A FAST ALGORITHM FOR REDUCED-COMPLEXITY PROGRAMMABLE DSP IMPLEMENTATION OF THE IFFT/FFT IN DMT SYSTEMS

An-Yeu Wu  Tsan-Shan Chen  Bowen Wang*

Department of Electrical Engineering
National Central University, Chung-Li, 32004, Taiwan, ROC
*200 CCL/TTRI Chutung, Hsinchu, Taiwan 310, ROC

Abstract - The discrete multitone (DMT) modulation/demodulation scheme is the standard transmission technique in the application of asymmetric digital subscriber lines (ADSL). 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 S12-point IFFT/FFT as the modulation/demodulation kernel. The large block size results in heavy computational load in running programmable digital signal processors (DSPs). In this paper, we derive computationally efficient fast algorithms for the IFFT/FFT. The proposed algorithm requires minimum number of multiplication operations compared with existing approaches. Also, it can avoid complex-domain operations that are inevitable in conventional IFFT/FFT computation. The resulting software function requires less MIPS count. Hence, it is very suitable for DSP-based DMT implementation. The proposed algorithm can also be applied to the technology of Orthogonal Frequency Division Multiplexing (OFDM) which is the processing kernel of the Digital Audio/Video Broadcasting (DAB/DVB) systems.

1. INTRODUCTION

Recent progress of Internet access has a strong demand on high-speed data transmission. Among various transmission techniques, the DMT can achieve rate-adaptive data transmission by employing several advanced DSP techniques such as dynamic bit allocation, multi-dimensional tone encoding, frequency-domain equalization, etc. As a consequence, the DMT has been chosen as the physical-layer transmission standard by the ADSL standardization committee [1] [2] [3].

One major disadvantage of the DMT scheme is its high computational complexity. In particular, the large block size of the IFFT/FFT (S12-point) consumes lots of computing power in running programmable DSPs. In [4], 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 modules in DSPs. By making use of the symmetric/anti-symmetric 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 [5] to derive novel recursive algorithms and modified butterfly structures for the modified DCT and DST.

The new scheme can avoid redundant complex-domain operations of the IFFT/FFT, and requires only $O(N \log_2 N)$ computational complexity. Since only real-valued operations are used, it can avoid the special data structure to run complex-domain additions/multiplications. Our analysis shows that we need only 17% multiplications compared with conventional butterfly implementation of the IFFT/FFT [6] [7, Chap. 9]. The low computational complexity as well as real-domain operations makes it very suitable for firmware coding in DSPs, and helps to reduce the million-instructions-per-second (MIPS) count. Also, the DSP program can be written in recursive form and apply the technique of in-place computation to save the ROM/RAM storage space.

![Figure 1: The IFFT/FFT block diagram in the DMT system.](image)

2. IFFT MODULE

At the transmitter side of the DMT system (see Fig. 1(a)), to ensure the IFFT generates only real-valued outputs, the inputs of the IFFT have the constraint [2]

$$x(n) = x^*(2N - n), \text{ for } n = 0, 1, ..., N - 1,$$

(1)
where \( N = 256 \) and \( x(n) \triangleq x_r(n) + jx_i(n) \) are the input encoded complex symbols with \( x(0) = x(N) = 0 \). In [4], we made use of the conjugate-symmetric property in (1), and simplified the IFFT as

\[
X(k) = 2 \sum_{n=0}^{N-1} x_r(n) \cos \frac{2\pi nk}{2N} + \sum_{n=0}^{N-1} x_i(n) \sin \frac{2\pi nk}{2N} \]
\[
= 2 \{ MDC\!(T)(k) + M\!D\!S\!T(k) \},
\]  
(2)

for \( k = 0, 1, \ldots, 2N - 1 \). From (2), we can see that the computation of the IFFT is decomposed into two real-value transform operations, the \( \text{modified DCT (MDCT)} \) and the \( \text{modified DST (MDST)} \). Furthermore, it can be shown that

\[
\text{MDCT}(k) = \text{MDCT}(2N - k),
\]
\[
\text{MDST}(k) = -\text{MDST}(2N - k),
\]

for \( k = 0, 1, \ldots, 2N - 1 \). The symmetric/anti-symmetric properties can save an additional 50\% computational complexity.

For the special case of \( k = N \), the \( \text{MDCT} \) and \( \text{MDST} \) can be simplified as

\[
\text{MDCT}(N) = \sum_{n=0}^{N-1} x_r(n)(-1)^n, \quad \text{MDST}(N) = 0.
\]

The implementation issue of the IFFT is to exploit the fast algorithms to realize the \( \text{MDCT} \) and \( \text{MDST} \) in a cost-efficient way. Then, we can just combine the results of the \( \text{MDCT} \) and \( \text{MDST} \) to obtain the IFFT results based on (2).

2.1. Modified butterfly structure of the IFFT

Define \( C_{2N}^{k} \triangleq \cos \frac{2\pi nk}{2N} \). By following the derivation in [5], it can be shown that the \( \text{MDCT}(k) \) can be written as

\[
\text{MDCT}(k) = \sum_{n=0}^{N/2-1} x_r(n)C_{2N}^{kn} = \sum_{n=0}^{N/2-1} x_r(n)C_{2N}^{kn}
\]
\[
+ \frac{1}{2C_{2N}^{k}} \left( \sum_{n=0}^{N/2-1} [x_r(2n + 1) + x_r(2n - 1)]C_{2N}^{k}\right)
\]
\[
+ x_r(N - 1)(-1)^k
\]

and

\[
\text{MDCT}(N - k) = \sum_{n=0}^{N/2-1} x_r(n)C_{2N}^{kn}
\]
\[
+ \frac{1}{2C_{2N}^{k}} \left( \sum_{n=0}^{N/2-1} [x_r(2n + 1) + x_r(2n - 1)]C_{2N}^{k}\right)
\]
\[
+ x_r(N - 1)(-1)^k.
\]

The special case \( \text{MDCT}(N/2) \) needs to be computed separately, and can be simplified as

\[
\text{MDCT}(N/2) = \sum_{n=0}^{N-1} x_r(n)C_{2N}^{nN/2} = \sum_{n=0}^{N-1} x_r(n) \cos \frac{n\pi}{2}.
\]

(8)

The mapping of (6) is shown in Fig. 2(a). As we can see, the \( N \)-point \( \text{MDCT} \) is decomposed into two \( N/2 \)-point \( \text{MDCT} \) (\( g(k) \) and \( h(k) \)) plus the pre-processing and post-processing modules. We can apply the technique of "divide-and-conquer" to recursively expand the \( N/2 \)-point \( \text{MDCT} \) until 1-point \( \text{MDCT} \) (see Fig. 2(b)) is formed. Then we can obtain a modified butterfly structure for the \( \text{MDCT} \).

Similarly, defining \( S_{2N}^{kn} \triangleq \sin \frac{2\pi nk}{2N} \), we can derive the \( \text{MDST}(k) \)

\[
\text{MDST}(k) = \sum_{n=0}^{N/2-1} x_r(n)S_{2N}^{kn}
\]
\[
+ \frac{1}{2S_{2N}^{k}} \sum_{n=0}^{N/2-1} [x_r(2n + 1) + x_r(2n - 1)]S_{2N}^{k},
\]

(9)

and

\[
\text{MDST}(N - k) = -\sum_{n=0}^{N/2-1} x_r(n)S_{2N}^{kn}
\]
\[
+ \frac{1}{2S_{2N}^{k}} \sum_{n=0}^{N/2-1} [x_r(2n + 1) + x_r(2n - 1)]S_{2N}^{k},
\]

(10)

for \( k = 0, 1, \ldots, N/2 - 1 \). It is worth noting that the injected item is zero in the \( \text{MDST} \). Besides, the \( \text{MDST} \) also has a special case for index \( N/2 \)

\[
\text{MDST}(N/2) = \sum_{n=0}^{N-1} x_r(n)S_{2N}^{nN/2} = \sum_{n=0}^{N-1} x_r(n) \sin \frac{n\pi}{2}.
\]

(11)

The mapping of the \( \text{MDST} \) structure is similar to the \( \text{MDCT} \) structure in Fig. 2 except that minimum processing block is \( 2 \)-point \( \text{MDST} \), and the injected items do not exist in the \( \text{MDST}(k) \) implementation.

The overall IFFT structure is shown in Fig. 3. It consists of the \( \text{MDCT/MDST} \) structure and a post-processing module. The post-processing module expands the \( N \)-point \( \text{MDCT/MDST} \) outputs to the \( 2N \)-point \( \text{MDCT/MDST} \) by using the symmetric/anti-symmetric properties in Eqs. (3) and (4), respectively. Based on (2), we combine the \( \text{MDCT} \) and \( \text{MDST} \) results, together with the scaling operation (which is achieved by shifting left by 1 bit), to obtain the IFFT results.

2.2. Matrix Notation of the \( \text{MDCT/MDST} \)

By following the notations in [8], we denote \( x_r(n), n = 0, 1, \ldots, N - 1, \) and \( \text{MDCT}(k), k = 0, 1, \ldots, N - 1 \) as

\[
[x_r(0), x_r(1), \ldots, x_r(N - 1)]^T.
\]

\[
\text{MDCT}(k) \triangleq [x_r(0) \ x_r(1) \ \cdots \ x_r(N - 1)]^T.
\]

(13)

In coding the C90-based C programs, we expand the recursive structure to 8-point \( \text{MDCT} \) to save the recursive call depth.
where $[A_{N/2}]$ is defined as

$$[A_{N/2}] = \begin{bmatrix} 1 & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{N/2 \times N/2}$$

(18)

Note that the special case of the $MDCT$ can be represented as

$$MDCT(N/2) = [S_N][x_r(n)N_2],$$

(19)

where

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

(20)

Based on Eq. (6) (7) and above matrix definition, $[T_N]$ can be expressed in the matrix form as

$$[T_N] = \begin{bmatrix} [T_{N/2}] & [S_{N/2}] ([T_{N/2}] [S_{N/2}] + [S_{N/2}] [T_{N/2}] + [S_{N/2}]) \\ \vdots & \vdots & \ddots & \vdots \\ [T_{N/2}] & [S_{N/2}] \end{bmatrix}_{N/2 \times N/2}$$

(21)

where $[J_{N/2}]$ denotes the anti-diagonal identity matrix which is used to reorder the second half outputs. We can also represent Eqs. (14) and (21) using the matrix form as shown in Fig. 4. Similarly, the matrix notation for transform kernel of the $MDST$ can be represented as

$$[T_N] = \begin{bmatrix} [T_{N/2}] & [S_{N/2}] [T_{N/2}] [S_{N/2}] [T_{N/2}] [S_{N/2}] \end{bmatrix}_{N/2 \times N/2}$$

(22)

Note that the $MDST$ is similar to the $MDCT$ expect that there is no injected items. Also, the special case matrix has to be modified as

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

(23)

The block diagram of the $MDST$ is shown in Fig. 5.
3. FFT MODULE

At the receiver side, the 512-point FFT is used to demodulate the received signals, which is given by

$$\tilde{x}(n) = \frac{1}{2N} \sum_{k=0}^{2N-1} \hat{X}(k)W_{2N}^{-nk}, \quad n = 0, 1, ..., 2N - 1, \quad (24)$$

where $\tilde{x}(k), k = 0, 1, ..., 2N - 1,$ are real-valued received data. Note that in the DMT system, the lower $N$-point FFT outputs are conjugate-symmetric to the upper $N$-point outputs (see Fig. 1(b)). Hence, we can neglect the outputs $\tilde{x}(n), \ n = N, N + 1, ..., 2N - 1,$ which can save 50\% of the complexity.

To have an efficient computation, we first decompose input sequence into a symmetric sequence, $\hat{X}_s(k),$ plus an anti-symmetric sequence, $\hat{X}_a(k),$ where

$$\hat{X}_s(k) = \frac{1}{2} [\hat{X}(k) + \hat{X}(2N - k)], \quad (25)$$

$$\hat{X}_a(k) = \frac{1}{2} [\hat{X}(k) - \hat{X}(2N - k)], \quad (26)$$

for $k = 1, 2, ..., N - 1.$ Hence, we have

$$\hat{X}(k) = \hat{X}_s(k) + \hat{X}_a(k), \quad (27)$$

$$\hat{X}(2N - k) = \hat{X}_s(k) - \hat{X}_a(k), \quad (28)$$

for $k = 1, 2, ..., N - 1.$ By substituting (27)-(28) into (24), we can simplify (24) as

$$\tilde{x}(n) = \frac{1}{2N} (\tilde{X}(0) + \tilde{X}(N) (-1)^n + 2 \sum_{k=0}^{N-1} \hat{X}_s(k) \cos \frac{2\pi nk}{2N} + j \sum_{k=0}^{N-1} \hat{X}_a(k) \sin \frac{2\pi nk}{2N}) = \frac{1}{2N} \{\tilde{X}(0) + \tilde{X}(N) (-1)^n \} + j \sum_{k=0}^{N-1} \hat{X}_a(k) \sin \frac{2\pi nk}{2N} \quad (29)$$

for $n = 0, 1, ..., N - 1,$ where $\tilde{X}_s(0) = 0$ and $\tilde{X}_a(0) = 0.$

As with the derivations in Section 2, we can derive the $MDCT(n)/MDST(n)$ as

$$MDCT(n) = \sum_{k=0}^{N/2-1} \tilde{X}_s(2k) C_N^{n2k} \quad (29)$$

$$+ \frac{1}{2C_N^2} \sum_{k=0}^{N/2-1} [\tilde{X}_s(2k + 1) + \tilde{X}_s(2k - 1)] C_N^{n2k} \quad (30)$$

$$+ \tilde{X}_s(N - 1) (-1)^n \quad (31)$$

$$+ \tilde{X}_a(n) \quad (32)$$

$$+ \tilde{X}_a(n) \quad (33)$$

$$+ \tilde{X}_a(n) \quad (34)$$

$$+ \tilde{X}_a(n) \quad (35)$$

The mapping of the $MDCT(n)/MDST(n)$ structure is similar to the $MDCT(k)/MDST(k)$ structure, respectively. Then we just combine the $MDCT(n)$ and $MDST(n)$ outputs, followed by adding the $\tilde{X}(0)$ and $\tilde{X}(N) (-1)^n,$ to obtain the FFT results based on (29). The overall structure of the FFT is shown in Fig. 6. Note that the pre-processing
module is used to decompose input sequence into a symmetric sequence, $X_s(k)$, plus an anti-symmetric sequence, $X_a(k)$.

The matrix-representation of the $MDCT(n)$ and $MDST(n)$ are very similar to the $MDCT(k)$ and $MDST(k)$ in the previous section. The difference is that it requires a pre-processing module to compute the $X_s(k)$ and $X_a(k)$. The block diagrams of the $MDCT$ and $MDST$ are shown in Figs. 7 and 8, respectively.

To compare the SNR performance of the proposed recursive algorithm with the conventional butterfly approach in running fixed-point DSPs, we conduct extensive computer simulation for finite-wordlength 512-point IFFT/FFT structure. Figure 9 shows the averaged SNR of 200 random inputs for each channel of the IFFT. We can see that the SNR performance of our approach is comparable to the conventional butterfly approach. Figure 10 shows the SNR performance with assigned wordlength $B = 8, 16, 32$ bits. We can also observe that the SNR performance with $B = 16$ bits is good enough in practical fixed-point implementations.
5. DSP IMPLEMENTATION

To verify the effectiveness of the proposed fast algorithm, we implement the IFFT/FFT algorithm on TI TMS320C30 evaluation board (EVM). We write the algorithm in C language in recursive form. That is, the program will call the MDCF and MDST functions recursively until the 8-point MDCT/MDST are formed. To make a fair comparison with the conventional FFT/IFFT algorithm, we use the FFT program in [14, Chap. 4] as the test bench. The C program in [14, Chap. 4] employs the Cooley-Tukey algorithm to perform the general-purpose FFT function in the complex domain. We compile both C programs using TI C30 C compiler under the same working environment. No assembly-level tricks are played. Hence, we can see clearly how the algorithmic-level simplification can reduce the clock cycles.

Table 2 shows the comparison between the proposed algorithm and the conventional FFT in terms of clock cycles. As we can see, for the 512-point FFT, the recursive program requires only 36% clock cycles compared with the Cooley-Tukey program. The results are consistent with our observation in Table 1. The saving in computational complexity is also significant for 128 and 256 point FFT.

<table>
<thead>
<tr>
<th></th>
<th>128-point</th>
<th>256-point</th>
<th>512-point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cooley-Tukey FFT[14]</td>
<td>516729</td>
<td>1108114</td>
<td>2367335</td>
</tr>
<tr>
<td>Ours (Recursive form)</td>
<td>155333</td>
<td>387078</td>
<td>866548</td>
</tr>
<tr>
<td>Clock cycle ratio</td>
<td>30%</td>
<td>34%</td>
<td>36%</td>
</tr>
</tbody>
</table>

Table 2: Comparison of clock cycles for Cooley-Tukey FFT [14] and the proposed recursive algorithm.

6. CONCLUSIONS

In this paper, we develop a computationally efficient fast algorithm for the implementation of the IFFT/FFT kernel in the DMT system. The proposed algorithm provides a good solution in reducing MIPS count in running DSPs for the DMT transceiver systems. Furthermore, the problem of larger block size \( N \) needs to be handled in the OFDM-based DAB/DVB systems [12][13]. Table 1 shows that, compared with the conventional butterfly implementation, our approach can gain more complexity savings as \( N \) increases. Hence, our approach provides very cost-efficient solution for the OFDM-based applications.

Acknowledgement

The authors would like to thank Mr. Jung-Yu Yeh for coding the C30-based C programs.

References