Unified VLSI Lattice Architectures for Discrete Sinusoidal Transforms

K.J. Ray Liu
Electrical Engr. Dept.
Institute of Systems Research
University of Maryland
College Park, MD.

C.T. Chiu
Electrical Engr. Dept.
National Taiwan
Institute of Technology
Taipei, Taiwan.

ABSTRACT

The problems of unified efficient computations of the discrete cosine transform (DCT), discrete sine transform (DST), discrete Hartley transform (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 approach, 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 a linear function of the transform size N. 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 for high-speed applications such as ISDN networks and HDTV systems. It is also shown in this paper 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.

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 high definition television (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. 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 [1, 2, 3]. It was shown by Jain 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 [4, 5]. In 1983, Bracewell introduced the DHT [6] 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 [6, 8, 24, 28].

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 the 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 the 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 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 the fundamental relation between the DCT and DST. In the following, the time-recursive relation for the DCT will be considered first.

This work is partially supported by the NSF grant ECD-8803012.

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

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

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.* $k = 1,2,\dots,N-1.$) 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 $[x(t+1), x(t+2),\dots,x(t+N)]$. From the definition, it is given by

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

This can be rewritten as

$$X_c(k,t+1) = \overline{X}_c(k,t+1)\cos\left(\frac{\pi k}{N}\right) + \overline{X}_s(k,t+1)\sin\left(\frac{\pi k}{N}\right), \tag{3}$$

where

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

and

$$\overline{X}_s(k,t+1) = \frac{2}{N}\sum_{n=t+1}^{t+N} x(n)\sin\left[\frac{\pi[2(n-t)+1]k}{2N}\right]. \tag{5}$$

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

2.2 Time-Recursive Discrete Sine Transform

There are several definitions for the DST. Here we prefer the definition made by Wang in [20] since it is of the form completed with the following derivations. The 1-D DST of a data vector $[x(t), x(t+1),\dots,x(t+N-1)]$ is defined as

$$X_s(k,t) = \frac{2C(k)}{N}\sum_{n=t}^{t+N-1} x(n)\sin\left[\frac{\pi[2(n-t)+1]k}{2N}\right], \quad k = 1,\dots,N. \tag{6}$$

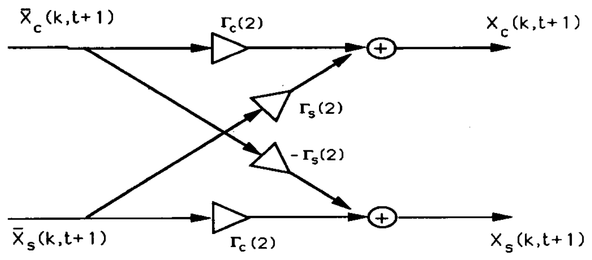
Note that the range of k is from 1 to N. Again, we consider those cases that $D(k)$'s equal one first, *i.e.*

$$X_s(k,t) = \frac{2}{N}\sum_{n=t}^{t+N-1} x(n)\sin\left[\frac{\pi[2(n-t)+1]k}{2N}\right]. \tag{7}$$

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

$$\begin{aligned} X_s(k,t+1) &= \frac{2}{N}\sum_{n=t+1}^{t+N} x(n)\sin\left[\frac{\pi[2(n-t-1)+1]k}{2N}\right] \\ &= \overline{X}_s(k,t+1)\cos\left(\frac{\pi k}{N}\right) - \overline{X}_c(k,t+1)\sin\left(\frac{\pi k}{N}\right). \end{aligned} \tag{8}$$

Here the terms $\overline{X}_s(k,t+1)$ and $\overline{X}_c(k,t+1)$ that are used in (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.



$\Gamma_c(n)=\cos(\pi kn/2N)$, $\Gamma_s(n)=\sin(\pi kn/2N)$

Figure 1: The lattice module.

2.3 The Lattice Structures

From (3) and (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. 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 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

$$\begin{aligned} \overline{X}_c(k,t+1) &= X_c(k,t) - x(t)\left(\frac{2}{N}\right)\cos\left(\frac{\pi k}{2N}\right) \\ &\quad + x(t+N)\left(\frac{2}{N}\right)\cos\left[\frac{\pi(2N+1)k}{2N}\right] \\ &= X_c(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\cos\left(\frac{\pi k}{2N}\right), \end{aligned} \tag{9}$$

and

$$\begin{aligned} \overline{X}_s(k,t+1) &= X_s(k,t) - x(t)\left(\frac{2}{N}\right)\sin\left(\frac{\pi k}{2N}\right) \\ &\quad + x(t+N)\left(\frac{2}{N}\right)\sin\left[\frac{\pi(2N+1)k}{2N}\right] \\ &= X_s(k,t) + \left[-x(t) + (-1)^k x(t+N)\right]\left(\frac{2}{N}\right)\sin\left(\frac{\pi k}{2N}\right). \end{aligned} \tag{10}$$

From (3), (8), (9), and (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

$$X_c(k,t+1) = \overline{X}_c(k,t+1)\cos\left(\frac{\pi k}{N}\right) + \overline{X}_s(k,t+1)\sin\left(\frac{\pi k}{N}\right), \tag{11}$$

and

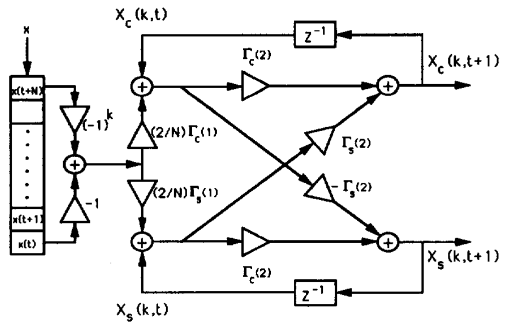
$$X_s(k,t+1) = \overline{X}_s(k,t+1)\cos\left(\frac{\pi k}{N}\right) - \overline{X}_c(k,t+1)\sin\left(\frac{\pi k}{N}\right), \tag{12}$$

where

$$\overline{X}_c(k,t{+}1) = X_c(k,t) + \left[-x(t) + (-1)^k x(N+t)\right] C(k)\left(\frac{2}{N}\right)\cos\left(\frac{\pi k}{2N}\right) \tag{13}$$

and

$$\overline{X}_s(k,t{+}1) = X_s(k,t) + \left[-x(t) + (-1)^k x(N+t)\right] C(k)\left(\frac{2}{N}\right)\sin\left(\frac{\pi k}{2N}\right) \tag{14}$$



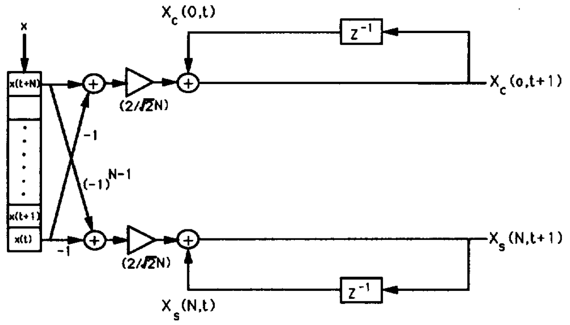


Figure 2: The lattice structure for the DCT and DST with coefficients $C(k)$'s and $D(k)$'s, $k = 0, 1, 2, ..., N-1, N$.

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

$$X_c(0, t+1) = \frac{2}{N\sqrt{2}} \sum_{n=t+1}^{t+N} x(n). \tag{15}$$

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

$$X_c(0, t+1) = X_c(0, t) + \frac{2}{N\sqrt{2}}[-x(t) + x(t+N)]. \tag{16}$$

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

$$\begin{aligned} X_s(N, t+1) &= \frac{2}{N\sqrt{2}} \sum_{n=t+1}^{t+N} x(n)(-1)^{n-t-1} \\ &= X_s(N, t) + \frac{2}{N\sqrt{2}}\left[-x(t) + (-1)^{N-1}x(t+N)\right]. \end{aligned} \tag{17}$$

The complete time-recursive lattice modules for $(k = 0, 1, 2, .., N-1.)$ are shown in Fig. 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. Then the output signals $X_c(k, t)$ and $X_s(k, t)$, $k = 0, 1, ..., N-1, N$, are updated recursively according to (11), (12), (16) and (17). 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.

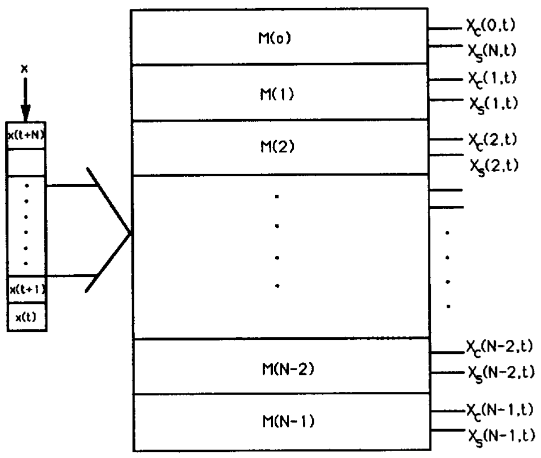


Figure 3: The parallel lattice structure for the DCT and DST.

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. 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 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 [14, 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 [21, 18] 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. This parallel lattice structure requires $6N-4$ multipliers and $5N-1$ adders.

3 Inverse Discrete Cosine Transform (IDCT) and Inverse Discrete Sine Transform (IDST)

3.1 Time-Recursive IDCT

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

$$x_c(n,t) = \sum_{k=t}^{t+N-1} C(k-t)X(k)\cos\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1. \tag{18}$$

The coefficients $C(k)$'s are given in (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

$$x_c(n,t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k)\cos\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right]. \tag{19}$$

Similar to the previous sections, we can decompose (19) into

$$x_c(n,t+1) = \overline{x}_c(n,t+1)\cos\left[\frac{\pi(2n+1)}{2N}\right] + \overline{x}_{as}(n,t+1)\sin\left[\frac{\pi(2n+1)}{2N}\right], \tag{20}$$

where

$$\overline{x}_c(n,t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k)\cos\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \tag{21}$$

and

$$\overline{x}_{as}(n,t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k)\sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right]. \tag{22}$$

In order to be a dually generated pair of the IDCT given in (18), we define the auxiliary inverse discrete sine transform (AIDST) as

$$x_{as}(n,t) = \sum_{k=t}^{t+N-1} C(k-t)X(k)\sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1. \tag{23}$$

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 (28) , 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

$$x_{as}(n,t+1) = \sum_{k=t+1}^{t+N} C(k-t-1)X(k)\sin\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right]. \tag{24}$$

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

$$x_{as}(n,t+1) = \overline{x}_{as}(n,t+1)\cos\left[\frac{\pi(2n+1)}{2N}\right] - \overline{x}_c(n,t+1)\sin\left[\frac{\pi(2n+1)}{2N}\right]. \tag{25}$$

3.1.1 Lattice Structure for IDCT

Combining (20) and (25), 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.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 (18) and (21) as well as (22) and (23) 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

$$\overline{x}_c(n,t+1) = x_c(n,t) - \frac{1}{\sqrt{2}}X(t) + \left(\frac{1}{\sqrt{2}} - 1\right)\cos\left[\frac{\pi(2n+1)}{2N}\right]X(t+1), \tag{26}$$

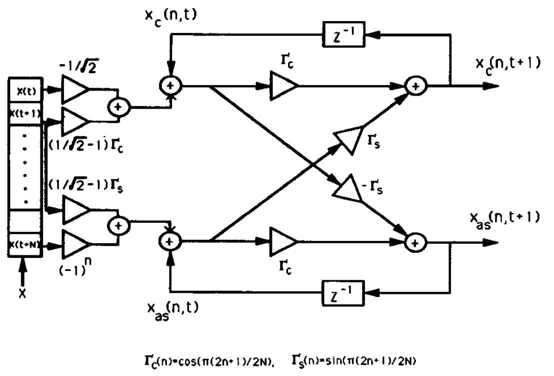


Figure 4: The lattice structure for the IDCT and AIDST.

and

$$\overline{x}_{as}(n,t+1) = x_{as}(n,t) + (-1)^n X(t+N) + \left(\frac{1}{\sqrt{2}} - 1\right) \cdot \sin\left[\frac{\pi(2n+1)}{2N}\right]X(t+1). \tag{27}$$

The complete lattice module for the IDCT and AIDST is shown in Fig.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.

3.2 Time-Recursive IDST

From the definition of the DST in (6), the IDST for the transform domain sequence $[X(t+1), X(t+2), ..., X(t+N)]$ is given by

$$x_s(n,t) = \sum_{k=t+1}^{t+N} D(k-t)X(k)\sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1. \tag{28}$$

The coefficients $D(k)$'s are given in (6). Analogous to Section 3.1, we define the auxiliary inverse discrete cosine transform (AIDCT)

$$x_{ac}(n,t) = \sum_{k=t+1}^{t+N} D(k-t)X(k)\cos\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \quad n = 0, 1, ..., N-1, \tag{29}$$

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

$$x_s(n,t+1) = \sum_{k=t+2}^{t+N+1} D(k-t-1)X(k)\sin\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right], \tag{30}$$

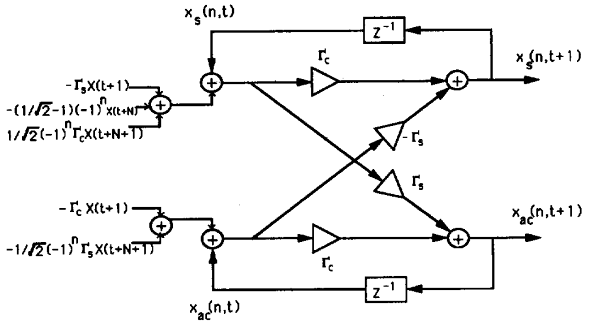


Figure 5: The pre-lattice structure for the IDST and AIDCT.

and

$$x_{ac}(n,t+1) = \sum_{k=t+2}^{t+N+1} D(k-t-1)X(k)\cos\left[\frac{\pi(2n+1)(k-t-1)}{2N}\right]. \tag{31}$$

Same as before, we can decompose (30) and (31) to

$$x_s(n,t+1) = \overline{x}_s(n,t+1)\cos\left[\frac{\pi(2n+1)}{2N}\right] - \overline{x}_c(n,t+1)\sin\left[\frac{\pi(2n+1)}{2N}\right], \tag{32}$$

and

$$x_{ac}(n,1) = \overline{x}_c(n,t+1)\cos\left[\frac{\pi(2n+1)}{2N}\right] + \overline{x}_s(n,t+1)\sin\left[\frac{\pi(2n+1)}{2N}\right], \tag{33}$$

where

$$\overline{x}_{ac}(n,t+1) = \sum_{k=t+2}^{t+N+1} D(k-t-1)X(k)\cos\left[\frac{\pi(2n+1)(k-t)}{2N}\right], \tag{34}$$

and

$$\overline{x}_s(n,t+1) = \sum_{k=t+2}^{t+N+1} D(k-t-1)X(k)\sin\left[\frac{\pi(2n+1)(k-t)}{2N}\right]. \tag{35}$$

3.2.1 Lattice Structure for IDST and AIDCT

If we employ the time-recursive derivation in the previous section to exploit the relations between $x_s(n,t+1)$ and $\overline{x}_s(n,t+1)$ as well as $x_{ac}(n,t+1)$ and $\overline{x}_{ac}(n,t+1)$, the results are

$$\begin{aligned}\overline{x}_s(n,t+1) &= -\sin\left[\frac{\pi(2n+1)}{2N}\right]X(t+1) \\ &\quad -\left(\frac{1}{\sqrt{2}}-1\right)(-1)^n X(t+N) \\ &\quad +\frac{1}{\sqrt{2}}(-1)^n\cos\left[\frac{\pi(2n+1)}{2N}\right]X(t+N+1) \\ &\quad +x_s(n,t),\end{aligned} \tag{36}$$

and

$$\begin{aligned}\overline{x}_{ac}(n,t+1) &= x_{ac}(n,t) - \cos\left[\frac{\pi(2n+1)}{2N}\right]X(t+1) \\ &\quad -\frac{1}{\sqrt{2}}(-1)^n\sin\left[\frac{\pi(2n+1)}{2N}\right]X(t+N+1).\end{aligned} \tag{37}$$

Equations (32), (33), (38) and (38) reveal that to dually generate the IDST and AIDCT requires 9 multipliers and 7 adders, more than that required for the IDCT and AIDST. The result is shown in Fig. 5. To reduce the number of multipliers and adders, substitute (38) and (38) into (32) and (33) and rearrange (32) and (33), we have

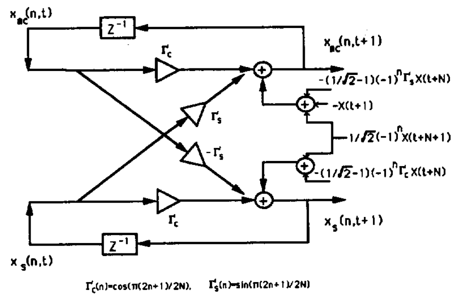

$x_{ac}(n,t)$
Z^{-1}
Γ_c'
$x_{ac}(n,t+1)$
$-(1/\sqrt{2}-1)(-1)^n\Gamma_s' X(t+N)$
$-X(t+1)$
Γ_s'
$1/\sqrt{2}(-1)^n X(t+N+1)$
$-\Gamma_s'$
$-(1/\sqrt{2}-1)(-1)^n\Gamma_c' X(t+N)$
Γ_c'
$x_s(n,t+1)$
Z^{-1}
$x_s(n,t)$
$\Gamma_c'(n)=\cos(\pi(2n+1)/2N)$, $\Gamma_s'(n)=\sin(\pi(2n+1)/2N)$

Figure 6: The post-lattice structure for the IDST and AIDCT.

$$\begin{aligned}x_s(n,t+1) &= \cos\left[\frac{\pi(2n+1)}{2N}\right]x_s(n,t) - \sin\left[\frac{\pi(2n+1)}{2N}\right]x_{ac}(n,t) \\ &\quad -(\frac{1}{\sqrt{2}}-1)(-1)^n\cos\left[\frac{\pi(2n+1)}{2N}\right]X(t+N) \\ &\quad +(\frac{1}{\sqrt{2}})(-1)^n X(t+N+1),\end{aligned} \tag{38}$$

and

$$\begin{aligned}x_{ac}(n,t+1) &= \cos\left[\frac{\pi(2n+1)}{2N}\right]x_{ac}(n,t) + \sin\left[\frac{\pi(2n+1)}{2N}\right]x_s(n,t) \\ &\quad -(\frac{1}{\sqrt{2}}-1)(-1)^n\sin\left[\frac{\pi(2n+1)}{2N}\right]X(t+N) \\ &\quad +(\frac{1}{\sqrt{2}})(-1)^n X(t+N+1) - X(t+1).\end{aligned} \tag{39}$$

The lattice module of this rearranged IDST and AIDCT is shown in Fig. 6. This structure differs from all the previous lattice modules in that the input 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

$$\begin{aligned}x_{as}(n,t+1) &= \cos\left[\frac{\pi(2n+1)}{2N}\right]x_{as}(n,t) - \sin\left[\frac{\pi(2n+1)}{2N}\right]x_c(n,t) \\ &\quad -(\frac{1}{\sqrt{2}})\sin\left[\frac{\pi(2n+1)}{2N}\right]X(t) \\ &\quad +(-1)^n X(t+N)\cos\left[\frac{\pi(2n+1)}{2N}\right],\end{aligned} \tag{40}$$

and

$$\begin{aligned}x_c(n,t+1) &= \cos\left[\frac{\pi(2n+1)}{2N}\right]x_c(n,t) + \sin\left[\frac{\pi(2n+1)}{2N}\right]x_{as}(n,t) \\ &\quad -(\frac{1}{\sqrt{2}})\cos\left[\frac{\pi(2n+1)}{2N}\right]X(t) - X(t+1) \\ &\quad +(-1)^n\sin\left[\frac{\pi(2n+1)}{2N}\right]X(t+N) \\ &\quad +(\frac{1}{\sqrt{2}}-1)X(t+1).\end{aligned} \tag{41}$$

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

$$\begin{aligned} X_c(k,t+1) &= \cos\left(\frac{\pi k}{N}\right) X_c(k,t) + \sin\left(\frac{\pi k}{N}\right) X_s(k,t) \\ &+ (\frac{2}{N})\cos\left(\frac{\pi k}{2N}\right)\left[-x(t) + (-1)^k x(t+N)\right], \quad (42) \end{aligned}$$

and

$$\begin{aligned} X_s(k,t+1) &= \cos\left(\frac{\pi k}{N}\right) X_s(k,t) - \sin\left(\frac{\pi k}{N}\right) X_s(k,t) \\ &- (\frac{2}{N})\sin\left(\frac{\pi k}{2N}\right)\left[-x(t) + (-1)^k x(t+N)\right]. \quad (43) \end{aligned}$$

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

4 Discrete Hartley Transform (DHT)

According to Bracewell's definition of the DHT in [6], the data sequence $x(n)$ and the DHT transformed data $X(k)$ have the following relation

$$\begin{aligned} X_h(k,t) &= \frac{1}{N}\sum_{n=t}^{t+N-1} x(n) cas\left(\frac{2\pi k(n-t)}{N}\right) \\ &= \frac{1}{N}\sum_{n=t}^{t+N-1} x(n)\left[\cos\left(\frac{2\pi k(n-t)}{N}\right) + \sin\left(\frac{2\pi k(n-t)}{N}\right)\right], \\ & k = 0,1,\ldots,N-1. \quad (44) \end{aligned}$$

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:

$$X_h(k,t) = \acute{X}_c(k,t) + \acute{X}_s(k,t), \quad (45)$$

where

$$\acute{X}_c(k,t) = \frac{1}{N}\sum_{n=t}^{t+N-1} x(n)\left[\cos\left(\frac{2\pi k(n-t)}{N}\right)\right], \quad (46)$$

and

$$\acute{X}_s(k,t) = \frac{1}{N}\sum_{n=t}^{t+N-1} x(n)\left[\sin\left(\frac{2\pi k(n-t)}{N}\right)\right]. \quad (47)$$

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 [22]. 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 $[x(t+1), x(t+2), \ldots, x(t+N)]$ are

$$\acute{X}_c(k,t+1) = \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\cos\left[\frac{2\pi k(n-t-1)}{N}\right], \quad (48)$$

and

$$\acute{X}_s(k,t+1) = \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\sin\left[\frac{2\pi k(n-t-1)}{N}\right]. \quad (49)$$

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

$$\acute{X}_c(k,t+1) = \overline{X}_c(k,t+1)\cos\left(\frac{2\pi k}{N}\right) + \overline{X}_s(k,t+1)\sin\left(\frac{2\pi k}{N}\right), \quad (50)$$

and

$$\acute{X}_s(k,t+1) = \overline{X}_s(k,t+1)\cos\left(\frac{2\pi k}{N}\right) - \overline{X}_c(k,t+1)\sin\left(\frac{2\pi k}{N}\right), \quad (51)$$

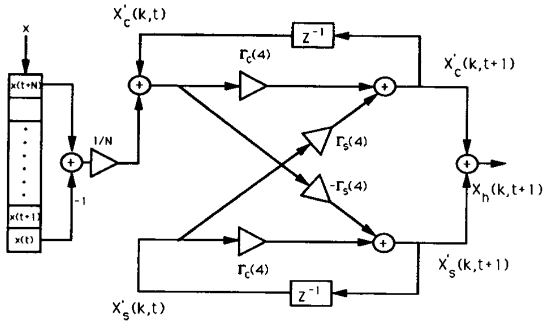


Figure 7: The lattice structure for the DHT for $k = 1, 2, \ldots, N-1$.

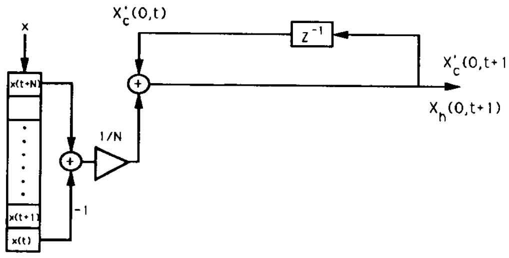


Figure 8: The lattice structure for the DHT for $k = 0$.

where

$$\begin{aligned} \overline{X}_c(k,t+1) &= \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\cos\left(\frac{2\pi k(n-t)}{N}\right) \\ &= \acute{X}_c(k,t) + \frac{1}{N}[-x(t) + x(t+N)], \quad (52) \end{aligned}$$

and

$$\begin{aligned} \overline{X}_s(k,t+1) &= \frac{1}{N}\sum_{n=t+1}^{t+N} x(n)\sin\left(\frac{2\pi k(n-t)}{N}\right) \\ &= \acute{X}_s(k,t). \quad (53) \end{aligned}$$

The lattice module for the DHT for $k = 1, \ldots, N-1$ is shown in Fig. 7 and Fig. 8. From Fig. 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.

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.

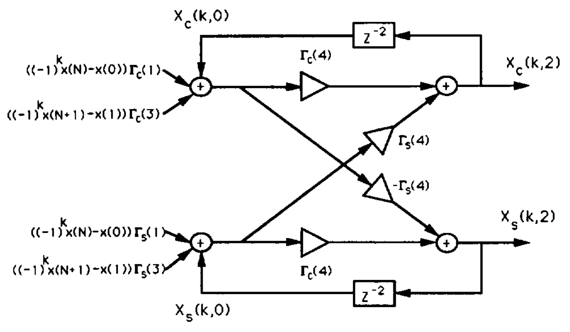


Figure 9: The lattice structure for block-size-two operation on the DCT and DST.

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 (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,

$$X_c(k,2)=\sum_{n=2}^{N+1}x(n)\cos\left\{\frac{\pi[2(n-2)+1]k}{2N}\right\}, \qquad (54)$$

and

$$X_s(k,2)=\sum_{n=2}^{N+1}x(n)\sin\left\{\frac{\pi[2(n-2)+1]k}{2N}\right\}. \qquad (55)$$

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

$$X_c(k,2)=\overline{X}_c(k,2)\cos\left(\frac{2\pi k}{N}\right)+\overline{X}_s(k,2)\sin\left(\frac{2\pi k}{N}\right), \qquad (56)$$

and

$$X_s(k,2)=\overline{X}_s(k,2)\cos\left(\frac{2\pi k}{N}\right)-\overline{X}_c(k,2)\sin\left(\frac{2\pi k}{N}\right), \qquad (57)$$

where

$$\begin{aligned}\overline{X}_c(k,2) &= \sum_{n=2}^{N+1}x(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right]\\ &= X_c(k,0)+[-x(0)+(-1)^k x(N)]\cos\left(\frac{\pi k}{2N}\right)\\ &\quad +[-x(1)+(-1)^k x(N+1)]\cos\left(\frac{3\pi k}{2N}\right), \qquad (58)\end{aligned}$$

and

$$\begin{aligned}\overline{X}_s(k,2) &= \sum_{n=2}^{N+1}x(n)\sin\left[\frac{\pi(2n+1)k}{2N}\right]\\ &= X_s(k,0)+[-x(0)+(-1)^k x(N)]\sin\left(\frac{\pi k}{2N}\right)\\ &\quad +[-x(1)+(-1)^k x(N+1)]\sin\left(\frac{3\pi k}{2N}\right). \qquad (59)\end{aligned}$$

The lattice module for the block-size-two update is shown in Fig. 9. There are two more multipliers in the lattice, *i.e.*, the total number of multipliers is eight. 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

$$X_c(k,m)=\sum_{n=m}^{N+m-1}x(n)\cos\left[\frac{\pi[2(n-m)+1]k}{2N}\right], \qquad (60)$$

and

$$X_s(k,m)=\sum_{n=m}^{N+m-1}x(n)\sin\left[\frac{\pi[2(n-m)+1]k}{2N}\right]. \qquad (61)$$

Applying the same procedure in the case of block-size-two update, we can write (61) and (62) as

$$X_c(k,m)=\overline{X}_c(k,m)\cos\left(\frac{m\pi k}{N}\right)+\overline{X}_s(k,m)\sin\left(\frac{m\pi k}{N}\right), \qquad (62)$$

and

$$X_s(k,m)=\overline{X}_s(k,m)\cos\left(\frac{m\pi k}{N}\right)-\overline{X}_c(k,m)\sin\left(\frac{m\pi k}{N}\right). \qquad (63)$$

where

$$\begin{aligned}\overline{X}_c(k,m) &= \sum_{n=m}^{N+m-1}x(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right]\\ &= X_c(k,0)-\sum_{n=0}^{m-1}x(n)\cos\left(\frac{\pi(2n+1)k}{2N}\right)\\ &\quad +\sum_{n=N}^{N+m-1}x(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right]\\ &= X_c(k,0)\\ &\quad +\sum_{n=0}^{m-1}[-x(n)+(-1)^k x(N+n)]\cos\left[\frac{\pi(2n+1)k}{2N}\right] \qquad (64)\end{aligned}$$

and

$$\begin{aligned}\overline{X}_s(k,m) &= \sum_{n=m}^{N+m-1}x(n)\sin\left[\frac{\pi(2n+1)k}{2N}\right]\\ &= X_s(k,0)-\sum_{n=0}^{m-1}x(n)\sin\left[\frac{\pi(2n+1)k}{2N}\right]\\ &\quad -\sum_{n=N}^{N+m-1}x(n)\sin\left[\frac{\pi(2n+1)k}{2N}\right]\\ &= X_c(k,0)\\ &\quad +\sum_{n=0}^{m-1}[-x(n)+(-1)^k x(N+n)]\sin\left[\frac{\pi(2n+1)k}{2N}\right] \qquad (65)\end{aligned}$$

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. 10. To obtain the transform data $X(k)$ in parallel, N lattice modules of Fig. 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.

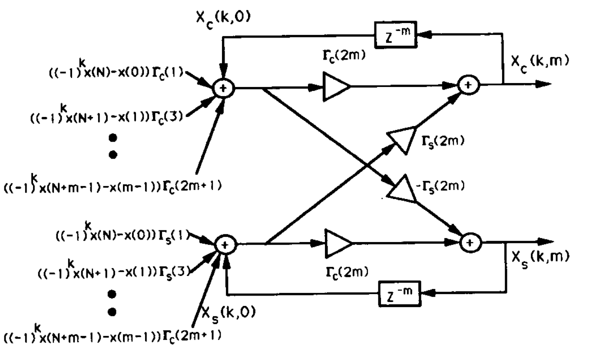

$X_c(k,0)$
z^{-m}
$\Gamma_c(2m)$
$X_c(k,m)$
$((-1)^k x(N)-x(0))\Gamma_c(1)$
$((-1)^k x(N+1)-x(1))\Gamma_c(3)$
$((-1)^k x(N+m-1)-x(m-1))\Gamma_c(2m+1)$
$\Gamma_s(2m)$
$-\Gamma_s(2m)$
$((-1)^k x(N)-x(0))\Gamma_s(1)$
$((-1)^k x(N+1)-x(1))\Gamma_s(3)$
$\Gamma_c(2m)$
$X_s(k,m)$
z^{-m}
$X_s(k,0)$
$((-1)^k x(N+m-1)-x(m-1))\Gamma_c(2m+1)$

Figure 10: The lattice structure for block-size-m operation on the DCT and DST.

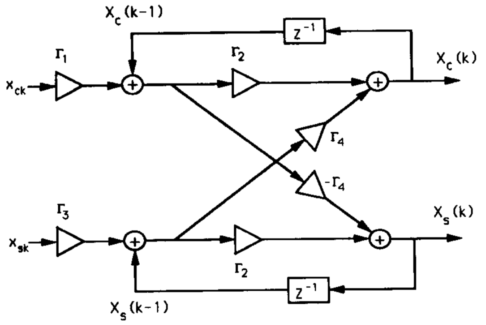


Figure 11: The general lattice module.

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.

6.1 *SISO* Approach

Let us consider this problem through a general lattice structure as shown in Fig. 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

$$\begin{aligned} X_c(t) &= [X_c(t-1)+\Gamma_1 x_{ct}]\Gamma_2 + [X_s(t-1)+\Gamma_3 x_{st}]\Gamma_4, \\ X_s(t) &= [X_s(t-1)+\Gamma_3 x_{st}]\Gamma_2 - [X_c(t-1)+\Gamma_1 x_{ct}]\Gamma_4. \quad (66) \end{aligned}$$

By dividing both equations by Γ_4, we have

$$\begin{aligned} X_c(t)/\Gamma_4 &= [X_c(t-1)+\Gamma_1 x_{ct}]\Gamma_2/\Gamma_4 + [X_s(t-1)+\Gamma_3 x_{st}], \\ X_s(t)/\Gamma_4 &= [X_s(t-1)+\Gamma_3 x_{st}]\Gamma_2/\Gamma_4 - [X_c(t-1)+\Gamma_1 x_{ct}].(67) \end{aligned}$$

The lattice structure manifesting the above relations is shown in Fig. 12. It is noted that only four multipliers exist in this structure and the outputs obtained differ from the original one by a factor Γ_4. To

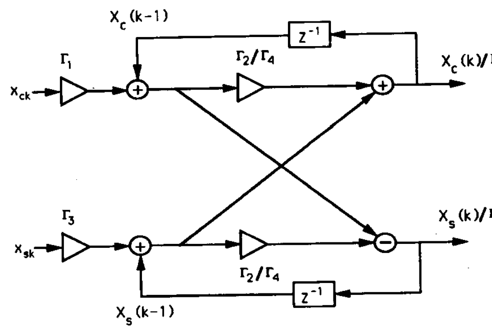


Figure 12: The model of multiplier-reduction.

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

$$\begin{aligned} X_c(1)/\Gamma_4 &= [X_c(0)+\Gamma_1 x_{c1}]\Gamma_2/\Gamma_4 + [X_s(0)+\Gamma_3 x_{s1}], \\ X_s(1)/\Gamma_4 &= [X_s(0)+\Gamma_3 x_{s1}]\Gamma_2/\Gamma_4 - [X_c(0)+\Gamma_1 x_{c1}]. \quad (68) \end{aligned}$$

For $t=2$

$$\begin{aligned} X_c(2)/\Gamma_4 &= [X_c(1)+\Gamma_1 x_{c2}]\Gamma_2/\Gamma_4 + [X_s(1)+\Gamma_3 x_{s2}], \\ X_s(2)/\Gamma_4 &= [X_s(1)+\Gamma_3 x_{s2}]\Gamma_2/\Gamma_4 - [X_c(1)+\Gamma_1 x_{c2}]. \quad (69) \end{aligned}$$

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 (69) 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

$$\begin{aligned} X_c(2)/\Gamma_4^2 &= [X_c(1)/\Gamma_4 + (\Gamma_1/\Gamma_4)x_{c2}]\Gamma_2/\Gamma_4 \\ &\quad + [X_s(1)/\Gamma_4 + (\Gamma_3/\Gamma_4)x_{s2}], \\ X_s(2)/\Gamma_4^2 &= [X_s(1)/\Gamma_4 + (\Gamma_3/\Gamma_4)x_{s2}]\Gamma_2/\Gamma_4 \\ &\quad - [X_s(1)/\Gamma_4 + (\Gamma_1/\Gamma_4)x_{c2}]. \quad (70) \end{aligned}$$

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

$$\begin{aligned} X_c(N)/\Gamma_4^N &= \left[X_c(N-1)/\Gamma_4^{N-1} + (\Gamma_1/\Gamma_4^{N-1})x_{cN}\right]\Gamma_2/\Gamma_4 \\ &\quad + \left[X_s(N-1)/\Gamma_4^{N-1} + (\Gamma_3/\Gamma_4^{N-1})x_{sN}\right], \\ X_s(N)/\Gamma_4^N &= \left[X_s(N-1)/\Gamma_4^{N-1} + (\Gamma_3/\Gamma_4^{N-1})x_{sN}\right]\Gamma_2/\Gamma_4 \\ &\quad - \left[X_c(N-1)/\Gamma_4^{N-1} + (\Gamma_1/\Gamma_4^{N-1})x_{cN}\right]. \quad (71) \end{aligned}$$

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. 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. 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

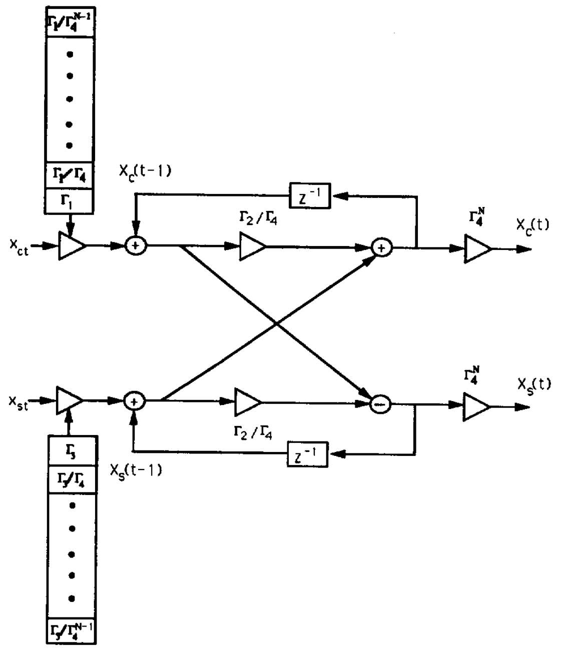


Figure 13: The multiplier-reduced lattice module.

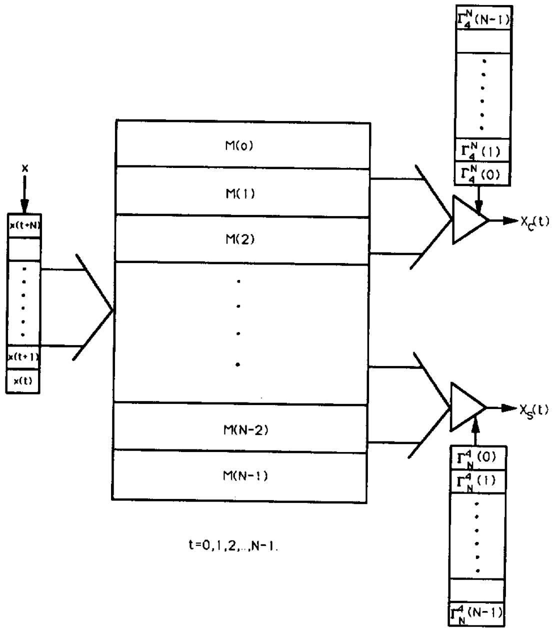


Figure 14: The complete parallel multiplier-reduced lattice structure.

$t = 1, 2, .., N-1$, the multiplier-reduced lattice structure is shown in Fig. 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 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.

6.2 Double-lattice Approach

Generally, a post-lattice structure has the following forms

$$\begin{aligned} X_c(k) &= \Gamma_2 X_c(k-1) + \Gamma_4 X_s(k-1) + \Gamma_1 x_{ck}, \\ X_s(k) &= \Gamma_2 X_s(k-1) - \Gamma_4 X_c(k-1) + \Gamma_3 x_{sk}. \end{aligned} \tag{72}$$

Based on the relationships shown below

$$\begin{aligned} &\Gamma_2 X_c(k-1) + \Gamma_4 X_s(k-1) = \\ &\quad \frac{1}{2}(\Gamma_2 + \Gamma_4)[X_c(k-1) + X_s(k-1)] \\ &\quad + \frac{1}{2}(\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)], \end{aligned} \tag{73}$$

and

$$\begin{aligned} &\Gamma_2 X_s(k-1) - \Gamma_4 X_c(k-1) = \\ &\quad \Gamma_2 X_s(k-1) + \Gamma_4 X_c(k-1) \\ &\quad -2\Gamma_4 X_c(k-1) \\ &\quad = \frac{1}{2}(\Gamma_2 + \Gamma_4)[X_c(k-1) + X_s(k-1)] \\ &\quad -\frac{1}{2}(\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)] \\ &\quad -2\Gamma_4 X_c(k-1), \end{aligned} \tag{74}$$

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

$$\begin{aligned} X_c(k) &= \frac{1}{2}(\Gamma_2 + \Gamma_4)[X_c(k-1) + X_s(k-1)] \\ &\quad + \frac{1}{2}(\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)] + \Gamma_1 x_{ck} \\ &= \frac{1}{2}\{t1 + t2\} + \Gamma_1 x_{ck}, \\ X_s(k) &= \frac{1}{2}(\Gamma_2 + \Gamma_4)[X_c(k-1) + X_s(k-1) \\ &\quad -\frac{1}{2}(\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)] \\ &\quad -2\Gamma_4 X_c(k-1) + \Gamma_3 x_{ck} \\ &= \frac{1}{2}\{t1 - t2\} - 2\Gamma_4 X_c(k-1) + \Gamma_3 x_{ck}. \end{aligned} \tag{75}$$

where

$$t1 = (\Gamma_2 + \Gamma_4)[X_c(k-1) + X_s(k-1)], \tag{76}$$

and

$$t2 = (\Gamma_2 - \Gamma_4)[X_c(k-1) - X_s(k-1)]. \tag{77}$$

The operational flow graph of (77) is illustrated in Fig. 15. Instead of calculating the outputs from (74) directly (that requires 6 multipliers and 4 adders), the first lattice adds and subtracts $X_c(k-1)$

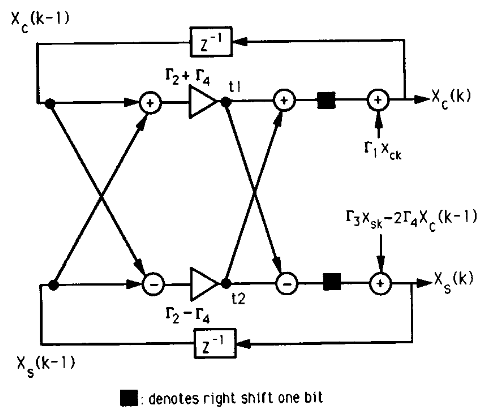


Figure 15: The double-lattice form of the post-lattice realization.

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 (78) and (79). 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 double-lattice structure can save N multipliers, but requires $3N$ more adders. The latency is N clock cycles and the system remains *SIPO*.

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.

Here, we would like to compare our lattice structures of the DCT and DST with those proposed in [14, 15, 7]. The architecture in [14] 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 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, Table 3, and Table 4.

The lattice architecture with six multipliers in the module as shown in Fig. 2 is called Liu-Chiu1 structure, the one in Fig. 14 is called Liu-Chiu2, and the parallel structure with the double-lattice modules as shown in Fig. 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

	Liu-Chiu1	chen *et al.*	Lee	Hou
No. of Multiplier	$6N-4$	$N ln(N)$ $-3N/2+4$	$(N/2)\ln(N)$	$N-1$
latency	N	$N/2$	$\frac{1}{2}[\ln(N)$ $*(\ln(N)-1)]$	$3N/2$
limitation on N	no	2^n	2^n	2^n
com.	local	global	global	global
I/O	*SIPO*	*PIPO*	*PIPO*	*SIPO*

Table 1: Comparison of different DCT algorithms

NO	Liu-Chiu1	Liu-Chiu2	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: 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 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 4: Comparision of the latency

	Liu-Chiu	Sorenson	Chaitali-JaJa
No. of Multipliers	$4N$	$N\ln(N)$ $-3N+4$	$N1+N2$
No. of adders	$5N-2$	$3N\ln(N)-3N+4$ $-3N+4$	$4N+\sqrt{N1}$ $\sqrt{N1}$
latency	N	$N\ln(N)$	$N1+N2$
limitation on N	no	2^n	$N=N1*N2$, $N1,N2$ mutual prime
com.	local	global	local
I/O	$SIPO$	$PIPO$	$SISO$

Table 5: Comparision of different DHT algorithms

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 [14, 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. 7 and different DHT algorithms [23, 18] is listed in Table 5. The architecture in [23], a representative fast algorithm, is developed base on the existing FFT method. Chaitali-JaJa's algorithm in [18] 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.

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. 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. 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.

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

$$x_c(n) = \sum_{k=0}^{N-1} C(k)X_c(k)\cos\left[\frac{\pi k(2n+1)}{2N}\right]. \tag{78}$$

Since in the synthesis bank different transform components are sent to independent synthesis modules, we therefore focus on a specific

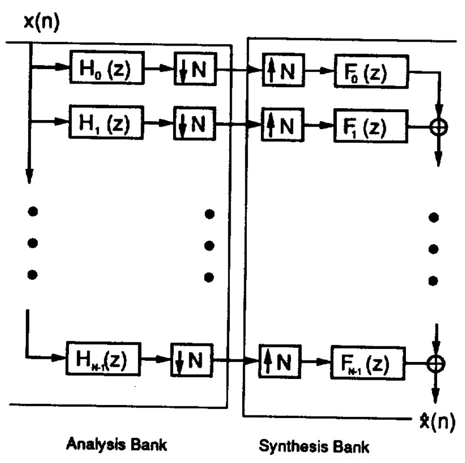


Figure 16: The filter bank structure.

transform component. Denote $\bar{x}_c(n,k)$ as the output signal generated by a specific synthesis module

$$\bar{x}_c(n,k) = C(k)X_c(k)\cos\left[\frac{\pi k(2n+1)}{2N}\right]. \tag{79}$$

The time-recursive concept can be applied here to update $\bar{x}_c(n,k)$ recursively. Use the result in section 3.1 that IDCT and AIDST can be generated from each other recursively and denote $\bar{x}_{as}(n,k)$ as the auxiliary inverse sine transform generated by a specific synthesis filter. We can obtain the following recursive-generated relations for $\bar{x}_c(n,k)$ and $\bar{x}_{as}(n,k)$ as

$$\begin{aligned}\bar{x}_c(n+1,k) &= C(k)X_c(k)\cos\left[\frac{\pi k[2(n+1)+1]}{2N}\right]\\ &= \bar{x}_c(n,k)\cos\left(\frac{\pi k}{2N}\right) - \bar{x}_{as}(n,k)\sin\left(\frac{\pi k}{2N}\right),\end{aligned} \tag{80}$$

and

$$\begin{aligned}\bar{x}_{as}(n+1,k) &= C(k)X_c(k)\sin\left[\frac{\pi k[2(n+1)+1]}{2N}\right]\\ &\bar{x}_{as}(n,k)\cos\left(\frac{\pi k}{2N}\right) + \bar{x}_c(n,k)\sin\left(\frac{\pi k}{2N}\right).\end{aligned} \tag{81}$$

(82) and (83) suggest that the $\bar{x}_c(n+1,k)$ and $\bar{x}_{as}(n+1,k)$ can be dually generated from the previous values $\bar{x}_c(n,k)$ and $\bar{x}_{as}(n,k)$ in a lattice form as shown in Fig. 17. Because the initial values for $x_c(0,k)$ and $x_{as}(0,k)$ are

$$\bar{x}_c(0,k) = X_c(k)\cos\left[\frac{\pi k}{2N}\right], \tag{82}$$

and

$$\bar{x}_{as}(0,k) = X_{as}(k)\sin\left[\frac{\pi k}{2N}\right], \tag{83}$$

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 $\bar{x}_{as}(n,k)$ output is reset every N clock cycles. The synthesis module diagram for the DCT is plotted in Fig. 17. The inverse transform is obtained by summing all the outputs of the synthesis modules.

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.

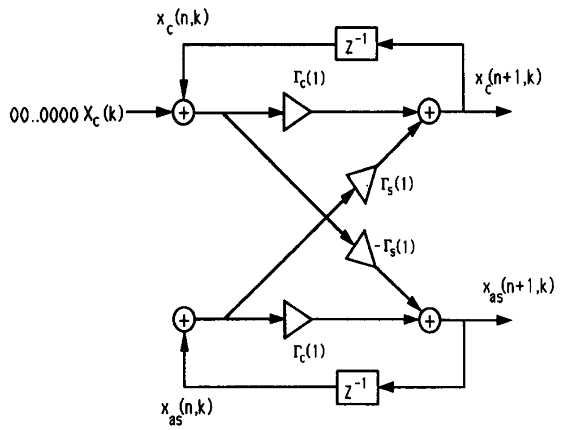


Figure 17: The synthsis bank structure of the DCT.

By using the dual generation concept, the operation of the synthesis module for the DST is

$$\begin{aligned}\bar{x}_s(n+1,k) &= D(k)X_s(k)\sin\left[\frac{\pi k[2(n+1)+1]}{2N}\right] \\ &= \bar{x}_s(n,k)\cos\left(\frac{\pi k}{2N}\right) + \bar{x}_{ac}(n,k)\sin\left(\frac{\pi k}{2N}\right),\end{aligned} \tag{84}$$

and

$$\begin{aligned}\bar{x}_{ac}(n+1,k) &= D(k)X_s(k)\cos\left[\frac{\pi k[2(n+1)+1]}{2N}\right] \\ &= \bar{x}_{ac}(n,k)\cos\left(\frac{\pi k}{2N}\right) - \bar{x}_s(n,k)\sin\left(\frac{\pi k}{2N}\right).\end{aligned} \tag{85}$$

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

$$\begin{aligned}x_h(n) &= \sum_{k=0}^{N-1} X_h(k)cas\left(\frac{2\pi kn}{N}\right) \\ &= \sum_{k=0}^{N-1} X_h(k)\left[\cos\left(\frac{2\pi kn}{N}\right) + \sin\left(\frac{2\pi kn}{N}\right)\right], \\ & \quad n = 0, 1, ..., N-1.\end{aligned} \tag{86}$$

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

$$x_h(n) = x_c'(n) + x_s'(n). \tag{87}$$

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

$$x_c'(n+1,k) = x_c'(n,k)\cos\left(\frac{2\pi k}{N}\right) - x_s'(n,k)\sin\left(\frac{2\pi k}{N}\right), \tag{88}$$

and

$$x_s'(n+1,k) = x_s'(n,k)\cos\left(\frac{2\pi k}{N}\right) - x_c'(n,k)\sin\left(\frac{2\pi k}{N}\right). \tag{89}$$

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.

9 Conclusions

In this paper, 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 [22, 33]. 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) [29, 30] which is known as an efficient method for the computation of orthogonal rotations and trigonometric functions.

References

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

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

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

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

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[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] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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. Acous., Speech, Signal Processing, vol. ASSP-34, pp. 979-989, Aug. 1986.

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

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

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

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

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

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

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

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

[30] 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.

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

[32] 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.

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