A NOVEL MEMORY-BASED FFT PROCESSOR FOR DMT/OFDM APPLICATIONS

Ching-Hsien Chang, Chin-Liang Wang, and Yu-Tai Chang

Department of Electrical Engineering, National Tsing Hua University
Hsinchu, Taiwan 300, Republic of China

Email: clwang@ee.nthu.edu.tw

ABSTRACT

This paper presents a novel VLSI architecture for computing the N-point discrete Fourier transform (DFT) based on a radix-2 fast algorithm, where N is a power of two. The architecture consists of one complex multiplier, two complex adders, and some special memory units. It can compute one transform sample every \( \log_2 N + 1 \) clock cycles in average. For the case of \( N = 512 \), the chip area required is about 5742x5222 \( \mu \)m\(^2\) and the throughput is up to 4M transform samples per second under 0.6 \( \mu \)m CMOS technology. Such area-time performance makes the proposed design rather attractive for use in long-length DFT applications, such as ADSL and OFDM systems.

1. INTRODUCTION

The discrete Fourier transform (DFT) is an important tool in the area of digital signal processing. Among many possible DFT applications, multichannel modulation [1] has shown to be an effective way to achieve reliable, efficient data transmission. There are two forms of multicarrier modulation that have received great attention in the communications industry. One is called discrete multitone (DMT) modulation [2], and the other is called orthogonal frequency division multiplexing (OFDM) [3]. DMT technology has been selected as standards for asymmetric digital subscriber lines (ADSL) service on ordinary phone lines [4], [5], while OFDM technology has been extensively investigated for broadcast applications and wireless communications [6], [7]. Both ADSL and OFDM applications involve long-length DFT computation, where the transform length is up to 512 or more. Since such long-length DFT computation is rather time-consuming, special fast Fourier transform (FFT) processors are necessary to meet the real-time requirements. A number of previous FFT architectures could be considered for this purpose (see, for example, [8]-[11]). However, most of them are not suitable for single-chip implementation due to their use of a lot of complex multipliers.

In this paper, we propose a new FFT processor for ADSL/OFDM applications. The proposed processor realizes a radix-2 FFT algorithm by using only a complex multiplier, two complex adders, one ROM, and some special memory-based buffers. A prototype chip for 512-point DFT is given to demonstrate its usefulness.

2. AN ALGORITHM FOR FFT COMPUTATION

Consider the N-point DFT defined by

\[
y_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N}, \quad k=0,1,2,\ldots,N-1,
\]

where \( N \) is a power of two and \( W_N = e^{-j2\pi/N} \). Letting \( X=[x_0, x_1, \ldots, x_{N-1}]^T \) be the input vector and \( Y=[y_0, y_1, \ldots, y_{N-1}]^T \) be the output vector, we can rewrite (1) as follows:

\[
Y = W(N)X
\]

where

\[
W(N) = \begin{bmatrix}
1 & 1 & \cdots & 1 \\
1 & W_N^1 & \cdots & W_N^{N-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & W_N^{N-1} & \cdots & W_N^{N-1(N-1)} \\
\end{bmatrix}
\]

is called a transform matrix. With the property of \( W_N^{-1} = (-1)^{k\ell} \), we can partition the matrix \( W(N) \) into four quadrants by shifting all its even-numbered rows to the upper half portion. This is equivalent to multiplying both sides of (2) by a permutation matrix \( Q(N) = [e_0, e_2, e_4, \ldots, e_{N/2-1}, e_1, e_3, e_5, \ldots, e_{N-1}]^T \) with \( e_0 \) being a unit Nx1 column vector whose \((n+1\text{-th})\text{ element is 1}, i.e.,

\[
Q(N)Y = Q(N)W(N)X = \begin{bmatrix}
E(N/2) & F(N/2) \\
D(N/2) & D(N/2) \\
\end{bmatrix}X
\]

where \( E(N/2), D(N/2), \) and their relationship are given as follows:

\[
E(N/2) = \begin{bmatrix}
1 & 1 & 1 \\
1 & W_N^{2}\ell & W_N^{2(\ell+1)} & \cdots & W_N^{2(N/2-1)} \\
\vdots & \vdots & \ddots & \vdots \\
1 & W_N^{2(N/2-1)} & W_N^{2(2N/2-1)} & \cdots & W_N^{N/2(2N/2-1)} \\
\end{bmatrix}
\]

\[
D(N/2) = \begin{bmatrix}
1 & W_N^{\ell} & W_N^{2\ell} & \cdots & W_N^{\ell(N/2-1)} \\
1 & W_N^{\ell} & W_N^{2\ell} & \cdots & W_N^{\ell(N/2-1)} \\
\vdots & \vdots & \ddots & \vdots \\
1 & W_N^{\ell(N/2-1)} & W_N^{2\ell(N/2-1)} & \cdots & W_N^{N/2\ell(N/2-1)} \\
\end{bmatrix}
\]

\[
F(N/2) = \begin{bmatrix}
W_N^\ell & 0 & \cdots & 0 \\
0 & W_N^\ell & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & W_N^{N/2-1} \\
\end{bmatrix}
\]

Substituting (7) into (4) yields

\[
Q(N)Y = \begin{bmatrix}
E(N/2) & F(N/2) \\
D(N/2) & D(N/2) \\
\end{bmatrix}X = \begin{bmatrix}
E(N/2) & 0 \\
0 & E(N/2) \\
\end{bmatrix}P(N)X
\]

where

\[
P(N) = \begin{bmatrix}
I_{N/2} & I_{N/2} \\
F(N/2) & -F(N/2) \\
\end{bmatrix} = \begin{bmatrix}
I_{N/2} & 0 \\
0 & F(N/2) \\
I_{N/2} & -I_{N/2} \\
\end{bmatrix} \begin{bmatrix}
I_{N/2} & I_{N/2} \\
F(N/2) & -F(N/2) \\
\end{bmatrix} \begin{bmatrix}
I_{N/2} & 0 \\
0 & F(N/2) \\
I_{N/2} & -I_{N/2} \\
\end{bmatrix}
\]
Since $W_{c_{(N/2)}}=W_{N/2}$, we can observe from (3) and (5) that $E(N/2) = W(N/2)$. Thus, (9) can be rewritten as

$$Q(N)Y = \begin{bmatrix} W(N/2) & 0 \\ 0 & W(N/2) \end{bmatrix} P(N)X$$  \hspace{1cm} (11)

For ease of presentation, we define $W_{N/2}(M)$ as the direct sum of $W_{N/2}$ over $\mathbb{R}^M$.

$$W_{2M}(M) = W_1(M) \oplus W_2(M) \oplus \ldots \oplus W_M(M)$$

$$= \begin{bmatrix} W_1(M) & 0 & \ldots & 0 \\ 0 & W_1(M) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & W_M(M) \end{bmatrix}$$  \hspace{1cm} (12)

With similar direct-sum definitions for matrices $Q_{N/2}(M)$ and $P_{N/2}(M)$, we can express (11) as follows:

$$Q_{N/2}(N)Y = W_{N/2}[P_{N/2}(N)X]$$  \hspace{1cm} (13)

Note that (13) can be regarded as an $N$-point transform with transform matrix $W_{N/2}$, input vector $P_{N/2}(N)X$, and output vector $Q_{N/2}(N)Y$. Since $W_{N/2}(M) = W_{N/2}(M) \oplus W_{N/2}(M)$, this $N$-point transform can be partitioned into two $N/2$-point DFT’s with transform matrix $W_{N/2}$ each. Using the permutation matrix $Q_{N/2}(N) = [e_0 e_1 e_2 e_3 e_4 e_5 e_6 e_7]$, each $N/2$-point DFT (i.e., multiplying both sides of (13) by $Q_{N/2}(N) = Q_{N/2}(N)Q_{N/2}(2)$) and following the procedure of $Q_{N/2}(2)$, we can further decompose each $N/2$-point DFT into two $N/4$-point DFT’s with transform matrix $W_{N/4}$ each. This can be expressed as

$$Q_{N/2}(N/2)Q_{N/4}(4) = Q_{N/2}(N)Y = P_{N/2}(N/2)P_{N/4}(N)X$$  \hspace{1cm} (14)

where $P_{N/2}(N/2) = P_{N/2}(N/2)P_{N/4}(N/2)$. Repeating such a decomposition process until $W_{N/2}(M)$ is an identity matrix, we have

$$Q_{N/2}(N/2)Q_{N/4}(4) = \ldots = Q_{N/2}(N)Y = P_{N/2}(2)P_{N/4}(4) \ldots P_{N/2}(N)X$$  \hspace{1cm} (15)

This equation implies that we can compute the 1-D DFT by performing a series of matrix-vector multiplications.

It is interesting to see that performing the matrix-vector multiplication with $P_{N/2}(2)$ in (15) is equivalent to realizing the $(\log_2 M + 1)$th stage of the well-known radix-2 $\log_2 N$-input/decimation-in-frequency MTF algorithm [13] and the resulting output vector (denoted by $Z$) is the bit-reversed version of $Y$. That is,

$$Z = [y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7]^T$$

$$= P_{N/2}(2)P_{N/4}(4) \ldots P_{N/2}(N/2)P_{N/2}(N)X$$  \hspace{1cm} (16)

It should also be noted that the matrix formulation given here is similar to that given by Pease [14], with the difference that each of the coefficient matrices $(P_{N/2}(M))$ for the former is further decomposed into two matrices as expressed in (10).

3. REALIZATION OF THE FFT ALGORITHM

3.1 Architecture

Fig. 1 shows an architecture for computing the $N$-point DFT based on the FFT algorithm described above. It mainly consists of three two-port RAM’s, a complex multiplier, two complex adders, and two multiplexers, where the signal parameters used in RAM’s are: RA: read address, WA: write address, DO: data output, DI: data input, and CK: clock signal for triggering the read/write actions. Each two-port RAM has $N/2$ addresses and is able to read-and-then-write at an assigned address in one clock cycle. Initially, the input data vector $X$ (consisting of $N$ data samples) are loaded into the system via a multiplexer sample by sample, where the selection signal (I/O CTRL) of the multiplexer is 0 during this initial phase. The first $N/2$ data samples are stored on RAM-1 during the first $N/2$ cycle periods, and the other $N/2$ data samples are stored on RAM-2 during the second $N/2$ cycle periods. Here RAM-1 has addresses 0 to $N/2-1$, and RAM-2 has addresses $N/2$ to $N-1$. Once the initial data loading is finished, these $N$ data samples will be read out from RAM-1 and RAM-2 to realize the $1^s$-stage matrix-vector multiplication (with coefficient matrix $P_1(N)$) of the FFT algorithm on the two adders, RAM-3, and the multiplier in next $N$ cycles. The temporary results generated at this stage are then fed back to RAM-1 and RAM-2 via a multiplexer for performing the $2^s$-stage matrix-vector multiplication with coefficient matrix $P_2(N/2)$. Note that RAM-3 acts as a first-in-first-out (FIFO) buffer, which can delay a data sequence by $N/2$ cycle periods for the $1^s$-stage operations, by $N/4$ cycle periods for the $2^s$-stage operations, and so on. Moreover, a ROM is required to store all the transform coefficients for the FFT algorithm. This is not shown in Fig. 1. Realization of the $2^s$-stage matrix-vector multiplication as well is rather similar to that of the $1^s$-stage matrix-vector multiplication. The main difference is in the arrangement of addressing/control sequences.

For clearly understanding the operations of the proposed FFT architecture, let us consider how an 8-point DFT will be computed on it. In this case, (16) becomes

$$Z = P_{N/2}(2)P_{N/4}(4)P_{N/8}(8)X$$

$$= P_{N/2}(2)P_{N/4}(4)A$$

$$= P_{N/2}(2)B$$

where $X=[x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7]^T$ and $Z=[z_0 z_1 z_2 z_3 z_4 z_5 z_6 z_7]^T$. There will be four (general log$_2 N$+1) phases of operations involved in the transform process. They are summarized as follows:

1. Phase 1 (cycles 0 to 7): The input data vector $X$ is loaded into RAM-1 and RAM-2. The addressing/control sequences required are given in Table I (a).

2. Phase 2 (cycles 8 to 15): The $1^s$-stage matrix-vector multiplication, i.e., $A=[a_0 a_1 a_2 a_3]^T = P_{N/2}(2)X$, is performed. The addressing/control sequences required are given in Table I (b). In this phase, RAM-3 acts as a 4-cycle delay buffer.

3. Phase 3 (cycles 16 to 23): The $2^s$-stage matrix-vector multiplication, i.e., $B=[b_0 b_1 b_2 b_3]^T = P_{N/4}(4)A$, is performed. The addressing/control sequences required are given in Table I (c). In this phase, RAM-3 acts as a 2-cycle delay buffer.

4. Phase 4 (cycles 24 to 31): The $3^s$-stage matrix-vector multiplication, i.e., $Z=[z_0 z_1 z_2 z_3 z_4 z_5 z_6 z_7]^T = P_{N/8}(8)B$, is performed. The addressing/control sequences required are given in Table I (d). In this phase, RAM-3 acts as a 1-cycle delay buffer. The final output sequence is the bit-reversed version of the desired transform sequence, i.e., $Z=[z_0 z_1 z_2 z_3 z_4 z_5 z_6 z_7]^T = [y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7]^T$.

With little effort, one can check from Table I that the proposed architecture realizes the N-point FFT algorithm described in Section 2 in N log$_2 N$ cycle periods. Looks like the addressing/control sequences required for each phase is different, and one might ask how to provide them using simple circuitry. Fortunately, we found that this can easily be achieved by using a log$_N$-bit counter. For $N=8$, the most significant bit (MSB) $b_0$ of a 3-bit counter can generate the CTRL sequence for Phases 1, the middle bit $b_1$ can generate that for Phase 2, and the least significant bit (LSB) $b_2$ can generate that for Phase 3. Similarly, the RA and WA sequences for RAM-1, RAM-2, and RAM-3 at
each phase can also be derived from bits of a 3-bit counter. All of these are detailed in Table II. For N=512, a 9-bit counter can be used to generate all the control/addressing sequences in a manner as shown in Table III.

3.2 Chip Design

Fig. 2 shows a chip layout of the proposed architecture for a 512-point DFT, where the external data wordlength is 16 bits and the internal data wordlength is 24 bits. The design contains a complex-valued multiplier including four real-valued multipliers. It is based on a standard cell library for 0.6 μm CMOS technology. The chip requires a die size of 574x5222 μm² (containing about 400,000 transistors) and is able to operate at a clock rate up to 40 MHz for reaching a throughput of 4M transform samples per second in average. Such speed performance meets the requirements of standard DMT-based ADSL transceivers [4], [5].

4. SUMMARY

A novel VLSI architecture has been proposed for N-point DFT computation, where N is a power of two. It realizes a matrix formulation of the radix-2 DIF FFT algorithm using only one complex multiplier, three special two-port RAM’s of N/2 words each, two complex adders, one ROM, and some simple logical circuits. The proposed design is able to provide a throughput of several Mega or more transform samples per second for most practical cases of interest, but still retains the low-complexity feature. It is rather attractive for use in real-time long-length DFT applications, such as ADSL/OFDM transmission systems.

5. REFERENCES

### TABLE I Arrangement of the Control/Addressing Sequences for the Four Phases of the 8-Point FFT Algorithm

<table>
<thead>
<tr>
<th>CTRL</th>
<th>RAM-1</th>
<th>RAM-2</th>
<th>RAM-3</th>
<th>RAM-1 or RAM-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RA</td>
<td>DO</td>
<td>DO</td>
<td>WA</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

I/O CTRL = 0

### TABLE II Generation of the Control/Addressing Sequences for the 8-Point FFT Algorithm Using a 3-Bit Counter

<table>
<thead>
<tr>
<th>Up Counter</th>
<th>CTRL for Phase</th>
<th>RA of RAM-1 for Phase</th>
<th>RA of RAM-2 for Phase</th>
<th>RA/WA of RAM-3 for Phase</th>
<th>WA of RAM-1 or RAM-2 for Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>000</td>
<td>X 0 0 0 X 000 000 000</td>
<td>X 100 100 100</td>
<td>X 00 00 00</td>
<td>000 000 000 X</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>X 0 0 1 X 001 001 000</td>
<td>X 101 101 100</td>
<td>X 01 01 00</td>
<td>001 001 100 X</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>X 0 1 0 X 010 000 001</td>
<td>X 110 100 101</td>
<td>X 10 10 00</td>
<td>010 100 001 X</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>X 0 1 1 X 011 001 001</td>
<td>X 111 101 101</td>
<td>X 11 10 00</td>
<td>011 110 101 X</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>X 1 0 0 X 000 010 010</td>
<td>X 100 110 110</td>
<td>X 10 00 00</td>
<td>010 110 100 X</td>
</tr>
<tr>
<td></td>
<td>000</td>
<td>X 1 0 1 X 010 010 011</td>
<td>X 111 101 111</td>
<td>X 11 00 10</td>
<td>011 110 110 X</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>X 1 1 0 X 011 011 011</td>
<td>X 111 111 111</td>
<td>X 11 01 00</td>
<td>011 111 110 X</td>
</tr>
<tr>
<td></td>
<td>000</td>
<td>X 1 1 1 X 100 000 000</td>
<td>X 110 110 110</td>
<td>X 11 00 00</td>
<td>011 111 111 X</td>
</tr>
</tbody>
</table>

### TABLE III Generation of the Control/Addressing Sequences for the 512-Point FFT Algorithm Using a 9-Bit Counter

<table>
<thead>
<tr>
<th>Phase Number</th>
<th>RA of RAM-1</th>
<th>RA of RAM-2</th>
<th>RA/WA of RAM-3</th>
<th>CTRL</th>
<th>WA of RAM-1 or RAM-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (Initial)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 (Computation with $P_1$)</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>2 (Computation with $P_2$)</td>
<td>0 $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>3 (Computation with $P_3$)</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>4 (Computation with $P_4$)</td>
<td>0 $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>5 (Computation with $P_5$)</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>6 (Computation with $P_6$)</td>
<td>0 $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>7 (Computation with $P_7$)</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>8 (Computation with $P_8$)</td>
<td>0 $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
<tr>
<td>9 (Computation with $P_9$)</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>1 $a_0$ $a_0$ $a_0$</td>
<td>0 $a_0$ $a_0$ $a_0$</td>
<td>$a_0$</td>
<td></td>
</tr>
</tbody>
</table>