EURASIP Journal on Applied Signal Processing 2002:9, 961–974

A Reduced-Complexity Fast Algorithm for Software Implementation of the IFFT/FFT in DMT Systems

Tsun-Shan Chan
VXIS Technology Corporation, Hsin-chu, Taiwan, ROC
Email: tscan@vxis.com

Jen-Chih Kuo
Department of Electrical Engineering, Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, ROC
Email: jj@access.ee.ntu.edu.tw

An-Yeu (Andy) Wu
Department of Electrical Engineering, Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, ROC
Email: andywu@cc.ee.ntu.edu.tw

Received 31 August 2001 and in revised form 15 May 2002

The discrete multitone (DMT) modulation/demodulation scheme is the standard transmission technique in the application of *asymmetric digital subscriber lines (ADSL)* and *very-high-speed digital subscriber lines (VDSL)*. Although the DMT can achieve higher data rate compared with other modulation/demodulation schemes, its computational complexity is too high for cost-efficient implementations. For example, it requires 512-point IFFT/FFT as the modulation/demodulation kernel in the ADSL systems and even higher in the VDSL systems. The large block size results in heavy computational load in running programmable digital signal processors (DSPs). In this paper, we derive computationally efficient fast algorithm for the IFFT/FFT. The proposed algorithm can avoid complex-domain operations that are inevitable in conventional IFFT/FFT computation. The resulting software function requires less computational complexity. We show that it acquires only 17% number of multiplications to compute the IFFT and FFT compared with the Cooly-Tukey algorithm. Hence, the proposed fast algorithm is very suitable for firmware development in reducing the MIPS count in programmable DSPs.

Keywords and phrases: FFT, IFFT, DMT, software implementation.

1. INTRODUCTION

Recent progress of Internet access has a strong demand on high-speed data transmission. To overcome the transmission bottleneck over the conventional twisted-pair telephone lines, several sophisticated modulation/demodulation schemes have been proposed, including carrierless-amplitude-phase (CAP) modulation [1], discrete multitone modulation (DMT) [2, 3, 4, 5] and QAM technology [6]. Among these advanced modulation schemes, the DMT can achieve highest transmission rate since it incorporates lots of advanced DSP techniques such as dynamic bit allocation, multidimensional tone encoding, frequency-domain equalization, and so forth. As a consequence, the DMT has been chosen as the physical layer transmission standard by the ADSL standardization committee.

One major disadvantage of the DMT scheme is its high computational complexity. In particular, the large block size of the IFFT/FFT consumes lots of computing power in running programmable DSPs [7]. In [8], we have considered a cost-efficient lattice VLSI architecture to realize the IFFT/FFT in integrated circuits. In this paper, we propose computationally efficient fast algorithms to run the IFFT/FFT function in software implementation such as programmable DSP processors (DSPs). By making use of the symmetric/antisymmetric properties of the Fourier transform, we first decompose the IFFT/FFT into a combination of two new real-domain transform kernels—the *Modified DCT* and *Modified DST*. These two transform functions are used to replace the complex-domain IFFT/FFT. Then we employ the divide-and-conquer approach in [9] to derive novel recursive algorithms and butterfly architectures for the modified DCT DST.

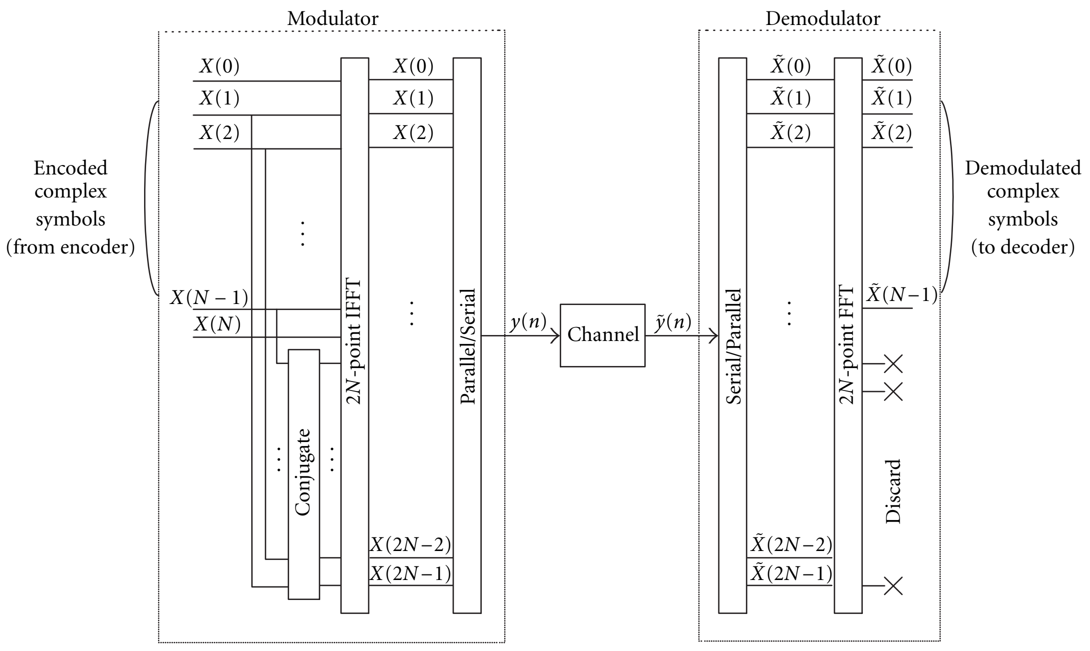


Figure 1: The IFFT/FFT block diagram in the DMT system.

The new scheme can avoid redundant complex-domain of the IFFT/FFT. That is, it involves only real-valued operations to compute the IFFT/FFT. Hence, we can avoid the special data structure in software programming to run complex-domain addition/multiplication operations in computing the IFFT/FFT. In addition, our analysis shows that we need only 17% and multiplications in computing the IFFT and FFT compared with Cooly-Tukey algorithm [10]. The low computational complexity as well as real-domain operations makes it very suitable for firmware coding in DSPs, which helps to save the MIPS counts. Also, the DSP program can be written in recursive form which requires less ROM/RAM program storage space to implement the IFFT/FFT.

The rest of this paper is organized as follows. Section 2 shows the derivation of the IFFT algorithm. In Section 3, the derivation of the FFT algorithm is discussed. The computation complexity comparison is shown in Section 4. The finite precision effect of our algorithm is also discussed. Finally, we conclude our work in Section 5.

2. REDUCED-COMPLEXITY IFFT ALGORITHM

2.1. The IFFT derivation

The IFFT/FFT block diagram in the DMT system is showed in Figure 1. At the transmitter side, to ensure the IFFT generates only real-valued outputs, the inputs of the IFFT in the DMT standard have the constraint [11],

$$\begin{gathered} X(0) = X(N) = 0, \\ X(k) = X^*(2N-k) \quad \text{for } k = 1, 2, \ldots, N-1, \end{gathered} \tag{1}$$

where $X(k) \triangleq X_r(k) + j \cdot X_i(k)$ are encoded complex symbols. As defined in [12, Chapter 9], the IFFT of a finite-length sequence of length $2N$ is

$$x(n) = \frac{1}{2N} \cdot \left[\sum_{k=0}^{2N-1} X(k) W_{2N}^{-nk} \right], \quad \text{for } n = 0, 1, \ldots, 2N-1, \tag{2}$$

where

$$W_{2N}^{nk} \triangleq \exp\left(-j \frac{2\pi nk}{2N} \right) = \cos \frac{2\pi nk}{2N} - j \sin \frac{2\pi nk}{2N}. \tag{3}$$

By decomposing n into the first half and the second half, (2) becomes

$$x(n) = \frac{1}{2N} \cdot \left[\sum_{k=0}^{N-1} X(k) W_{2N}^{-nk} + \sum_{k=N}^{2N-1} X(k) W_{2N}^{-nk} \right]. \tag{4}$$

Next, by substituting (3) into (4), and using (1), we can simplify (4) as (see Appendix A)

$$\begin{aligned} x(n) &= \frac{1}{N} \cdot \left[\sum_{k=0}^{N-1} X_r(k) \cos \frac{2\pi nk}{2N} - \sum_{k=0}^{N-1} X_i(k) \sin \frac{2\pi nk}{2N} \right] \\ &= \frac{1}{N} \cdot [\mathrm{MDCT}(n) - \mathrm{MDST}(n)], \\ &\qquad \text{for } n = 0, 1, \ldots, 2N-1. \end{aligned} \tag{5}$$

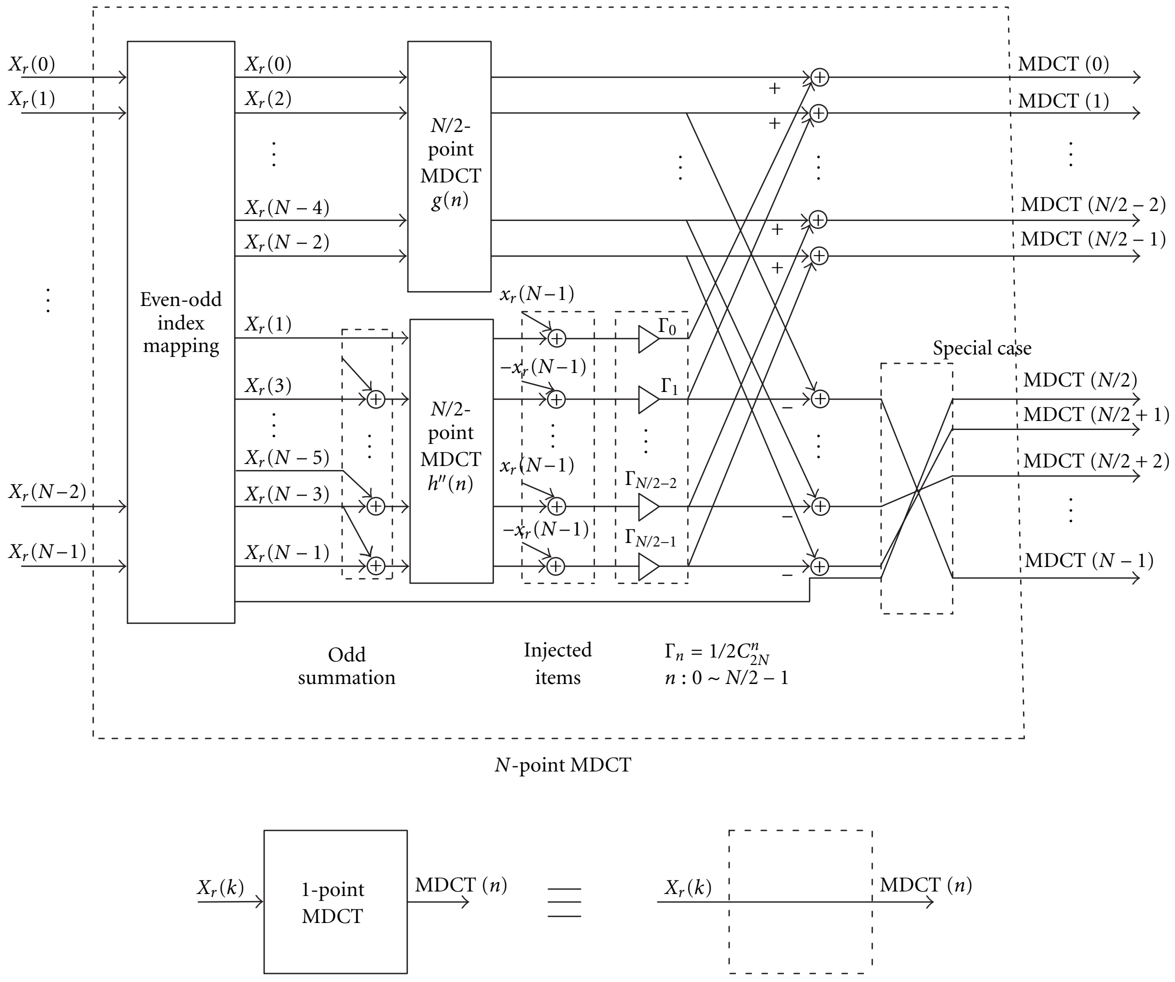


FIGURE 2: N-point MDCT(n) butterfly structure, where 1-point MDCT is the minimum-sized processing block.

From (5), we can see that the computation of the IFFT is decomposed into two real-valued operations. One is a discrete cosine transform DCT-like operation with $X_r(k)$, $k = 0, 1, 2, \ldots, N-1$, as the inputs. The other is a discrete sine transform DST-like operation with $X_i(k)$, $k = 0, 1, 2, \ldots, N-1$, as the inputs. We will name the first term *Modified DCT (MDCT)*, and the second term *Modified DST (MDST)*. Note that the MDCT and MDST involve only real-valued operators. Furthermore, it can be shown that

$$\text{MDCT}(n) = \text{MDCT}(2N-n), \quad \text{for } n = 0, 1, \ldots, N-1, \tag{6}$$

$$\text{MDST}(n) = -\text{MDST}(2N-n), \quad \text{for } n = 0, 1, \ldots, N-1. \tag{7}$$

Hence, we can focus on computing MDCT(n) and MDST(n) for $n = 0, 1, \ldots, N-1$. Then, expand the results for $n = N+1, N+2, \ldots, 2N-1$. For the special cases of $n = 0$ and $n = N$, the MDCT and MDST can be simplified as

$$\begin{aligned}
\text{MDCT}(0) &= \sum_{k=0}^{N-1} X_r(k)\cos\frac{2\pi 0k}{2N} = \sum_{k=0}^{N-1} X_r(k),\\
\text{MDST}(0) &= \sum_{k=0}^{N-1} X_i(k)\sin\frac{2\pi 0k}{2N} = 0,\\
\text{MDCT}(N) &= \sum_{k=0}^{N-1} X_r(k)\cos\frac{2\pi Nk}{2N} = \sum_{k=0}^{N-1} X_r(k)(-1)^k,\\
\text{MDST}(N) &= \sum_{k=0}^{N-1} X_i(k)\sin\frac{2\pi Nk}{2N} = 0,
\end{aligned} \tag{8}$$

respectively. These simple relationships can help us to save additional computation complexity.

2.2. MDCT/MDST operations of the IFFT

From the preceding discussion, we can see that the implementation issue of the IFFT is to realize MDCT and MDST in a cost-efficient way. Then, we can just combine the results of the MDCT and MDST to obtain the IFFT results based on (5). Here, we first consider the implementation of the MDCT. We follow the derivation in [9] and define $C_{2N}^{nk} \triangleq \cos(2\pi nk/2N)$. Then, the MDCT can be written as

$$\mathrm{MDCT}(n) = \sum_{k=0}^{N-1} X_r(k) C_{2N}^{nk}, \quad \text{for } n = 0, 1, \ldots, N-1. \tag{9}$$

Decompose the MDCT into even and odd indices of k, then (9) can be rewritten as

$$\mathrm{MDCT}(n) = g(n) + h'(n), \quad \text{for } n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{10}$$

where

$$\begin{aligned} g(n) &\triangleq \sum_{k=0}^{N/2-1} X_r(2k) C_{2N}^{n(2k)} = \sum_{k=0}^{N/2-1} X_r(2k) C_N^{nk}, \\ h'(n) &\triangleq \sum_{k=0}^{N/2-1} X_r(2k+1) C_{2N}^{n(2k+1)}. \end{aligned} \tag{11}$$

Define $h(n) \triangleq 2C_{2N}^{n} h'(n)$. Following the derivation in Lee's algorithm [9], we can find

$$\mathrm{MDCT}(n) = g(n) + h'(n) = g(n) + \frac{1}{2C_{2N}^{n}} h(n). \tag{12}$$

That is,

$$\begin{aligned} \underbrace{\sum_{k=0}^{N-1} X_r(k) C_{2N}^{nk}}_{N\text{-point MDCT}} &= \underbrace{\sum_{k=0}^{N/2-1} X_r(2k) C_N^{nk}}_{N/2\text{-point MDCT, } g(n)} \\ &\quad + \frac{1}{2C_{2N}^{n}} \Bigg[\underbrace{\sum_{k=0}^{N/2-1} \left[X_r(2k+1) + X_r(2k-1)\right] C_N^{nk}}_{N/2\text{-point MDCT, } h''(n)} \\ &\quad + \underbrace{X_r(N-1)(-1)^n}_{\text{injected item}} \Bigg], \\ &\quad \text{for } n = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{13}$$

On the other hand, by replacing index n with $(N-n)$ in (12), it can be shown that

$$\mathrm{MDCT}(N-n) = g(n) - h'(n) = g(n) - \frac{1}{2C_{2N}^{n}} h(n). \tag{14}$$

The special case MDCT($N/2$) needs to be computed separately, which can be simplified as

$$\mathrm{MDCT}\left(\frac{N}{2}\right) = \sum_{k=0}^{N-1} X_r(k) C_{2N}^{k(N/2)} = \sum_{k=0}^{N-1} X_r(k) \cos\frac{k\pi}{2}. \tag{15}$$

The mapping of (13), (14), and (15) is shown in Figure 2. As we can see, the N-point MDCT is decomposed into two $N/2$-point MDCT ($g(n)$ and $h''(n)$) plus some pre-processing and post-processing modules. Then we can apply the technique of *divide-and-conquer* to recursively expand the $N/2$-point MDCT until 1-point MDCT is formed. That is, we repeat the decomposition in (10) and (11) until $N = 1$.

Next, we consider the recursive implementation of the MDST. We define $S_{2N}^{nk} \triangleq \sin(2\pi nk/2N)$. As with the derivation in (10), (11), (12), (13), and (14), we can find

$$\begin{aligned} \mathrm{MDST}(n) &= \sum_{k=0}^{N/2-1} X_i(2k) S_N^{nk} \\ &\quad + \frac{1}{2C_{2N}^{n}} \sum_{k=0}^{N/2-1} \left[X_i(2k+1) + X_i(2k-1)\right] S_N^{nk}, \\ \mathrm{MDST}(N-n) &= -\sum_{k=0}^{N/2-1} X_i(2k) S_N^{nk} \\ &\quad + \frac{1}{2C_{2N}^{n}} \sum_{k=0}^{N/2-1} \left[X_i(2k+1) + X_i(2k-1)\right] S_N^{nk}, \\ &\quad \text{for } n = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{16}$$

It is worth noting that the injected item is zero in the MDST. Besides, the MDST also has a special case for index $N/2$ as

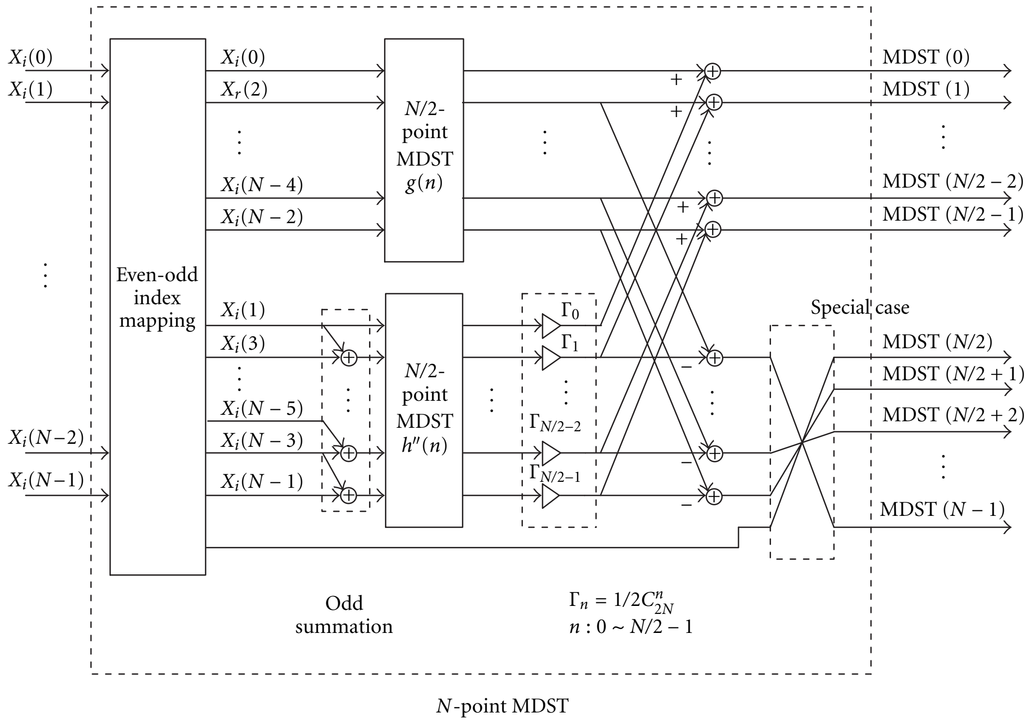
$$\mathrm{MDST}\left(\frac{N}{2}\right) = \sum_{k=0}^{N-1} X_i(k) S_{2N}^{k(N/2)} = \sum_{k=0}^{N-1} X_i(k) \sin\frac{k\pi}{2}. \tag{17}$$

The mapping of the MDST structure in Figure 3 is similar to the MDCT structure, except that minimum processing block is 2-point MDST (see Figure 3) and the injected items do not exist in the MDST implementation. That is, we repeat the decomposition in (16) until $N = 2$. Note that the 1-point MDST is always equal to zero.

2.3. *Overall IFFT computation procedures*

The overall IFFT computation flow is shown in Figure 4. It consists of the MDCT/MDST operations and a post-processing operation. The operations in Figure 4 are as follows:

(1) set the butterfly operation to MDCT mode;
(2) $X_r(k)$, $k = 0, 1, \ldots, N-1$, are first fed into the butterfly architecture to obtain the MDCT(n), for $n = 0, 1, \ldots, N-1$;
(3) the post-processing operation expands the N-point MDCT outputs to $2N$-point MDCT using the symmetric property in (6);
(4) set the butterfly operation to MDST mode;
(5) repeat the computation in Steps 2 and 3 using $X_i(k)$, $k = 0, 1, \ldots, N-1$ as inputs, and obtain the MDST(n), for $n = 0, 1, \ldots, N-1$;
(6) the post-processing operation expands the N-point MDST outputs to $2N$-point MDST by using the antisymmetric property in (7);



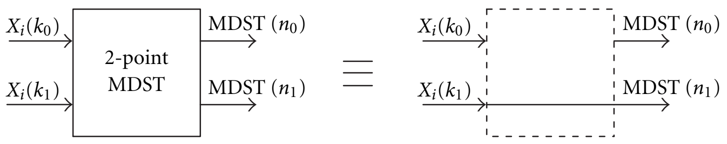


Figure 3: N-point MDST(n) butterfly structure, where 2-point MDST is the minimum-sized processing block.

(7) based on (5), we combine the MDCT and MDST results together with the scaling operation (which is achieved by shifting right by $\log_2(N)$ bits) to obtain the IFFT results. This is done in the post-processing operation.

2.4. Matrix notation of the MDCT/MDST

In this section, we present the matrix notation of the proposed fast IFFT algorithm. The matrix form can help to see the divide-and-conquer nature of our approach. By following the notation in [13], we rewrite $X_r(k)$ and MDCT(n) as

$$[X_r(k)_N] \triangleq \begin{bmatrix} X_r(0) & X_r(1) & \cdots & X_r(N-1) \end{bmatrix}^T, \tag{18}$$

$$[\,\mathrm{MDCT}(n)_N] \triangleq \begin{bmatrix} \mathrm{MDCT}(0) & \mathrm{MDCT}(1) & \cdots & \mathrm{MDCT}(N-1) \end{bmatrix}^T, \tag{19}$$

respectively. Then (9) can be represented as

$$[\,\mathrm{MDCT}(n)_N] = [T_{N,\mathrm{MDCT}}][X_r(k)_N], \tag{20}$$

where $[T_{N,\mathrm{MDCT}}]$ denotes the transform kernel matrix of the MDCT operation. Next, the injected items of (13) can be represented as

$$\text{Injected} = [O_{N/2}]X_r(N-1), \tag{21}$$

where

$$[O_{N/2}] = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & \cdots & -1 \end{bmatrix}^T. \tag{22}$$

We define the *odd-summation matrix* as

$$[L_{N/2}] = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix} \tag{23}$$

and the *scaling matrix* as

$$[\Phi_{N/2}] = \mathrm{diag}\left\{\frac{1}{2C_{2N}^n}\right\}, \quad \text{for } n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{24}$$

The special case of the MDCT in (15) can be represented as

$$\mathrm{MDCT}\left(\frac{N}{2}\right) = [S_N][X_r(k)_N], \tag{25}$$

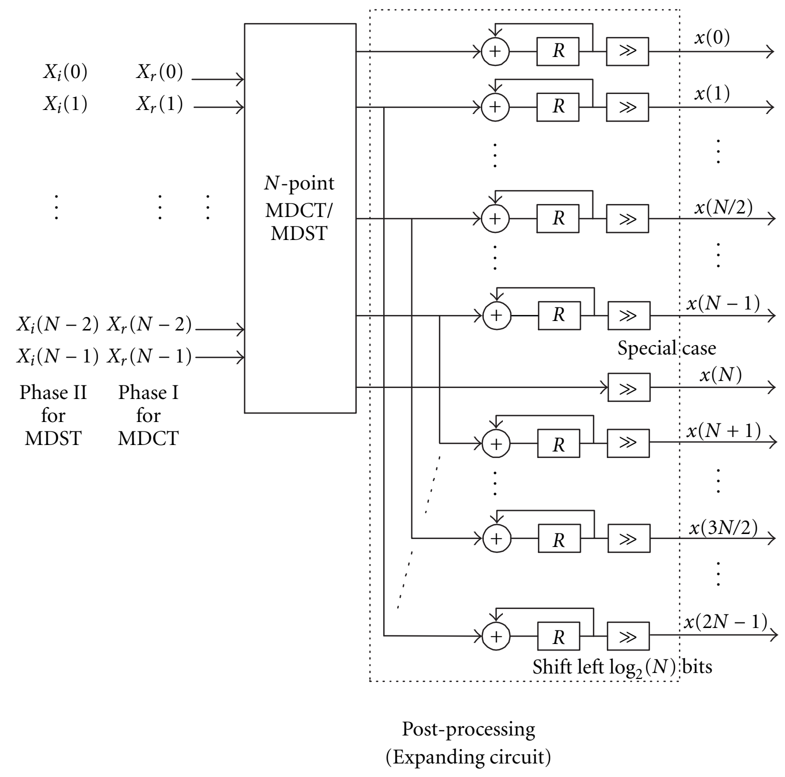


FIGURE 4: The proposed IFFT architecture.

where

$$[S_N] = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & \cdots & 0 \end{bmatrix}. \tag{26}$$

Based on (12), (13), (14), (21), (22), (23), (24), (25), and (26), the $[T_{N,\text{MDCT}}]$ can be expressed in the matrix form as

$$[T_{N,\text{MDCT}}] = \begin{bmatrix} [T_{N/2}] & [\psi_{N,\text{MDCT}}] \\ [T_{N/2}][J_{N/2}] & [\psi_{N,\text{MDCT}}][J_{N/2}] \end{bmatrix}, \tag{27}$$

where $[\psi_{N,\text{MDCT}}] = [\Phi_{N/2}]([L_{N/2}][T_{N/2}] + [O_{N/2}])\|[S_N]$ and $[J_{N/2}]$ denotes the opposite-diagonal identity matrix. We can also represent (20) and (27) in the recursive form as shown in Figure 5. Following the above derivations, the matrix notation of transform kernel of the MDST can be derived as

$$[T_{N,\text{MDST}}] = \begin{bmatrix} [T_{N/2}] & [\psi_{N,\text{MDST}}] \\ -[T_{N/2}][J_{N/2}] & [\psi_{N,\text{MDST}}][J_{N/2}] \end{bmatrix}, \tag{28}$$

where $[\psi_{N,\text{MDST}}] = [\Phi_{N/2}][L_{N/2}][T_{N/2}][S_N]$. Note that the MDST is similar to the MDCT except that there is no injected items. Also, the special case matrix can be modified as

$$[S_N] = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & \cdots & -1 \end{bmatrix}. \tag{29}$$

The block diagram of the MDST in the matrix form is shown in Figure 6.

3. REDUCED-COMPLEXITY FFT ALGORITHM

3.1. The FFT derivation

At the receiver side (see Figure 1), the 512-point FFT is used to demodulate the received signals, which is given by

$$\tilde{X}(k) = \sum_{n=0}^{2N-1} \tilde{x}(n) W_{2N}^{nk}, \quad \text{for } k = 0, 1, \ldots, 2N-1, \tag{30}$$

where

$$W_{2N}^{nk} \triangleq \exp\left(-j\frac{2\pi nk}{2N}\right) = \cos\frac{2\pi nk}{2N} - j \cdot \sin\frac{2\pi nk}{2N}. \tag{31}$$

Note that $\tilde{x}(n)$, $n = 0, 1, \ldots, 2N-1$, are real-valued numbers. Hence, (30) can be rewritten as

$$\begin{aligned} \tilde{X}(k) &= \sum_{n=0}^{2N-1} \tilde{x}(n)\cos\frac{2\pi nk}{2N} - j\sum_{n=0}^{2N-1} \tilde{x}(n)\sin\frac{2\pi nk}{2N} \\ &= \text{MDCT}(k) - j \cdot \text{MDST}(k), \quad \text{for } k = 0, 1, \ldots, 2N-1. \end{aligned} \tag{32}$$

Equation (32) shows that the computation of the FFT is

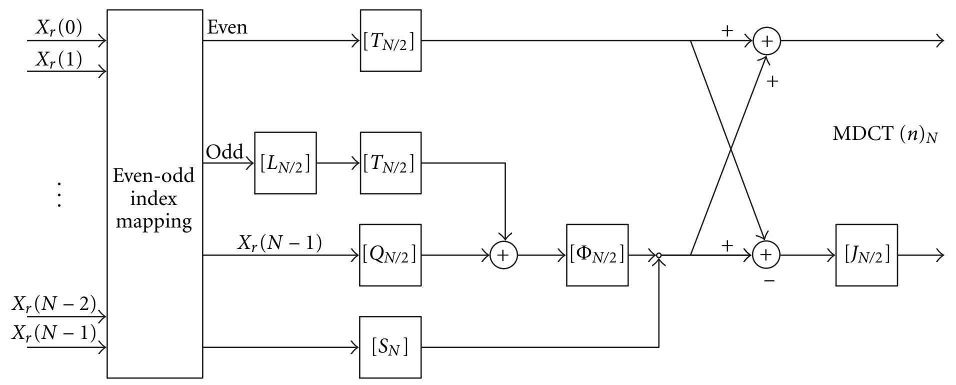


FIGURE 5: Block diagram of the MDCT operation in matrix form.

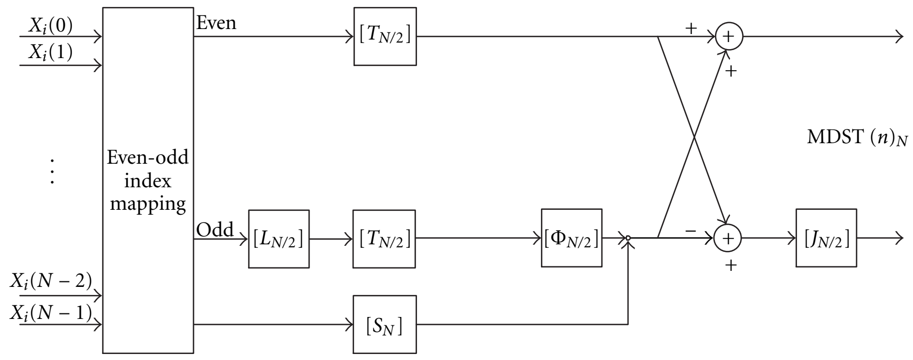


FIGURE 6: Block diagram of the MDCT operation in matrix form.

decomposed into a combination of two real-domain kernels—MDCT(k) and MDST(k). Both MDCT and MDST use $\tilde{x}(n)$, $n = 0, 1, \dots, 2N-1$, as the inputs. Hence, we only employ two real-valued kernels (MDCT and MDST), thus no complex-valued operations are required in computing the FFT. In addition, in the DMT system, the lower N-point FFT outputs are conjugate-symmetric to the upper N-point outputs. We are only interested in N-point data for $k = 0, 1, \dots, N-1$. Hence, we can neglect the outputs $\tilde{X}(k)$, for $k = N, N+1, \dots, 2N-1$.

3.2. MDCT/MDST operations of the FFT

In (32), the transform kernels are $2N$-point MDCT(k) and MDST(k). Here, we propose a novel approach to further reduce the computational complexity. Hence, we only need to perform N-point MDCT/MDST.

We first decompose input sequence into a *symmetric sequence*, $\tilde{x}_c(n)$, plus an *antisymmetric sequence*, $\tilde{x}_s(n)$, where

$$\tilde{x}_c(n) \triangleq \frac{1}{2}[\tilde{x}(n) + \tilde{x}(2N-n)],$$
$$\tilde{x}_s(n) \triangleq \frac{1}{2}[\tilde{x}(n) - \tilde{x}(2N-n)], \quad \text{for } n = 1, 2, \dots, N-1. \tag{33}$$

Hence, we have

$$\tilde{x}(n) = \tilde{x}_c(n) + \tilde{x}_s(n), \tag{34}$$

$$\tilde{x}(2N-n) = \tilde{x}_c(n) - \tilde{x}_s(n), \quad \text{for } n = 1, 2, \dots, N-1. \tag{35}$$

By substituting (34) and (35) into (30), we can simplify (30) as (see Appendix B)

$$\begin{aligned}\tilde{X}(k) &= \Bigg\{\tilde{x}(0) + \tilde{x}(N)(-1)^k \\ &\quad + 2\left[\sum_{n=0}^{N-1} \tilde{x}_c(n)\cos\frac{2\pi nk}{2N} - j\sum_{n=0}^{N-1} \tilde{x}_s(n)\sin\frac{2\pi nk}{2N}\right]\Bigg\} \\ &= \{\tilde{x}(0) + \tilde{x}(N)(-1)^k + 2[\text{MDCT}(k) - j\,\text{MDST}(k)]\}, \\ &\qquad \text{for } k = 0, 1, \dots, N-1,\end{aligned} \tag{36}$$

where $\tilde{x}_c(0) = 0$ and $\tilde{x}_s(0) = 0$. Since the block size is reduced from $2N$-point (see (32)) to N-point (see (36)).

Next, following the derivations of the IFFT in Section 2, we can have

$$\begin{aligned}\text{MDCT}(k) &= g(k) + \frac{1}{2C_{2N}^k}h(k)\\ &= \underbrace{\sum_{n=0}^{N/2-1}\tilde{x}_c(2n)C_N^{nk}}_{N/2\text{-point MDCT, } g(k)}\\ &\quad + \frac{1}{2C_{2N}^k}\left[\underbrace{\sum_{n=0}^{N/2-1}\left[\tilde{x}_c(2n+1)+\tilde{x}_c(2n-1)\right]C_N^{nk}}_{N/2\text{-point MDCT, } h''(k)}\right.\\ &\quad \left. + \underbrace{\tilde{x}_c(N-1)(-1)^k}_{\text{injected item}}\right],\end{aligned} \tag{37}$$

$$\begin{aligned}\text{MDCT}(N-k) &= g(k) - \frac{1}{2C_{2N}^k}h(k)\\ &= \sum_{n=0}^{N/2-1}\tilde{x}_c(2n)C_N^{nk}\\ &\quad - \frac{1}{2C_{2N}^k}\left[\sum_{n=0}^{N/2-1}\left[\tilde{x}_c(2n+1)\right.\right.\\ &\quad \left.+\tilde{x}_c(2n-1)\right]C_N^{nk}\\ &\quad \left. + \tilde{x}_c(N-1)(-1)^k\right],\\ &\quad \text{for } k = 0,1,\dots,\frac{N}{2}-1.\end{aligned} \tag{38}$$

Similarly, for the MDST(k), we have

$$\begin{aligned}\text{MDST}(k) &= g(k) + \frac{1}{2C_{2N}^k}h(k)\\ &= \sum_{n=0}^{N/2-1}\tilde{x}_s(2n)S_N^{nk}\\ &\quad + \frac{1}{2C_{2N}^k}\sum_{n=0}^{N/2-1}\left[\tilde{x}_s(2n+1)+\tilde{x}_s(2n-1)\right]S_N^{nk},\end{aligned} \tag{39}$$

$$\begin{aligned}\text{MDST}(N-k) &= -g(k) + \frac{1}{2C_{2N}^k}h(k)\\ &= -\sum_{n=0}^{N/2-1}\tilde{x}_s(2n)S_N^{nk}\\ &\quad + \frac{1}{2C_{2N}^k}\sum_{n=0}^{N/2-1}\left[\tilde{x}_s(2n+1)+\tilde{x}_s(2n-1)\right]S_N^{nk},\\ &\quad \text{for } k = 0,1,\dots,\frac{N}{2}-1.\end{aligned} \tag{40}$$

The two special cases for index $N/2$ are

$$\begin{aligned}\text{MDCT}\left(\frac{N}{2}\right) &= \sum_{n=0}^{N-1}\tilde{x}_c(n)\cos\frac{n\pi}{2},\\ \text{MDST}\left(N2\right) &= \sum_{n=0}^{N-1}\tilde{x}_s(n)\sin\frac{n\pi}{2}.\end{aligned} \tag{41}$$

The block diagram of the MDCT(k) is shown in Figure 7. The mapping of the MDST structure is similar to the MDCT structure in Figure 7 except that minimum processing block is 2-point MDST and the injected items do not exist in the MDST(k) implementation (see Figure 8). Then we can just combine the MDCT(k) and MDST(k) outputs, followed by adding $\tilde{x}(0)$ and $\tilde{x}(N)(-1)^k$, to obtain the FFT results based on (36).

3.3. Overall FFT computation procedures

The overall computation flow of the FFT is shown in Figure 9. The operations are as follows.

(1) The received signals $\tilde{x}(n)$, $n = 0,1,\dots,2N-1$, are decomposed to $\tilde{x}_c(n)$ and $\tilde{x}_s(n)$, $n = 0,1,\dots,N-1$, through the pre-processing operation.

(2) In the first phase, the generated $\tilde{x}_c(n)$ are fed into recursive butterfly operation to obtain the MDCT(k) outputs.

(3) In the second phase, we repeat the computation by using the $\tilde{x}_s(n)$ as inputs into recursive butterfly operation to obtain the MDST(k) outputs.

(4) We combine the MDCT(k) and MDST(k) results then add $\tilde{x}(0)$ and $\tilde{x}(N)(-1)^k$ together to obtain the FFT results based on (36). This is done in the post-processing operation.

3.4. Matrix notation of the MDCT/MDST

Based on (19), (20), (21), (22) (23), (24), (25), and (26), we can represent (37), (38), and (39) as

$$\left[T_{N,\text{MDCT}}\right] = \begin{bmatrix}\left[T_{N/2}\right] & \left[\psi_{N,\text{MDCT}}\right]\\ \left[T_{N/2}\right]\left[J_{N/2}\right] & -\left[\psi_{N,\text{MDCT}}\right]\left[J_{N/2}\right]\end{bmatrix}, \tag{42}$$

where $[\psi_{N,\text{MDCT}}] = [\Phi_{N/2}]([L_{N/2}][T_{N/2}] + [O_{N/2}])\|[S_N]$,

$$\left[T_{N,\text{MDST}}\right] = \begin{bmatrix}\left[T_{N/2}\right] & \left[\psi_{N,\text{MDST}}\right]\\ -\left[T_{N/2}\right]\left[J_{N/2}\right] & \left[\psi_{N,\text{MDST}}\right]\left[J_{N/2}\right]\end{bmatrix}, \tag{43}$$

where $[\psi_{N,\text{MDST}}] = [\Phi_{N/2}][L_{N/2}][T_{N/2}][S_N]$, of the MDCT$(k)$/ MDST$(k)$, respectively. The block diagrams of the MDCT(k) and MDST(k) are very similar to the MDCT(n) and MDST(n) in Section 2. The difference is that it requires a pre-processing to compute the $\tilde{x}_c(n)$ and $\tilde{x}_s(n)$. The block diagrams of the MDCT and MDST are shown in Figures 10 and 11, respectively.

4. COMPLEXITY COMPARISON AND FINITE-PRECISION EFFECT

4.1. Comparison of hardware complexity

In this section, we compare the computation complexity of the proposed algorithm with the traditional Cooly-Tukey

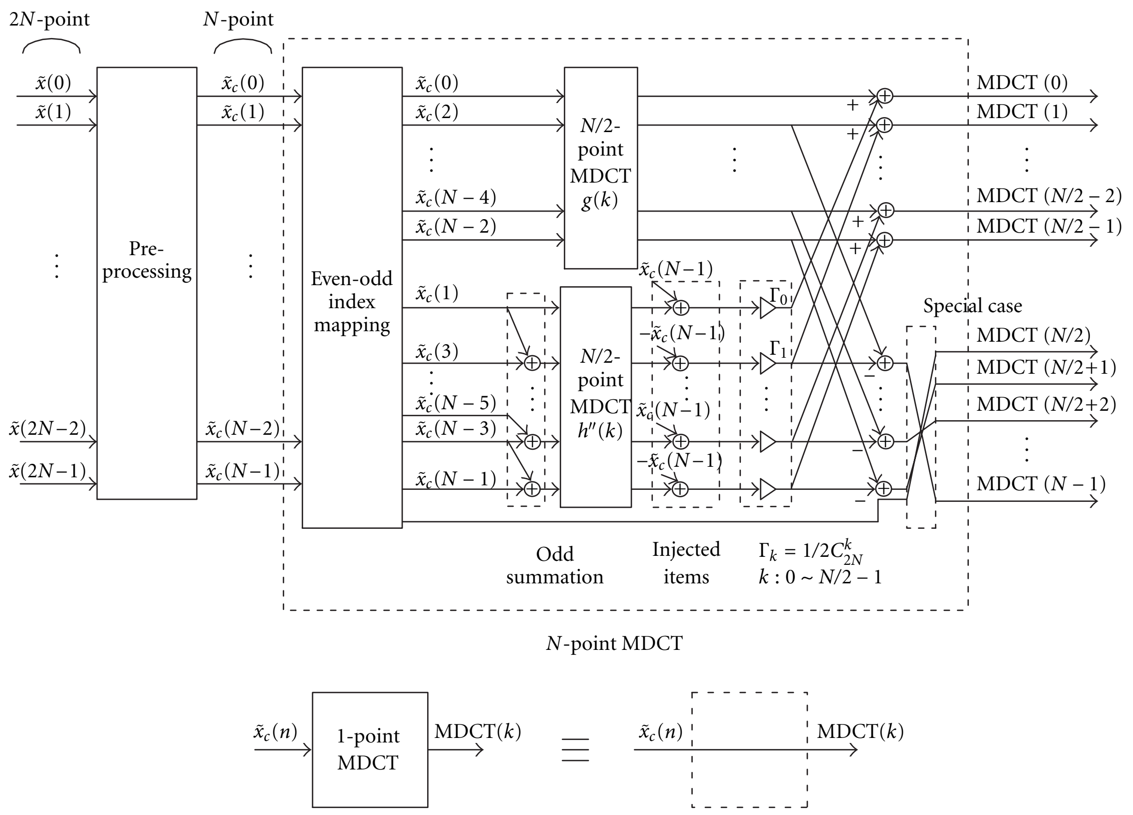


FIGURE 7: N-point MDCT(k) butterfly structure, where 1-point MDCT is the minimum-sized processing block of the FFT module.

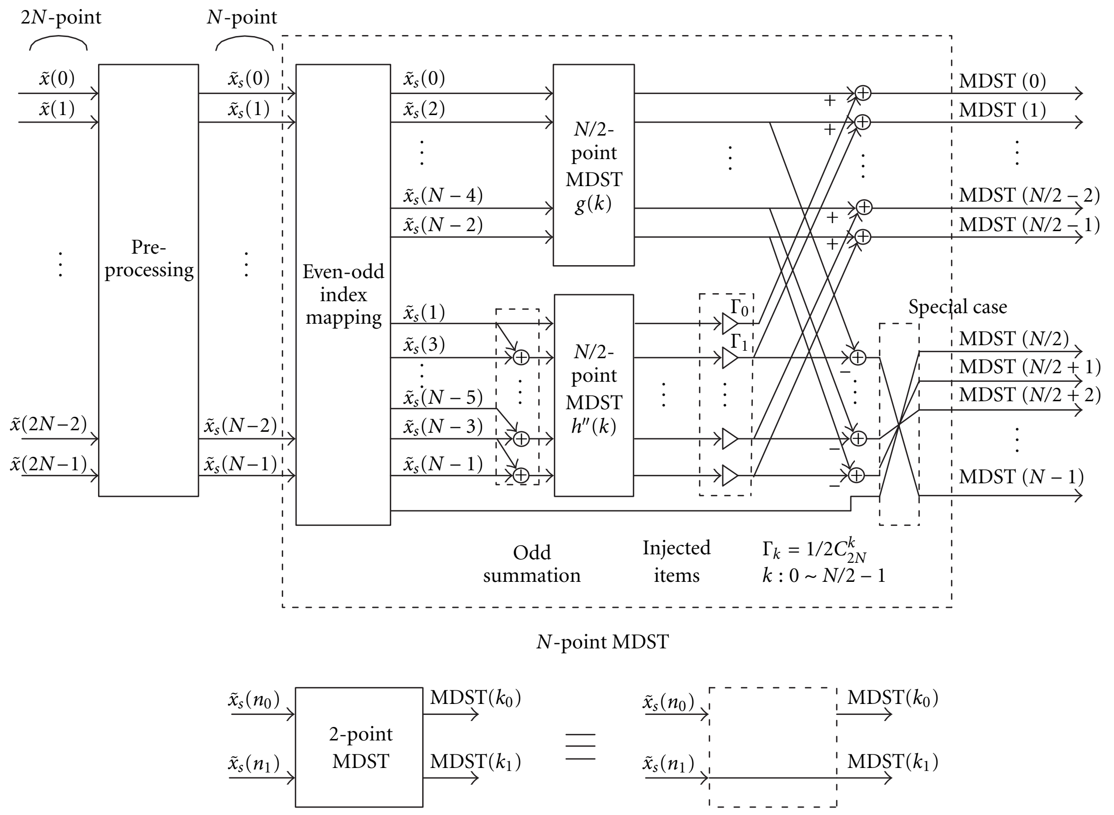


FIGURE 8: N-point MDST(k) butterfly structure, where the 2-point MDST is the minimum-sized processing block of the FFT module.

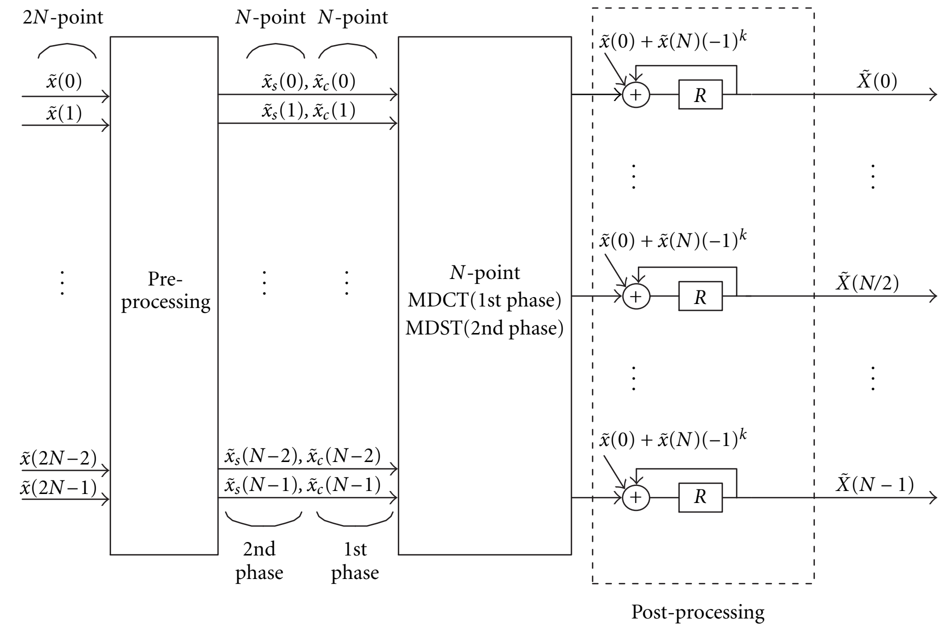

2N-point
N-point N-point
$\tilde{x}(0)$
$\tilde{x}(1)$
$\tilde{x}(2N-2)$
$\tilde{x}(2N-1)$
Pre-processing
$\tilde{x}_s(0), \tilde{x}_c(0)$
$\tilde{x}_s(1), \tilde{x}_c(1)$
$\tilde{x}_s(N-2), \tilde{x}_c(N-2)$
$\tilde{x}_s(N-1), \tilde{x}_c(N-1)$
2nd phase
1st phase
N-point MDCT(1st phase) MDST(2nd phase)
$\tilde{x}(0)+\tilde{x}(N)(-1)^k$
R
$\tilde{X}(0)$
$\tilde{X}(N/2)$
$\tilde{X}(N-1)$
Post-processing

FIGURE 9: The proposed FFT architecture.

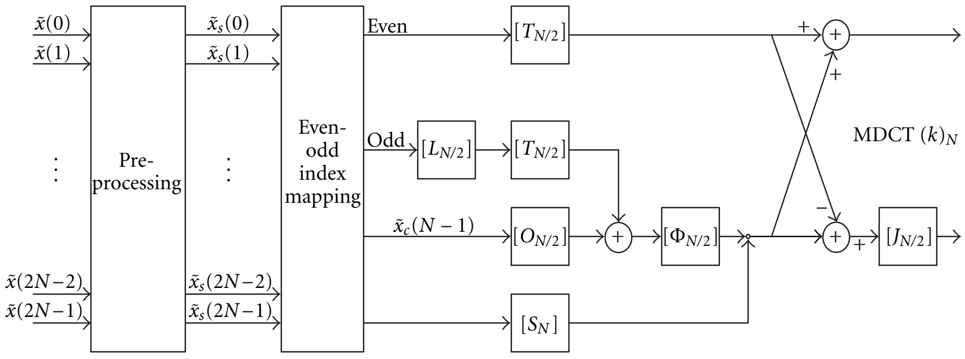


FIGURE 10: Block diagram of the MDCT in matrix form for the FFT operation.

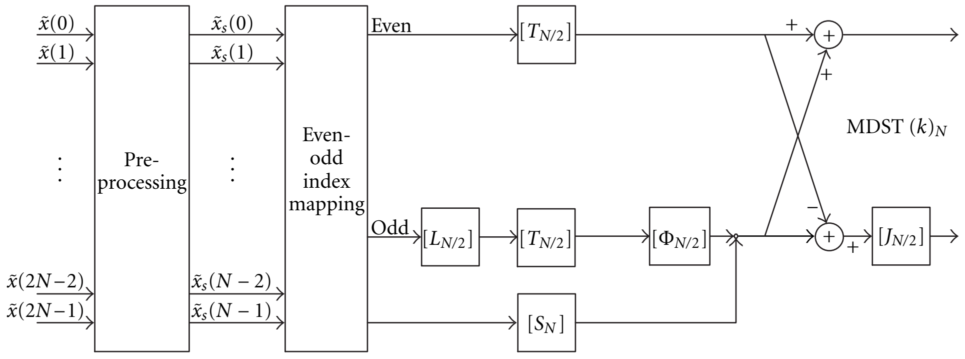


FIGURE 11: Block diagram of the MDST in matrix form for the FFT operation.

algorithm. The corresponding butterfly architecture requires $\log_2(2N)$ stages in the $2N$-point IFFT/FFT. Each stage consists of N multiplications and $2N$ additions. Because input sequences are complex data, the IFFT/FFT kernels are complex in nature. Hence, it requires 4 real-valued multiplications and 2 real-valued additions for 1 complex

TABLE 1: Comparison of computational complexity for $2N$-point IFFT/FFT.

	IFFT			FFT		
	Cooly-Tukey [10] (O_1)	Chan et al. (O_2)	CR	Cooly-Tukey [10] (O_1)	Chan et al. (O_2)	CR
N	$4N\log_2 2N$	$N\log_2 N - 2N + 2$		$4N\log_2 2N$	$N\log_2 N - 2N + 2$	
256	9216	1538	0.169	9216	1538	0.169
512	20480	3586	0.175	20480	3586	0.175
1024	45056	8194	0.182	45056	8194	0.182
2048	98304	18434	0.188	98304	18434	0.186
4096	212992	40962	0.192	212992	40962	0.192
8192	458752	90114	0.196	458752	90114	0.196

(a) Number of multiplication operations.

	IFFT			FFT		
	Cooly-Tukey [10] (O_1)	Chan et al. (O_2)	CR	Cooly-Tukey [10] (O_1)	Chan et al. (O_2)	CR
N	$6N\log_2 2N$	$(9/2)N\log_2 N + N + 1$		$6N\log_2 2N$	$(9/2)N\log_2 N + N$	
256	13824	9473	0.685	13824	9472	0.685
512	30720	21249	0.692	30720	21248	0.692
1024	67584	47105	0.697	67584	47104	0.697
2048	147456	103425	0.701	147456	103424	0.701
4096	319488	225281	0.705	319488	225281	0.705
8192	688128	487425	0.708	688128	487424	0.708

(b) Number of addition operations.

multiplication. Also, it takes 2 real additions to realize a complex addition. As a result, the direct approach requires a total of $4N\log_2(2N)$ real multiplications and $6N\log_2(2N)$ real additions. The large computation complexity are not suitable for cost-effective realization of the IFFT/FFT modules in the DMT system.

The complexity comparison for $2N$-point IFFT/FFT are listed in Table 1. The *complexity ratio (*CR*)* is defined as

$$\mathrm{CR} \triangleq \frac{O_2}{O_1}, \tag{44}$$

where O_1 and O_2 are the number of multiplications (or additions) in other fast algorithms and our approach, respectively. We can see that the complexity ratio of the multiplication is only 17% for $N = 256$ compared with conventional IFFT/FFT. Table 1 also shows that our approach can gain more computation savings as N gets larger in the VDSL systems [14].

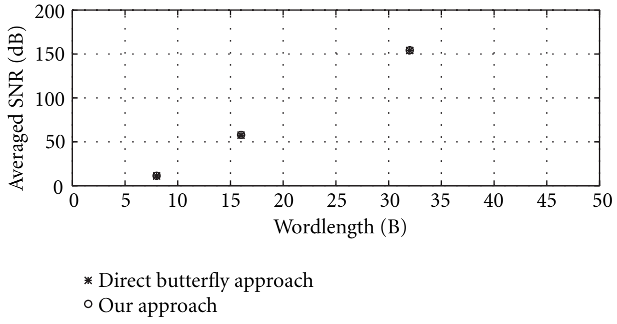
4.2. *Experiment results*

There are lots of DSP processors on the market. Due to the variety or hardware structure, coding styles, compliers, and so forth, we are not trying to do the detail optimization for specific processors. On the other hand, we would like to compare the proposed algorithm with Cooly-Tukey's algorithm, which is a baseline of the FFT realization. The implementation platform is TI TMS320C54 evaluation board, http://www.ti.com. Both algorithms are written in C language without any assembly-level programming tricks. During compilation, the TI C54X C complier is used without adding special compilation options, neither.

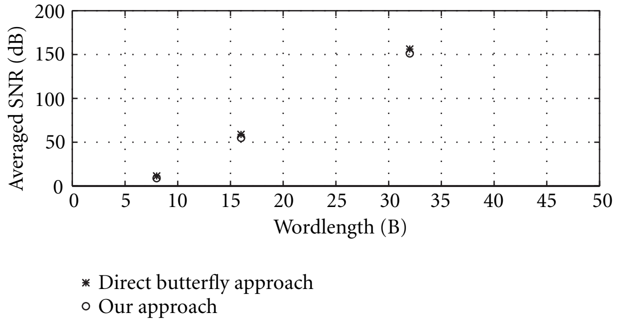
Table 2 shows the comparison of the proposed algorithm and the conventional FFT in terms of clock cycles. As we can see, the proposed algorithm requires only about 30% clock cycles of the Cooly-Tukey's. The result is very consistent with our observation in Table 1.

TABLE 2: Comparison of clock cycle for Cooley-Tukey FFT and proposed recursive algorithm.

	128-point	256-point	512-point
Cooley-Tukey FFT	16,485	37,118	82,347
Proposed	11,869	25,726	55,435
Clock cycle Ratio	28%	31%	33%



(a)



(b)

FIGURE 12: Averaged SNR versus wordlength for the 512-point ($2N$ value) (a) IFFT. (b) FFT.

4.3. *Finite-precision effect*

In fixed-point implementation of the IFFT/FFT kernels, it is important to consider the effects of finite register length in the IFFT/FFT calculations (see [12, Chapter 9] and [15]). To compare the butterfly approach and our approach in fixed-point implementation, we conduct extensive computer simulation by using MATLAB for finite-wordlength IFFT/FFT architecture. Figure 12 shows the SNR performance with assigned wordlength $B = 8, 16, 32$ bits. We observe that the SNR performance with $B = 16$ bits is good enough in practical fixed-point implementations. From the simulation results, we can see that the SNR performance of our approach is comparable to the traditional butterfly approach under the same wordlength.

5. CONCLUSIONS

In this paper, we develop a computationally efficient fast algorithm for the software implementation of the IFFT/FFT kernel in the DMT system. We reformulate the IFFT/FFT functions so as to avoid complex-domain operations. The complexity ratio of the multiplications is only 17% compared with the direct butterfly implementation approach. The proposed algorithm provides a good solution in reducing MIPS count in programmable DSP implementation for the applications of the DMT transceiver systems.

APPENDICES

A. DERIVATION OF (4)

Decomposing (4) into the first half and second half with the fact that $X(0) = X(N) = 0$, (4) can be represented as

$$x(n) = \frac{1}{2N} \cdot \left[\sum_{k=1}^{N-1} X(k) W_{2N}^{-nk} + \sum_{k=N+1}^{2N-1} X(k) W_{2N}^{-nk} \right]. \quad (\text{A.1})$$

Use $k' = 2N - k$ to replace the variable in the second term. Then, we have

$$x(n) = \frac{1}{2N} \cdot \left[\sum_{k=1}^{N-1} X(k) W_{2N}^{-nk} + \sum_{k'=N-1}^{1} X(2N-k') W_{2N}^{-(2N-k')n} \right]. \quad (\text{A.2})$$

Because k' is a dummy variable, we can rewrite (A.2) as

$$\begin{aligned} x(n) &= \frac{1}{2N} \cdot \left[\sum_{k=1}^{N-1} X(k) W_{2N}^{-nk} + \sum_{k=1}^{N-1} X(2N-k) W_{2N}^{-(2N-k)n} \right] \\ &= \frac{1}{2N} \cdot \left[\sum_{k=1}^{N-1} X(k) W_{2N}^{-nk} + \sum_{k=1}^{N-1} X(2N-k) W_{2N}^{-2Nn} W_{2N}^{nk} \right]. \end{aligned} \quad (\text{A.3})$$

By using the facts that

$$\begin{aligned} W_{2N}^{2Nn} &= 1, \\ W_{2N}^{nk} &= \exp\left(-j\frac{2\pi nk}{2N}\right) = \cos\frac{2\pi nk}{2N} - j\sin\frac{2\pi nk}{2N}, \\ W_{2N}^{-nk} &= \exp\left(j\frac{2\pi nk}{2N}\right) = \cos\frac{2\pi nk}{2N} + j\sin\frac{2\pi nk}{2N}, \\ X(0) &= X(N) = 0, \end{aligned} \quad (\text{A.4})$$

we can rearrange (A.3) to

$$x(n) = \frac{1}{N} \cdot \left[\sum_{k=0}^{N-1} \left(X_r(k) \cos\frac{2\pi nk}{2N} - X_i(k) \sin\frac{2\pi nk}{2N} \right) \right]. \quad (\text{A.5})$$

B. DERIVATION OF (30)

Equation (30) can be represented as

$$\tilde{X}(k) = \tilde{x}(0)+\tilde{x}(N)(-1)^k+\left[\sum_{n=1}^{N-1}\tilde{x}(n)W_{2N}^{nk}+\sum_{n=N+1}^{2N-1}\tilde{x}(n)W_{2N}^{nk}\right]. \tag{B.1}$$

Use $n' = 2N - n$ to replace the variable in the second term. Then, we have

$$\tilde{X}(k) = \tilde{x}(0)+\tilde{x}(N)(-1)^k + \left[\sum_{n=1}^{N-1}\tilde{x}(n)W_{2N}^{nk}+\sum_{n'=N-1}^{1}\tilde{x}(2N-n')W_{2N}^{k(2N-n')}\right]. \tag{B.2}$$

Because n' is a dummy variable, we can rewrite (B.2) as

$$\begin{aligned}\tilde{X}(k) &= \tilde{x}(0)+\tilde{x}(N)(-1)^k + \left[\sum_{n=1}^{N-1}\tilde{x}(n)W_{2N}^{nk}+\sum_{n=1}^{N-1}\tilde{x}(2N-n)W_{2N}^{k(2N-n)}\right]\\ &= \tilde{x}(0)+\tilde{x}(N)(-1)^k + \left[\sum_{n=1}^{N-1}\tilde{x}(n)W_{2N}^{nk}+\sum_{n=1}^{N-1}\tilde{x}(2N-n)W_{2N}^{2kN}W_{2N}^{-nk}\right].\end{aligned} \tag{B.3}$$

By using the fact that $W_{2N}^{2kN} = 1$ and applying the assumption of the input data in (35), we can rearrange (B.3) as

$$\tilde{X}(k) = \tilde{x}(0)+\tilde{x}(N)(-1)^k + 2\left[\sum_{n=1}^{N-1}\tilde{x}_c(n)\cos\frac{2\pi nk}{2N} - j\sum_{n=1}^{N-1}\tilde{x}_s(n)\sin\frac{2\pi nk}{2N}\right]. \tag{B.4}$$

ACKNOWLEDGMENT

T. S. Chan is with the VXIS Tech. Corp. Hsin-Chu, Taiwan, ROC. This work is supported in part by the National Science Council, ROC, under Grant NSC 87-2213-E-008-011.

REFERENCES

[1] G. H. Im, D. D. Harman, G. Huang, A. V. Mandzik, M. H. Nguyen, and J. J. Werner, "51.84 Mb/s 16-CAP ATM LAN standard," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, pp. 620–632, 1995.

[2] J. S. Chow, J. C. Tu, and J. M. Cioffi, "A discrete multitone transceiver system for HDSL applications," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 6, pp. 895–908, 1991.

[3] K. Sistanizadeh, P. Chow, and J. M. Cioffi, "Multi-tone transmission for asymmetric digital subscriber lines (ADSL)," in *Proc. IEEE International Conf. on Communications*, vol. 2, pp. 756–760, Geneva, Switzerland, 1993.

[4] I. Lee, J. S. Chou, and J. M. Cioffi, "Performance evaluation of a fast computation algorithm for the DMT in high-speed subscriber loop," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 9, pp. 1560–1570, 1995.

[5] T. N. Zogakis, J. T. Aslanis Jr., and J. M. Cioffi, "A coded and shaped discrete multitone system," *IEEE Trans. Communications*, vol. 43, no. 12, pp. 2941–2949, 1995.

[6] B. Daneshrad and H. Samueli, "A 1.6 Mbps digital-QAM system for DSL transmission," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 9, pp. 1600–1610, 1995.

[7] B. R. Wiese and J. S. Chow, "Programmable implementations of xDSL transceiver systems," *IEEE Communications Magazine*, vol. 38, no. 5, pp. 114–119, 2000.

[8] A.-Y. Wu and T. S. Chan, "Cost-efficient parallel lattice VLSI architecture for the IFFT/FFT in DMT transceiver technology," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 3517–3520, Seattle, Wash, USA, May 1998.

[9] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1243–1245, 1984.

[10] J. W. Cooly and J. W. Tukey, "An algorithm for the machine calculation of the complex Fourier series," *Math. Comp.*, vol. 19, pp. 297–301, April 1965.

[11] ANSI Standard T1.413, "Network and customer installation interface-Asymmetric digital subscriber line (ADSL) metallic interface," 1995.

[12] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.

[13] H. D. Yun and S. U. Lee, "On the fixed-point-error analysis of several fast DCT algorithms," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 3, no. 1, pp. 27–41, 1993.

[14] T1E1.4/2000-013R3, "Very-high-speed digital subscriber lines (VDSL) metallic interface, part 3: Technical specification of a multi-carrier modulation transceiver," 2000.

[15] K. J. R. Liu, A.-Y. Wu, A. Raghupathy, and J. Chen, "Algorithm-based low-power and high-performance multimedia signal processing," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1155–1202, 1998, Special Issue on Multimedia Signal Processing.

Tsun-Shan Chan was born in Chang-Hui, Taiwan, ROC, in 1973. He received his M.S. degree in electrical engineering from the National Central University, Taiwan, in 1998. During 1998–1999, he worked on communication applications in Industrial Technology Research Institute, Hsin-Chu, Taiwan. Since 1999, he has been serving as a system engineer of the video processing projects in VXIS Technology Corporation.

Jen-Chih Kuo received his B.S. degree in electrical engineering from the Nation Taiwan University, Taiwan, in 2000. He is now in the Graduate Institute of Electronics Engineering of the same school. His research interests include VLSI architectures for DSP algorithms, adaptive signal processing, and digital communication systems.

An-Yeu (Andy) Wu received his B.S. degree from National Taiwan University in 1987, and the M.S. and Ph.D. degrees from the University of Maryland, College Park in 1992 and 1995, respectively, all in electrical engineering. During 1987–1989, he served as a signal officer in the Army, Taipei, Taiwan, for his mandatory military service. During 1990–1995, he was a graduate teaching and research assistant with the Department of Electrical Engineering and Institute for Systems Research at the University of Maryland, College Park. From August 1995 to July 1996, he was a Member of Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, working on high-speed transmission IC designs. From 1996 to July 2000, he was with the Electrical Engineering Department of National Central University, Taiwan. He is currently an Associate Professor with the Department of Electrical Engineering Department and Graduate Institute of Electronics Engineering of National Taiwan University, Taiwan. His research interests include low-power/high-performance VLSI architectures for DSP and communication applications, adaptive signal processing, and multirate signal processing.