}C(k)C(l)\sum_{m=t+1}^{t+N}\sum_{n=0}^{N-1}x(m,n)\cos\left[\frac{\pi[2(m-t-1)+1]k}{2N}\right] \cdot\cos\left[\frac{\pi(2n+1)l}{2N}\right]. \quad (3.3)$$

By using trigonometric function expansions on $\cos\left[\frac{\pi[2(m-t-1)+1]k}{2N}\right]$, (3.3) can be rewritten as

$$X_c(k,l,t+1) = \overline{X}_c\cos\left(\frac{\pi k}{N}\right) + \overline{X}_{sc}\sin\left(\frac{\pi k}{N}\right), \quad (3.4)$$

where

$$\overline{X}_c = \frac{4}{N^2}C(k)C(l)\sum_{m=t+1}^{t+N}\sum_{n=0}^{N-1}x(m,n)\cos\left[\frac{\pi[2(m-t)+1]k}{2N}\right]\cos\left[\frac{\pi(2n+1)l}{2N}\right], \quad (3.5)$$

and

$$\overline{X}_{sc} = \frac{4}{N^2}C(k)C(l)\sum_{m=t+1}^{t+N}\sum_{n=0}^{N-1}x(m,n)\sin\left[\frac{\pi[2(m-t)+1]k}{2N}\right]\cos\left[\frac{\pi(2n+1)l}{2N}\right]. \quad (3.6)$$

We can view the term $\sin\left[\frac{\pi[2(m-t)+1]k}{2N}\right]\cos\left[\frac{\pi(2n+1)l}{2N}\right]$ in (3.6) as a new transform kernel, and define a $N \times N$ 2-D discrete sine-cosine transform (DSCT) sequence $\{X_{sc}(k,l,t) : k = 1,2,...,N; l = 0,1,...,N-1.\}$ of the 2-D data sequence $\{x(m,n) : m = t,t+1,...,N+t-1; n = 0,1,...,N-1.\}$ as

$$X_{sc}(k,l,t) = \frac{4}{N^2}C(k)C(l)\sum_{m=t}^{N+t-1}\sum_{n=0}^{N-1}x(m,n)\sin\left[\frac{\pi[2(m-t)+1]k}{2N}\right] \quad (3.7)$$

$$\cdot \cos\left[\frac{\pi(2n+1)l}{2N}\right].$$

Here we extend the definition of $C(k)$ to $C(N)$ and define $C(N) = \frac{1}{\sqrt{2}}$. Similarly, we are interested in the 2-D DSCT of the $(t+1)$'s frame. According to the definition, it is

$$\begin{aligned} X_{sc}(k,l,t+1) &= \frac{4}{N^2}C(k)C(l)\sum_{m=t+1}^{t+N}\sum_{n=0}^{N-1}x(m,n) \qquad (3.8)\\ &\quad \cdot \sin\left[\frac{\pi[2(m-t-1)+1]k}{2N}\right]\cos\left[\frac{\pi(2n+1)l}{2N}\right] \\ &= \overline{X}_{sc}\cos\left(\frac{\pi k}{N}\right) - \overline{X}_{c}\sin\left(\frac{\pi k}{N}\right), \qquad (3.9) \end{aligned}$$

where the terms $\overline{X}_c$ and $\overline{X}_{sc}$ used in (3.4) to generate $X_c(k,l,t+1)$ appear again. This suggests that the 2-D DCT and 2-D DSCT can be dually generated from each other.

3.2.2 Lattice structures for frame-recursive 2D-DCT

We will show in this section that 2D-DCT can be generated by using two lattice arrays. From (3.4) and (3.8), it is noted that the new 2-D DCT and DSCT transforms can be obtained from the intermediate values $\overline{X}_c$ and $\overline{X}_{sc}$ in a lattice form as shown in Fig. 3.2. A similar relation also exists in the dual generations of the 1-D DCT and 1-D DST [17]. The intermediate data $\overline{X}_c$ and $\overline{X}_{sc}$ differ from the original signal $X_c(k,l,t)$ and $X_{sc}(k,l,t)$ only in the t'th row and the $(t+N)$'th row of the input data frames. So, the intermediate data $\overline{X}_c$ and $\overline{X}_{sc}$ can be obtained from $X_c(k,l,t)$ and $X_{sc}(k,l,t)$ by removing the t'th row of the transformed data and updating the $(t+N)$'th row of the transformed data. That is,

$$\overline{X}_c = X_c(k,l,t) - \frac{4}{N^2}C(k)C(l)\sum_{n=0}^{N-1}x(t,n)\cos\left[\frac{\pi(2n+1)l}{2N}\right]\cos\left(\frac{\pi k}{2N}\right)$$

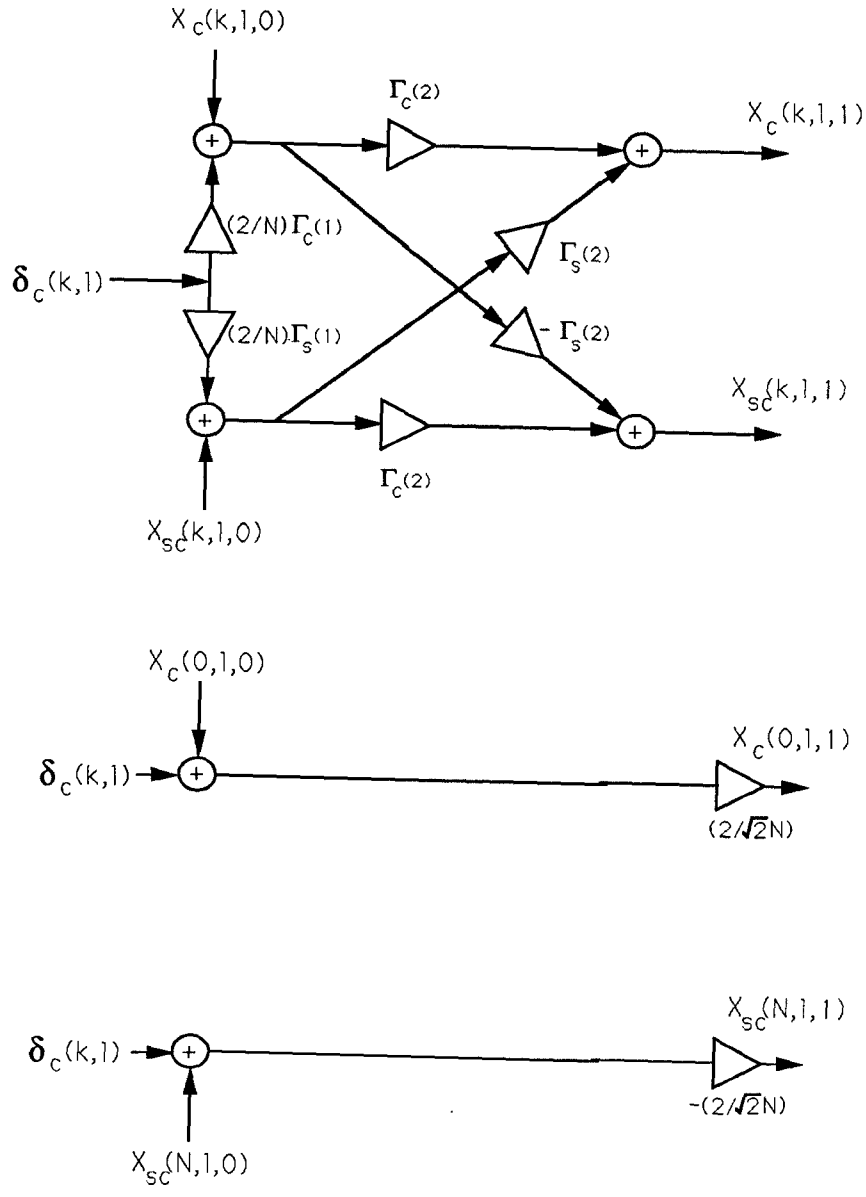


Figure 3.2: The lattice module of lattice array II.

$$+\frac{4}{N^2}C(k)C(l)\sum_{n=0}^{N-1}x(t+N,n)\cos\left[\frac{\pi(2n+1)l}{2N}\right]\cos\left[\frac{\pi(2N+1)k}{2N}\right] \quad (3.10)$$

and

$$\overline{X}_{sc} = X_{sc}(k,l,t) - \frac{4}{N^2}C(k)C(l)\sum_{n=0}^{N-1}x(t,n)\cos\left[\frac{\pi(2n+1)l}{2N}\right]\sin\left(\frac{\pi k}{2N}\right)$$
$$+\frac{4}{N^2}C(k)C(l)\sum_{n=0}^{N-1}x(t+N,n)\cos\left[\frac{\pi(2n+1)l}{2N}\right]\sin\left[\frac{\pi(2N+1)k}{2N}\right] \quad (3.11)$$

These two equations can be further simplified as

$$\overline{X}_c = X_c(k,l,t) + \delta_c(k,l,t)\frac{2}{N}C(k)\cos\left(\frac{\pi k}{2N}\right), \quad (3.12)$$

and

$$\overline{X}_{sc} = X_{sc}(k,l,t) + \delta_c(k,l,t)\frac{2}{N}C(k)\sin\left(\frac{\pi k}{2N}\right), \quad (3.13)$$

where

$$\delta_c(k,l,t) = \frac{2}{N}C(l)\sum_{n=0}^{N-1}\left[(-1)^k x(t+N,n) - x(t,n)\right]\cos\left[\frac{\pi(2n+1)l}{2N}\right]. \quad (3.14)$$

By substituting $\overline{X}_c$ and $\overline{X}_{sc}$ in (3.12) and (3.13) into the updated transformed signal $X_c(k,l,t+1)$ and $X_{sc}(k,l,t+1)$ in (3.4) and (3.8), the relation between the updated transformed signal and previous transformed signal for $k = 1, 2, .., N-1$ are represented in a lattice form as shown in the upper part of Fig. 3.2. Since the multiplications can be reduced to addition and subtraction for $k = 0$ in the 2-D DCT and $k = N$ in the DSCT respectively, these two cases can be simplified as shown in the lower part of Fig. 3.2. What is worth noticing is that $\delta_c(k,l,t)$ in (3.14) is the 1-D DCT of the data vector which is the difference between the parity of the t'th row and $(t+N)$'th row of the 2-D input sequence. As described in [17], $\delta_c(k,l,t)$ can be generated in a lattice form as shown in Fig. 3.3. We call this as lattice array I (LAI) and that in Fig. 3.2 as lattice array II ($LAII$).

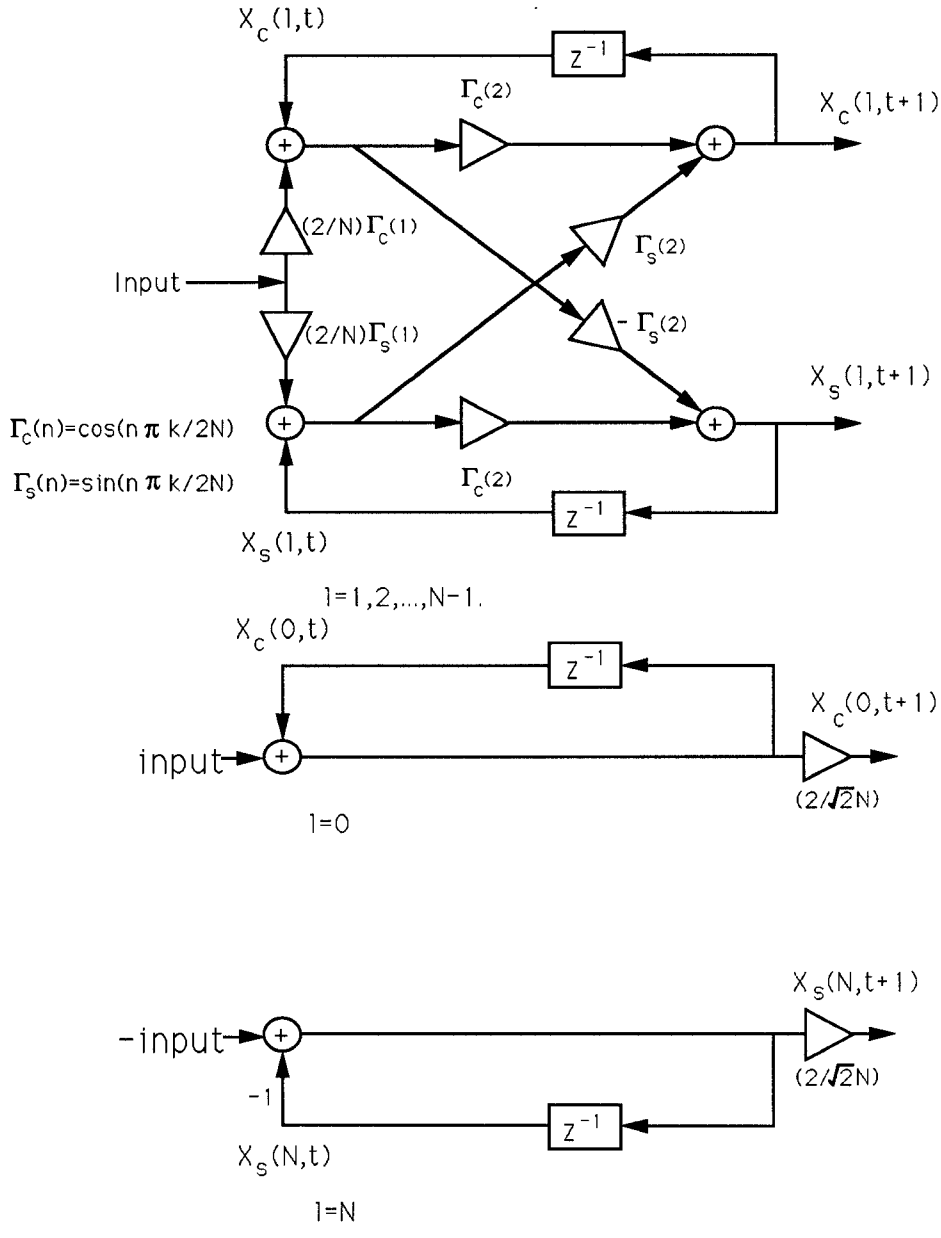


Figure 3.3: The lattice module of lattice array I.

Comparing these two structures, we observe that these two lattice modules have the same architecture except that LAI feedbacks the outputs directly through a delay stage to add with the inputs.

The 2-D DCT and DSCT are produced by applying input data frames to LAIs which generate the $\delta_c(k,l,t)$. After obtaining the $\delta_c(k,l,t)$, the updated transformed signal can be obtained recursively by feeding $\delta_c(k,l,t)$ into the lattice array II. We observe that the 2-D DCT can be obtained by using two 1-D DCT lattice arrays. It will be shown in the next section that the 2-D DCT obtained by this time-recursive approach is fully-pipelined and no transposition is required. This is because that by using the frame-recursive approach, we start from the transformed 2-D DCT directly and avoid calculating the 2-D DCT indirectly from the 1-D DCT. Our architectures are efficient since it is a direct 2-D approach. This method can also obtain the 2-D DCT and 2-D DSCT simultaneously. In contrast to processing the input 2-D data sequence by rows, the input data can be updated by columns. In this case, 2-D DCT and 2-D discrete cosine-sine transform (DCST) are dually generated, and all other results are similar and can be easily derived.

3.3 Architectures of Frame-Recursive Lattice 2D-DCT and 2-D DSCT

The fully-pipelined parallel lattice structures for successive frame and block 2-D DCT and DSCT are described in this section. As we know, most of the 2-D DCT architectures are implemented by the row-column decomposition method [25, 24, 26]. It is due to the fact that the architectures of most fast direct 2-

D algorithms are usually irregular and globally connected, therefore it is not practical for VLSI implementation. Another reason is that it is beneficial to generate a 2-D DCT system from existing 1-D DCT circuit rather than to build a new multiplier-saved 2-D DCT architecture which may not be compatible with the 1-D DCT system. By using the frame-recursive method, the 2-D DCT can be implemented by 1-D DCT lattice arrays which are regular, modular and suitable for VLSI implementation. So the difficulties mentioned above are avoided. We will discuss two architectures, the moving frame 2-D DCT architecture and the block 2-D DCT architecture. The moving frame 2-D DCT architecture is used to calculate the 2-D DCT of sequential frames. For example, the 2-D DCTs of the 0'th frame, first frame, second frame and so on. The block 2-D DCT architecture computes the 2-D DCT of an $N \times N$ input data matrix for every N frames, *i.e.*, the 0'th frame, the N'th frame, the $2N$'th frame and so on.

3.3.1 The Moving Frame 2-D DCT Architectures

The moving frame 2-D DCT architectures generate the 2-D DCT of successive input frames. From the frame-recursive concept derived in section 3.2, the 2-D DCT recursive lattice structures can be constructed according to (3.4), (3.8), and (3.14). Although the intermediate values $\delta_c(k, l, t)$ in (3.14) are functions of both k and l, it is noted that the effect due to the index k is equivalent to sign changes in the input data sequences. Using this property, we will show two approaches, the pre-matrix method and the post-matrix method, to implement the moving frame 2-D DCT architectures. The pre-matrix method will be discussed first.

The Pre-Matrix Method

In this method, the intermediate values of $\delta_c(k,l,t)$ are realized directly from (3.14). As we can see, the index k in (3.14) affect only the sign of the new input data sequence. Thus, there are only two possible input sequence combinations: $\{\ x(t+N,n) - x(t,n)\}$ and$\{\ -x(t+N,n) - x(t,n)\}$. The resulting pre-matrix moving-frame 2-D DCT architecture is shown in Fig. 3.4 which includes a circular shift matrix I, two adders, two LAIs, one $LAII$, and two circular shift arrays and shift register arrays. Except for the LAI, $LAII$, and adders, all other components are shift registers. We will describe the functions of every blocks first, then demonstrate how the system works.

The circular shift matrix I ($CSMI$) is an $(N+1)\times N$ shift registers connected as shown in the upper part of Fig. 3.5. When a new input datum $x(m,n)$ arrives every clock cycle, all the data are shifted in the way as indicated in Fig. 3.5. Both of the first elements in the t'th row and $(t+N)$'th row are sent to the adders for summation and subtraction as shown in Fig. 3.4. The pre-matrix architecture contains two LAIs which includes N lattice modules as shown in Fig. 3.3. The upper and lower LAIs execute the 1-D DCT on the rows of the 2-D input data for the even and odd transformed components k respectively. Because the length of the input vector is N and only the discrete cosine transformed data are needed, the 1-D DCT transformed data $\delta_c(k,l,t)$ generated by the LAIs are sent out every N clock cycles [17]. Due to the time-recursive approach used, the initial values $X_c(l,t)$ and $X_s(l,t)$ in the LAIs (see Fig. 3.3) are reset to zeros every N clock cycles.

The circular shift array in the middle of the system is an $N \times 1$ shift register array as shown in Fig. 3.6. This special shift register array loads an $N \times 1$ data

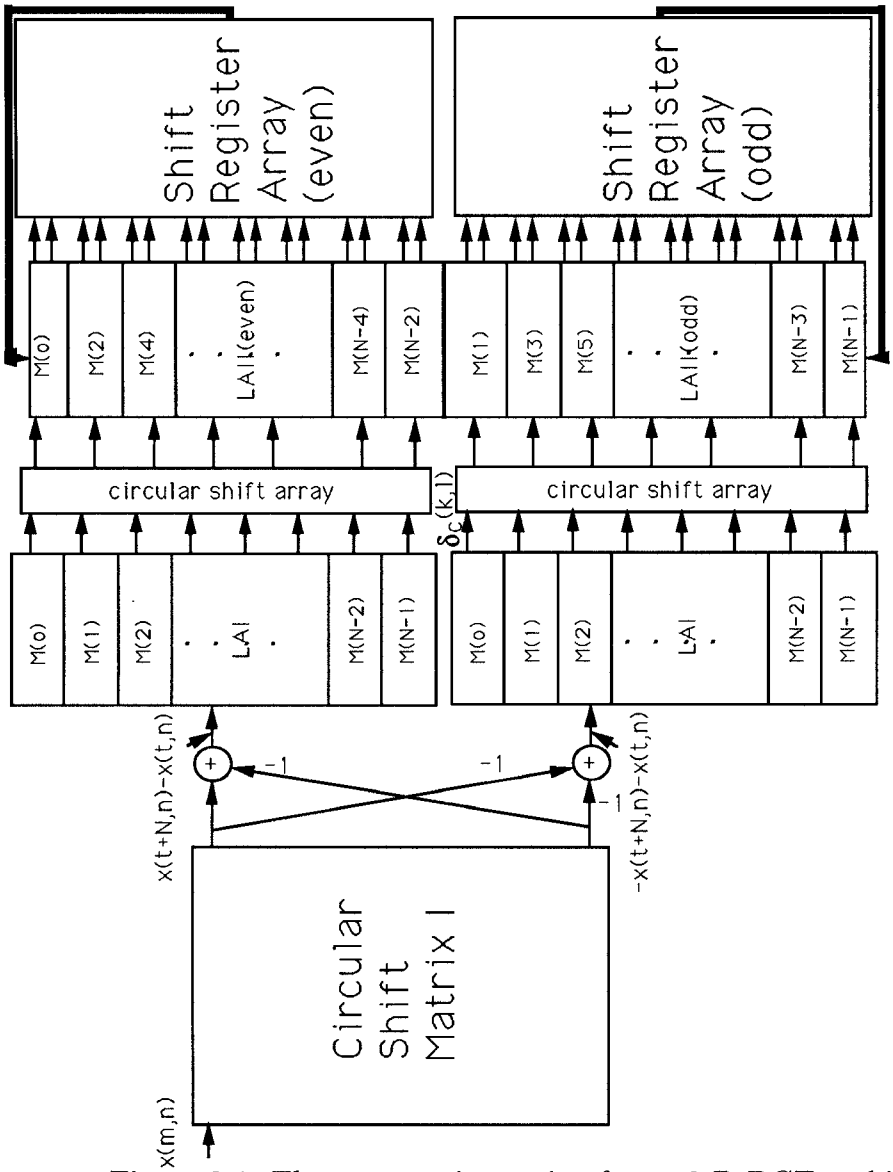


Figure 3.4: The pre-matrix moving frame 2-D DCT architecture.

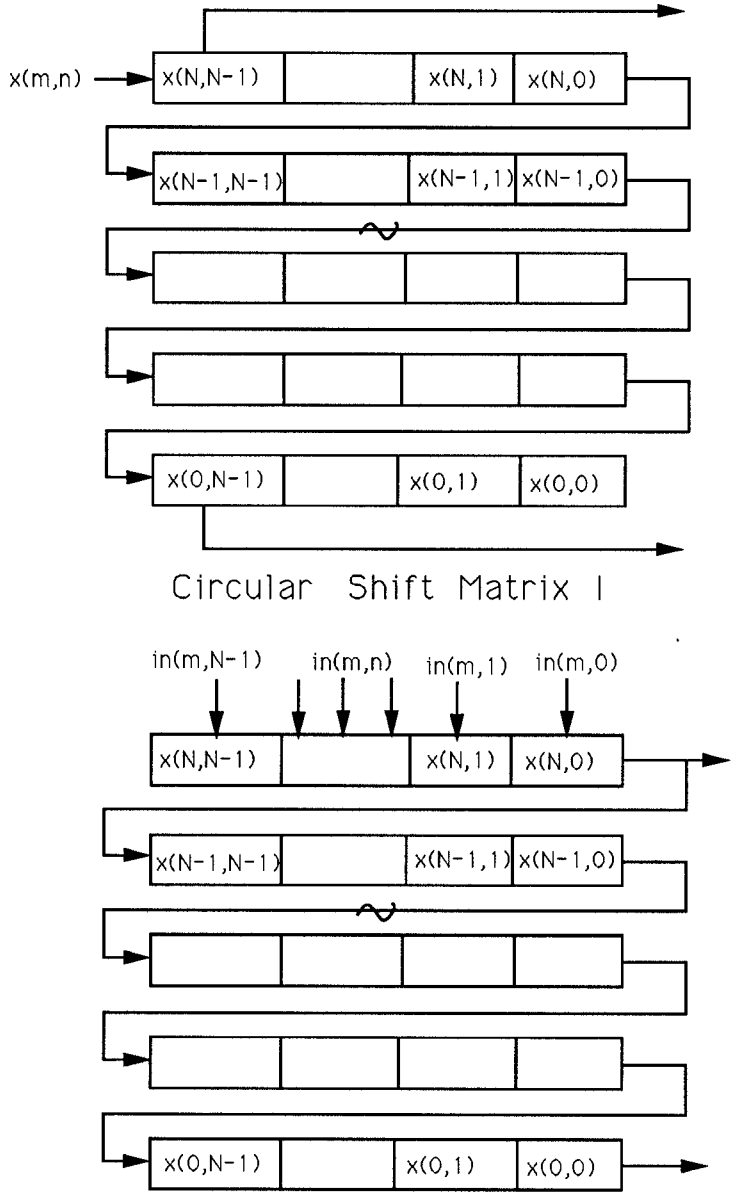


Figure 3.5: The circular shift matrix (CSM) I and II.

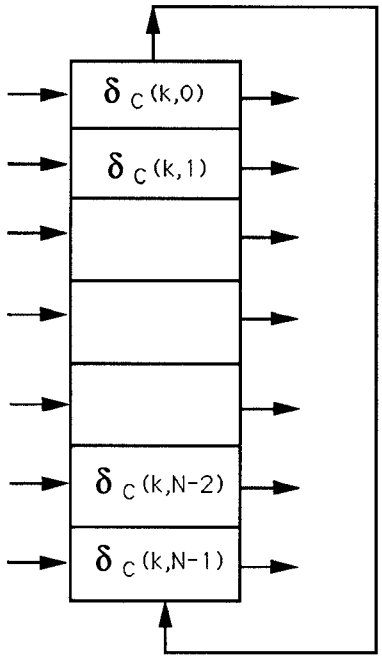


Figure 3.6: The circular shift array (CSA).

vector from the LAI every N clock cycles, then it will shift the data circularly and send the data to the $LAII$ every clock cycle. There are three inputs in $LAII$, $\delta_c(k,l,t)$, $X_c(k,l,t)$ and $X_s(k,l,t)$, where the $\delta_c(k,l,t)$ comes from the circular shift array, and $X_c(k,l,t)$ and $X_s(k,l,t)$ from the shift register arrays located behind the $LAII$. We divide the $LAII$ into two groups: the $LAII_{even}$ and $LAII_{odd}$. Each includes $N/2$ lattice modules as shown in Fig. 3.2. The $LAII_{even}$ contains only those lattice modules for even transformed components k, while $LAII_{odd}$ contains only the odd lattice modules. It should be noticed that this system contains two LAI and only one $LAII$. The shift register array contains $2N \times N$ registers. Their operations are shown in Fig. 3.7.

The following is to illustrate how this parallel lattice structure works to obtain the 2-D DCT and DSCT of 2-D input successive frames. All the initial values of the circular shift matrix I ($CSMI$), circular shift array, and shift register array

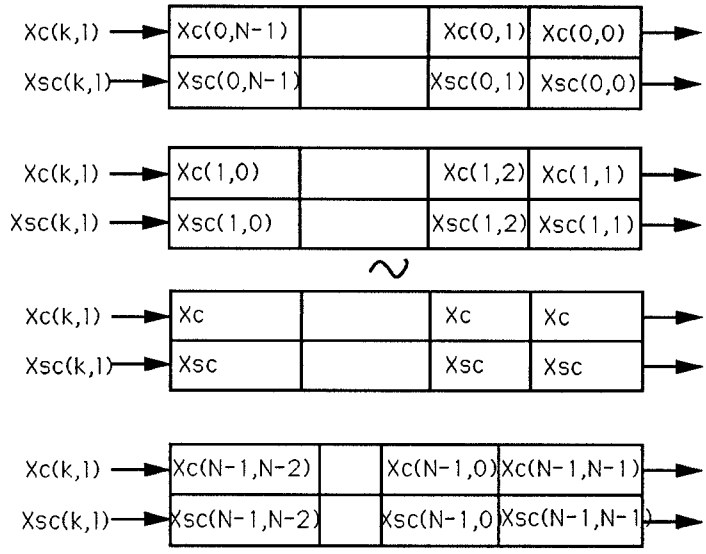


Figure 3.7: The Shift Register Array.

are set to zeros. The input data sequence $x(m,n)$ sequentially shifts row by row into the $(N+1) \times N$ $CSMI$. First we calculate the difference between the t'th row and the $(t+N)$'th row data vector of the $CSMI$. The two resulting combinations of the input sequences, $x(t+N,n)-x(t,n)$ and $-x(t+N,n)-x(t,n)$ for $n = 0,1,2,..,N-1$, are used as the input sequences for the lattice array Is, which consist of $2N$ lattice modules to calculate the 1-D DCT for $\{x(t+N,n)-x(t,n)\}$ and $\{-x(t+N,n)-x(t,n)\}$. The upper LAI is for the even transformed components k and the lower one for odd k. Suppose the data of the input vectors arrive serially per clock cycle, it takes N clock cycles to obtain the $\delta_c(k,l,t)$ for both of the input sequences. At the N'th cycle, the N transformed data $\delta_c(k,l,t)$ are loaded into the circular shift arrays, CSA, which will shift circularly and send the data out of the register into the $LAII$ for different k components every clock cycle. In $LAII$, $X_c(k,l,t+1)$ and $X_{sc}(k,l,t+1)$ are evaluated according to (3.4)

and (3.8). Since $LAII_{even}$ and $LAII_{odd}$ have only $N/2$ modules, every $\delta_c(k,l,t)$ is floating for $N/2$ clock cycles. It is noted that a specific 2-D transform data $X_c(k,l,t+1)$ and $X_{sc}(k,l,t+1)$ are updated recursively every N clock cycles from $X_c(k,l,t)$ and $X_{sc}(k,l,t)$. Therefore the outputs of the $LAII$ are sent into the shift register array (SRA) where data are delayed by N clock cycles. Each SRA contains $N/2$ shift registers each with length N. The data in the rightest registers are sent back as the $X_c(k,l,t)$ and $X_{sc}(k,l,t)$ of $LAII$. At the N^2 clock cycle, the 2-D DCT and DSCT of the 0'th frame are available. After this, the 2-D transformed data of successive frames can be obtained every N clock cycles.

We observe two interesting results in the pre-matrix method. First, both LAI and $LAII$ can be viewed as filter banks. It is because every lattice module itself is an independent digital filter with different frequency components $k,l = 0,1,...,N-1$. Moreover, all the lattice modules in this architecture have the same structure. Second, the system requires 3 1-D DCT array and is fully pipelined with throughput rate N clock cycles. From the above discussion, transposition for the row-column decomposition method is unnecessary in this realization. According to the 1-D DCT architecture proposed in [17] (Liu-Chiu2 architecture), the total multipliers required in the 2-D DCT is $12N$ and total number of adders is $15N$. Due to the goal to pipe out the results every N clock cycles, it requires three 1-D DCT structures in the system. We will show how to use only two 1-D DCT lattice arrays to attain the results at the same throughput rate in the post-matrix method.

The Post-Matrix Method

The intermediate value $\delta_c(k,l,t)$ in (3.14) can be rewritten as

$$\begin{aligned}\delta_c(k,l,t) &= (-1)^k \frac{2}{N} C(l) \sum_{n=0}^{N-1} x(t+N,n) \cos\left[\frac{\pi(2n+1)l}{2N}\right] \\ &\quad -\frac{2}{N} C(l) \sum_{n=0}^{N-1} x(t,n) \cos\left[\frac{\pi(2n+1)l}{2N}\right] \\ &= (-1)^k \acute{X}_c(t+N,l) - \acute{X}_c(t,l). \end{aligned} \tag{3.15}$$

That is, we can calculate the 1-D DCT of the t'th row and $(t+N)$'th row of the input frame individually, then perform the summation later on. Our approach is to send the input sequence $x(m,n)$ row by row directly into the LAI. It takes N clock cycles for the LAI to complete the 1-D DCT of one row input vector, then the array sends the 1-D DCT data in parallel to the $CSMII$ as shown in Fig. 3.8. The operations of $CSMII$ are shown in the lower part of Fig. 3.5. At the output of the $CSMII$, the 1-D transformed data of the $(t+N)$'th row and t'th row are added together according to (3.15) depending on the sign of the k components (see Fig.3.8). Then the results are sent to CSAs, $LAII$, and SRAs, whose operations remain intact as in the pre-matrix method. The whole structure is demonstrated in Fig. 3.8. Therefore, by transforming the input data first, we can implement the 2-D DCT by using only two 1-D DCT lattice arrays and retain the same pipeline rate. The total numbers of multipliers and adders needed for the post-matrix method are $8N$ and $10N$ respectively.

3.3.2 The Block 2-D DCT Architecture

In most image processing applications, the 2-D DCT are executed block by block instead of in successive frames [35, 37]. We will show how to apply the frame-recursive concept to obtain the block 2-D DCT. It will be easier to understand if

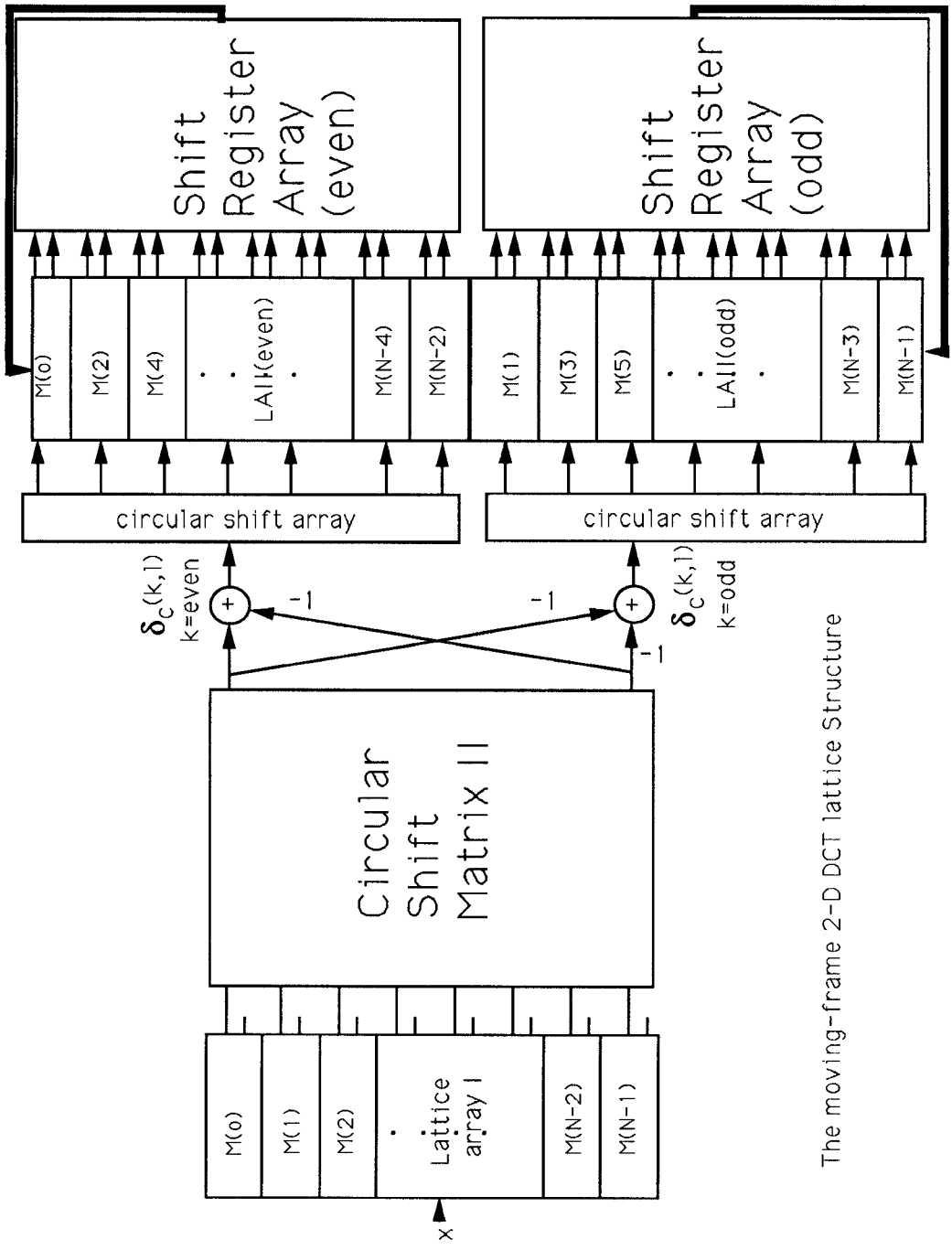


Figure 3.8: The post-matrix moving frame 2-D DCT architecture.

we use an example to show how to obtain the 0'th frame 2-D DCT in the post-matrix moving frame 2-D DCT architecture. Recall that the $CSMII$ in Fig. 3.8 is used to store the transformed data $\acute{X}_c(t,l)$ of the previous input row vectors. Since the 0'th frame is the first input data frame, all the values in the $CSMII$ are set to zeros. Corresponding to (3.15), this means that the second terms $\acute{X}_c(t,l)$ are zeros. When the $(N-1)$'th row data vector (the last row of the 0'th frame) arrive, the 2-D DCT of the 0'th input data frame is obtained. During this initial stage, the 2-D DCT of the 0'th frame obtained by the moving frame approach is equal to the block 2-D DCT of the 0'th frame. Therefore, if we want to compute the 2-D DCT of the N'th frame, then all the values in the $CSMII$ are resetting to zeros when the first datum in the N'th data frame (*i.e.* $x(N,0)$) arrives. That is, we can obtain the block 2-D DCT by reset the values of the $CSMII$ every N^2 cycles. This means that the $CSMII$ in Fig. 3.8 is redundant and the second terms $\acute{X}_c(t,l)$ in (3.15) are zeros. So, the intermediate value of $\delta_c(k,l,t)$ can be rewritten as

$$\delta_c(k,l,t) = (-1)^k \frac{2}{N} C(l) \sum_{n=0}^{N-1} x(t+N,n) \cos\left[\frac{\pi(2n+1)l}{2N}\right]. \tag{3.16}$$

The block 2-D DCT architecture is shown in Fig. 3.9. Corresponding to our frame-recursive algorithm, we obtain another block of input data every N^2 clock cycles. Note that this is also the total time required for all the N^2 data to arrive in a transmission system.

The following is an example to calculate block 2-D DCT for the 0'th frame. When row data vectors arrive, the LAI performs the 1-D DCT on them. Every N clock cycles, after the last datum of each row $x(m,N-1)$ is available, the LAI completes the 1-D DCT for every row and sends the N 1-D DCT transformed data to the two length-N CSAs. The upper CSA translates the intermediate

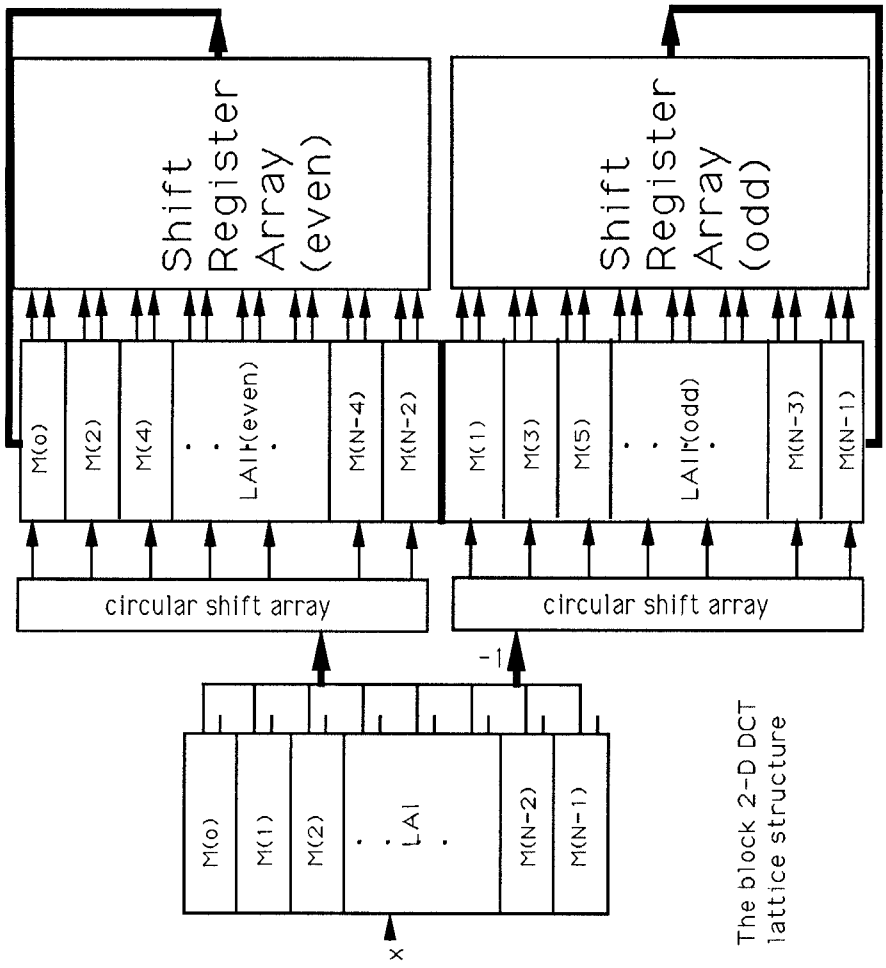


Figure 3.9: The block 2-D DCT architecture.

value $\delta_c(k,l,t)$ to the lattice array II_{even}, as do the lower CSA except that the signs of the output of the CSA are changed before being sent to the lattice array II_{odd}. The operations of the lattice array II and the SRA are the same as those in the previous methods. As we can see, the hardware complexity of our block 2-D DCT architecture is as simple as the row-column methods. Moreover, our system can be operated in a fully pipelined manner.

3.4 Comparisons

Since most of the 2-D DCT algorithms proposed are based on manipulating $N \times N$ block signals, we compare only the 2-D block architecture as described in section 3.2. with other algorithms. The block 2-D DCT architecture is a fully-pipelined serial input parallel output ($SIPO$) system with throughput rate every N clock cycles and in terms of hardware complexity, it requires only two 1-D DCT architectures without transpositions. It is attractive, therefore, not only for its efficiency in term of system throughput but also for its hardware simplicity and regularity.

In the following section, the comparisons between our 2-D DCT block structure and those of others are based on the number of multipliers, adders and speed. For the sake of clarity, we divide the algorithms into two groups: parallel input parallel output ($PIPO$) and serial input parallel output ($SIPO$). The fast algorithms presented by Vetterli and Nussbaumer [9, 79], Duhamel and Guillemot [80], and Cho and Lee [55] belong to the former class. Vetterli's algorithm [79] mapped the 2-D DCT into a 2-D cosine DFT and sine DFT through a number of rotations, and the 2-D DFT are computed by Polynomial Transform

(PT) methods [79]. Vetterli's method can reduce the number of multipliers more than 50% in comparison to the row-column method based on Chen's algorithms [13] and has a comparable amount of additions . Duhamel *et al* [80] present a PT-based algorithm which uses the direct DCT approach. This direct PT 2-D DCT method provides a 10% improvement in both the numbers of additions and multiplications compared to Vetterli's result [79] but it requires complex computations. Cho and Lees' algorithm is a direct 2-D method which employs the properties of trigonometric functions. The number of multipliers are the same as that of Duhamel's, but the structure is more regular, and only real arithmetic is involved. Up to now, the best results for the first $PIPO$ systems in terms of the number of multipliers are $(N^2 + 2N + 2)$, which were obtained by Duhamel and Guillemot, as well as by Cho and Lee. But assuming that all the N^2 input data arrive at the same time is not practical in communication systems. The data waiting time is N^2 which is always neglected in these approaches.

The systolic array approaches proposed by Lee and Yasuda [15], Ma [18], and Liu-Chiu belong to the $SIPO$ method. Lee-Yasuda presented a 2-D systolic DCT/DST array algorithm based on an IDFT version of the Goertzel algorithm via Horner's rule in [15]. The latest systolic array algorithm for 2-D DCT was proposed by Ma [18], where he showed two systolic architectures of 1-D DCT arrays based on the indirect approach proposed by Vetterli-Nussbaumer [79], then he exploited the 2-D DCT systolic array by using the features of the two 1-D DCT systolic arrays. This method requires $(N + 1)$ 1-D DCT structures and the total number of time steps is $(N^2 + 2N + 2)$ [18]. We call the block 2-D DCT structure shown in Fig. 3.9, based on the Liu-Chiu2 module [17], Liu-Chiu2D. This needs only two 1-D DCT, and the total time steps are N^2.

	R-C method on Chen	Duhamel *et al.*	Cho-Lee	Ma	Ours
No. of multipliers	$2N^2\ln(N)$ $-6N^2/2+8N$	N^2 $+2N+2$	N^2 $+2N+2$	$4N$ $*N+1$	$8N$
Throughput	$N+$ transposition	$2N$	N	$2N+1$	N
Limitation on N	power of 2	power of 2	power of 2	no	no
Commun.	global	global	global	local	local
I/O	*PIPO*	*PIPO*	*PIPO*	*SIPO*	*SIPO*
Approach	indirect	direct	direct	indirect	direct

Table 3.1: Comparisons of different 2-D DCT algorithms.

The comparisons regarding their inherent characteristics are given in Table 3.1. In addition, the quantities comparisons in terms of the number of multipliers and adders are given in Table 3.2 and Table 3.3. In general, the *SIPO* method is more workable in hardware implementations. Our structure requires fewer multipliers than Ma's structure and is highly regular, systematic, and uses only local communications. In addition, this lattice 2-D DCT architecture can be generated from the 1-D DCT lattice array without modifications.

3.5 Applications to the HDTV systems

In recent years, the focus of video signal processing research has been concentrated on high-definition television (HDTV) which will become future standard

NO	row-column method bsed on Chen in[2]	Duhamel[15] *et al.*	Cho-Lee [10]	Ma [7]	Liu-Chiu2D
8	256	96	96	288	64
16	1408	512	512	1088	128
32	7424	2560	2560	4224	256
64	37376	12288	12288	16640	512

Table 3.2: Comparisions of the number of multipliers.

No	row-column method bsed on Chen in[2]	Duhamel[15] *et al.*	Cho-Lee [10]	Ma [7]	Liu-Chiu2D
8	416	484	466	432	78
16	2368	2531	2530	1632	158
32	12416	12578	12738	6336	318
64	61696	60578	42461	24960	638

Table 3.3: Comparisions of the number of adders.

for the next generation television [40]. According to CCIR Recommendation 601, the bit rate for transmitting an uncompressed digital HDTV is about 1Gbps. This bit rate is too high even for broadband ISDN (BISDN) [40]. Furthermore, video signals contain a great deal of redundancy when psychological and visual effects are considered. To make HDTV systems practical, bit rate reduction and data compression are indispensable. In the past decades, many studies have been conducted on differential pulse code modulation (DPCM), subband coding, and transform coding (especially DCT) to achieve bit rate reduction [42, 38, 43]. The DCT has obtained most attention due to its diverse attractive features. The DCT approaches the statistical optimal transform, Karhunen-Loeve Transform (KLT), which minimize the mean square errors, for highly correlated signals [1]. Additionally, the DCT has superior energy compaction properties for transform coding. Many HDTV systems based on the DCT coding schemes show satisfactory speed and promising performance [29, 35, 36, 30, 41, 42, 34]. A commonly used encoder configuration of the DCT based source coding is shown in Fig. 3.10 [41, 34]. The DCT is performed on the 2-D video signals with block size 8×8, which is widely used due to its acceptable SNR and implementation complexity [41].

Although there is no uniform standard for HDTV, the interlaced mode with 1080 active lines per frame, 30Hz frame frequency, and 2:1 interlaced ratio is presently under widely investigations due to its reasonable data rate [32]. With an assumption of coding each pel (luminance and chrominance) with 2 bits, the bit rate required for transmission of the video signal under interlaced modes is 119.232 Mbits/s which satisfies the requirement of 140Mbits/s H4 hierarchy level [29], and allows sufficient margin for error protection and auxiliary data. The

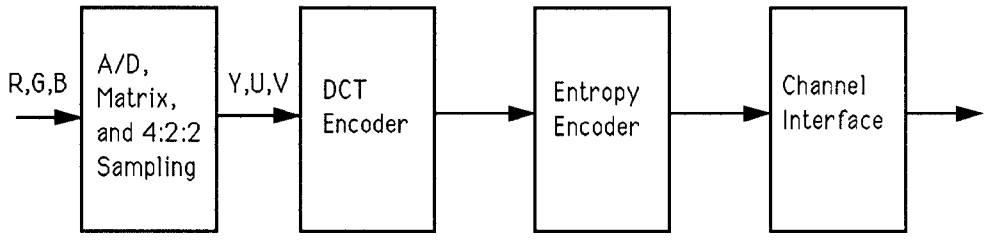


Figure 3.10: System diagram of the DCT based HDTV coder.

4:2:2 YUV signals are obtained from RGB signal by A/D converters and coordinate rearrangements. The intrafield 2-D DCT is used for data compression. The transformed signal is processed by an entropy encoder, which is usually combined with the run-length coder and variable length encoder. The run-length coder can reduce the bit-rate by coding every sequence of zeros with a single codeword. The variable length coder encodes the DCT coefficients with a variable length code adapted to their probability density function (pdf) distribution.

Most of the 2-D DCT implementations are based on the row-column decompositions methods. Although fast algorithms exist for the 1-D DCT, the second 1-D DCT cannot start until all the first 1-D DCTs are completed. To speed up the operations, one method is to in parallel execute the first 1-D DCT. For the 8×8 case, there are 8 1-D DCT blocks to perform the first transform simultaneously. Assuming that each signal is 10-bit long, in order to to satisfy the

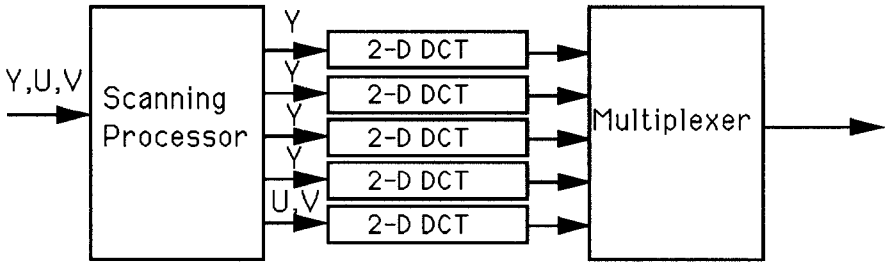


Figure 3.11: The block diagram of the DCT encoder.

precision, then the total number of bits required in the input is 640 bits, which is not practical in the circuit realizations. From this point of view, our serial input 2-D DCT system is more practical in hardware implementations. Moreover, if the speed of the circuit components, such as the ROM and adder, is high enough, our 2-D DCT system can be executed as fast as the sample clocking rate.

Although our 2-D DCT implementations are effective, transforming a video frame of 1080×1920 still requires intensive computations. Therefore, we designed a 2-D DCT architecture suitable for the HDTV system to achieve higher performance. The block diagram of the 2-D DCT encoder is shown in Fig. 3.11, where five 2-D DCT chips are included. Five chips were used because the ratio of pixel numbers per line for luminance signal Y and color difference signals U and V is 4:2:2. As the sampling frequency of HDTV is very high, the pixels of Y are divided into four groups, in order to carry out DCT in parallel. Additionally, the color difference signal Y and U are switched alternatively to another DCT coder. The scanning processor shown in Fig. 3.11 is used to divide the

signal into four luminance components and one color difference component. The outputs of the 2-D DCT transformed data are sent to the entropy encoder in parallel or through multiplexers. Since the transform block size is 8×8, we divided the frame into 135×240 blocks and 240 channels as shown in Fig. 3.12. The 2-D DCT are executed on each channel whose scanning pattern is shown in Fig. 3.12. This scanning pattern reflects the fact that our system is based on row by row scanning order and is fully pipelined. Thus, such a scanning method would maximize the system throughput.

3.6 Summary

In this Chapter, we propose a new 2-D DCT algorithm based on a frame-recursive approach. The resulting 2-D DCT architectures can be obtained by using only two 1-D DCT arrays, at the same time, the transposition procedure is eliminated. It, therefore, does not have the drawback of the row-column decomposition method in which a transposition is needed between the first and the second 1-D DCT. In addition, this algorithm generates the 2-D DCT and the 2-D DSCT or the 2-D DCST simultaneously. There are two methods, the pre-matrix method and the post-matrix method, to realize the moving frame 2-D DCT architecture. From the post-matrix method, the block 2-D DCT architecture is developed. These architectures are fully-pipelined with throughput rate N clock cycles for an $N \times N$ input frame. As to the hardware complexity, the structures contain two 1-D DCT arrays, each with N lattice modules which are modified normal form digital filters with different multiplying coefficients. The total number of multipliers required in the system is $8N$. Because of the regular-

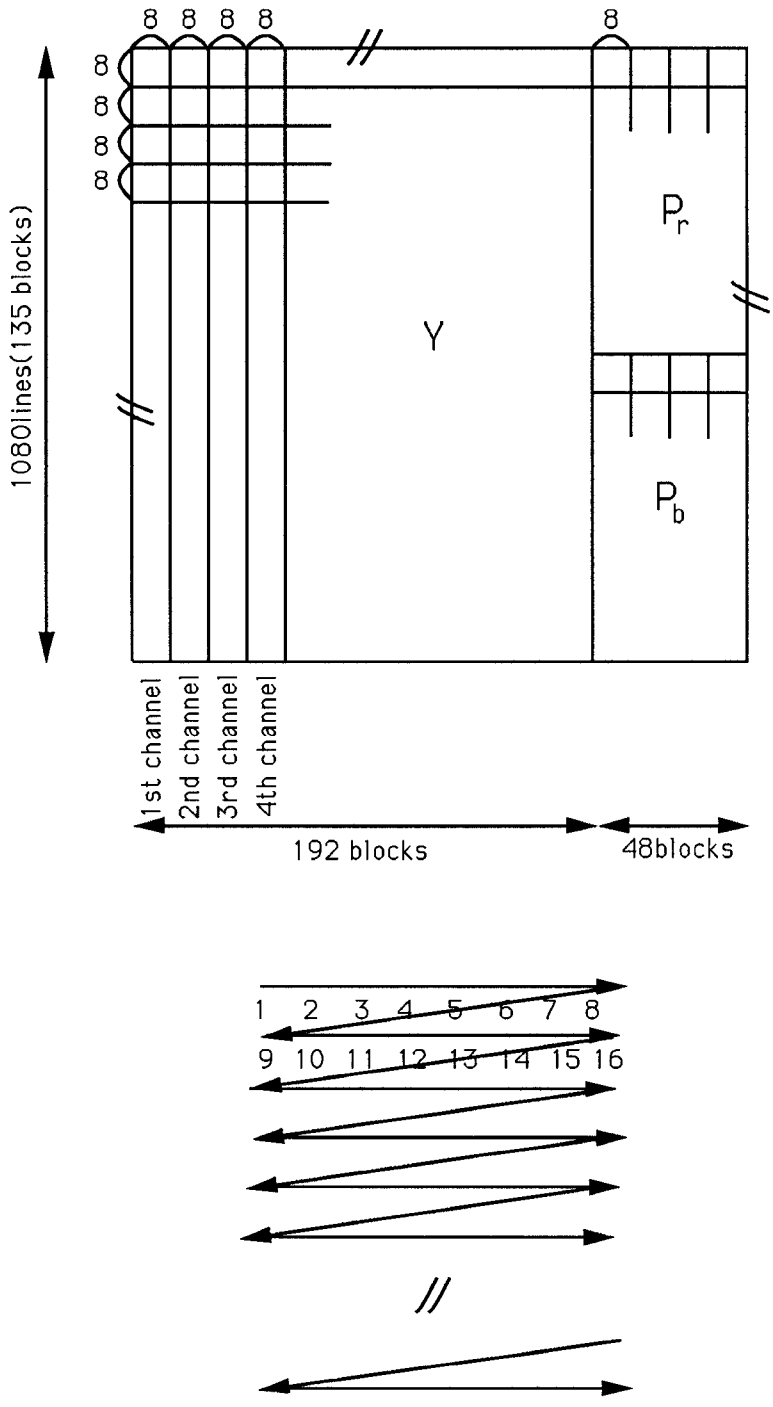


Figure 3.12: Block construction of a Video frame and proposed scanning pattern.

ity and efficiency of the systems, they are very suitable for VLSI implementation for the high speed HDTV systems.

Many HDTV systems based on the DCT coding shows satisfactory speed and promising performance since the DCT has superior energy compaction property. However, most of the 2-D DCT portion of the HDTV systems are still implemented by row-column decomposition methods. To speed up the throughput, the first DCT operation in the row-column method has to be performed in parallel, while it is not practical in circuit realizations because of the hardware complexity. In view of these facts, our serial input 2-D DCT system is more feasible in HDTV applications. The parallel 2-D DCT architecture and the scanning pattern proposed in Section 6 can process the video data in real time and eliminate the waiting time in the DCT codings so that the system performance can be maximized. Consequencely, our real-time parallel and fully-pipelined 2D-DCT structure is very attractive in high speed transmission system where every arrived data can be processed immediately.

Chapter 4

Optimal Unified IIR Architectures

An optimal unified architecture that can efficiently compute the Discrete Cosine, Sine, Hartley, Fourier, Lapped Orthogonal, and Complex Lapped transforms for a continuous input data stream is proposed. This structure uses only half as many multipliers as the previous best known scheme [17]. The proposed architecture is regular, modular, and has only local interconnections in both data and control paths. There is no limitation on the transform size N and only $2N-2$ multipliers are needed for the DCT. The throughput of this scheme is one input sample per clock cycle. We provide a theoretical justification by showing that any discrete transform whose basis functions satisfy the *Fundamental Recurrence Formula* has a second-order autoregressive structure in its filter realization. We also demonstrate that dual generation transform pairs share the same autoregressive structure. We extend these time-recursive concepts to multi-dimensional transforms. The resulting d-dimensional structures are fully-pipelined and consist of only d 1-D transform arrays and shift registers.

4.1 Introduction

Discrete sinusoidal transforms play significant roles in various digital signal processing applications, such as spectrum analysis, image and speech signal processing, computer tomography, data compression, and signal reconstruction[22, 60, 68, 51]. Among different discrete sinusoidal transforms, the discrete cosine transform (DCT)[69, 85, 55, 56], the discrete sine transform (DST)[49, 56], the discrete Hartley transform (DHT)[52, 53, 48, 69], and the discrete Fourier transform (DFT)[73, 60] are widely used because of their efficient performance[51, 7, 3, 9, 11]. Recently, the Lapped Orthogonal Transform (LOT)[70], and the Complex Lapped transform (CLT)[62] were introduced for transform coding with significantly reduced blocking effects and for motion estimation.

In real-time signal processing applications, especially in speech/image communications and radar/sonar signal processing, input data arrive serially. In traditional FFT based algorithms, the serial data is buffered and then transformed using the FFT scheme of complexity $O(N \log N)$[60]. Buffering the serial data requires $O(N)$ time. In this Chapter, we describe a novel architecture that merges the buffering and transform operations into a single unit of total hardware complexity $O(N)$. Unlike the FFT, this architecture has only local interconnections and is better suited for VLSI implementations. It is important to note that the proposed architectures generate time-recursive transforms, not just block transforms, *i.e.,* the transform of the N points $[x(t+1), x(t+2), \ldots, x(t+N)]$ is generated one clock cycle after the transform of $[x(t), x(t+1), \ldots, x(t+N-1)]$ is generated. To generate time-recursive transforms, the traditional fast algorithms based architectures require $O(logN)$ time using $O(N \log N)$ hardware, while the architectures we propose require only a constant time with $O(N)$ hard-

ware. Time-recursive transforms are currently gaining widespread use in motion estimation, in video signal processing, and in reducing blocking effects in data compression.

We have shown in [17] that when discrete transforms are performed on segments of a continuously incoming data stream, transforms can be realized by a unified lattice structure with a data throughput rate of one input sample per clock cycle. This architecture is regular, modular, and free of global interconnections. Unlike the many fast algorithms for DFT, DCT, and DHT, there is no constraint on the transform size N. Table 2.1 [17] summarizes a comparison of the time-recursive approach with other well-known fast algorithms. A time-recursive lattice 2-D DCT structure with applications to the HDTV systems is also given in [65]. This 2-D DCT structure requires only two 1-D DCT blocks and is fully-pipelined with no transposition.

In this Chapter, we describe an optimal unified filter structure, which preserves the advantages of the lattice architecture, while reducing the hardware complexity in half. In the time-recursive lattice architecture, two transforms called the dual generated pairs, are obtained simultaneously. The unified filter structure is more suitable for applications where only one transform is required. We develop a systematic approach to derive the time-recursive unified filter architecture for any discrete transform. We show that all the resulting unified filter architectures have a similar second-order autoregressive structure with minimum number of multipliers. A theoretical basis for this fact is provided. We also demonstrate that the time-recursive concept can be generalized to multi-dimensional transforms by using only the one-dimensional transform architecture and simple shift registers. An area-time complexity analysis is also

provided to show that the proposed approach is asymptotically optimal in speed and area.

The rest of this Chapter is organized as follows. The unified lattice structure for sinusoidal transforms is described in Section 4.2. The derivation of the optimal unified filter structure from the transfer function of the discrete sinusoidal transforms is discussed in Section 4.3. The architectures of the inverse discrete sinusoidal transforms based on the IIR filter realization are presented in Section 4.4. In Section 4.5, the characteristics of these architectures are discussed from a theoretical point of view. The unified architectures for time-recursive based multi-dimensional discrete sinusoidal transforms are discussed in Section 4.6. Finally, we give a conclusion in Section 4.7.

4.2 Lattice Structure for Discrete Sinusoidal Transforms

The time-recursive approach has been shown to be efficient in both hardware and computational complexity for the computation of discrete sinusoidal transforms (DXT), (such as the DCT, DST, and DHT), for time series input data stream [17]. In this section, we will extend this approach to the problems of computing the transforms DFT, LOT and CLT and we provide a unified view of lattice structures for time-recursive approach.

Denote the discrete sinusoidal transform DXT of a data sequence of length N $[x(t), x(t+1), ..., x(t+N-1); t = 0, 1, 2, ...]$ at time t as

$$X(k,t) = C(k) \sum_{n=t}^{t+N-1} x(n) P_{n-t}(k), k = 0, 1, ..., N-1, \tag{4.1}$$

where $P_{n-t}(k)$ are transform basis functions and $C(k)$ are constants used for normalization. It was shown in [17] that most discrete sinusoidal transforms have dual generated pairs. That is, the lattice structure used for generating one transform automatically generates its dual. For example, the dual of the DCT is the DST. Both the transform and its dual have similar updating relations. Let us denote the dual generated pairs by $X_{xc}(k,t)$ and $X_{xs}(k,t)$. Then, the time-recursive relation between $X_x(k,t)$ and $X_x(k,t+1)$ can be obtained by eliminating the effect of the first term of the previous sequence and updating the effect of the last term of the current sequence. In general, the dual generation properties between the transform pairs $X_{xc}(k,t)$ and $X_{xs}(k,t)$ are given by the following equations [17]:

$$\begin{aligned} X_{xc}(k,t+1) &= e(k)\{\left[X_{xc}(k,t)+[x(t+N)(-1)^k-x(t)]D_c\right]\Gamma_c \\ &\quad +\left[X_{xs}(k,t)+[x(t+N)(-1)^k-x(t)]D_s\right]\Gamma_s\} \end{aligned} \tag{4.2}$$

and

$$\begin{aligned} X_{xs}(k,t+1) &= f(k)\{\left[X_{xs}(k,t)+[x(t+N)(-1)^k-x(t)]D_s\right]\Gamma_c \\ &\quad -\left[X_{xc}(k,t)+[x(t+N)(-1)^k-x(t)]D_c\right]\Gamma_s\}, \end{aligned} \tag{4.3}$$

where D_c and D_s are the associated cos and sin transform kernels of the DXT with fixed index n. Coefficients $e(k)$ and $f(k)$ depend on the definition of the transforms and are always equal to one except for the two transforms LOT and CLT. That is, two transforms can be dually generated from each other in a lattice form as shown in Fig. 4.1. Here, we will briefly describe the definition of the DFT, LOT, and CLT and state the equations corresponds to (4.2) and (4.3).

$$X_f(k,t)=\frac{1}{\sqrt{N}}\sum_{n=t}^{t+N-1}x(n)\exp\{-j2(n-t)\frac{\pi k}{N}\},\quad k=0,1,\ldots,N-1. \tag{4.4}$$

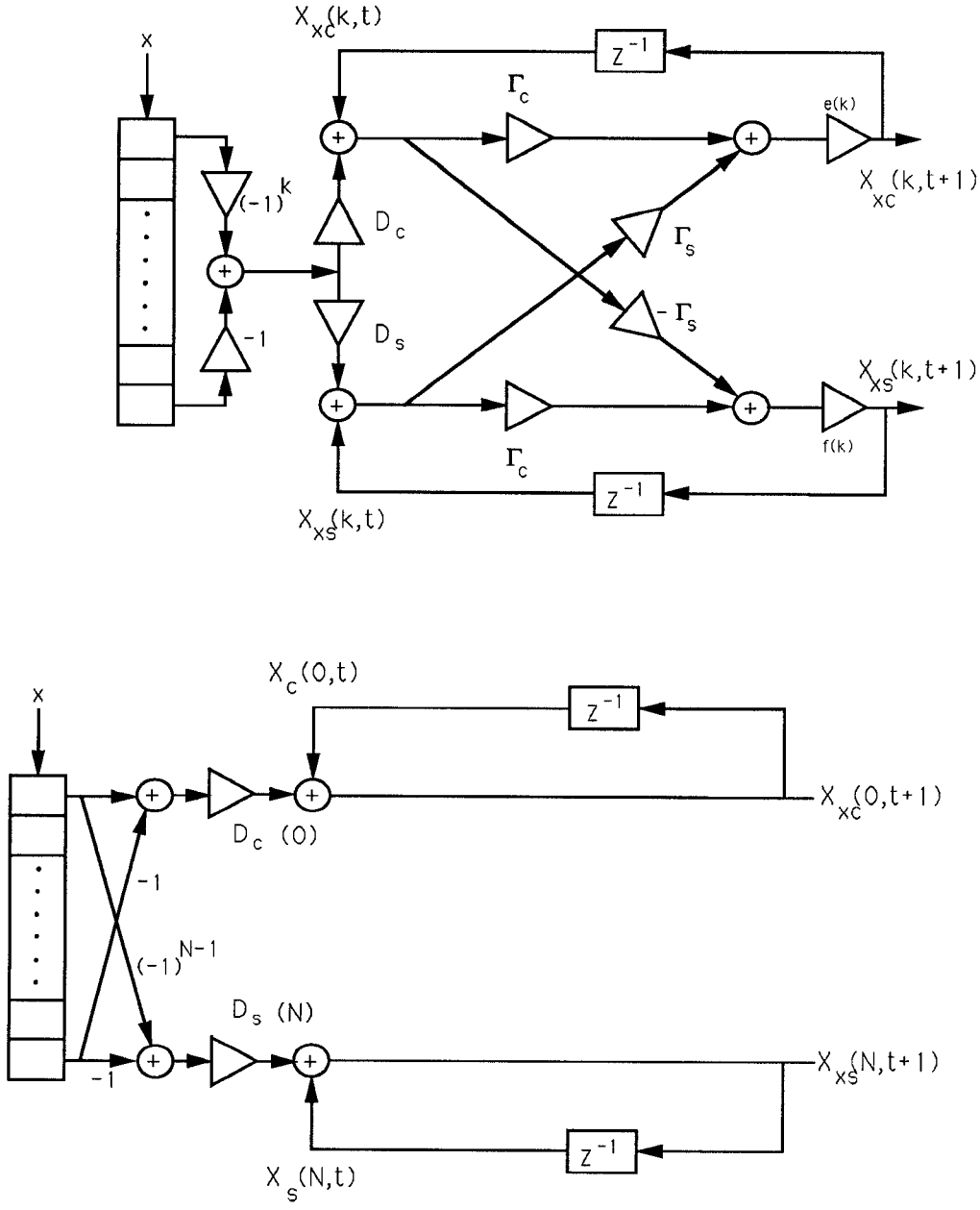


Figure 4.1: The universal lattice module.

	DCT/DST	DHT/DFT	LOT/CLT
Γ_c	$\cos(\pi k/N)$	$\cos(2\pi k/N)$	$\cos(\pi/2N)$
Γ_s	$\sin(\pi k/N)$	$\sin(2\pi k/N)$	$\sin(\pi/2N)$
D_c	$C(k)\sqrt{\frac{2}{N}}\cos(\pi k/2N)$	$\sqrt{\frac{1}{N}}$	$\sqrt{\frac{1}{N}}(-1)^k j \cdot$ $\exp -j\theta_k \sin(\pi/4N)$
D_s	$C(k)\sqrt{\frac{2}{N}}\sin(\pi k/2N)$	0	$\sqrt{\frac{1}{N}}(-1)^k j \cdot$ $\exp -j\theta_k \cos(\pi/4N)$
$e(k)$	1	1	$\exp j2\theta_k$
$f(k)$	1	1	$\exp j2\theta_k$

Table 4.1: Coefficients of the Lattice structure for the DXT

The Complex Lapped Transform (CLT) [62] of $2N$ samples $[x(t-N+\frac{1}{2}), x(t-N+\frac{3}{2}), \ldots, x(x+N-\frac{1}{2})]$ is defined as

$$X_{clt}(k,t) = \frac{1}{\sqrt{N}} \sum_{n=t-(N-\frac{1}{2})}^{t+(N-\frac{1}{2})} x(n) \exp\{-j\frac{(2k+1)(n-t)\pi}{2N}\} \cos\frac{(n-t)\pi}{2N}, \quad k = 0, 1, \ldots, N-1. \tag{4.5}$$

The Lapped Orthogonal Transform (LOT) [70, 62] of $2N$ samples $[x(t-N+\frac{1}{2}), x(t-N+\frac{3}{2}), \ldots, x(t+N-\frac{1}{2})]$ is defined as

$$X_{lot}(k,t) = \begin{cases} \sqrt{\frac{2}{N}} \sum_{n=t-(N-\frac{1}{2})}^{t+(N-\frac{1}{2})} x(n) \cos\frac{(2k+1)(n-t)\pi}{2N} \cos\frac{(n-t)\pi}{2N} + \alpha_k, \\ \qquad k = 0, 2, \ldots, (N-2), \text{even part of the CLT} \\ \sqrt{\frac{2}{N}} \sum_{n=t-(N-\frac{1}{2})}^{t+(N-\frac{1}{2})} x(n) \sin\frac{(2k+1)(n-t)\pi}{2N} \cos\frac{(n-t)\pi}{2N} + \beta_{nk}, \\ \qquad k = 1, 3, \ldots, (N-1), \text{odd part of the CLT} \end{cases} \tag{4.6}$$

where $\alpha_k = \beta_{nk} = 0$, except for $\alpha_0 = -(\sqrt{2}-1)/(2\sqrt{2})$, and $\beta_{n(N-1)} = (-1)^{n+\frac{1}{2}}\alpha_0$.

Since the LOT is obtained from the even and odd value of k, we focus on the discussion of the dual generation for the CLT only. Define an Auxiliary Complex Lapped Transform (ACLT) of $2N$ samples $[x(t-N+\frac{1}{2}), x(t-N+\frac{3}{2}), \ldots, x(x+N-\frac{1}{2})]$ as

$$X_{clt}(k,t) = \frac{1}{\sqrt{N}} \sum_{n=t-(N-\frac{1}{2})}^{t+(N-\frac{1}{2})} x(n) \exp\{-j\frac{(2k+1)(n-t)\pi}{2N}\} \sin\frac{(n-t)\pi}{2N}, \quad k = 0, 1, \ldots, N-1. \tag{4.7}$$

Then, the CLT and ACLT can be dually generated from (4.2) and (4.3) with the corresponding coefficients listed in Table 4.1. All the transforms mentioned above can be realized by a lattice structure as shown in Fig. 4.1. This lattice structure is a modified normal form digital filter. Table 4.1 lists the coefficients in the unified lattice structure for different transforms. Here θ_k associated with the LOT/CLT equals $\frac{(2k+1)\pi}{4N}$.

4.3 Optimal Time-Recursive Architectures

4.3.1 Transfer Function Approach

Input data arrive serially in most real-time signal processing applications. If we can view the transform operation as a linear shift invariant (LSI) system which transforms the input sequence of samples into their transform coefficients, then it is similar to a filtering operation. The general approach to tackle a digital filter problem is to look at its transfer function. The transfer functions of the DXT can be derived using several approaches. We will derive them from the unified time-

recursive lattice structures as shown in Fig. 4.1. The time difference equations [1] for the dually generated pairs are

$$y_{xc,k}(t) = e(k)\left\{\Gamma_c\left[D_c\tilde{x}(t) + y_{xc,k}(t-1)\right] + \Gamma_s\left[D_s\tilde{x}(t) + y_{xs,k}(t-1)\right]\right\} \quad (4.8)$$

and

$$y_{xs,k}(t) = f(k)\left\{\Gamma_c\left[D_s\tilde{x}(t) + y_{xs,k}(t-1)\right] - \Gamma_s\left[D_c\tilde{x}(t) + y_{xc,k}(t-1)\right]\right\}, \quad (4.9)$$

where

$$\tilde{x}(t) = (-1)^k x(t+N) - x(t), \quad (4.10)$$

and $y_{xc,k}(t)$ and $y_{xs,k}(t)$ corresponds to $X_{xc}(k,t)$ and $X_{xs}(k,t)$ in (4.2) and (4.3). The z transform deduced from the above difference equations are

$$Y_{xc,k}(z) = e(k)\left\{\left(D_c\Gamma_c + D_s\Gamma_s\right)\tilde{X}(z) + \Gamma_c Y_{xc,k}(z)z^{-1} + \Gamma_s Y_{xs,k}(z)z^{-1}\right\} \quad (4.11)$$

and

$$Y_{xs,k}(z) = f(k)\left\{\left(D_s\Gamma_c - D_c\Gamma_s\right)\tilde{X}(z) + \Gamma_c Y_{xs,k}(z)z^{-1} - \Gamma_s Y_{xc,k}(z)z^{-1}\right\}. \quad (4.12)$$

$Y_{xs,k}(z)$ can be expressed in terms of $Y_{xc,k}(z)$ and $\tilde{X}(z)$ as

$$Y_{xs,k}(z) = \frac{f(k)\{\left(D_s\Gamma_c - D_c\Gamma_s\right)\tilde{X}(z) - \Gamma_s Y_{xc,k}(z)z^{-1}\}}{1 - f(k)\Gamma_c z^{-1}}, \quad (4.13)$$

it follows that the transfer function for $\frac{Y_{c,k}(z)}{X(z)}$ is given by

$$H_{xc,k}(z) = \frac{\left((-1)^k - z^{-N}\right)\left(e(k)[D_c\Gamma_c + D_s\Gamma_s] - e(k)f(k)D_c z^{-1}\right)}{1 - (e(k) + f(k))\Gamma_c z^{-1} + e(k)f(k)z^{-2}}. \quad (4.14)$$

Similarly, the transfer function for $\frac{Y_{s,k}(Z)}{X(z)}$ is given by

$$H_{xs,k}(z) = \frac{\left((-1)^k - z^{-N}\right)\left(f(k)[D_s\Gamma_c - D_c\Gamma_s] - e(k)f(k)D_s z^{-1}\right)}{1 - (e(k) + f(k))\Gamma_c z^{-1} + e(k)f(k)z^{-2}}. \quad (4.15)$$

[1] The time index t is an integer parameter.

From Table 4.1 and the transfer functions derived above, the transfer functions of different discrete sinusoidal transforms are given by

$$H_c(z) = \sqrt{\frac{2}{N}} C(k) \left((-1)^k - z^{-N}\right) \left(\cos \frac{\pi k}{2N}\right) \frac{(1 - z^{-1})}{\left(1 - 2\left(\cos \frac{\pi k}{N}\right) z^{-1} + z^{-2}\right)}, \tag{4.16}$$

$$H_s(z) = -\sqrt{\frac{2}{N}} C(k) \left((-1)^k - z^{-N}\right) \left(\sin \frac{\pi k}{2N}\right) \frac{(1 + z^{-1})}{\left(1 - 2\left(\cos \frac{\pi k}{N}\right) z^{-1} + z^{-2}\right)}, \tag{4.17}$$

$$H_h(z) = \frac{1}{\sqrt{N}} \left(1 - z^{-N}\right) \left(\frac{\cos \frac{2\pi k}{N} - \sin \frac{2\pi k}{N} - z^{-1}}{1 - 2\left(\cos \frac{2\pi k}{N}\right) z^{-1} + z^{-2}}\right), \tag{4.18}$$

$$H_f(z) = \frac{1}{\sqrt{N}} \left(1 - z^{-N}\right) \left(\frac{\cos \frac{2\pi k}{N} + j \sin \frac{2\pi k}{N} - z^{-1}}{1 - 2\left(\cos \frac{2\pi k}{N}\right) z^{-1} + z^{-2}}\right). \tag{4.19}$$

Because the size of the input data is $2N$ in the CLT, the updating vector is $1 - z^{2N}$ instead of $1 - z^N$. The transfer function is obtained by substituting the corresponding coefficients in Table 4.1 to (4.8), resulting in

$$H_{clt}(z) = \left(1 - z^{-2N}\right) \frac{1}{\sqrt{N}} j(-1)^{k+1} \frac{\left(\sin \frac{\pi}{4N}\right) e^{j\theta} \left(1 + e^{j2\theta} z^{-1}\right)}{1 - e^{j2\theta} 2\left(\cos \frac{\pi}{2N}\right) z^{-1} + e^{j4\theta} z^{-2}} \tag{4.20}$$

$$\theta = \frac{(2k+1)\pi}{4N}.$$

It follows that for the LOT,

$$H_{lote}(z) = evenpartof\{H_{clt}\} \tag{4.21}$$

$$H_{loto}(z) = oddpartof\{H_{clt}\}. \tag{4.22}$$

We know from (4.1) that the transfer functions of these transforms are of finite impulse response. Hence, the poles in the denominator will be cancelled by the zeros of $((-1)^k - z^{-N})$ in the nominator. We observe that when the updating vector $(1 - z^{-N})$ is factored out, the basic structure of all the transforms is

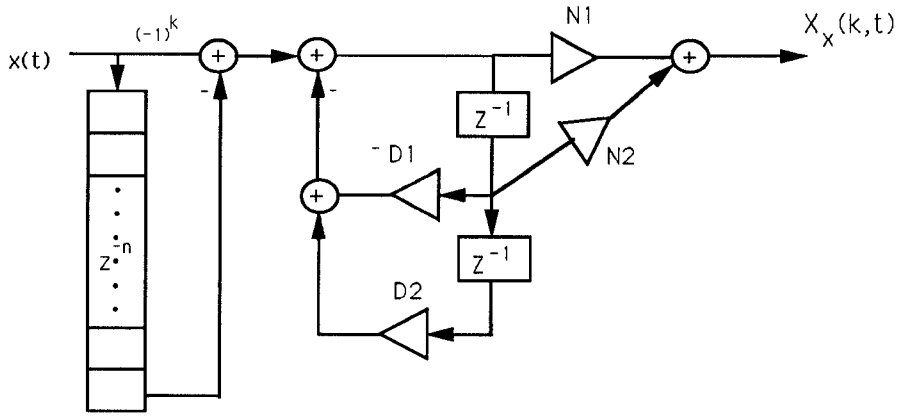


Figure 4.2: The universal IIR filter module.

composed of a FIR and an IIR filter with a second order denominator and a first order numerator, *i.e.* we are using an IIR filter to realize a FIR filter. This realization can greatly reduce the hardware complexity compared with the implementation consisting of FIR structures.

4.3.2 The Unified IIR Filter Architectures

From the transfer functions derived above, we observe that the DXT can be realized using a single universal filter module consisting of a shift register array and a second order IIR filter. This structure is depicted in Fig. 4.2. The coefficients of the universal IIR module for different transforms are listed in Table 4.2.

We note from (4.16) and (4.17) that the DCT and the DST share the same denominator and can be simultaneously generated using an IIR filter structure with three multipliers as depicted in Fig. 4.3. Compared with the lattice struc-

	k	n	$D1$	$D2$	$N1$	$N2$
DCT	k	N	$2\cos(\pi k/N)$	1	$C(k)\sqrt{\frac{2}{N}}$ $\cos(\pi k/2N)$	$-C(k)\sqrt{\frac{2}{N}}$ $\cos(\pi k/2N)$
DST	k	N	$2\cos(\pi k/N)$	1	$-C(k)\sqrt{\frac{2}{N}}$ $\sin(\pi k/2N)$	$-C(k)\sqrt{\frac{2}{N}}$ $\sin(\pi k/2N)$
DHT	0	N	$2\cos(2\pi k/N)$	1	$\sqrt{\frac{1}{N}}[\cos(2\pi k/N)$ $-\sin(2\pi k/N)]$	$\sqrt{\frac{1}{N}}$
DFT	0	N	$2\cos(2\pi k/N)$	1	$\sqrt{\frac{1}{N}}[\cos(2\pi k/N)$ $+j\sin(2\pi k/N)]$	$\sqrt{\frac{1}{N}}$
CLT	0	$2N$	$\exp j2\theta_k$ $2\cos(\pi/2N)$	$\exp j4\theta_k$	$\sin(\pi/4N)^2$ $\exp j\theta_k$	$\sin(\pi/4N)$ $\cos(\pi/4N)$ $(-1)^k$ $\exp j4\theta_k$

Table 4.2: Coefficients of the universal IIR filter structure for the DXT.

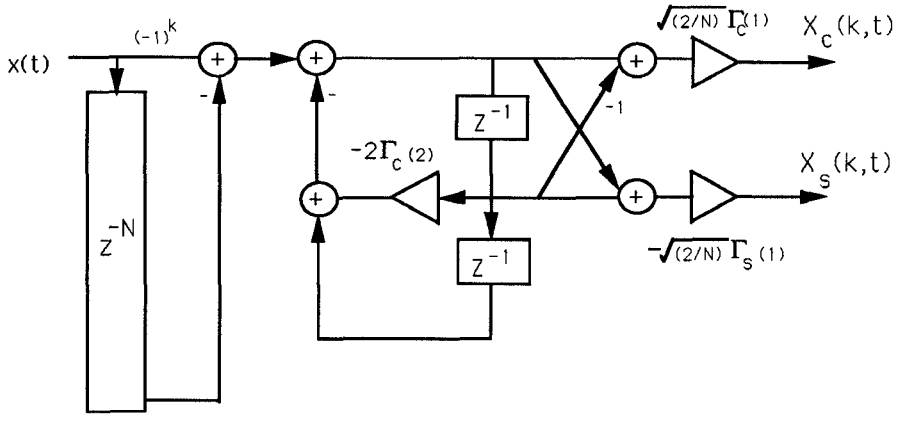


Figure 4.3: The IIR filter structure for the DCT and DST.

ture for the DCT and DST[17], the IIR realization requires only half as many multipliers. The difference is that the IIR structure implements the denominator of the transfer function in the direct form, while the lattice structure implements the poles in the normal form. From (4.18) and (4.19), we also observe that a single unified filter structure can be used to generate both the DHT and the DFT. This structure is depicted in Fig. 4.4.

The transfer function derived in (4.21) is in complex form. The IIR filter architecture for the LOT and the CLT is shown in Fig. 4.5.

We will show in the following how to realize the CLT using real operations. The definition of the CLT in (4.5) can be rewritten as

$$X_{clt}(k) = (-1)^k j \frac{1}{\sqrt{N}} \sum_{n=0}^{2N-1} x(n) \exp\{-j\frac{(2k+1)(2n+1)\pi}{4N}\} \sin\frac{(2n+1)\pi}{4N}, \quad k = 0, 1, \ldots, N-1. \tag{4.23}$$

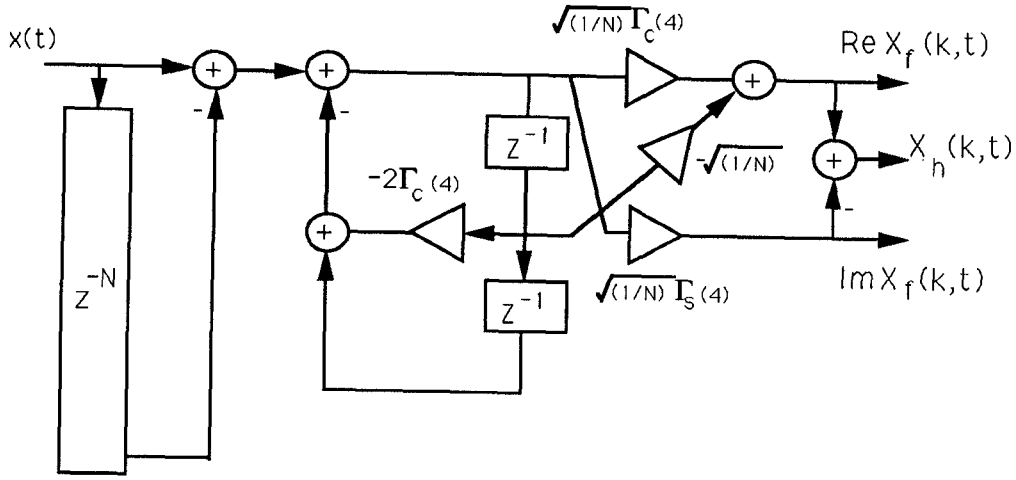


Figure 4.4: The IIR filter structure for the DHT and DFT.

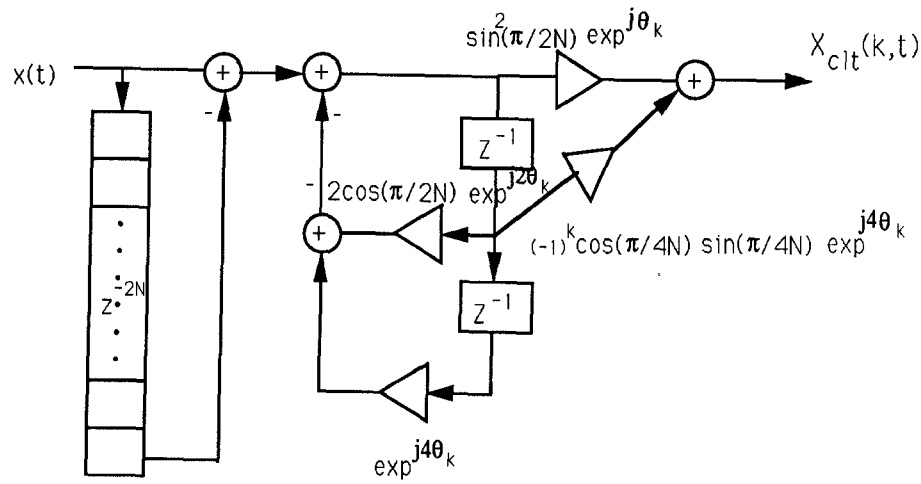


Figure 4.5: The IIR filter structure for the LOT and CLT.

Transforms	multipliers	adders
DCT	$2N-2$	$3N+2$
DST	$2N-2$	$3N+2$
DHT	$2N$	$3N+1$
DFT	$3N-2$	$3N+1$
LOT*	$4N$	$4N$
CLT*	$4N$	$4N$
DCT and DST	$3N-3$	$4N+2$
DHT and DFT	$3N-2$	$4N+1$

Table 4.3: Number of multipliers and adders for different transforms with IIR filter realizations(Here * denotes complex operations).

If we define another transform with basis functions only length N,

$$\begin{aligned} t_{nk} &= \frac{1}{N}\exp\frac{j(2n+1)k\pi}{2N} \\ &= \frac{1}{N}\{DCT_{nk} - j\cdot DST_{nk}\}, \qquad n,k = 0,1,...,N-1. \end{aligned} \tag{4.24}$$

then the CLT can be expressed in the form of [62]

$$\begin{aligned} X_{clt}(k) = \frac{1}{2}(-1)^k \sum_{n=0}^{N-1} x(n)[t_{nk} - t_{n(k+1)}] \\ +\frac{1}{2}(-1)^k \sum_{n=N}^{2N-1} x(n)[t_{nk} + t_{n(k+1)}], \end{aligned} \tag{4.25}$$

This leads to the CLT architecture as shown in Fig. 4.6, in which the t_{mn} are generated by using the DCT and DST dual generating circuit as depicted in Fig. 4.3. The number of multipliers and adders required for these IIR filter structures are summarized in Table 4.3.

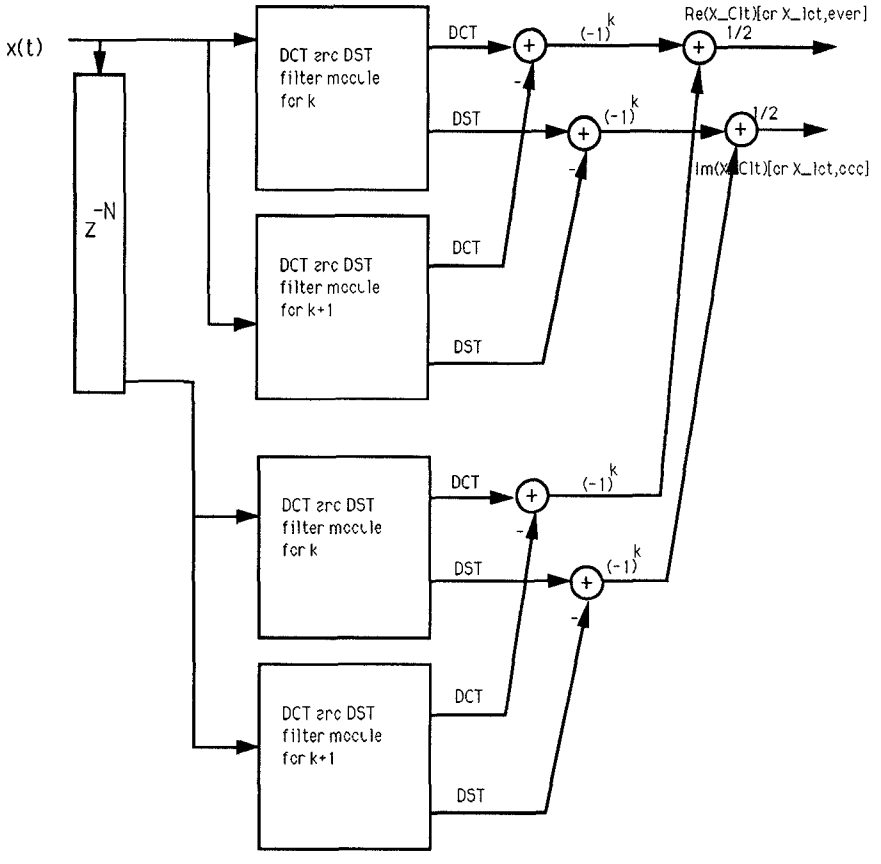


Figure 4.6: The IIR filter structure for real operation of the LOT and CLT.

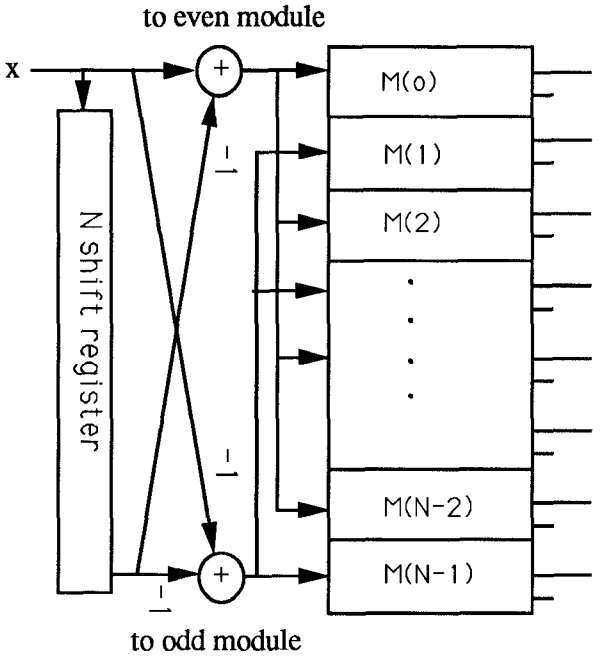


Figure 4.7: The parallel IIR filter structure for 1-D DXT.

The architecture to generate 1-D DXT is depicted in Fig. 4.7. This parallel structure consists of a shift register array of size N, two adders, and N IIR filter modules. Two sets of inputs $x(t+N)-x(t)$ and $-x(t+N)-x(t)$ are generated for the even and odd filter modules respectively. When a new datum $x(t)$ arrives, a new set of transform coefficients are obtained in O(1) time, *i.e.*, the throughput rate is O(1).

4.4 Architectures for Inverse Transforms

Inverse transforms are important in retrieving original information in digital communication systems. The inverse DHT and DFT are given by

$$x_h(n,t) = \frac{1}{\sqrt{N}} \sum_{k=t}^{t+N-1} X_h(k) cas\left(2n\frac{\pi(k-t)}{N}\right) \quad n=0,1,...,N-1. \tag{4.26}$$

$$x_f(n,t) = \frac{1}{\sqrt{N}} \sum_{k=t}^{t+N-1} X_f(k) \exp\{j2n\frac{\pi(k-t)}{N}\}, \quad n=0,1,\ldots,N-1. \tag{4.27}$$

We observe that the transfer function of the inverse DHT(IDHT) is exactly the same as its forward transform. The transfer function of the inverse DFT(IDFT) is given by

$$H_f(z) = \frac{1}{\sqrt{N}}\left(1 - z^{-N}\right)\left(\frac{\cos\frac{2\pi k}{N} - j\sin\frac{2\pi k}{N} - z^{-1}}{1 - 2\cos\frac{2\pi k}{N}z^{-1} + z^{-2}}\right), \tag{4.28}$$

which is same as (4.19) except that the imaginary part is negated. Therefore, the IDHT and IDFT can be realized by using the same architecture as those depicted in Fig. 4.4 except that we have to add an inverter at the output of the $ImX_f(k,t)$.

The inverse DCT and DST (IDCT and TDST) are defined as follows:

$$x_c(n,t) = \sqrt{\frac{2}{N}}\sum_{k=t}^{t+N-1} C(k-t)X_c(k)\cos\left[(n+\frac{1}{2})\frac{(k-t)\pi}{N}\right], \tag{4.29}$$

$$n = 0, 1, ..., N-1. \tag{4.30}$$

$$x_s(n,t) = \sqrt{\frac{2}{N}}\sum_{k=t+1}^{t+N} C(k-t)X_s(k)\sin\left[(n+\frac{1}{2})\frac{(k-t)\pi}{2N}\right], \tag{4.31}$$

$$n = 0, 1, ..., N-1. \tag{4.32}$$

Because $C(k)$ is inside the transform, the architectures require some modification. Since $C(k) = 1$ except for $k = 0$ or $k = N$, we can rewrite (4.29) as

$$x_c(n,t) = \sqrt{\frac{2}{N}}\sum_{k=t}^{t+N-1} X(k)\cos\left[(n+\frac{1}{2})\frac{(k-t)\pi}{N}\right] + \sqrt{\frac{2}{N}}(\sqrt{\frac{1}{2}} - 1)X(t). \tag{4.33}$$

We observe that the first part of the above equation is of the same form as that of DCT except for the leading constant coefficients $C(k)$. Hence, the transfer function of the IDCT is that of the DCT plus one delay term specified as follows

$$\begin{aligned} H_{ic}(z) &= \sqrt{\frac{2}{N}}\left((-1)^k - z^{-N}\right)\cos\frac{\pi k}{2N}\frac{(1-z^{-1})}{\left(1 - 2\cos\frac{\pi k}{N}z^{-1} + z^{-2}\right)} \\ &\quad + \sqrt{\frac{2}{N}}(\sqrt{\frac{1}{2}} - 1)z^{-(N-1)} \end{aligned} \tag{4.34}$$

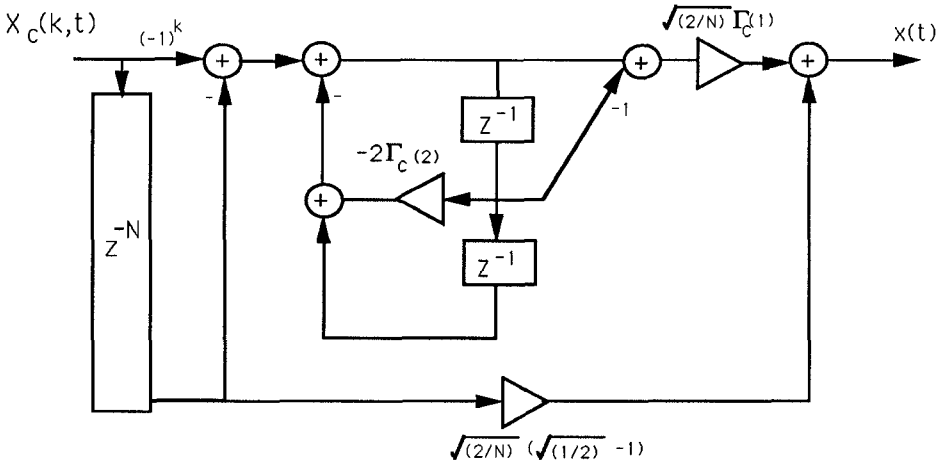


Figure 4.8: The IIR filter structure for the IDCT.

This implies that the IDCT can be implemented by using the same architecture as the DCT and adding the compensated term $\sqrt{\frac{2}{N}}(\sqrt{\frac{1}{2}}-1)X(t)$. The architecture is shown in Fig. 4.8.

Similarly, the IDST can be rewritten as

$$x_s(n,t) = \sum_{k=t}^{t+N-1} X(k)\sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right] + \sqrt{\frac{2}{N}}(\sqrt{\frac{1}{2}}-1)X(t+N-1) \quad (4.35)$$

whose transfer function is

$$H_{is}(z) = \sqrt{\frac{2}{N}}\left((-1)^k - z^{-N}\right)\sin\frac{\pi k}{2N}\frac{(1-z^{-1})}{\left(1-2\cos\frac{\pi k}{N}z^{-1}+z^{-2}\right)} + \sqrt{\frac{2}{N}}(\sqrt{\frac{1}{2}}-1) \quad (4.36)$$

The architecture for IDST is shown in Fig. 4.9.

The Inverse Complex Lapped Transform (ICLT) [62] of samples $[X(t), X(t+1), ..., X(t+N-1), X(t+N), ..., X(t+2N-1)]$ is defined as

$$x_{clt}(k,t) = \frac{1}{2\sqrt{N}}\sum_{k=t}^{t+N-1}[X(k)+X(k-1) \quad (4.37)$$

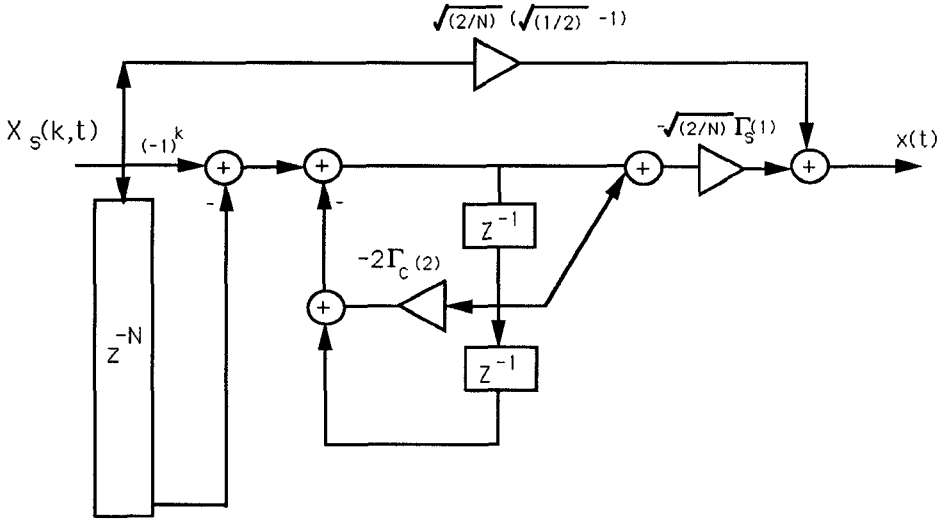


Figure 4.9: The IIR filter structure for the IDST.

$$+(-1)^k(X(k+N)+X(k+N-1))]\exp\{j\frac{(2n+1)(k-t)\pi}{2N}\}.$$

The architecture of the IDCLT is depicted in Fig. 4.10. From the previous derivation, we see that the inverse transforms can be obtained by using the same architectures of the forward transforms with one additional branch of multiplier.

4.5 Theoretical Basis

The basis functions of all the discrete sinusoidal transforms mentioned above corresponds to a set of orthogonal polynomials. One of the important characteristics of orthogonal polynomials is that any three consecutive polynomials are related by the *Fundamental Recurrence Formula* [71] given by

$$P_n(k) = (k - c_n)P_{n-1}(k) - \lambda_n P_{n-2}(k). \tag{4.38}$$

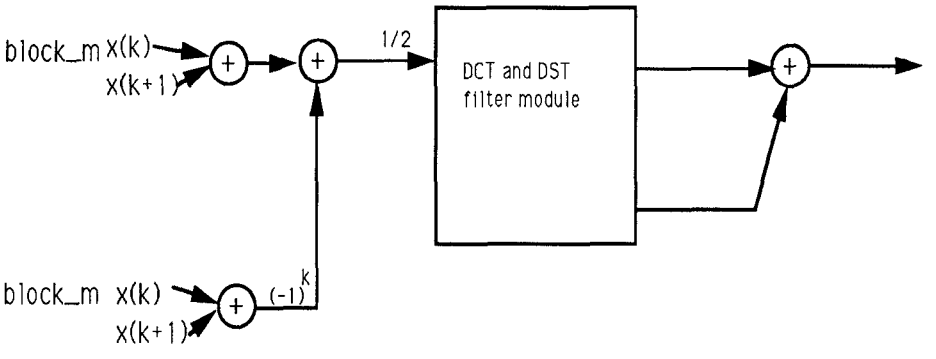


Figure 4.10: The IIR filter structure for the IDCLT.

The discrete transforms discussed in the previous section satisfy a simpler version of the recurrence relation. More precisely, the parameters c_n and λ_n are independent of n and the basis function $P_n(k)$ is periodic in n and k of period N. In these cases, the *Fundamental Recurrence Formula* can be rewritten as

$$P_n(k) = (k-c)P_{n-1}(k) - \lambda P_{n-2}(k), \qquad n = 0,1,..,N-1, \qquad k = 0,1,2,..,N-1. \tag{4.39}$$

For different discrete sinusoidal transforms, the corresponding parameters k, c, λ in the *Fundamental Recurrence Formula* are stated in Table 4.4.

Lemma 1 *For all discrete transforms whose basis functions satisfy the Fundamental Recurrence Formula (4.39), the z-transform of the basis functions* $\{P_n(k)\}$ *can be expressed as a rational function with a second order denominator that is the characteristic equation of the Fundamental Recurrence Formula.*

	DCT	DST	DHT	DFT	CLT
k	2 $\cos(\pi k/N)$	$\cos(\pi k/N)$ $\cdot 2$	2 $\cos(2\pi k/N)$	$\cos(2\pi k/N)$ 2	$2\exp j2\theta_k$ $\cos(\pi/2N)$
c	0	0	0	0	0
λ	1	1	1	1	$\exp -j4\theta_k$
P_0	$\cos(\pi k/2N)$	$\sin(\pi k/2N)$	1	1	1
P_{-1}	P_0	$-P_0$	$\cos(2\pi k/N)$ $\sin(2\pi k/N)$	$\cos(2\pi k/N)$ $-j\sin(2\pi k/N)$	$\exp j2\theta_k$ $\cos(\pi/2N)$
P_{N-1}	$(-1)^k P_0$	$(-1)^{k+1} P_0$	P_{-1}	P_{-1}	$(-1)^k j$ $\sin(\pi/2N)$ $\exp j2\theta_k$
P_N	P_0	P_0	1	1	$(-1)^k j$

Table 4.4: Corresponding coefficients in the Recurrence Formula for different DXT.

Proof: Since any $P_n(k)$ depends only on the previous two terms, the first two polynomial terms, $P_{-1}(k)$ and $P_{-2}(k)$, uniquely specify the entire set of basis functions.

Apply z transform on index n to both sides of (4.39),

$$\begin{aligned} P(z,k) &= \sum_{n=0}^{N-1} z^{-n} P_n(k) \qquad (4.40) \\ &= \sum_{n=0}^{N-1} \{(k-c)z^{-n}P_{n-1}(k) - \lambda z^{-n}P_{n-2}(k)\} \\ &= (k-c)[P_{-1}(k) + z^{-1}\sum_{n=0}^{N-1} z^{-n}P_n(k) - z^{-N}P_{N-1}(k)] - \\ &\lambda[P_{-2}(k) + z^{-1}P_{-1}(k) + z^{-2}\sum_{n=0}^{N-1} z^{-n}P_n(k)] \\ &\lambda[-z^{-N}P_{N-2}(k) - z^{-(N+1)}P_{N-1}(k)] \\ &= (k-c)z^{-1}P(z,k) - \lambda z^{-2}P(z,k) + [(k-c)P_{-1}(k) - \lambda P_{-2}(k)] \\ &-z^{-N}[(k-c)P_{N-1}(k) - \lambda P_{N-2}(k)] \\ &-\lambda z^{-1}P_{-1}(k) + \lambda z^{-(N+1)}P_{N-1}(k) \\ &k = 1, 2, .., N-1. \end{aligned}$$

Factoring out $P(z,k)$, we obtain

$$\begin{aligned} P(z,k) &= \frac{z^{-(N-1)}\lambda P_{N-1}(k) - P_N(k)z^{-(N-2)} - \lambda P_{-1}(k)z^1 + P_0(k)z^2}{\lambda - (k-c)z + z^2} \qquad (4.41) \\ &= \frac{z^2(P_0(k) - P_N(k)z^{-N}) - \lambda z(P_{-1}(k) - P_{N-1}(k)z^{-N})}{\lambda - (k-c)z + z^2}. \end{aligned}$$

Because of the second-order recurrence relation, the denominators of the z-transform of the basis functions are second-order polynomials in z.

The characteristic equation of the *Fundamental Recurrence Formula* (4.39) is obtained by solving the homogeneous solutions of the difference equation (4.39). The homogeneous equation is given by

$$P(z,k) = (k-c)P(z,k)z^{-1} - \lambda P(z,k)z^{-2}. \qquad (4.42)$$

Combining both sides of the equation, we have

$$P(z,k)z^{-2}(\lambda-(k-c)z+z^2)=0. \tag{4.43}$$

Since $P(z,k)$ dose not equal to zero, we have that $(\lambda-(k-c)z+z^2)$ equals to zero and hence the characteristics equation is $(\lambda-(k-c)z+z^2)$, which is the denominator. □

The transfer function of the discrete transforms (DXT) is derived from (4.1), that can be rewritten as

$$X(k,t)=C(k)\sum_{n=0}^{N-1}x(n+t)P_n(k), t=0,1,2,.... \tag{4.44}$$

Performing the z-transform on the index t on both sides of the above equation, we have

$$H_X(z) \;=\; C(k)z^{-(N-1)}\sum_{n=0}^{N-1}z^nP_n(k)=C(k)z^{-(N-1)}P(z^{-1},k) \tag{4.45}$$

which is the z-transform of the basis orthogonal polynominals with index z replaced by z^{-1} and multiplied by $C(k)z^{-(N-1)}$. That is, the transfer function of the discrete transform can also be expressed as a rational function with a second order denominator

$$H_x(z)=C(k)\frac{\left(\lambda P_{N-1}(k)-P_N(k)z^{-1}-\lambda P_{-1}(k)z^{-N}+P_0(k)z^{-(N+1)}\right)}{(\lambda-(k-c)z^{-1}+z^{-2})}. \tag{4.46}$$

Here we illustrate another way to derive the transfer function of the discrete sinusoidal transforms. Substituting the coefficients listed in Table 5 to (4.46), we obtain the transfer functions derived in Section 4.3.1.

Lemma 2 *To compute the discrete sinusoidal transforms time recursively, we have to factor out the updating component* $(1-z^{-N})$ *or* $(1+z^{-N})$ *in the filter*

realization. There exists an updating component $(1 + z^{-N})$ *or* $(1 - z^{-N})$ *in the nominator of the transfer function of the discrete sinusoidal transform, if and only if the boundary conditions of the basis function satisfy* $P_0 = \pm P_N$ *and* $P_{-1} = \pm P_{N-1}$.

Proof: If the updating vector can be realized by $(1 + z^{-N})$ or $(1 - z^{-N})$, then the nominator of (4.46) must contain the factor $(1 + z^{-N})$ or $(1 - z^{-N})$. That is, the nominator can be expressed as

$$\lambda P_{N-1}(k) - P_N(k)z^{-1} - \lambda P_{-1}(k)z^{-N} + P_0(k)z^{-(N+1)} = (1 \pm z^{-N})(a + bz^{-1}), \quad (4.47)$$

since it is a $(-N-1)$ degree polynominal. Expand the right side of the above equation, we have

$$\lambda P_{N-1}(k) - P_N(k)z^{-1} - \lambda P_{-1}(k)z^{-N} + P_0(k)z^{-(N+1)} = a + bz^{-1} \pm az^{-N} \pm bz^{-N-1}, \quad (4.48)$$

it follows that

$$\begin{aligned} a &= \mp\lambda P_{-1}(k) = \lambda P_{N-1}(k) \quad (4.49) \\ b &= \pm P_0(k) = -P_N(k), \end{aligned}$$

and

$$\begin{aligned} P_0(k) &= \pm P_N(k) \quad (4.50) \\ P_{-1}(k) &= \mp P_{N-1}(k). \end{aligned}$$

This proves the necessary condition. If $P_0 = \pm P_N$ and $P_{-1} = \pm P_{N-1}$, then the nominator in (4.46) becomes

$$\begin{aligned} &\lambda P_{N-1}(k) - P_N(k)z^{-1} - \lambda P_{-1}(k)z^{-N} + P_0(k)z^{-(N+1)} \quad (4.51) \\ &= \mp\lambda P_{-1}(k) \pm P_0(k)z^{-1} - \lambda P_{-1}(k)z^{-N} + P_0(k)z^{-(N+1)} \\ &= (1 \pm z^{-N})(\lambda P_0(k)z^{-1} \mp \lambda P_{-1}(k)), \end{aligned}$$

which means the nominator contains the factor $(1 \pm z^{-N})$. □

Lemma 3 *All the transforms that satisfies Lemma 1 and Lemma 2 can be realized by an updating FIR filter with transfer function* $(1 - z^{-N})$ *or* $(1 + z^{-N})$, *and an IIR filter with second order denominator and first order nominator whose coefficients are dependent on* $\lambda, (k - c), P_0$ *and* P_{-1}.

Proof: If Lemma 1 and 2 are satisfied, the transfer function can be expressed as

$$H_x(z) = C(k)\frac{(1 \pm z^{-N})(\lambda P_{N-1} - P_N z^{-1})}{(\lambda - (k-c)z^{-1} + z^{-2})}. \tag{4.52}$$

Therefore, the transform can be realized by the filter structure as shown in Fig. 4.2. The coefficients are

$$\begin{aligned} D1 &= (k - c) \\ D2 &= \lambda \\ N1 &= \lambda P_{N-1} \\ N2 &= -P_N. \end{aligned} \tag{4.53}$$

□

Lemma 3 implies that if a transform can be computed time-recursively, a maximum of four multipliers required to realize the transform. Fig. 4.2 shows a good example of this case.

Lemma 4 *For the discrete sinusoidal transforms, the roots of the characteristic equation belong to the set of the root of* $(1 \pm z^{-N})$.

Proof: Since the discrete sinusoidal transform is FIR in natural, the roots of the denominators should be cancelled by the zeros of the nominator. In general, the

roots of the denominator are complex conjugate poles because of $(k-c)^2-4\lambda < 0$. Therefore, the poles should be cancelled by the zeros of the $(1 \pm z^{-N})$, and the roots of the denominator

$$z1, z2 = \frac{(k-c) \pm \sqrt{(k-c)^2 - 4\lambda}}{2\lambda} \in \tag{4.54}$$
$$\left\{ \begin{array}{ll} \exp \frac{j2\pi n}{N} & n = 0, ..., N-1., z^N = 1 \\ \exp \frac{j\pi(2n+1)}{N} & n = 0, ..., N-1., z^N = -1. \end{array} \right\}$$

□

All the discrete sinusoidal transforms list in Table 4 satisfies Lemmas 1 through 4. Therefore, these transforms can be computed time recursively and can be realized by a FIR filter with transfer function $(1 \pm z^{-N})$ and an IIR filter with second order polynominals. These facts support the results obtained in Section 3 and 4.

Lemma 5 *If two transforms can be dually generated, then they share the same autoregressive model in their IIR filter structure.*

Proof: The basis polynomial p_n and q_n of the dual generated transform pairs satisfy the following equations

$$\begin{aligned} p_n &= D_{xc} p_{n-1} + D_{xs} q_{n-1} \\ q_n &= D_{xc} q_{n-1} - D_{xs} p_{n-1}. \end{aligned} \tag{4.55}$$

Since p_n and q_n are dually generated and from (4.56), they have the same characteristic equation. That is

$$I - Az^{-1} = 0, \tag{4.56}$$

where

$$A = \begin{bmatrix} D_{xc} & D_{xs} \\ -D_{xs} & D_{xc} \end{bmatrix}$$

As shown in Lemma 1, the roots of the denominators are the roots of the characteristics equation. Since p_n and q_n have the same characteristic equation, they have the same denominator. Hence, both transform have identical poles, and as a result, the same autoregressive filter form. □.

Example 1 *The DCT and DST are dual generated transform pairs and share the same second order denominator.*

As shown in [17], the DCT and DST satisfy

$$\begin{aligned} \cos\left[\frac{\pi(2(n+1)+1)k}{2N}\right] &= \cos\left[\frac{\pi k}{N}\right]\cos\left[\frac{\pi(2n+1)k}{2N}\right] \\ &\quad -\sin\left[\frac{\pi k}{N}\right]\sin\left[\frac{\pi(2n+1)k}{2N}\right] \\ \sin\left[\frac{\pi(2(n+1)+1)k}{2N}\right] &= \cos\left[\frac{\pi k}{N}\right]\sin\left[\frac{\pi(2n+1)k}{2N}\right] \\ &\quad +\sin\left[\frac{\pi k}{N}\right]\cos\left[\frac{\pi(2n+1)k}{2N}\right], \end{aligned} \tag{4.57}$$

it follows that

$$\begin{aligned} D_{xc} &= \cos\left[\frac{\pi k}{N}\right], \\ D_{xs} &= -\sin\left[\frac{\pi k}{N}\right]. \end{aligned} \tag{4.58}$$

From (4.56), the poles are the root the following equation $1 - 2\cos\left[\frac{\pi k}{N}\right] z^{-1} + z^{-2} = 0$, which is the same as the characteristic equation derived from the Lemma 1. This is why the DCT, DST and DFT, DHT share the same second order autoregressive structure. From Lemma 3, it is noted that a maximum of $4N$ multipliers is required to realize the transform. Due to the fact that $\lambda = 1$ and $P_N = \pm P_{N-1}$ for the case of the DCT and DST, we can see that $2N$ multipliers for the DCT and DST is minimum for this realization. Based on

Lemma 5, we can combine the denominator together for the dual generation of DCT and DST. This gives an average $1.5N$ multipliers to realize the DCT or DST. We believe that this is the best we can achieve for real time computation.

4.6 Time-Recursive Multi-dimensional Transforms

Multi-dimensional transforms provide powerful tools for multi-dimensional signal processing. Some of the important applications are in the areas of signal reconstruction, speech processing, spectrum analysis, tomography, image processing, and computer vision. Specifically, in multispectral imaging, interframe video imaging, and computer tomography, we have to work with three or (higher) dimensional data. It is difficult to generalize the existing fast 1-D algorithms to 3-D or higher dimensional transforms. However, our time-recursive concept can be easily extended to multi-dimensional transforms resulting in architectures that are simple, modular, and hence suitable for VLSI implementation. Since the 3-D DCT is very useful in processing interframe video imaging data, we first describe the filter architecture for the 3-D DCT, and then generalize it to any multi-dimensional discrete sinusoidal transform.

4.6.1 Time-Recursive Structures for 3-D DCT

The basic concept of time-recursive computation is to compute the new transform at time $(t+1)$ based on the transform at time t. The operations can be divided into two parts, one consists of computing the difference of the input data between time t and $(t+1)$ and the other consists of performing the recursive updating.

Looking at the basic architecture of computing 1-D DXT as shown in Fig. 4.2, the basic structure consists of three components: shift registers, adders, and IIR arrays. The shift register is used to store the input data from $x(t)$ to $x(t+N)$; adders are used to compute the difference between data $x(t)$ and $x(t+N)$ and the IIR arrays are used to perform the computation recursively. We will show that the d-D DXT can be computed by using d blocks consisting of shift registers, adders, and filter arrays, each performing the time-recursive computation along a dimension.

For 1-D time-recursive DXT, the input data window is moved one sample at a time. That is, the input data vector at time t is given by the vector $[x(t), ..., x(t+N-1)]$, and at time $(t+1)$ the input data consists of the vector $[x(t+1), ..., x(t+N)]$. The time-recursive relation for the 2-D transforms is based on updating the input data row by row [65]. A 2D-DCT for HDTV application based on the lattice structure as considered in [65]. The input data sequences for time-recursive 3-D transforms are as shown in Fig. 4.11. Here, we assume that input data is updated frame by frame in the third axis n_3, that is, the range of the input data $x(n_1, n_2, n_3)$ is $\{n_1 = 0, \cdots, N-1; n_2 = 0, \cdots, N-1; n_3 = 0, 1, 2, ...\}$. We call the input data frame $x(n_1, n_2, t)$ for a specific index t as the tth frame input data. The 3-D DCT of the tth frame input data is defined as

$$X_{c^3}(k_1, k_2, k_3, t) = C(k_1)C(k_2)C(k_3) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \sum_{n_3=t}^{t+N-1} x(n_1, n_2, n_3) \qquad (4.59)$$
$$\cdot \cos\left[\frac{\pi(2n_1+1)k_1}{2N}\right] \cos\left[\frac{\pi(2n_2+1)k_2}{2N}\right] \cos\left[\frac{\pi[2(n_3-t)+1]k_3}{2N}\right].$$

The 3-D DCT of the $(t+1)$ frame input data $\{n_1 = 0, \cdots, N-1; n_2 = 0, \cdots, N-1; n_3 = t+1, ..., t+N.\}$ is

$$X_{c^3}(k_1, k_2, k_3, t+1) = C(k_1)C(k_2)C(k_3) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \sum_{n_3=t+1}^{t+N} x(n_1, n_2, n_3)$$

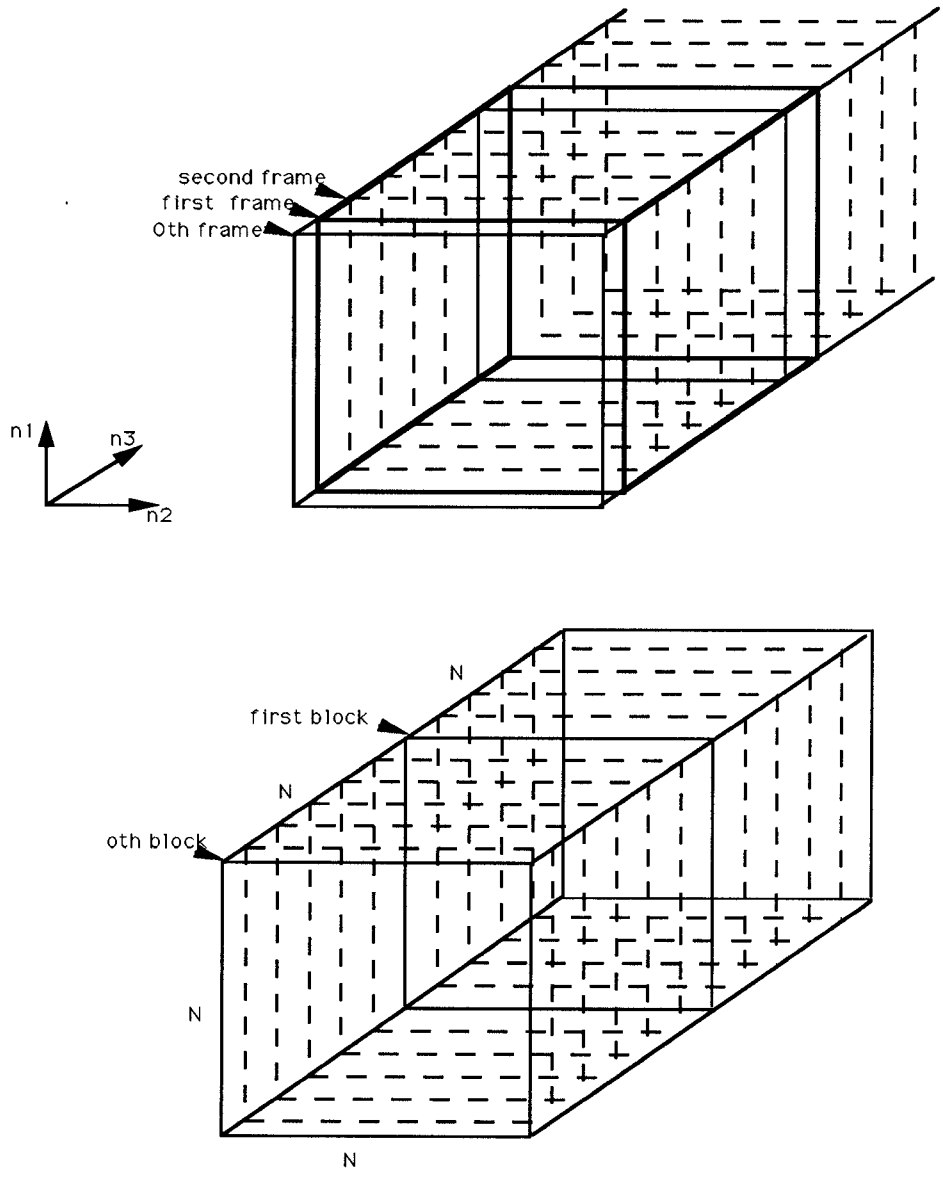


Figure 4.11: The input sequence of the time-recursive based 3-D transforms.

$$\cdot \cos\left[\frac{\pi(2n_1+1)k_1}{2N}\right] \tag{4.60}$$
$$\cdot \cos\left[\frac{\pi(2n_2+1)k_2}{2N}\right]$$
$$\cdot \cos\left[\frac{\pi[2(n_3-t-1)+1]k_3}{2N}\right].$$

The concept of time-recursive approach is to update the 3-D DCT of the $(t+1)$ frame input data based on $X_{c^3}(k_1,k_2,k_3,t)$. The time-recursive relations between the 3-D DCT $X_{c^3}(k_1,k_2,k_3,t+1)$ of the $(t+1)$th input frame and the 3-D DCT $X_{c^3}(k_1,k_2,k_3,t)$ of the (t)th input frame are

$$\begin{aligned} & X_{c^3}(k_1,k_2,k_3,t+1) \qquad (4.61)\\ &= \left[X_{c^3}(k_1,k_2,k_3,t) + X_{c^2}[k_1,k_2,t_\Delta]\frac{2}{N}C(k_3)\cos(\frac{\pi k_3}{2N})\right]\cos\left(\frac{\pi k_3}{N}\right)\\ &+ \left[X_{c^2s}(k_1,k_2,k_3,t) + X_{c^2}[k_1,k_2,t_\Delta]\frac{2}{N}C(k_3)\sin(\frac{\pi k_3}{2N})\right]\sin\left(\frac{\pi k_3}{N}\right). \end{aligned}$$

Here we introduce another 3-D transform $X_{c^2s}(k_1,k_2,k_3,t)$ defined as

$$\begin{aligned} X_{c^2s}(k_1,k_2,k_3,t) &= C(k_1)C(k_2)C(k_3)\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}\sum_{n_3=t}^{t+N-1}x(n_1,n_2,n_3) \qquad (4.62)\\ &\cos\left[\frac{\pi(2n_1+1)k_1}{2N}\right]\\ &\cdot\cos\left[\frac{\pi(2n_2+1)k_2}{2N}\right]\\ &\cdot\sin\left[\frac{\pi[2(n_3-t)+1]k_3}{2N}\right]. \end{aligned}$$

A similar relation exists between the updated transform $X_{c^2s}(k_1,k_2,k_3,t+1)$ and the previous transform $X_{c^2s}(k_1,k_2,k_3,t)$, that is

$$\begin{aligned} & X_{c^2s}(k_1,k_2,k_3,t+1) \qquad (4.63)\\ &= \left[X_{c^2s}(k_1,k_2,k_3,t) + X_{c^2}[k_1,k_2,t_\Delta]\frac{2}{N}C(k_3)\sin(\frac{\pi k_3}{2N})\right]\cos\left(\frac{\pi k_3}{N}\right)\\ &- \left[X_{c^3}(k_1,k_2,k_3,t) + X_{c^2}[k_1,k_2,t_\Delta]\frac{2}{N}C(k_3)\cos(\frac{\pi k_3}{2N})\right]\sin\left(\frac{\pi k_3}{N}\right), \end{aligned}$$

where t_Δ in $X_{c^2}[k_1, k_2, t_\Delta]$ implies that the input data (Here we denote as $\Delta(t+N,t)$) of the 2-D DCT are based on difference of two frames and $\Delta(t+N,t)$ is given by

$$\Delta(t+N,t) = (-1)^{k_3} x(n_1, n_2, t+N) - x(n_1, n_2, t), \tag{4.64}$$

which is a 2-D data frame obtained from the difference between the $(t+N)$th and the tth input data frames as shown in Fig. 4.11. This is the first part of the time-recursive computation. The 2-D DCT $X_{c^2}(k_1, k_2, \Delta(t+N,t))$ can be rewritten as

$$\begin{aligned} & X_{c^2}(k_1, k_2, \Delta(t+N,t)) \qquad (4.65) \\ & = (-1)^k C(k_1)C(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2, t+N) \cos\left[\frac{\pi(2n_1+1)k_1}{2N}\right] \\ & \cdot \cos\left[\frac{\pi(2n_2+1)k_2}{2N}\right] \\ & -C(k_1)C(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1, n_2, t) \cos\left[\frac{\pi(2n_1+1)k_1}{2N}\right] \qquad (4.66) \\ & \cdot \cos\left[\frac{\pi(2n_2+1)k_2}{2N}\right]. \end{aligned}$$

The above equation suggests that the 2-D DCT of each frame can be computed first and store it in a shift register array of size $(N+1) \times N^2$. The difference between the 2-D DCT of the tth frame and $(t+N)$th frame is then computed. Equations (4.62) and (4.64) indicate that the 3-D DCT can be generated by feeding the 2-D DCT of the updating vector into a lattice module as shown in Fig. 4.12. The size of the shift register in the lattice module is N^2 because for a specific k_3 there are N^2 values ($k_1 = 0, .., N-1; k_2 = 0, .., N-1$) to be updated. A similar updating relation exists for the 2-D DCT and the 1-D DCT [65]. The number of shift registers in the lattice module for 2-D and 1-D DCT are N and 1 respectively. In fact, any d-D DCT can be obtained from the 1-

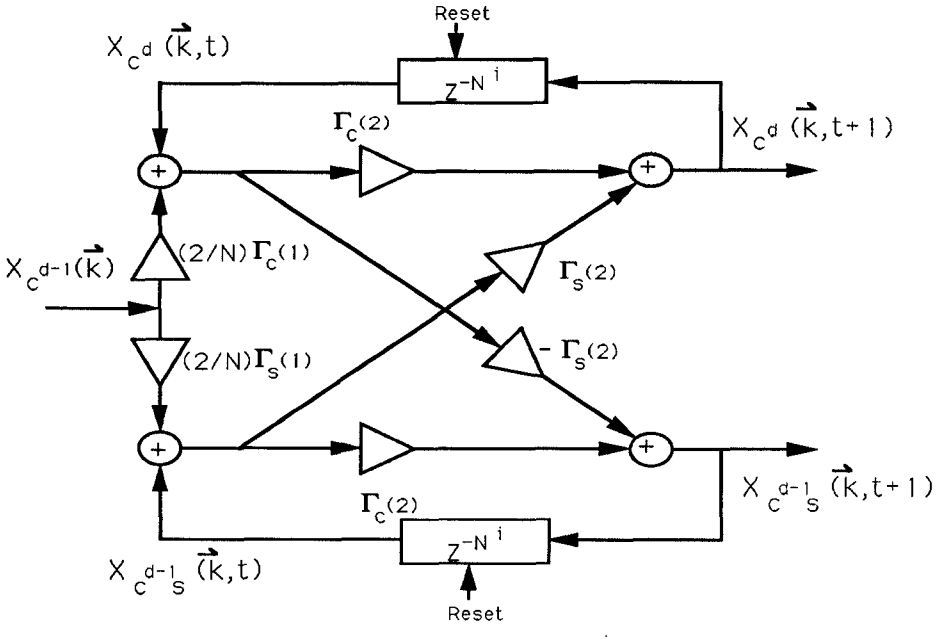


Figure 4.12: The lattice module.

D DCT by repeated application of equations (4.62) and (4.64). Therefore, the time-recursive 3-D DCT lattice structure consists of three lattice arrays which are used to produce the 1-D, 2-D and 3-D DCT individually. The 3-D DCT can be implemented using either the lattice or the filter structures as described below.

Lattice 3-D DCT architecture

The architecture of the frame-recursive lattice 3-D DCT is depicted in Fig. 4.13. It consists of three Lattice Array Blocks (LAB0, LAB1, and LAB2) whose configurations are depicted in Fig. 4.14. The lattice array LABi consists of a shift register array, two adders, and a lattice array; the shift register array is of size $(N+1) \times N^i$ and is used to store the intermediate values. The function of the adders is to update the effect of the new data and eliminate the effect of the

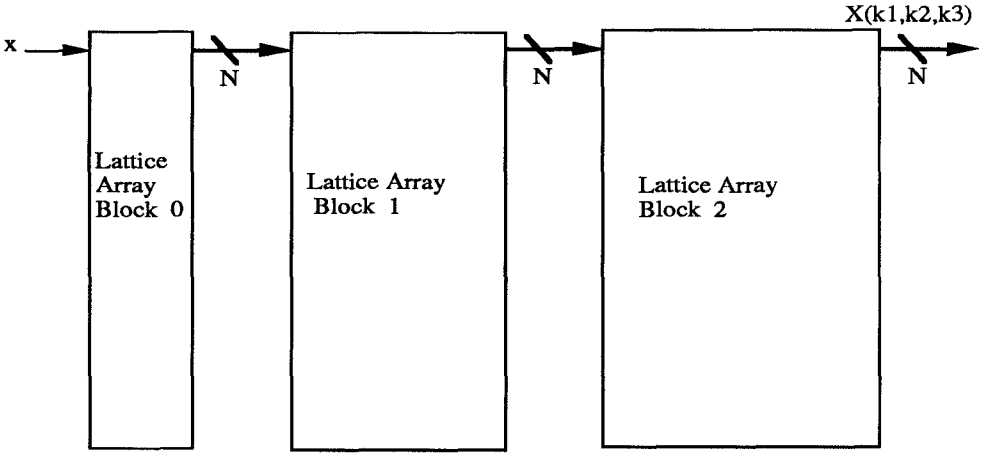


Figure 4.13: The architecture for the frame-recursive 3-D DCT.

previous data. The structure of the lattice array is shown in Fig. 4.12. The difference between different lattice arrays is only in the number of delays in the feedback loop. There are N^i delay elements in the ith lattice array.

The operation of this architecture can be viewed as follows. Input data is scanned row by row and frame by frame and sent to the first module LAB0 which generates the 1-D DCT of each row on every input frame. When the last datum of each row is available, the 1-D DCT of each input row vector is obtained. These N 1-D DCT transformed data are loaded in parallel into the second module LAB1 every N clock cycles. The LAB1 module is used to generate the 2-D DCT of each data frame. After N^2 clock cycles, when the last datum of the each frame arrives, the 2-D DCT of each frame is available. These values are loaded in parallel into the LAB2 module to generate the 3-D DCT recursively. The difference between the 2-D DCT of the parity of the $(t + N)$th and tth

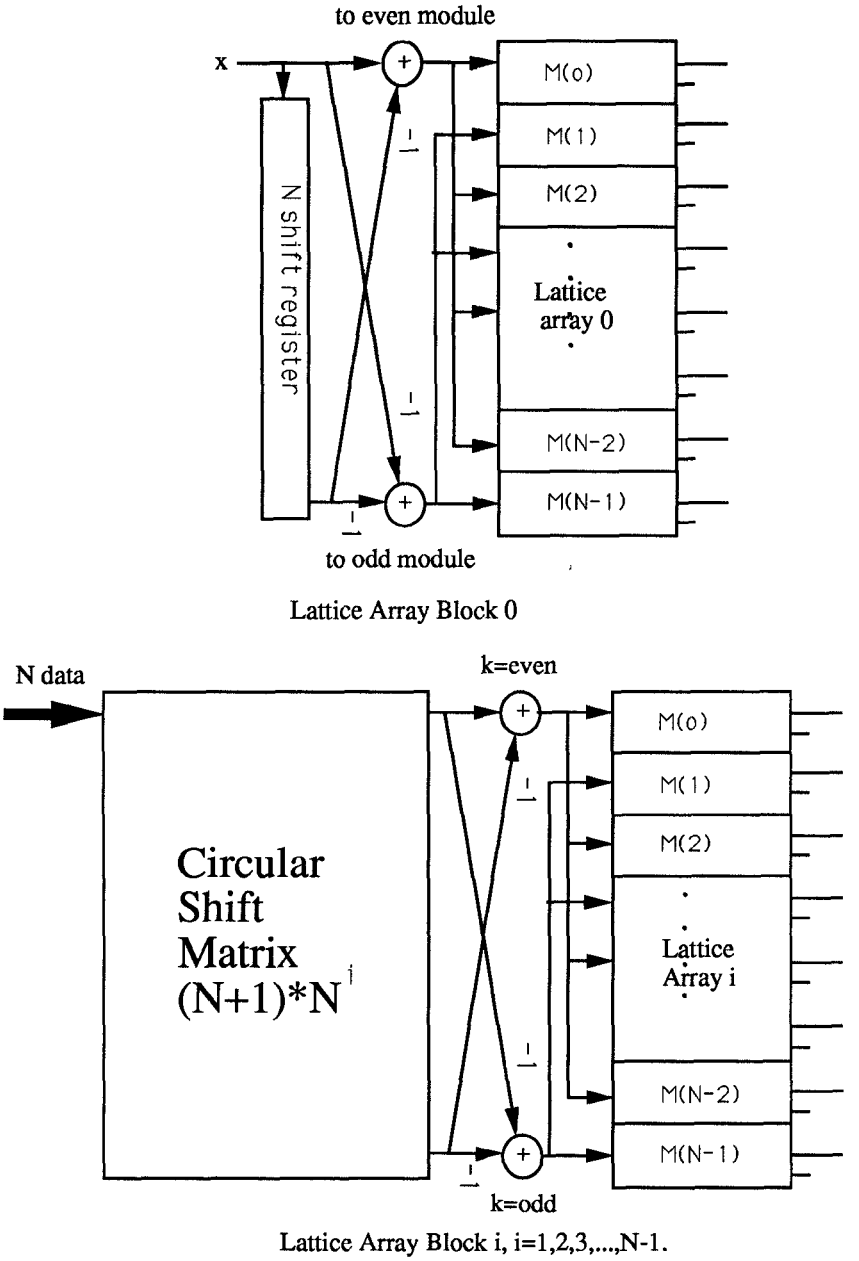


Figure 4.14: The structure for Lattice Array Blocks.

frame is used as the input to the LA2 module. There are N^2 shift registers in the feedback loop of $LA2$ to store the transformed data of each frame. It takes N^2 cycles to finish updating a new 3-D block and this is the period required to obtain a new 2-D DCT data block. It is easy to verify that the system is fully-pipelined.

In applications where only block multi-dimensional transforms are required, the above architecture can be simplified. Intermediate values stored in the shift registers are not necessary. The purpose of the shift registers required is to store the current data obtained from filter arrays, hence its size is reduced to N^i for Lattice Array Block i. Since the updating is unnecessary, the two adders can be eliminated. The lattice block 3-D DCT structure is shown in Fig. 4.15.

IIR 3-D DCT architecture

We have seen in Section 4.3 that the lattice structure can be realized as directly as a digital filter by considering the transfer function of each lattice module. This approach is used to convert the time-recursive lattice 3-D DCT structure into its direct form configuration. The only difference between lattice and IIR 3-D DCT architecture is that the lattice array i is replaced by direct form filter array i. The direct form of the lattice module in Fig. 4.12 is depicted in Fig. 4.16. The size of the shift register in direct form realization is the same as that of lattice modules. The configuration of the direct form filter 3-D block DCT is depicted in Fig. 4.16.

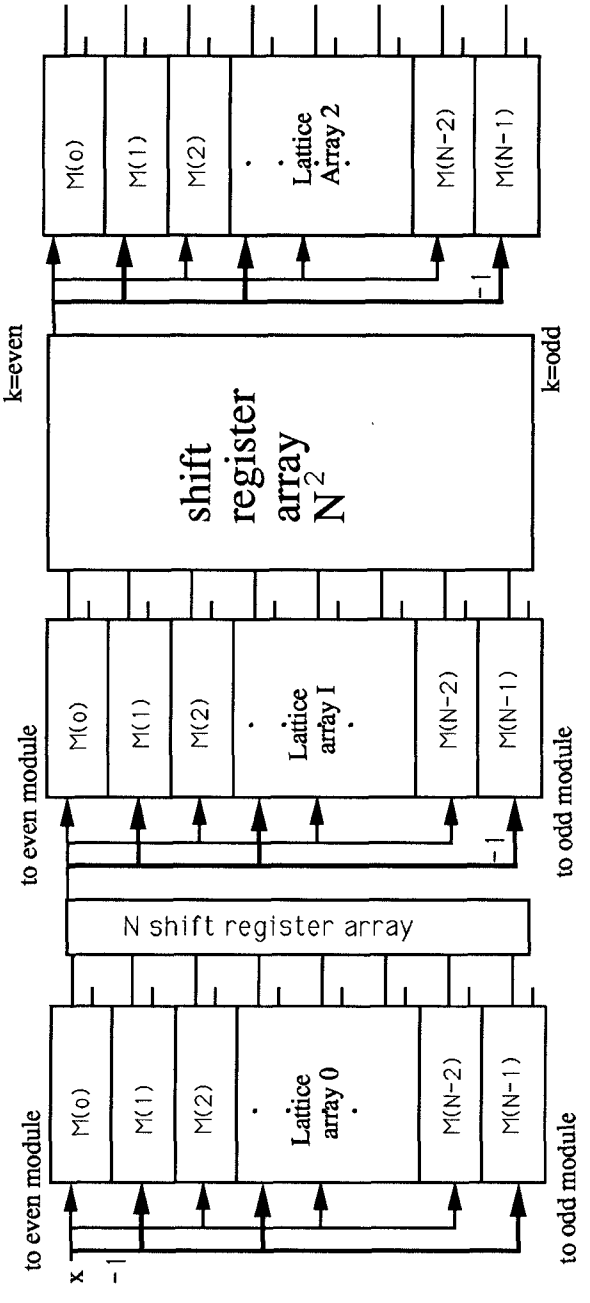


Figure 4.15: The architecture for block 3-D DCT.

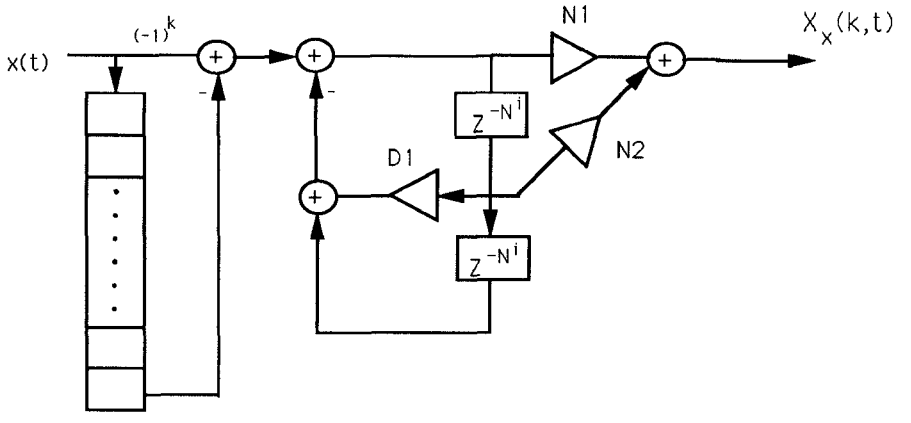


Figure 4.16: The configuration of the direct form filter 3-D block DCT.

4.6.2 Time-Recursive Structures for Multi-Dimensional DXT

In this section, we generalize the time-recursive concept to any multi-dimensional DXT and derive the fully-pipelined block structures. Denote by $[x(\vec{n_d}, t)]$ the input data file at time t, and by $[x(\vec{n_d}, t+1)]$ the data file at time $(t+1)$ which is obtained by shifting $[x(\vec{n_d}, t)]$ in a direction of one of the axes of $\vec{n_d}$ by one unit. For simplicity, let us assume that the data file is shifted in the direction of the last axis, n_d. The d-dimensional DXT of the input data file $[x(\vec{n_d}, t)]$ is defined as

$$X_{X^d}(\vec{k_d}, t) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \cdots \sum_{n_d=t}^{t+N-1} x(\vec{n_d}, t) P_{\vec{n_d}}(\vec{k_d}), \tag{4.67}$$

Here, we assume that the transform kernel $P_{\vec{n_d}}(\vec{k_d})$ is separable[2]. That is

$$P_{\vec{n_d}}(\vec{k_d}) = P_{n_1}(k_1)P_{n_2}(k_2)\cdots P_{n_d}(k_d). \tag{4.68}$$

From the analysis in Section 4.6.1, we see that the updated transform $X_{X^d}(\vec{k_d}, t+1)$ is related to the previous transform $X_{X^d}(\vec{k_d}, t)$ by the following equation [65]:

$$X_{X^x}(\vec{k_d}, t+1) = \left\{X_{X^d}(\vec{k_d}, t) + X_{x^{d-1}}[\vec{k}_{d-1}, \Delta(t+N, t)]D_x(k)\right\}\Gamma_x(k), \tag{4.69}$$

where $\Delta(t+N, t)$ is the difference between the data files at time t and $(t+N)$, and $D_x(k)$ and $\Gamma_x(k)$ are coefficients that depend only on the transform kernel and index k. The above equation indicates that the d-dimensional DXT can be updated recursively using the previous transformed data $X_{X^d}(\vec{k_d}, t)$ and the $(d-1)$-D DXT of $\Delta(t+N, t)$. This relation can be used recursively such that any d-D DXT can be generated from the 1-D DXT using d filter blocks as shown in Fig. 4.17.

As described in the previous section, there are two kinds of time-recursive DXT architectures, the moving-frame d-D DXT and the block d-D DXT. The structure of the basic building block in the moving-frame DXT is shown in Fig. 4.18, where the filter array can be either the lattice or the filter form. The function of each block is to shift the $(d-1)$-dimensional data into a data bank, then distribute the difference of the first and last frame of the data bank to the second stage DXT array. The dimension of the shift register array is $(N+1)\times N^i$ and the delay in filter array i is N^i. The time required to obtain the $(d-1)$-dimensional DXT is N^{d-1}, which is also the time required to obtain the N^d elements of the transformed data.

[2]This is true for all the discrete sinusoidal transforms considered here.

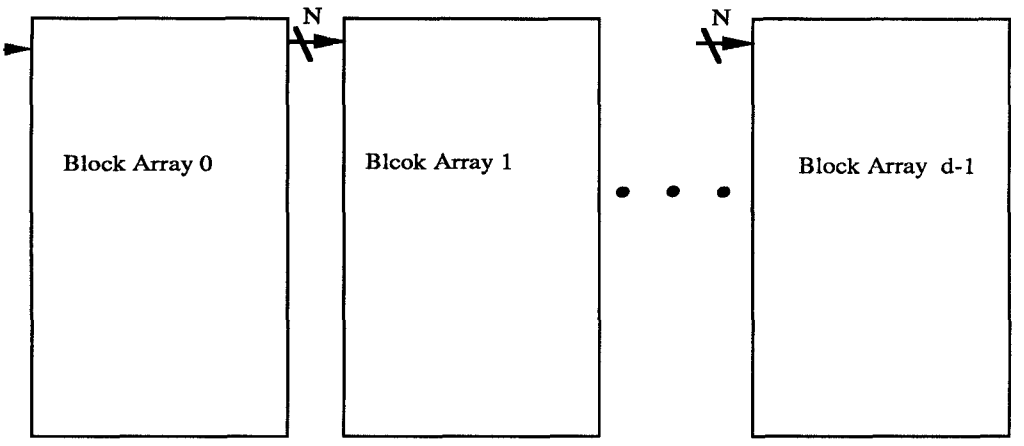


Figure 4.17: The block diagram of the d-D DXT.

In the case of block DXT, the size of the shift register array can be reduced and adders can be eliminated because intermediate transformed data do not have to be stored. The size of the shift register array is N^i. The structure of the LAB is shown in Fig. 4.19. The lattice array i is reset every N^{i+1} cycles.

Area-Time Complexity Analysis

Our architecture for computing the d-dimensional transform DXT over N^d points consists of d blocks, each block is composed of a shift register array followed by a one-dimensional lattice or IIR structure made up of N DXT modules. The ith shift register array is of size $(N+1) \times N^i b$, where $0 \leq i \leq d-1$ and b is the number of bits used to represent each number. The output is generated in a shift register array of size $N^d b$. Therefore the total number of multipliers and adders used is $O(dN) = O(N)$, and the total amount of memory is $O(N^d b)$. The

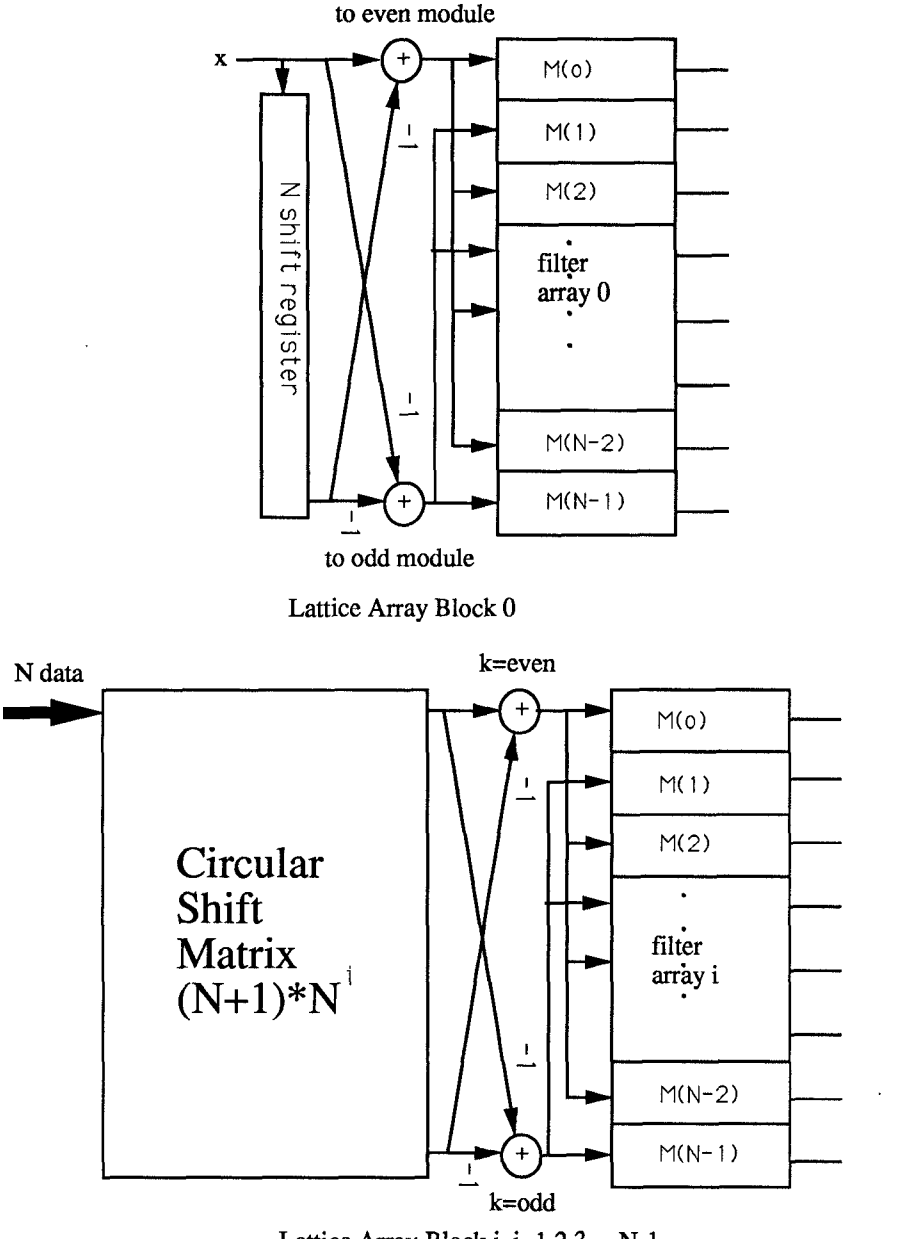
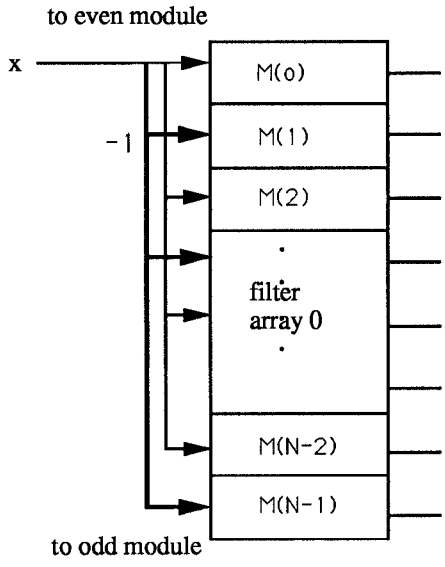
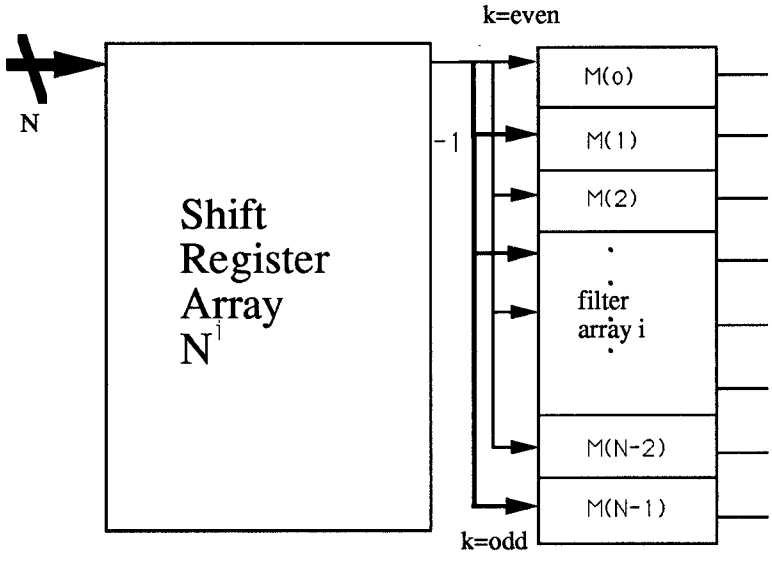


Figure 4.18: The basic building structure of the moving-frame DXT.



Filter Array Block 0



Filter Array BLock i, i=1,2,3,...,N-1.

Figure 4.19: The basic structure of the block DXT.

next lemma states that the area of any chip that computes the d-dimensional DFT transform must be $\Omega(N^d b)$, and hence our design asymptotically optimal in its use of area. The same holds true for the remaining transforms. We are using the standard VLSI model as introduced by Thompson[75].

Lemma 6 *Any VLSI system that computes the d-dimensional DFT on N^d points requires area $A = \Omega(N^d b)$, where b is the number of bits required to represent each input number.*

The proof of the lemma can be derived from a result in [74] in a straightforward way. Hence our design uses the least amount of memory asymptotically. The speed of our VLSI design cannot be improved asymptotically since it processes the input in real time. Hence our design is asymptotically optimal in both speed and area.

4.7 Summary

In this Chapter, we proposed optimal time-recursive unified architectures for computing the DCT, DST, DHT, DFT, LOT, and CLT using only half as many multipliers as the unified lattice structure described in [17]. In the lattice structure, two transforms are dually generated simultaneously, while this optimal architecture has the flexibility of generating either one transform or both together. The basic configuration of the optimal unified architectures has a second order autoregressive model. It is optimal in the sense that the number of the multipliers used is minimum and both speed and area are asymptotically optimal. We also gave a theoretical justification of the unified time-recursive architecture using the Fundamental Recurrence Formula. We show that to generate the DCT and

DST, only $2N-2$ multipliers are necessary, while in the case of dual generation of the DCT and DST, only $1.5N$ multipliers are required for each transform on average. Finally, we generalized the time-recursive concept to multi-dimensional transforms. The resulting architecture is fully-pipelined, modular, and regular. It requires only d 1-D arrays for computing a d-D DXT.

Chapter 5

VLSI Implementation of DCT/DST Lattice Structures

The alternate use [22] of the discrete cosine transform (DCT) and the discrete sine transform (DST) can achieve a higher data compression rate and less block effect in image processing. We will focus on the parallel lattice structure that can dually generate the 1-D DCT and DST proposed in Chapter 2. This architecture is especially suited for VLSI implementation because they are modular, regular, and have only local interconnections. The VLSI implementation of the lattice module using the distributed arithmetic approach is described. The chip was designed for real-time processing of 14.5-MHz sampled video data. Fabricated by using $2\mu m$ double-metal CMOS technology, the lattice module contains approximately 18,000 transistors, which occupy a $5.6 \times 3.4 mm^2$ area. The 40-pad die size is $6.8 \times 4.6 mm^2$. It has been tested to be fully functional.

5.1 Introduction

The DCT and DST are two of the most efficient transforms for video and image coding applications. The DCT approaches the optimal performance of the KLT transform for highly correlated signals, while the DST approaches the optimal performance of the KLT for signals with low correlation coefficients. If we are able to compute the DCT and DST simultaneously, this structure will be very useful, especially when we do not know the statistics of the incoming signal. Rose *et al.* showed in [22] that alternate use of the DCT and DST can achieve the removal of redundancies in the correlation between neighboring blocks, as well as the preservation of continuity across the block boundaries.

The most important block in the parallel 1-D lattice structures is the lattice module. The VLSI implementation of the lattice DCT/DST module is described in this Chapter. In order to achieve an efficient VLSI realization, we adopt the distributed-arithmetic method to implement the multipliers in the structure. The advantages of using this memory-oriented realization are the savings in chip area, better accuracy and higher speed [25, 90]. This chip has been fabricated using $2\mu m$ CMOS technology, and will be capable of processing 116 Mb/s data in real-time.

The rest of this Chapter is organized as follows. In Section 2, the algorithm and structure to generate the parallel 1-D DCT and DST are described. In Section 3, the VLSI implementation of the lattice module by employing the distributed-arithmetic method is discussed. The detailed architectures of the building blocks in the DCT/DST lattice module are described in Section 4. The VLSI realization of the chip and simulation results are given in Section 5. The finite-precision analysis is given in Section 6. Finally, the conclusion is given in

Section 7.

5.2 Distributed-Arithmetic based Implementation

Since the lattice module is the most important component in the DCT/DST lattice structures, we focus on the VLSI implementation of the lattice module. It has been shown in [63, 64, 86, 85, 82] that the DCT and other sinusoidal transforms can be implemented in the form of filter banks. Every lattice module in the DCT/DST structure is a modified normal form digital filter [28], which has the least roundoff noise and is less sensitive to coefficient inaccuracy. Due to the fact that the block 2-D DCT operation will reset all the outputs of the LAI and $LAII$ every N and N^2 clock cycles respectively, the roundoff errors will be further minimized [65]. In the following we will focus our discussion on the 8 point DCT with 8-bit input signals and 12-bit output signals using two's complement binary number system.

Suppose we want to construct the lattice module that is based on the Liu-Chiu2 module with 4 multipliers [17]. Then the total number of multipliers needed for the 8×8 2-D DCT is 64, which is quite excessive. In addition, the system throughput will be limited by the operational speed of the multipliers.

Several DCT processors has been proposed [88, 81, 89, 25]. Sun *et al.* [25, 28] proposed the first working 16×16 DCT chip which incorporates the distributed-arithmetic method. Using this memory-oriented structure, a high speed, high accuracy efficient hardware implementation of the 2-D DCT can be achieved. We adopt the distributed-arithmetic scheme in our VLSI implementations. The

multiplication results stored in the ROM are computed using double precision numbers on a SUN workstation. The lattice module can be redrawn as shown in Fig. 5.1. The dashed box in Fig. 5.1 can be implemented by using a single ROM with three inputs and four outputs. Using this realization, the roundoff errors due to multiplication are minimized since distributed-arithmetic transforms explicit multiplication into implicit multiplication. Therefore, the errors of the system are all due to the quantization errors resulting from finite precision implementation and addition operations. Under the 12-bit two's complement realization, the RMS error values are approximately $40d$B [25], which is satisfactory for most applications. Assuming that every input of the ROM is 2-bit long, the lattice module can be implemented using 6 ROMs and 22 adders as shown in Fig. 5.1. The ROM size for each lattice array is 18432 bits. By reducing the number of bits of every input of the ROM to one, the ROM size reduces to 4608 bits. This is one-fourth of the previous case, but the the number of adders needed is doubled.

Another way to implement the lattice module using ROMs is shown in Fig. 5.2. Each dashed box is realized using a ROM with one input and two outputs. Fig. 5.2 illustrates the realization of each ROM when the number of bits of the input signal is 6 bits. Using this method, the ROM size of each lattice array is 13824 bits and the number of adders needed is 10. When the number of bits of the input signal is reduced, the ROM size is reduced but the number of adders is increased. We implement our system based on the schematic diagram for each lattice module as shown in Fig. 5.2. The adder is a 12-bit carry lookahead full adder/subtracter which is constructed using three 4-bit carry-look-ahead adders. Since, ROMs need less area than general purpose multipliers

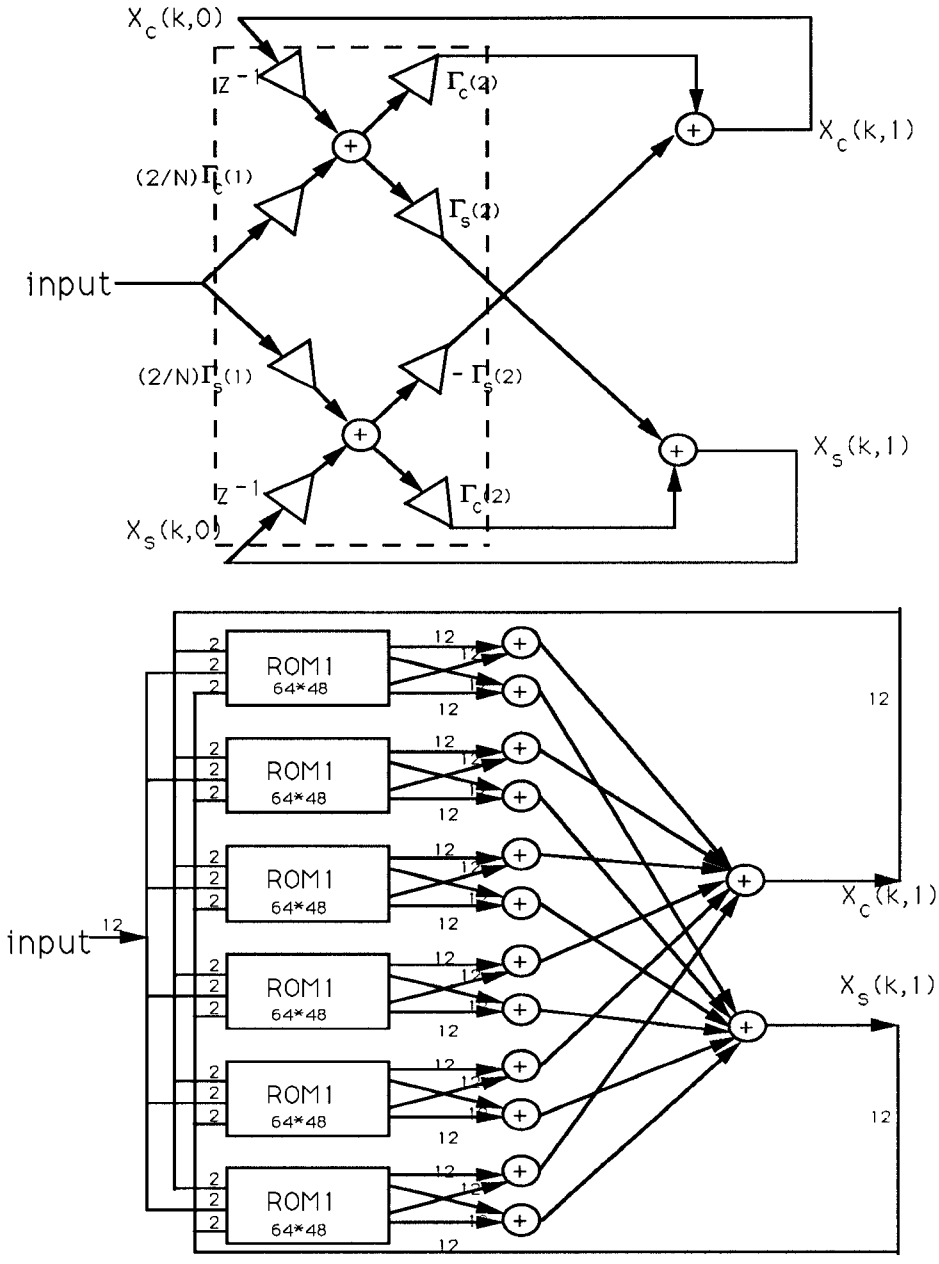


Figure 5.1: The realization of the lattice module using one ROM.

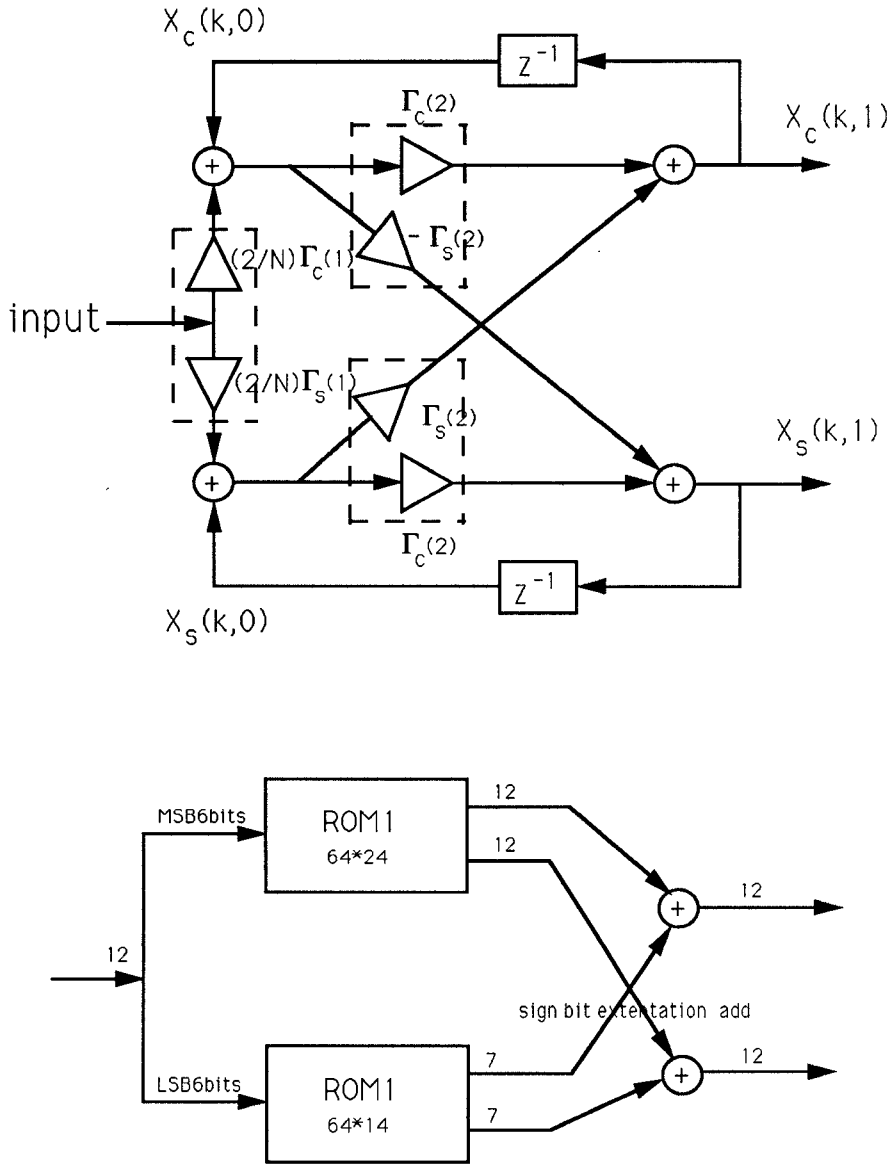


Figure 5.2: The realization of the lattice module using three ROMs.

and can achieve a higher speed, circuit implementations using this approach can be used for very high speed video signal processing.

5.3 Design of the Building Blocks

The main building blocks of the lattice structure are ROMs, adders, shift registers, and latches. These components will be described individually in this section. The critical path in our lattice modules is the feedback loop which includes one ROM and three adders. A traditional two-phase clocking scheme would use one phase to perform these computations and a second phase to latch the results. In order to make the two phases of the clock more symmetric, we perform computations on both phases of the clock. As shown in Fig. 5.3, we perform one addition and the ROM lookup during the first phase and two additions during the second phase. Intermediate results are stored in half-latches as described below.

5.3.1 ROM Implementation

As described in Section 4, the main component of the distributed-arithmetic based lattice structure is the ROM. Most existing ROMs are implemented based on the precharge concept, that is, the bit lines are precharged high during the precharge phase, and then the selected word lines discharge some of the bit lines according the coefficients stored during the evaluate phase. In order to reduce the ROM access time, we use a novel ROM design [66]. Fig. 5.4 shows the detail of each cell in the ROM. A simple inverter with a feedback transistor and a transmission gate controlled by phase ϕ_1 is used as a sense amplifier at the

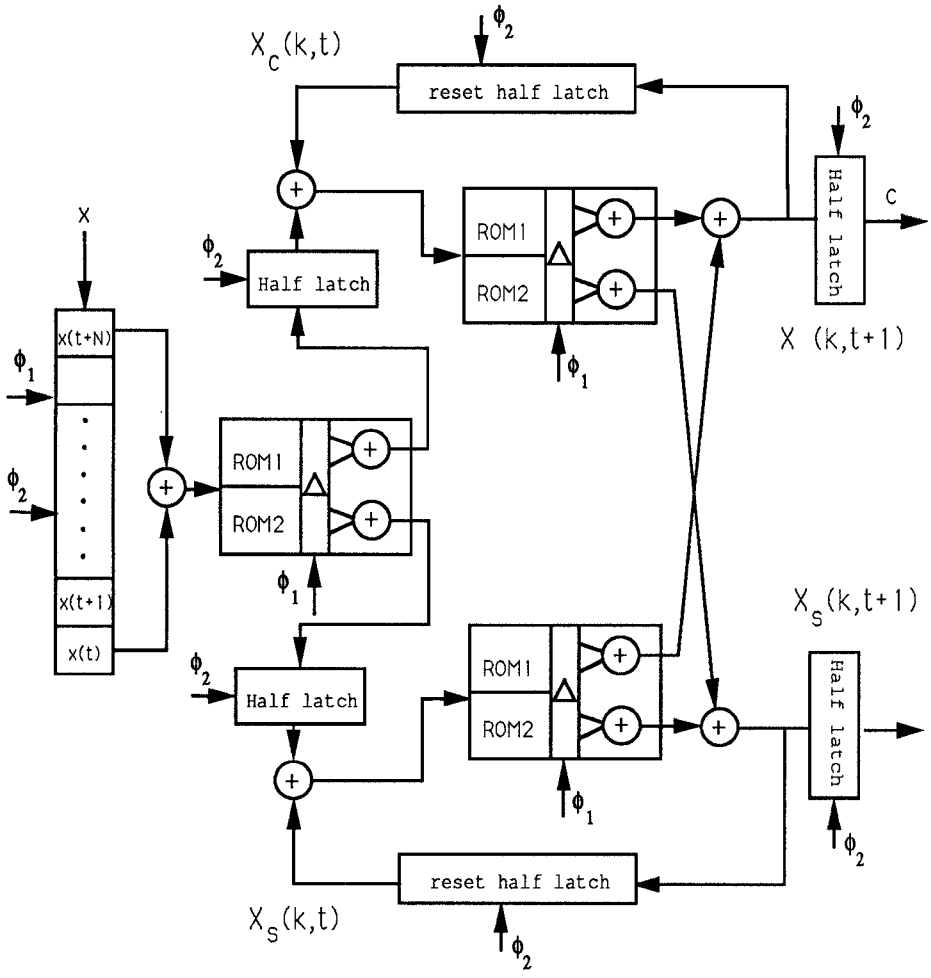


Figure 5.3: The bulding blocks of the lattice module with clocks.

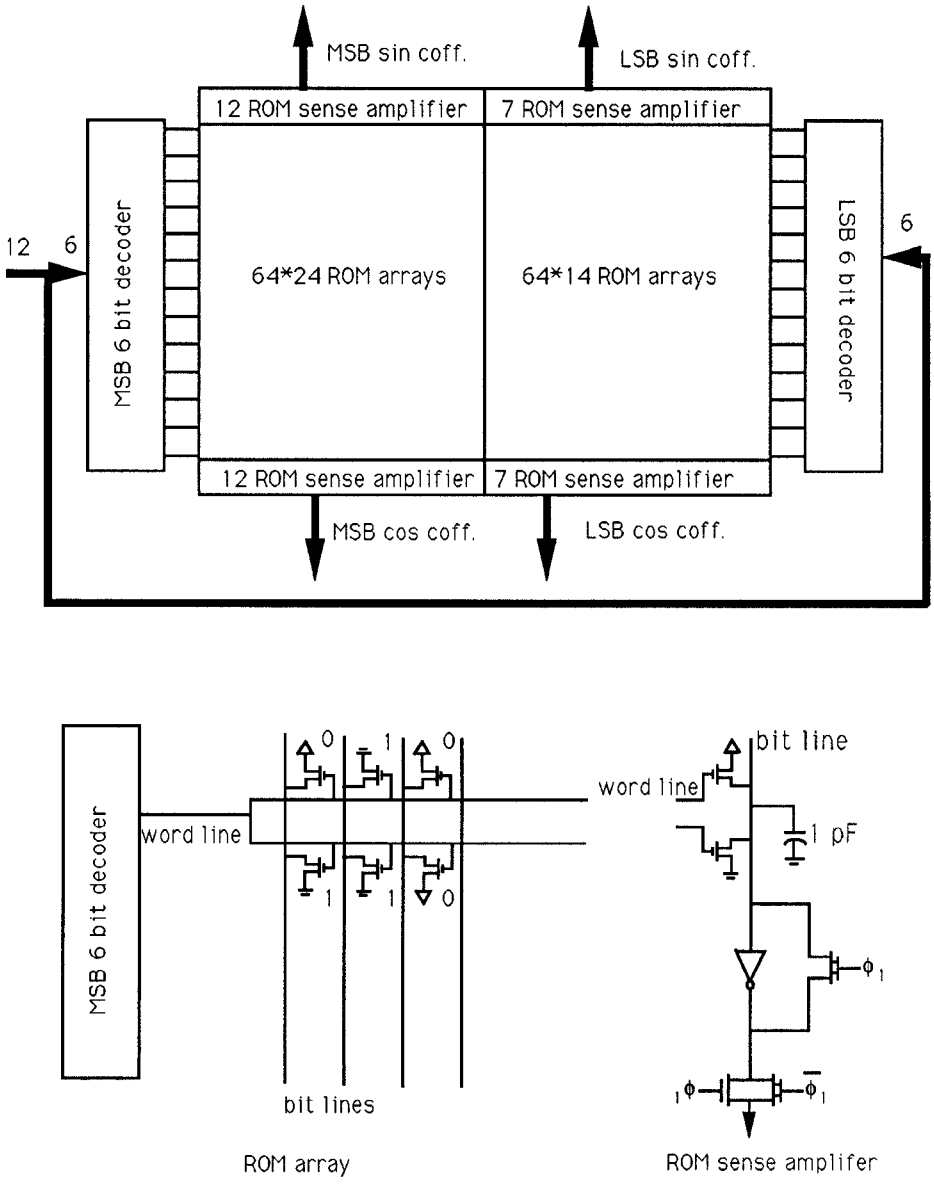


Figure 5.4: The logical diagram of ROM.

output of the bit-lines. We precharge the bit lines to an intermediate voltage between GND and Vdd, and use n-channel transistors to either charge the bit line from this intermediate voltage to Vdd-Vt or discharge it to GND, during the evaluate phase. In this scheme, the array is fully populated; . *i.e.* the number of n-channel transistors in the array is MN, where M is the number of word lines and N is the number of bit lines. A 'zero' is stored at a particular location, by connecting the n-channel transistor at that location to Vdd; a 'one' is stored by connecting the transistor to GND. The layout of the storage cells in the ROM array is shown in Fig. 5.5. The cell size is only $13\lambda \times 16\lambda$.

In our distributed-arithmetic scheme, the multiplication of the 12-bit input number with a 12-bit sine or cosine coefficient is performed by two ROMs each with 6-bit inputs and two adders. This reduces the chip area needed to implement multiplication with fixed coefficients. The ROM includes two 6-bit decoders and two small ROMs as shown in Fig. 5.3. The 12-bit input is divided into two parts; the most significant 6-bits of the input are used to generate the coefficients for small ROM1 and the least significant 6-bits are used for small ROM2. The final result of the multiplication is obtained by adding the outputs of small ROM1 with a shifted version of the outputs of small ROM2. We only store the most significant 7-bit result of the multiplication at ROM2. The sizes of small ROM1 and ROM2 are 64 words by 24 bits and 64 words by 14 bits respectively.

In order to improve the ROM access time, each 6-bit decoder is implemented as a tree consisting of two 3-bit decoders and a linear array of 64 AND gates. The delay time for this 6-bit decoder is 8.55ns, while a straightforward implementation would have a delay of 20.40ns. The outputs of the 64 AND gates form the word lines of the ROM array. The logical layout of the 6-bit decoder is

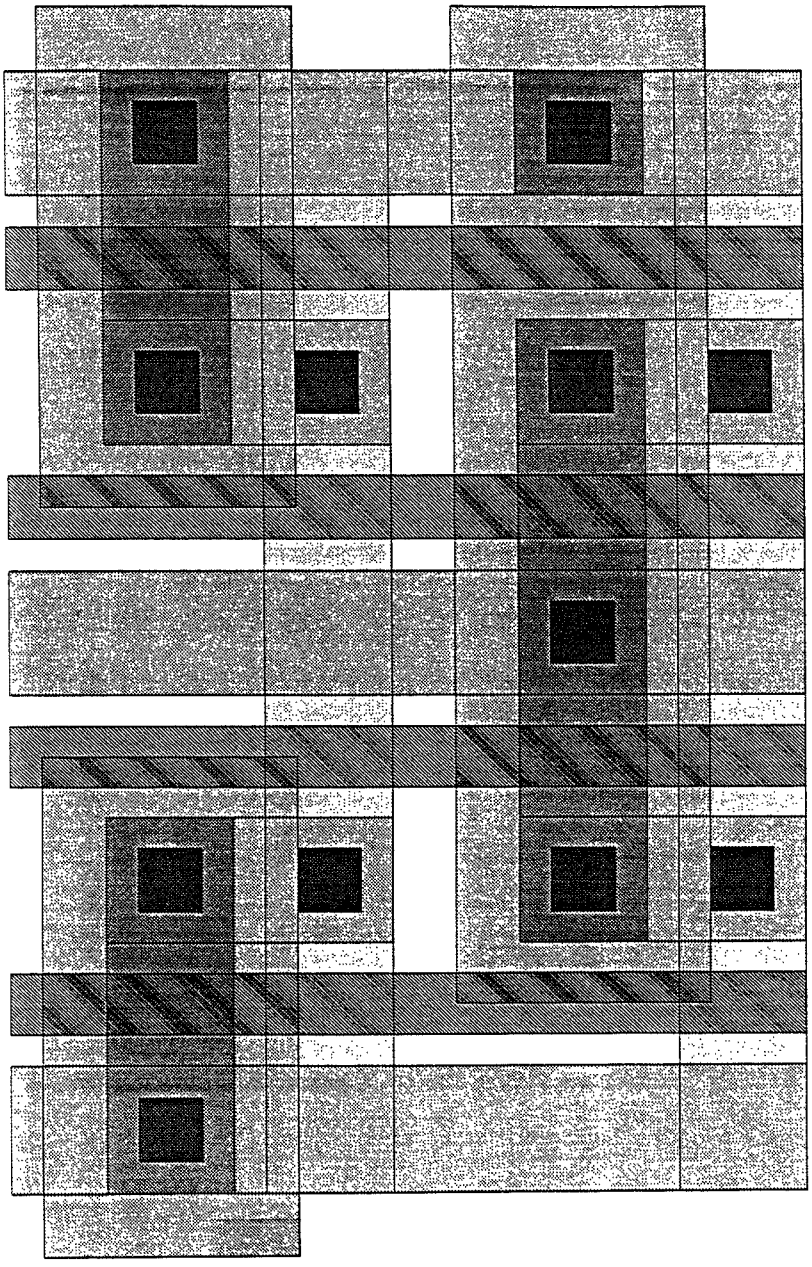


Figure 5.5: The layout of cells in the ROM array.

shown in Fig. 5.6. The physical size of the final ROM is $1292\lambda \times 1646\lambda$ which is much smaller than the area needed by a general purpose multiplier. The total ROM access time is 20ns.

5.3.2 Adders

Since the lattice modules are implemented based on a word-serial bit-parallel approach, high-speed bit-parallel adders are necessary. We build a 12-bit carry lookahead adder using three 4-bit carry lookahead adders [67]. Two large inverters are placed at the outputs of the adders to supply sufficient drive capability. The physical size of the 12-bit adder is $1022\lambda \times 256\lambda$ and the propagation delay through the adder is 18ns.

5.3.3 Shift Registers and Latches

There are two kinds of latches and one shift registers in the circuit. One of the latches is the half-latch which is controlled by phase ϕ_2 and is used to store the intermediate results obtained from the adders. The logical representation of the 12-bit half-latch is depicted in Fig. 5.7. The other latch is the reset controlled half-latch located in the feedback loop. Its logical circuit is shown in Fig. 5.7. When the reset signal goes low, the outputs from ROM2 and ROM3 are set to zero. The shift register located at the input stage of the system is a regular two phase shift register which delays the input sequence by eight clock cycles.

5.3.4 Control Unit

Only one control signal (reset) and two clock phases (ϕ_1 and ϕ_2) are required in this system. Phase ϕ_1 is used to latch the computational results from one adder

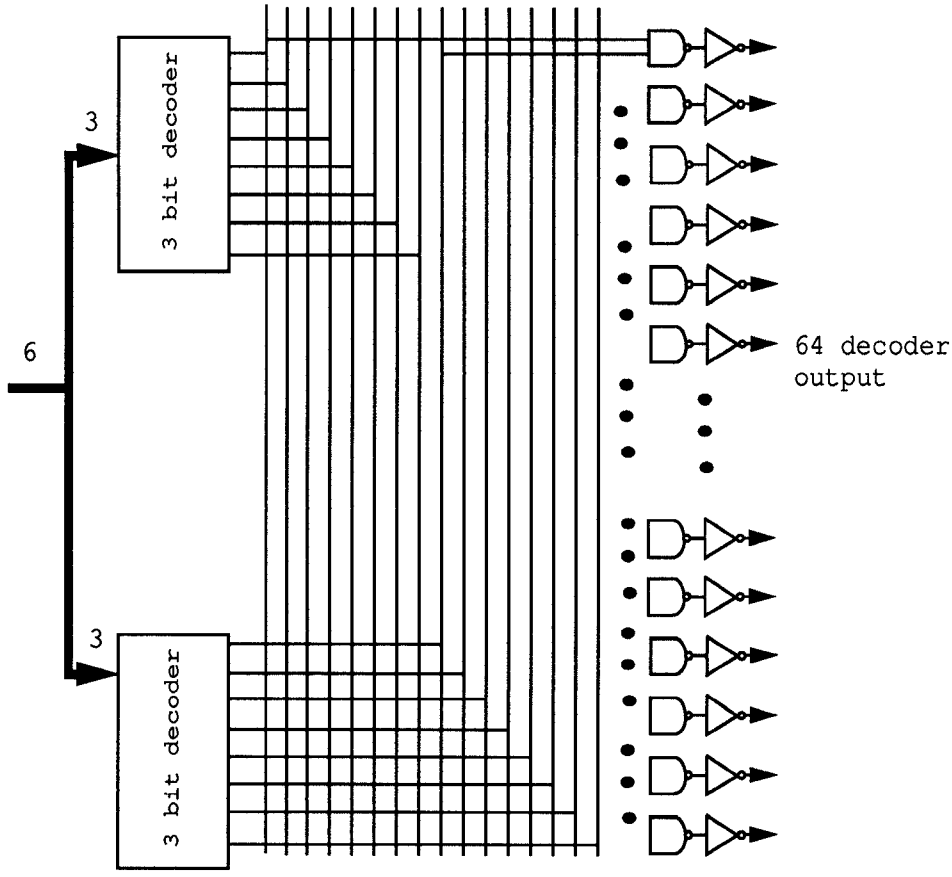


Figure 5.6: The logical diagram of the 6-bit decoder.

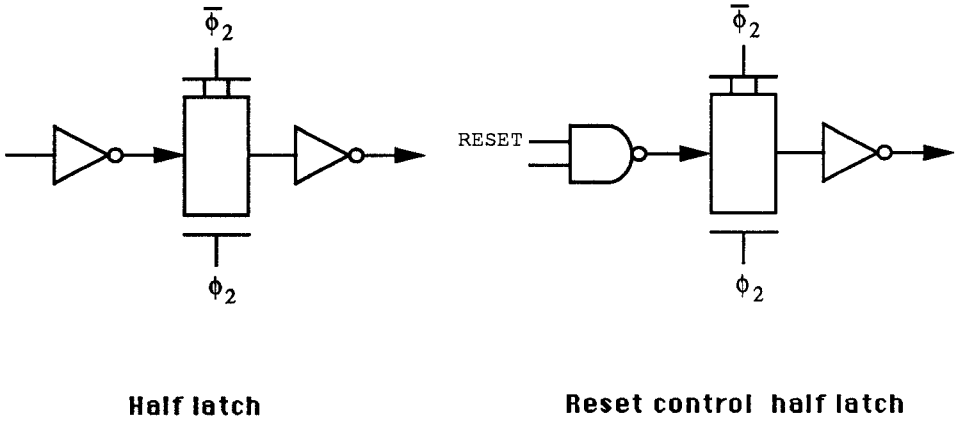


Figure 5.7: The logical diagram of half-latch and reset controlled half-latch.

and the ROM, while phase ϕ_2 is used to latch the results from the remaining adders. Since the propagation delay time for the ROM and the adder is approximately the same, we can make both clock phases symmetric to each other. The signal diagram of these two phases is depicted in Fig. 5.8. The reset signal is active low. One of the attractive features of this chip is the very simple control signals used. No additional logical control circuitry is needed in the design.

5.4 Chip Realization and Simulations

Having realized the symbolic layout of the individual blocks, the next issue is to integrate all these components efficiently. The floor plan of the lattice module is shown in Fig. 5.9. This includes three ROMs, eleven adders, four half-latches, two reset controlled half-latches and one shift register. ROM2 and

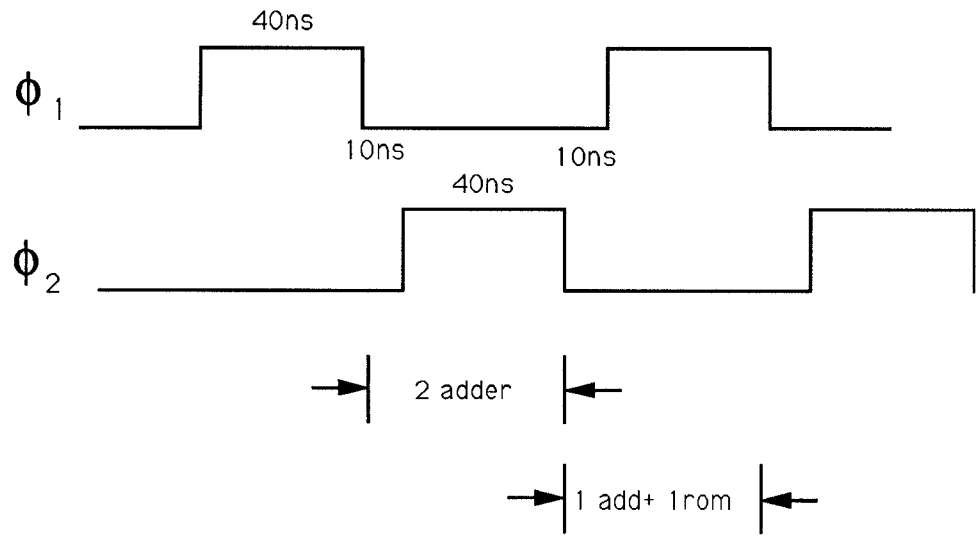


Figure 5.8: The signal diagram of the clock signals.

ROM3 are rotated by 90 and 270 degree respectively to simplify inter-component routing. This chip accepts 8-bit input signals and produces 12-bit DCT and DST coefficients every 100ns. The physical layout of the lattice module chip is depicted in Fig. 5.10. There are 18000 transistors in the chip, most of which are used in the three fully-populated ROMs. The total size of the active circuitry is $5400\lambda \times 3400\lambda$. This is fabricated in a die of size $6800\lambda \times 4600\lambda$ and packaged in a 40-pin chip.

Both logical and timing simulations were performed on the this chip. From the simulation results due to Sun *et. al* [25], the word length of the ROM must be at least 9 bits to ensure that the SNR is greater than 40dB. To reduce the error due to recursive computations, we increase the word length of the ROM to 12 bits. We used IRSIM to perform logic simulations on the layout of the chip. The results from IRSIM were compared with the DCT and DST of the same input

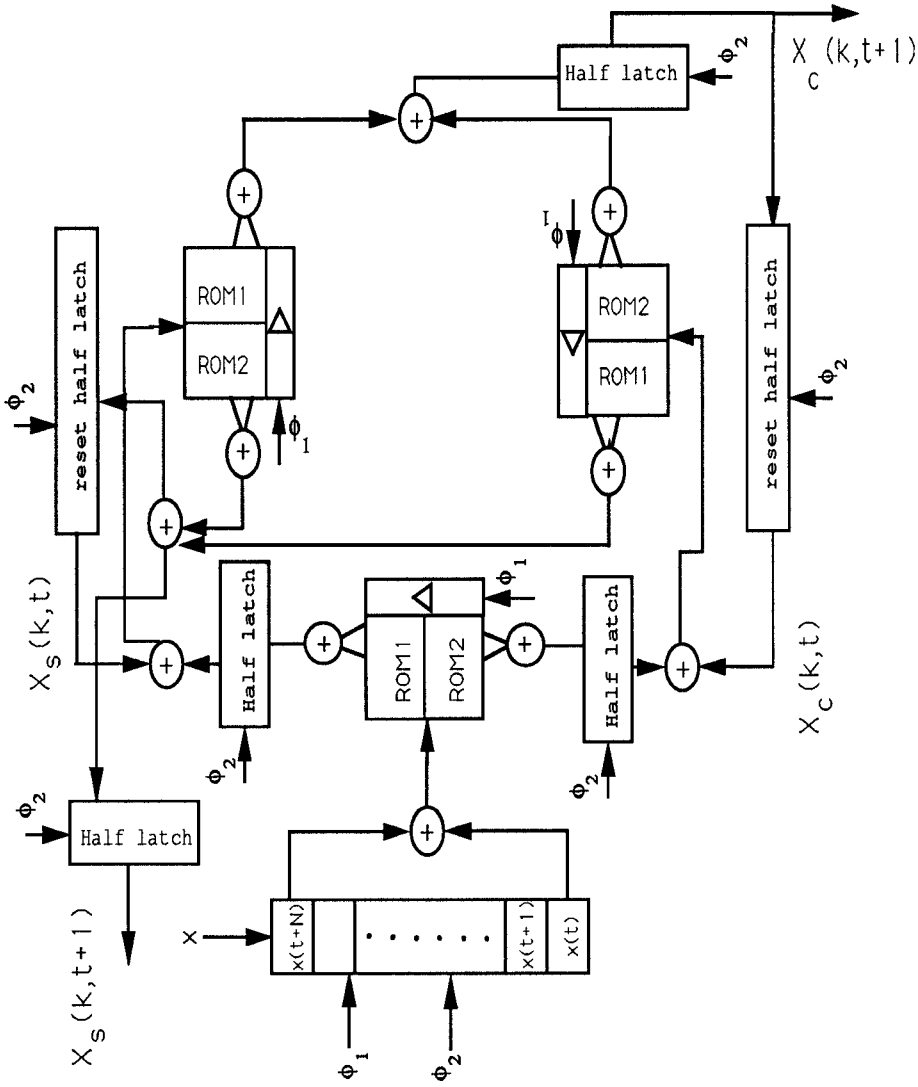


Figure 5.9: The floorplan diagram of the lattice module.

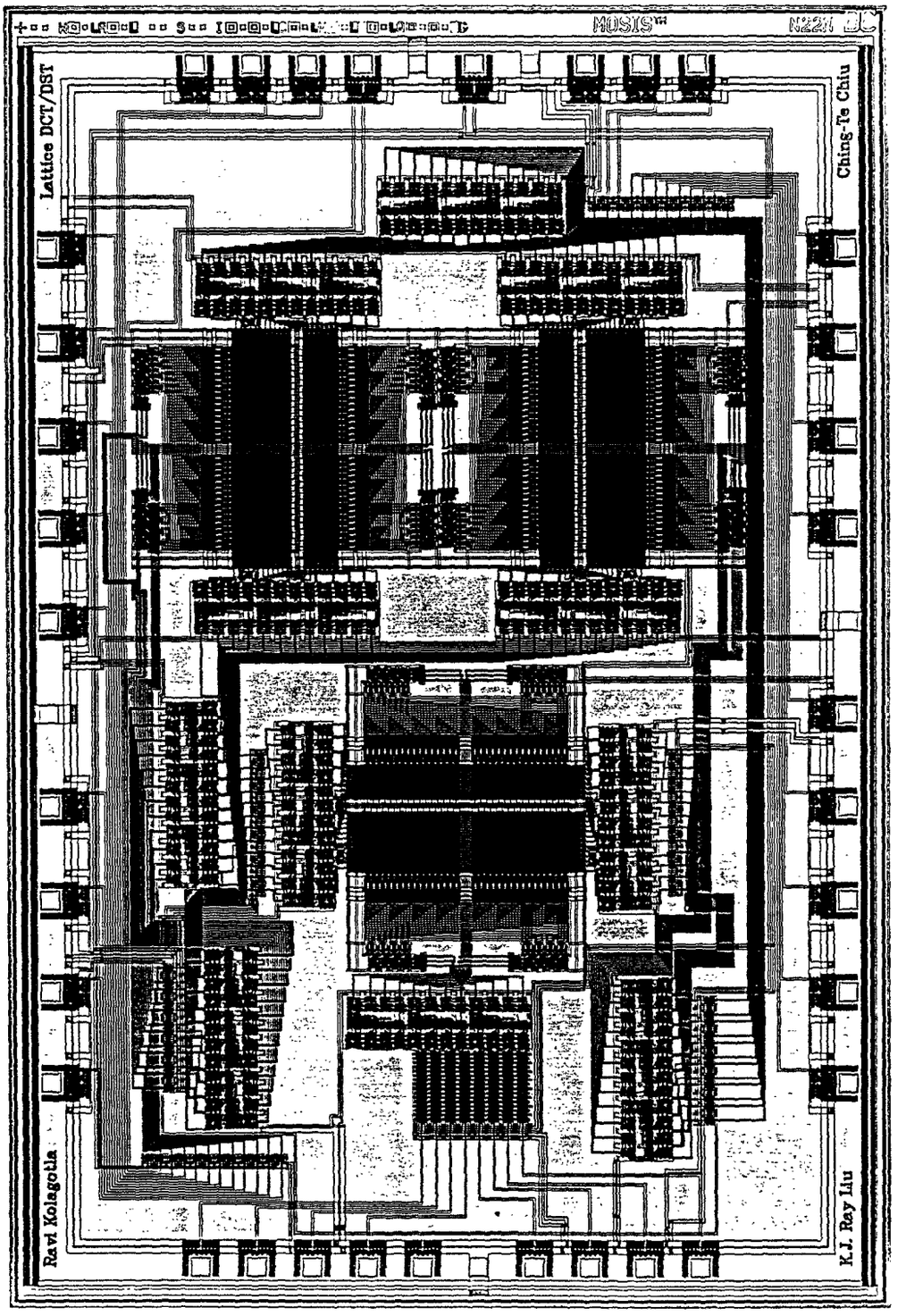


Figure 5.10: The physical layout diagram of the lattice module.

sequence obtained from a C program written on a SUN workstation. The results are accurate up to the least significant bit of the 12-bit representation. The SNR of this computation from simulations is about 41dB, which is satisfactory for image and video processing applications. SPICE simulations indicate that the worst case rise and fall time for the ROM bit-lines are 8ns and 9ns respectively. Fig. 5.11 shows the timing simulation for the entire ROM which includes the decoder, cell array, and sense amplifier. The sense amplifier, together with a feedback transistor is used to precharge the bit lines to an intermediate voltage between GND and Vdd. This decreases the voltage swing on the bit lines during the evaluate phase, and hence reduces the ROM access time. The upper part of Fig. 5.11 shows the case when the output is charged to high. The total delay time from input to output is 20ns. The lower part of Fig. 5.11 shows the case when the output discharges from this intermediate voltage to GND. It should be noted that although the bit line does not charge up to Vdd, the sense amplifier can still discharge the output to GND in a relatively short time. The delay time in this case is 15ns. The delay time for the three stage 12-bit carry lookahead adder is 20ns. The critical path in the structure is the feedback loop, which contains one ROM and three adders. Timing simulations indicate that the chip works at a frequency of 14.5MHz and hence input data at a rate of 116Mb/s can be processed in real-time. An example of the chip testing results using Integrated Measurement Systems V2000 machines is given in the following list. The exact DCT and DST results are compared with those obtained from the Irsim and chip testing. Compared with the DCT chip built by Sun *et. al* [25], our data throughput rate is about nine times faster than theirs.

Resource P1=Timing TXT

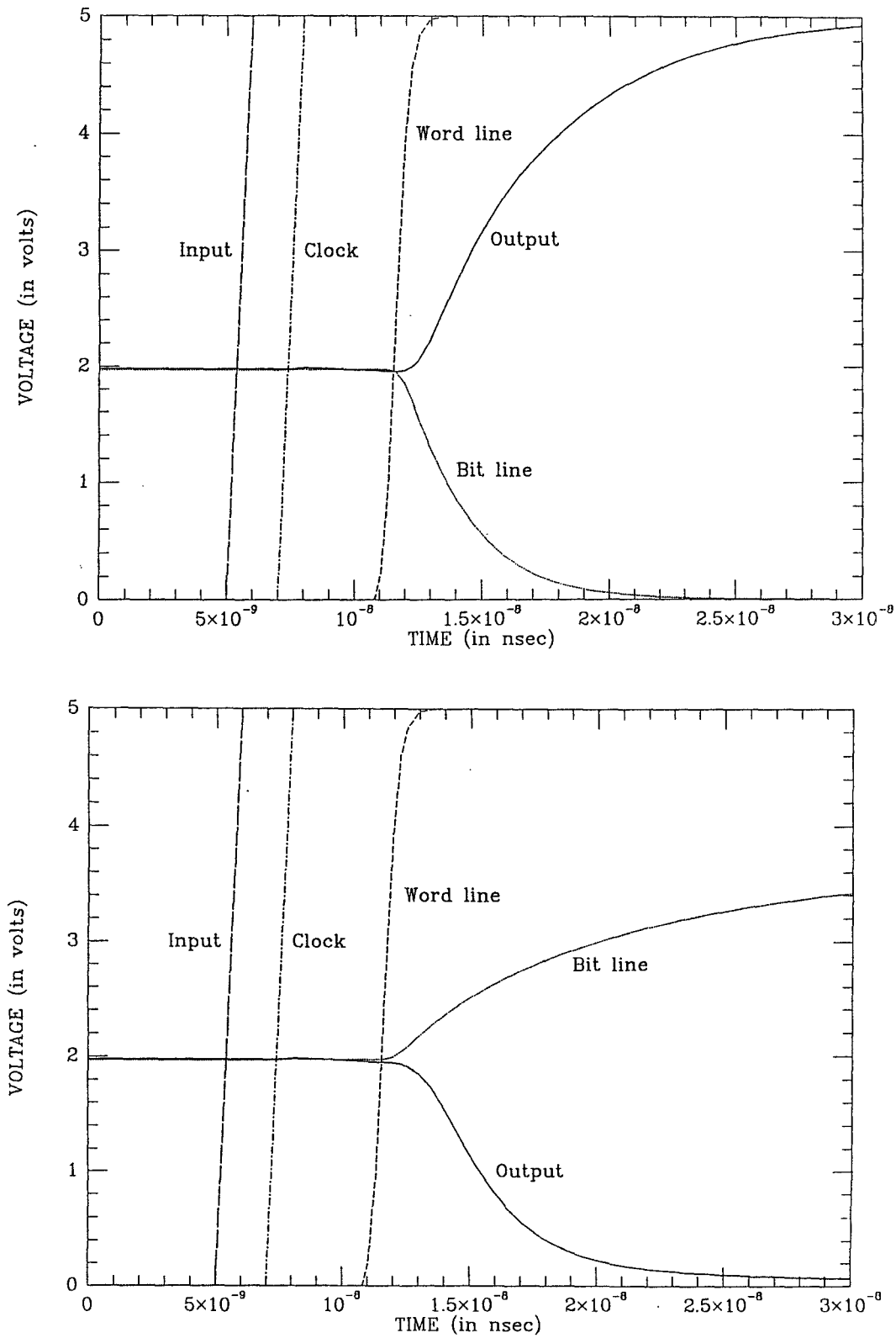


Figure 5.11: The timing simulation of the ROM using SPICE.

Technology	2-um double metal CMOS
Core size	$5.6 \times 3.4mm^2$
Die size	$6.8 \times 4.6mm^2$
Total no of transistors	18,000
Active signal pads	39
Speed	$14.5MHz$

Table 5.1: Summary of the DCT lattice module chip.

17 AT P1 12

Resource End

Radix P1=Bin

Polarity P1=Pos

Hidrive P1=5.00V

Lodrive P1=0V

Format P1=RZ,0s,30ns

Resource P2=Timing TXT

26 AT P2 10

Resource End

Radix P2=Bin

Polarity P2=Pos

Hidrive P2=5.00V

Lodrive P2=0V

Format P2=RZ,40.00ns,30ns

Resource DCT=Compare TXT

18 B3 Yc11 36

18 B2 Yc10 37

18 B1 Yc9 38

18 B0 Yc8 39

9 B7 Yc7 40

9 B6 Yc6 2

9 B5 Yc5 3

9 B4 Yc4 4

9 B3 Yc3 5

9 B2 Yc2 6

9 B1 Yc1 7

9 B0 Yc0 8

Resource End

Radix DCT=Bin

Polarity DCT=Pos

Threshold DCT=2.27V

Sample DCT=72.00ns

Resource DST=Compare TXT

15 B3 Ys11 13

15 B2 Ys10 14

15 B1 Ys9 15

15 B0 Ys8 16

3 B7 Ys7 17

3 B6 Ys6 18

3 B5 Ys5 19

3 B4 Ys4 20

3 B3 Ys3 22

3 B2 Ys2 23

3 B1 Ys1 24

3 B0 Ys0 25

Resource End

Radix DST=Bin

Polarity DST=Pos

Threshold DST=2.27V

Sample DST=72.00ns

Resource Vdd=Power TXT

25 V0 Vdd 11

Resource End

Power Vdd=5.00V,250mA,0s,HIZ

Aclk System

Fail 0

Mask "111111111111 111111111111"

Input R P1 P2 Irsim DCT Irsim DST Chip DCT Chip DST

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 000000001110 111111101000

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 000000000000 000000000000

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 111100000111 000011000101

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 111100000111 000011000101

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 111100000111 000011000101

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 111100000111 000011000101

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 111100000111 000011000101

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 111100000111 000011000101

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 111100000111 000011000101

" ",00000000 0 1 1 xxxxxxxxxxxx xxxxxxxxxxxx 000000000000 000000000000

" ",00000000 0 1 1 000000000000 000000000000 000000000000 000000000000

" ",00000000 0 1 1 000000000000 000000000000 000000000000 000000000000

" ",11111111 1 1 1 000000000000 000000000000 000000000000 000000000000

" ",00000000 1 1 1 111100000111 000011000101 111100000111 000011000101

" ",00000000 1 1 1 111100101100 001000110001 111100101100 001000110001

" ",00000000 1 1 1 111101110001 001101001010 111101110001 001101001010

" ",00000000 1 1 1 111111001100 001111100011 111111001100 001111100011

" ",00000000 1 1 1 000000101111 001111100100 000000101111 001111100100

" ",00000000 1 1 1 000010001010 001101010001 000010001010 001101010001

" ",10001000 1 1 1 000011010000 001000111111 000011010000 001000111111

" ",00010001 1 1 1 000001110010 000100111111 000001110010 000100111111

" ",11000000 1 1 1 111101111101 000101001011 111101111101 000101001011

" ",00111000 1 1 1 111011101010 001010001111 111011101010 001010001111

" ",00001110 1 1 1 111100001000 010000110001 111100001000 010000110001

" ",00111100 1 1 1 111101110100 010101100100 111101110100 010101100100

" ",00111101 1 1 1 111111001000 010111111101 111111001000 010111111101

" ",00000010 1 1 1 000000100100 011000001011 000000100100 011000001011

" ",00000001 1 1 1 000010110011 010101100000 000010110011 010101100000

: Mem End

5.5 Finite Precision Analysis

From the definition of the discrete sinusoidal transforms, we can see that the discrete sinusoidal transforms can be realized directly by using transversal FIR filters. To reduce the number of multipliers, we propose the lattice and direct form IIR architectures, Either the lattice or direct form structure is realized the FIR operations in terms of IIR filter structures. Due to quantization on the coefficients of poles under finite-precision implementation, the poles and zeros can not be cancelled exactly. Therefore, the problems of numerical stability, accuracy of recursive computations become interesting issues.

From the transfer functions of the discrete sinusoidal transforms, we observe that poles of the system belong to the roots of $(1 \pm z^{-N})$ (Lemma 4 of Chapter 4) because those transforms are FIR in natural. The poles are determined from the coefficients of those four multipliers on the butterfly of the lattice module and from the coefficients of the multipliers in the second order autoregressive loop of the direct IIR realization. The poles and zeros can not be cancelled exactly under finite-precision implementation. However, the locations of the poles are determined from the cos and sin coefficients only. We claim that the systems are numerical stable because we can always choose appropriate coefficients such that the poles of both architectures are inside the unit circle. The operation of the butterfly inside the lattice structure is basically a rotation operations like

the QR-Decomposition that has been shown to be numerical stable.

Next, we will discuss the accuracy problem under finite-precision implementation. Our approach is to use random input data sequences as input signals to our time-recursive lattice or IIR structure, then calculate the average SNR by using our time-recursive approach. The input sequence is a random sequence with integer values from 0 to 255. Here the noise or the error of the SNR is the difference between the result of time-recursive and direct DCT computation. The signal used in the computation of SNR is the transformed value obtained from the direct DCT computation. The transform size is assumed to be 8 for simplicity.

Under the cases of finite precision implementation, we write a program to simulate the number of word-length required to ensure the correct operation of the transforms. We perform the time-recursive DCT computation based on the lattice structure by using a random sequence with 1000 sets of input vectors. Before every set of input vector, the system will be reset. Under the case that every word is realized by using 17 bit 2's complement representation (the number of integer bit before binary point are $9, 9, 8, 8, 7, 7, 7, 7$ for $k = 0, 1, .., 7$ respectively, the average SNR of the DCT and DST is given in Table 5.2. For the case of 13 bit 2's complement realization and the same number of integer bit for different k as described above, the average SNR of DCT and DST is shown in Table 5.3. For the case without reset and computing for more than 2000 sets of input vector, the SNR of the DCT and DST for different channels is shown in Table 5.4. In our VLSI implementation, we realize the multiplication by table lookup ROM instead of multiplier. Under the ROM realization, we multiply the fix-point input with floating-point cos or sin coefficients then truncate the results

k	DCT	DST
0/1	44.942963	35.190636
1/2	34.184319	44.814548
2/3	39.264011	43.289104
3/4	41.745930	79.265343
4/5	79.027130	49.259743
5/6	48.230076	50.621094
6/7	53.087914	57.517517
7/8	57.517517	57.517517

Table 5.2: The SNR of different channels under 17 bit 2's commplement realization for lattice structure with reset.

to fix-point representation. The precision will be grealty improved under this realization. The SNR of the DCT and DST under ROM realization with 13 bit 2's complement realization is shown in Table 5.5. We can see that for ROM-based fix-point time-recursive DCT and DST computations with reset by the lattice structure have reasonable SNR under 13 bit 2's complement realization.

We perform the same finite precision simulation on the direct form IIR structure. Under the case that every word is realized by using $\{24, 23, 21, 21, 20, 20, 20, 21\}$ for $\{k = 0, 1, .., 7\}$ 2's complement realization, the average SNR of the DCT and DST computations with reset is given in Table 5.6. Under the same number of bits 2's complement realization for the above IIR case , the average SNR for the case without reset and computing for more than 2000 points is given in Table 5.7. We see that the direct IIR structure requires larger

k	DCT	DST
0/1	10.655185	16.459002
1/2	9.285774	10.730865
2/3	23.946426	18.457523
3/4	20.127630	24.292803
4/5	30.984459	25.592823
5/6	19.969921	34.406025
6/7	33.664532	27.156586
7/8	28.221741	24.243774

Table 5.3: The SNR of different channels under 12 bit 2's commplement realization for lattice structure with reset.

k	DCT	DST
0/1	44.987225	36.080139
1/2	29.888460	39.248798
2/3	39.477829	30.254431
3/4	24.502811	79.231575
4/5	79.267960	43.144184
5/6	40.664646	23.840471
6/7	24.687447	16.813284
7/8	16.019300	79.347824

Table 5.4: The SNR of different channels under 17 bit 2's commplement realization for lattice structure without reset.

k	DCT	DST
0/1	50.443138	53.527218
1/2	45.009701	49.953545
2/3	47.878632	53.429337
3/4	51.759617	62.241699
4/5	62.962940	59.318321
5/6	55.741486	57.282707
6/7	58.739319	60.233566
7/8	54.367374	61.524181

Table 5.5: The SNR of different channels under 12 bit 2's commplement realization for lattice structure with reset.

number of word-length to have the same range of SNR as the lattice structure.

From the simulation results, we observe some interesting phenomenon. First, the lattice structure has better numerical properties under finite-precision analysis than direct IIR realization. This is because the multiplication inside the feedback loop of the direct IIR structure will increase the dynamic range of internal values.. Therefore, it requires more number of bits to prevent overflow problem. Second, the dynamic range of different channels is different. The word-length to have same SNR for different channels will be different.

5.6 Summary

The algorithm and architecture of the first chip that can generate the 1-D DCT and DST simultaneously were described. We implemented the lattice module

k	DCT	DST
0/1	44.942963	27.735569
1/2	37.864502	70.388779
2/3	52.676773	38.406712
3/4	45.298786	45.232964
4/5	79.027130	63.747444
5/6	55.914196	52.004436
6/7	40.730568	46.265491
7/8	35.204231	45.141651

Table 5.6: The SNR of different channels for IIR realization with reset.

k	DCT	DST
0/1	10.665971	0.115382
1/2	12.286688	7.103189
2/3	11.586293	8.112682
3/4	13.489008	24.281227
4/5	18.664270	10.714873
5/6	12.423462	3.119943
6/7	11.605821	0.481012
7/8	10.191545	24.317970

Table 5.7: The SNR of different channels for direct form IIR structure without reset.

using the distributed arithmetic method with a data rate of 116 Mb/s under $2\mu m$ CMOS technology. Testing results clearly show that our VLSI implementation is a good candidate for real-time image processing. We are currently exploring further refinements and a full-implementation of a system that can handle 16×16 block sizes.

Chapter 6

Conclusions and Future Research

In this dissertation, we have studied the problem of developing efficient VLSI architectures for real-time computation of the discrete sinusoidal transforms, especially the DCT/DST/DHT/DFT/LOT/CLT. The architectures are based on the time-recursive approach which can obtain the transform immediately whenever a new datum arrives. This approach has been shown to be a very promising method for the applications of video communications since our structures can match the high speed requirements in real-time video systems. Moreover, the resulting architectures are regular, modular, local-connected, and very suitable for VLSI implementation. In the following, we will summary our contributions and suggest further research directions.

The major contribution of our work is to propose a new point of view in the VLSI implementation of the discrete sinusoidal transforms. Instead of using different implementation techniques to improve the realization of FFT-like algorithms, we start from the serially available property of signals in the transmission and scanning systems to improve the operation speed of the systems. We have shown that our system has higher throughput rate than others and satisfies the

speed requirement in video communication systems. From the time-recursive approach, we also provide an unified view of the discrete sinusoidal transforms. All these discrete sinusoidal transforms can be generated by using same lattice or IIR architectures with different multiplication coefficients. The fundamental dual generation relations between the DCT/DST, DHT/DFT and LOT/CLT can also be derived. The other important results are summaried as follows.

Based on the time-recursive concept, we first proposed the unified parallel lattice structures that can dually generate the DCT and DST simultaneously as well as the DHT are developed. These structures can obtain the transformed data for sequential input time-recursively with throughput rate one per clock cycle and the total number of multipliers required is $6N - 8$. Furthermore, there is no constraint on N. The resulting architectures are regular, modular, and without global communication so that they are very suitable for VLSI implementation. Every lattice module in the lattice array can be viewed as an analysis filter banks. It is because every lattice module itself is an independent digital filter with different frequency components $k, l = 0, 1, ..., N - 1$. We also proposed the synthesis filter bank structure for the DCT that is almost same as that of the analysis filter bank.

In addition to deriving unified parallel lattice structures for 1-D discrete sinusoidal transforms, we also employ the frame-recursive concept on the 2-D DCT for the successive input frames. A new real-time parallel 2-D DCT lattice structure is proposed. The system is fully-pipelined with throughput rate N clock cycles for $N \times N$ successive input data frame. Moreover, the 2-D DCT architecture is modular, regular, and requires only two 1-D DCT blocks which can be extended directly from the 1-D DCT. Parallel implementation of this

architecture for HDTV applications is also presented.

The lattice structures we proposed are superior to other architectures in all the qualitative comparisons except the number of multipliers is large when N is small. Hence, we proposed an optimal time-recursive unified architectures for computing the DCT, DST, DHT, DFT, LOT, and CLT using only half as many multipliers as the unified lattice structures. It is optimal in the sense that the number of the multipliers used is a minimum and both speed and area are asymptotically optimal. We show that to generate the DCT and DST, only $2N-2$ multipliers are necessary, while in the case of dual generation of the DCT and DST, only $1.5N$ multipliers are required for each transform on average. Two transforms are dually generated simultaneously in the lattice structure, while this optimal architecture has the flexibility of generating either one transform or both together. We also gave a theoretical justification of the unified time-recursive architecture using the fundamental recurrence formula. Finally, we generalized the time-recursive concept to multi-dimensional transforms. The resulting architecture is fully-pipelined, modular, and regular. It requires only d 1-D arrays for computing a d-D DXT.

The VLSI implementation of the lattice module has been fabricated by using $2\mu m$ double metal CMOS technology. The chip has been shown to be fully functional under 14.5MHz clock rate which can achieve a data processing rate of 116Mb/s that is satisfied for most video signal applications. Under current 1.2 or 0.8 μm CMOS technology, a clock rate of 40MHz and data rate of 320M Hz is easy to achieve that is very desirable in sophisticated video communications applications such as HDTV or multimedia systems.

Although a lot of work has been done on this research topic, a few prob-

lems are not studied completely and several new research topics can be further investigated. They are summaried as follows.

The finite-precision problem is an important issue for filter realizations. Several interesting issues, such as numerical sensitivity, stability, quantization effects under different realization models, the sensitivity of the SNR of different frequency components should be addressed. Theoretical analysis and numerical simulation of the quantization effects should be done to better understand these problems.

Fault tolerance is an important issue for VLSI systems. For example, in the 2-D DCT chip, there will be more than 100000 transistors. Therefore, designing an efficient algorithm-based fault-tolerance scheme is essential to ensure the correct operation of the chip.

Motion estimation is important in video signal processing and teleconferencing where only small motion involved between interframes. Instead of transmitting all the data in the frames, only motion vectors are estimated and transmitted. There are many ways to find motion vectors, such as block matching, phase correction and cross correlation approaches. Most of the phase correlation approachs are based on the DFT. However, the DCT other than the DFT is most used in the image data compression. Combining the DCT and motion estimation by the time-recursive and phase correction relation concepts will be an interesting research direction.

Bibliography

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," IEEE Trans. Comput., vol. C-23, pp. 90-93, Jan. 1974.

[2] R. N. Bracewell, "Discrete Hartley transform," J. opt. Soc. Amer., vol. 73, pp. 1832-1835, Dec. 1983.

[3] R. N. Bracewell, "The fast Hartley transform," Proc. IEEE, vol. 72, pp1010-1018, Aug. 1984.

[4] R. J. Clark, "Relation between the Karhunen-Loeve and cosine transform," IEE Proc., vol. 128, pt. F, no. 6, pp. 359-360, Nov. 1981.

[5] A. K. Jain, "A fast Karhunen-Loeve transform for a class of stochastic process," IEEE Trans., Commun., vol. COM-24, pp.1023-1029, 1976.

[6] R. Yip and K. R. Rao, "On the computation and effectiveness of discrete sine transform," Comput. Elec. Eng., vol. 7, pp. 45-55, 1980.

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

[8] B. D. Tseng, and W. C. Miller, "On computing the discrete cosine transform," IEEE Trans. on Computer, vol. C-27, pp. 966-968, July 1976.

[9] M. Vetterli and H. Nussbaumer, "Simple FFT and DCT algorithm with reduced number of operations," Signal Processing, vol. 6, no. 4, pp. 267-278, Aug. 1984.

[10] M. Vetterli, and A. Ligtenberg, "A discrete Fourier-Cosine transform chip," IEEE Journal on Selected Areas in Communications, vol. SAC-4, No. 1, pp. 49-61. Jan. 1986.

[11] H. W. Jones, D. N. Hein, and S. C. Knauer, "The Karhunen-Loeve, discrete cosine and related transform via the Hadmard transform," in Proc. Inc. Telemeter, Conf., Los Angeles, CA, pp. 87-98, Nov. 1978.

[12] M. J. Narashimha and A. M. Peterson, "On the computation of the discrete cosine transform," IEEE Trans. Communication, vol. COM-26, pp. 934-936, June 1978.

[13] W. H. Chen, C. H. Smith, and S. C. Fralick, ' 'A fast computational algorithm for the discrete cosine transform," IEEE Trans. Communication, vol. COM-25, pp. 1004-1009, Sept. 1977.

[14] D. E. Dudgeon, and R. M. Mersereau, **Multidimensional Digital Signal Processing**, Prentice Hall, 1984.

[15] B. G. Lee, "A new algorithm to compute the discrete cosine transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, pp 1243-1245, Dec. 1984.

[16] N. H. Lee and Y. Yasuda, "New 2-D systolic array algorithm for DCT/DST," Electron. Lett., 1989, 25, pp. 1702-1704.

[17] K. J. R. Liu, and C. T. Chiu, "Unified Parallel Lattice Structures for Time-Recursive Discrete Cosine/Sine/Hartley Transforms," to appear IEEE Trans. Signal processing, May 1993.

[18] W. Ma, "2-D DCT systolic array implementation," Electronics Letters, Vol. 27, No. 3, pp. 201-202, 31st Jan. 1991.

[19] H. G. Musmann, P. Pirsch, and H. G. Grallert, "Advances in picture coding," Proc. IEEE, vol. 73, pp. 523-548, Apr. 1985.

[20] H. J. Nussbaumer, and P. Quandale, "Fast Polynomial Transform Computation of 2-d DCT," Proc. Int. Conf. on Digital signal Processing, Florence, Italy, pp. 276-283, 1981.

[21] N. I. Cho and S. U. Lee, "Fast algorithm and implementation of 2-D Discrete Cosine Transform," IEEE Trans. on Circuits and Systems, vol. 38, No. 3, pp. 297-305, March. 1991.

[22] K. Rose, A. Heiman, and I. Dinstein, "DCT/DST Alternate-Transform Image Coding," IEEE Trans. on Communication, vol.38, No. 1, pp. 94-101, Jan. 1990.

[23] A. Rosenfeld, and A. C. Kak, "Digital Picture Processing," 2nd edition, Academic Press, 1982.

[24] B. Silkstrom, *et al.*, "A high speed 2-D Discrete Cosine-Transform," Integration, VLSI journal 5, pp. 159-163, 1987.

[25] M. T. Sun, T. C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16×16 Discrete Cosine Transform," IEEE Trans. on Circuits and Systems, vol. 36, No. 4, pp. 610-617, Apr. 1989.

[26] U. Totzek, F. Matthiesen, S. Wohlleben, and T. G. Noll,"CMOS VLSI Implementation of the 2D-DCT with Linear Processor Array," IEEE ICASSP Proc., pp. 937-940, May, 1990.

[27] S. A. White, "High-Speed Distributed-Arithmetic Realization of a Second-Order Normal-Form Digital Filter," IEEE Trans. on Circuits and Systems, vol. 33, No. 10, pp. 1036-1038, Oct. 1986.

[28] S. A. White, "Applications of Distributed-Arithmetic to Digital Signal Processing: A Tutorial Review," IEEE ASSP Magazine, pp. 4-19, July. 1989.

[29] M. Barbero, S. Cucchi, H. Bailon, "A Flexible Architecture for a HDTV Codec based on DCT", Signal Processing of HDTV, II, L. Chiariglione, ed., pp. 587-594, Elsevier Science Publishers B.V., 1990.

[30] W. Paik, "Digivipher, all digital, channel compatible, HDTV broadcast system," IEEE Trans. on Brocasting, Vol. 36, No. 4, pp. 245-254, Dec. 1990.

[31] S. Okubo, T. Omachi, and F. Ono, "International standardization on Picture Coding," IEICE Transactions, Vol. E.74, No 3, pp 533-539, March 1991.

[32] M. Barbero, S. Cucchi, M. Stroppiana, "Coding Strategies based on DCT for the Transmission of HDTV", Signal Processing of HDTV, L. Chiariglione, ed., pp. 503-508, Elsevier Science Publishers B.V., 1988.

[33] J. A. Bellisio and S. Chu, "Television coding for broadband ISDN," IEEE Globecom'86, vol. 2, pp. 894-890, Dec. 1-4, 1986, Houston Texas.

[34] W. H. Chen, and W. K. Pratt, "Scene Adaptive Coder,"IEEE Trans. on Communications, Vol. Com-32, No.3, pp. 225-232, Mar. 1984.

[35] S. Cucchi, and F. Molo, "DCT-based Television Codec for DS3 digital Transmission," SMPTE Journal, pp. 640-646, Sep. 1989.

[36] S. S. Dixit and J. B. Nardone, "A Variable Bit Rate Layered DCT Video Coder for Packet Switched (ATM) Networks," IEEE ICASSP Proc., pp. 2253-2256, May, 1990.

[37] D. L. Gall, H. Gaggioni, and C. T. Chen, "Transmission of HDTV signals under 140Mbits/s using a subband decomposition and discrete cosine transform coding," Proc. 2nd Int. Workshop on signal processing on HDTV, 29 Feb-2 Mar. 1988, L'Aquila Italy.

[38] R. K. Jurgen, "The Challenges of digital HDTV," IEEE Spectrum, pp. 28-30,71-73, Apr. 1991.

[39] K. Kinoshita, T. Nakahashi, and Eto, "130 M bit/s (H4 rate) HDTV Codec based on the DCT algorithms," Electronics Letters, Vol. 26, No. 16, pp. 1245-1246, 2nd Aug. 1990.

[40] R. L. Nickelson, "The evolution of HDTV in the work of the CCIR," IEEE Trans. Broadcasting, vol. 35, No. 3, pp. 250-258, Sep. 1989.

[41] J. Suzuki, M. Nomura, and S. Ono, "Comparative Study of transform coding for Super High Definition Images,"IEEE ICASSP Proc., pp. 2257-2260, May, 1990.

[42] K. H. Tzou, T. C. Chen, P. E. Fleischer, and M. L. Liou, "Compatible HDTV Coding for Broadband ISDN," Proc. Globecom '88, pp. 743-749, NOV. 1988.

[43] Y. Yashima, and K. Sawada, "100 Mbit/s HDTV Transmission using a High Efficiency Codec," Signal Processing of HDTV, II, L. Chiariglione, ed., pp. 587-594, Elsevier Science Publishers B.V., 1990.

[44] N. S. Jayant, and P. Noll, **Digital Coding of Waveform**, Prentice Hall, 1984.

[45] B. G. Kashef and A. Habibi, "Direct computation of higher-order DCT coefficients from lower-order DCT coefficients," SPIE 28th Annu. Int. Tech. Symp., San Diego, CA, Aug. 19-24, 1984.

[46] M. H. Lee, "On computing 2-D systolic algorithm for discrete cosine transform," IEEE Trans. on Circuit and Systems, vol-37, No. 10, pp. 1321-1323, Oct. 1990.

[47] C. Chakrabarti, and J. J'aJ'a, "Systolic architectures for the computation of the discrete Hartley and the discrete cosine transforms based on prime factor decomposition," IEEE Trans. on Computer, vol. 39, No. 11, pp. 1359-1368, Nov. 1990.

[48] H. S. Hou, "The fast Hartley transform algorithm," IEEE Trans. on Computer, vol. C-36, No. 2, pp. 147-156, Feb. 1987.

[49] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," IEEE Trans. Acoust., Speech, Signal processing, vol. ASSP-32, Aug. 1984.

[50] R. W. Owens and J. J'aJ'a, "A VLSI chip for the Winograd/Prime factor algorithm to compute the discrete Fourier transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-34, pp. 979-989, Aug. 1986.

[51] R. Yip and K. R. Rao, "On the shift property of DCT's and DST's," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, No. 3, pp. 404-406, March. 1987.

[52] H. V. Sorenson, *et. al.*, "On computing the discrete Hartley transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-33, No. 4, pp. 1231-1238, Oct. 1985.

[53] S. B. Narayanan and K. M. M. Prabhu, "Fast Hartley transform pruning," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-39, No. 1, pp. 230-233, Jan. 1991.

[54] W. Kou and J.W. Mark, "A new look at DCT-type transforms," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-37, No. 12, pp. 1899-1908, Dec. 1989.

[55] N. I. Cho and S. U. Lee, "DCT algorithms for VLSI parallel implementations," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-38, No. 1, . 1899-1908, Dec. 1989.

[56] L. W. Chang and M. C. Wu, "A unified systolic array for discrete cosine and sine transforms," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-39, No. 1, pp. 192-194, Jan. 1991.

[57] E. A. Jonckheere and C. Ma, "Split-radix fast Hartley transform in one and two dimensions," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-39, No. 2, pp. 499-503, Feb. 1991.

[58] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation," IEEE Trans. on Computers, vol. 40, No. 9, pp. 989-995, Sep. 1991.

[59] H. M. Ahmed, J. M. Delosme, and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing," IEEE Comput. Mag., vol. 15, No. 1, pp. 65-82, Jan. 1982.

[60] A. V. Oppenheim and R. W. Schafer, **Discrete-Time Signal Processing**, Prentice Hall, 1989.

[61] S. A. White, "High-Speed Distributed-Arithmetic Realization of a Second-Order Normal-Form Digital Filter," IEEE Trans. on Circuits and Systems, vol. 33, No. 10, pp. 1036-1038, Oct. 1986.

[62] R. Young, and N. Kingsbury, "Motion Estimation using Lapped Transforms," IEEE ICASSP Proc., pp. III 261-264, March, 1992.

[63] R. D. Koilpillai, and P. P. Vaidyanathan, "New Results on Cosine-Modulated FIR filter banks satisfying perfect reconstruction," IEEE ICASSP, 1991, pp. 1793-1796.

[64] R. Gluth, "Regular FFT-related transform kernels for DCT/DST based polyphase filter banks," IEEE ICASSP, 1991, pp. 2205-2208.

[65] C. T. Chiu, and K. J. R. Liu, "Real-Time Parallel and Fully-Pipelined Two-Dimensional DCT Lattice Structures with Application to HDTV Systems," IEEE Trans. on Circuits and Systems for Video Technology, pp. 25-32, March 1992.

[66] L. A. Glasser, and D. W. Dobberpuhl, "The Design and Analysis of VLSI Circuits," Addison Wesley, 1985.

[67] Neil H. E. Weste, and Kamran Eshraghian, "Principles of CMOS VLSI Design, A Systems Perspective," Addison Wesley, 1985.

[68] P. H. Ang, P. A. Ruetz, and D. Auld, "Video compression makes big gains," *IEEE Spectrum*, pp. 16–19, Oct. 1991.

[69] C. Chakrabarti, and J. JáJá, "Systolic architectures for the computation of the discrete Hartley and the discrete cosine transforms based on prime factor decomposition," IEEE Trans. on Computer, vol. 39, No. 11, pp. 1359-1368, Nov. 1990.

[70] H. S. Malvar and D. H. Staelin, "The LOT: Transform coding without blocking effects," *IEEE Trans. Acoust., Speech, Signal Processing*, pp. 553–559, Apr. 1989.

[71] T. S. Chihara, *An Introduction to Orthogonal Polynomials.* Gordon and Breach, 1978.

[72] K.J.R. Liu,"Novel parallel architectures for shot-time Fourier transform," SRC TR-92-4, University of Maryland, 1992.

[73] R. W. Owens and J. J'aJ'a, "A VLSI chip for the Winograd/Prime factor algorithm to compute the discrete Fourier transform," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-34, pp. 979-989, Aug. 1986.

[74] P. Duris et. al, "Tight chip area lower bounds for discrete Fourier and Walsh-Hadamard transformations," Information Processing letters 21, pp. 245-247, 1985.

[75] C. D. Thompson, "A complexity theory for VLSI," PH.D. Thesis, Department of Computer Science, Carnegie Mellon University, Pittsburgh, 1980.

[76] P. M. Cassereau, D. H. Staelin, and G. D. Jager,"Encoding of Imaages Based on a Lapped Orthogonal Transforms," IEEE Trans. on Communications, Vol. 37, No. 2, pp. 189-193, Feb. 1989.

[77] H. T. kung, "Why systolic architectures?" IEEE Computer, pp.37-46, 1982.

[78] S. Y. Kung, "VLSI Array Processores," Prentice Hall, 1988.

[79] M. Vetterli, "Fast 2-D discrete Cosine Transform," IEEE ICASSP Proc., pp. 1538-1541, March. 1985.

[80] P. Duhamel and C. Guillemot, "Polynomial Transform computation of the 2-D DCT," IEEE ICASSP Proc., pp. 1515-1518, March. 1990.

[81] D. Slawecki and W. Li, "DCT/IDCT Processor Design for High Data Rate Image Codings," IEEE Trans. on Circuits and Systems for Video Technology, pp. 135-146, June 1992.

[82] M. A. Haque, "A two-dimensional fast cosine transform," IEEE Trans. on Acoust. Speech, Signal Processing, vol. ASSP-33, no. 6, pp. 1532-1539, Dec. 1985.

[83] W. Li, "A new algorithm to compute the DCT and its inverse," IEEE Trans. on Signal Processing, vol. 39, n0. 6, pp. 1305-1313, June, 1991.

[84] K. K. Chau *et al.*, "VLSI implementation of a 2-D DCT in a complier," in Proc. IEEE ICASSP'91, pp. 1233-1236.

[85] W. Kou and J.W. Mark, "A new look at DCT-type transforms," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-37, No. 12, pp. 1899-1908, Dec. 1989.

[86] G. Peceli, "A Common Structure for Recursive Discrete Transforms," IEEE Trans. on Circuits and Systems, vol. 33, No. 10, pp. 1035-10386, Oct. 1986.

[87] J. A. Bellisio and S. Chu, "Television coding for broadband ISDN," IEEE Globecom'86, vol. 2, pp. 894-890, Dec. 1-4, 1986, Houston Texas.

[88] S. Uramoto *et al.*, " A 100-MHz 2-D discrete cosine transform core processor," in Proc. 1991 Symp. VLSI Circuits, Osio, Japan, May 30-June 1, 1991, pp. 35-36.

[89] "L64730 discrete cosine transform processor," LSI Logic, July 1990.

[90] C. S. Burrus, "Digital filter structures described by distributed arithmetic," IEEE Trans. Circuits Syst. vol. CAS-24, no. 12 pp. 674-680, Dec., 1974.

[91] F. Jutand *et al.*, "A single chip video rate 16 by 16 discrete cosine transform," in Proc. IEEE ICASSP'86, Tokyo, Japan, pp. 805-808.

[92] G. K. Wallace, "The JPEG still picture compression standard," Commun. ACM, vol. 34, no. 4, pp. 30-44, April, 1991.

[93] D. LeGall, "MPEG: A video compression standard for multimedia applications," Commun. ACM, vol. 34, no. 4, pp. 46-58, April, 1991.

[94] M. Liou, "Overview of the p by 64 kbits/s video coding standard," Commun. ACM, vol. 34, no. 4, pp. 59-63, April, 1991.