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 μm^2 and the throughput is up to 4M transform samples per second under 0.6 μm 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, multicarrier 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 architecture realizes a radix-2 FFT algorithm by using one 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

$$y_k = \sum_{n=0}^{N-1} x_n W_N^{nk}, \quad k=0,1,2,\ldots,N-1, \tag{1}$$

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

$$\mathbf{Y} = \mathbf{W}(N)\mathbf{X} \tag{2}$$

where

$$\mathbf{W}(N) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & \cdots & W_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \tag{3}$$

is called a transform matrix. With the property of $W_N^{mk}=(-1)^k W_N^{(m+N/2)k}$, we can partition the matrix $\mathbf{W}(N)$ 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 $\mathbf{Q}(N)=[\,\mathbf{e}_0\ \mathbf{e}_2\ \mathbf{e}_4\ \ldots\ \mathbf{e}_{N-2}\ \mathbf{e}_1\ \mathbf{e}_3\ \mathbf{e}_5\ldots\ \mathbf{e}_{N-1}]^T$ with $\mathbf{e}_n$ being a unit *N*x1 column vector whose (*n*+1)-th element is 1, i.e.,

$$\mathbf{Q}(N)\mathbf{Y} = \mathbf{Q}(N)\mathbf{W}(N)\mathbf{X} = \begin{bmatrix} \mathbf{E}(N/2) & \mathbf{E}(N/2) \\ \mathbf{D}(N/2) & -\mathbf{D}(N/2) \end{bmatrix}\mathbf{X} \tag{4}$$

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

$$\mathbf{E}(N/2) = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^{2\text{x}1} & W_N^{2\text{x}2} & \cdots & W_N^{2\text{x}(N/2-1)} \\ 1 & W_N^{4\text{x}1} & W_N^{4\text{x}2} & \cdots & W_N^{4\text{x}(N/2-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-2)\text{x}1} & W_N^{(N-2)\text{x}2} & \cdots & W_N^{(N-2)\text{x}(N/2-1)} \end{bmatrix} \tag{5}$$

$$\mathbf{D}(N/2) = \begin{bmatrix} 1 & W_N^{1\text{x}1} & W_N^{1\text{x}2} & \cdots & W_N^{1\text{x}(N/2-1)} \\ 1 & W_N^{3\text{x}1} & W_N^{3\text{x}2} & \cdots & W_N^{3\text{x}(N/2-1)} \\ 1 & W_N^{5\text{x}1} & W_N^{5\text{x}2} & \cdots & W_N^{5\text{x}(N/2-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)\text{x}1} & W_N^{(N-1)\text{x}2} & \cdots & W_N^{(N-1)\text{x}(N/2-1)} \end{bmatrix} \tag{6}$$

$$\mathbf{D}(N/2) = \mathbf{E}(N/2)\mathbf{F}(N/2) \tag{7}$$

$$\mathbf{F}(N/2) = \begin{bmatrix} W_N^0 & 0 & \cdots & 0 \\ 0 & W_N^1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_N^{(N/2-1)} \end{bmatrix} \tag{8}$$

Substituting (7) into (4) yields

$$\begin{aligned} \mathbf{Q}(N)\mathbf{Y} &= \begin{bmatrix} \mathbf{E}(N/2) & \mathbf{E}(N/2) \\ \mathbf{E}(N/2)\mathbf{F}(N/2) & -\mathbf{E}(N/2)\mathbf{F}(N/2) \end{bmatrix}\mathbf{X} \\ &= \begin{bmatrix} \mathbf{E}(N/2) & \mathbf{0} \\ \mathbf{0} & \mathbf{E}(N/2) \end{bmatrix}\mathbf{P}(N)\mathbf{X} \end{aligned} \tag{9}$$

where

$$\mathbf{P}(N) = \begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{I}_{N/2} \\ \mathbf{F}(N/2) & -\mathbf{F}(N/2) \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}(N/2) \end{bmatrix}\begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{I}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{I}_{N/2} \end{bmatrix}. \tag{10}$$

Since $W_N^{2kn}=W_{N/2}^{kn}$, we can observe from (3) and (5) that $\mathbf{E}(N/2)=\mathbf{W}(N/2)$. Thus, (9) can be rewritten as

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

For ease of presentation, we define $\mathbf{W}_{N/M}(N/2)$ as the direct sum [12] of N/M $\mathbf{W}$(M)'s of size MxM, i.e.,

$$\mathbf{W}_{N/M}(M)=\mathbf{W}(M)\oplus\mathbf{W}(M)\oplus\cdots\oplus\mathbf{W}(M)=\begin{bmatrix}\mathbf{W}(M) & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{W}(M) & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{W}(M)\end{bmatrix} \tag{12}$$

With similar direct-sum definitions for matrices $\mathbf{Q}_{N/M}(M)$ and $\mathbf{P}_{N/M}(M)$, we can express (11) as follows:

$$[\mathbf{Q}_1(N)\mathbf{Y}]=\mathbf{W}_2(N/2)[\mathbf{P}_1(N)\mathbf{X}] \tag{13}$$

Note that (13) can be regarded as an N-point transform with transform matrix $\mathbf{W}_2(N/2)$, input vector $\mathbf{P}_1(N)\mathbf{X}$, and output vector $\mathbf{Q}_1(N)\mathbf{Y}$. Since $\mathbf{W}_2(N/2)=\mathbf{W}(N/2)\oplus\mathbf{W}(N/2)$, this N-point transform can be partitioned into two $N/2$-point DFT's with transform matrix $\mathbf{W}(N/2)$ each. Using the permutation matrix $\mathbf{Q}(N/2)=[\,\mathbf{e}_0\,\mathbf{e}_2\,\mathbf{e}_4\,...\,\mathbf{e}_{N/2-2}\,\mathbf{e}_1\,\mathbf{e}_3\,\mathbf{e}_5...\,\mathbf{e}_{N/2-1}]^T$ for each $N/2$-point DFT (i.e., multiplying both sides of (13) by $\mathbf{Q}_2(N/2)=\mathbf{Q}(N/2)\oplus\mathbf{Q}(N/2)$) and following the procedure of (4)-(13), we can further decompose each $N/2$-point DFT into two $N/4$-point DFT's with transform matrix $\mathbf{W}(N/4)$ each. This can be described as

$$[\mathbf{Q}_2(N/2)\mathbf{Q}_1(N)\mathbf{Y}]=\mathbf{W}_4(N/4)[\mathbf{P}_2(N/2)\mathbf{P}_1(N)\mathbf{X}] \tag{14}$$

where $\mathbf{P}_2(N/2)=\mathbf{P}(N/2)\oplus\mathbf{P}(N/2)$. Repeating such a decomposition process until $\mathbf{W}_N(1)=\mathbf{I}_N$ (an identity matrix) appears, we have

$$[\mathbf{Q}_{N/2}(2)\mathbf{Q}_{N/4}(4)\cdots\mathbf{Q}_1(N)\mathbf{Y}]=\mathbf{P}_{N/2}(2)\mathbf{P}_{N/4}(4)\cdots\mathbf{P}_1(N)\mathbf{X}\,. \tag{15}$$

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 $\mathbf{P}_{N/M}(M)$ in (15) is equivalent to realizing the $(\log_2 N/M+1)$th stage of the well-known radix-2 decimation-in-frequency FFT algorithm [13] and the resulting output vector (denoted by $\mathbf{Z}$) is the bit-reversed version of $\mathbf{Y}$. That is,

$$\begin{aligned}\mathbf{Z}&=[y_0\ y_{N/2}\ y_{N/4}\ y_{3N/4}\ y_{N/8}\cdots y_{N-1}]^T\\ &=\mathbf{P}_{N/2}(2)\mathbf{P}_{N/4}(4)\cdots\mathbf{P}_2(N/2)\mathbf{P}_1(N)\mathbf{X}\end{aligned}\,. \tag{16}$$

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 ($\mathbf{P}_{N/M}(M)$) 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 $\mathbf{X}$ (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 ~ $N/2$-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 1st-stage matrix-vector multiplication (with coefficient matrix $\mathbf{P}_1(N)$) 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 2nd-stage matrix-vector multiplication with coefficient matrix $\mathbf{P}_2(N/2)$. 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 1st-satge operations, by $N/4$ cycle periods for the 2nd-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 2nd-stage matrix-vector multiplication et al. is rather similar to that of the 1st-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

$$\begin{aligned}\mathbf{Z}&=\mathbf{P}_4(2)\cdot\mathbf{P}_2(4)\cdot\mathbf{P}_1(8)\cdot\mathbf{X}\\ &=\mathbf{P}_4(2)\cdot\mathbf{P}_2(4)\cdot\mathbf{A}\\ &=\mathbf{P}_4(2)\cdot\mathbf{B}\end{aligned} \tag{17}$$

where $\mathbf{X}=[x_0, x_1, ... x_7]^T$ and $\mathbf{Z}=[z_0, z_1, ... z_7]^T$. There will be four (in general $\log_2 N+1$) phases of operations involved in the transform process. They are summarized as follows:

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

(2) Phase 1 (from cycles 8 ~ 15): The 1st-satge matrix-vector multiplication, i.e., $\mathbf{A}=[a_0, a_1, ... a_7]^T=\mathbf{P}_1(8)\mathbf{X}$, 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 2 (from cycles 16 ~23): The 2nd-stage matrix-vector multiplication, i.e., $\mathbf{B}=[b_0, b_1, ... b_7]^T=\mathbf{P}_2(4)\mathbf{A}$, 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 3 (from cycles 24 ~31): The 3rd-stage matrix-vector multiplication, i.e., $\mathbf{Z}=[z_0, z_1, ... z_7]^T=\mathbf{P}_4(2)\mathbf{B}$, 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., $\{z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7\}=\{y_0, y_4, y_2, y_6, y_1, y_5, y_3, y_7\}$.

With little effort, one can check from Table I that the proposed architecture realizes the N-point FFT algorithm described in Section 2 in $N+N\log_2 N$ 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 $\log_2 N$-bit counter. For N=8, the most significant bit (MSB) ω_2 of a 3-bit counter can generate the CTRL sequence for Phases 1, the middle bit ω_1 can generate that for Phase 2, and the least significant bit (LSB) ω_0 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 5747x5222 μm^2 (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

[1] J. A. C. Bingham, "Multicarrier modulation for data transmission: An idea whose time has come," *IEEE Commun. Mag.*, pp. 5-14, May 1990.

[2] J. S. Chow, J. C. Tu, and J. M. Cioffi, "A discrete multitone transceiver system for HDSL applications," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 895-908, Aug. 1991.

[3] B. Le Floch, M. Alard, and C. Berrou, "Coded orthogonal frequency-division multiplexing," Proc. IEEE, vol. 83, pp. 982-996, June 1995.

[4] American National Standard - T1.413-1995, "ADSL Metallic Interface Specification," New York, NY, 1995.

[5] J. M. Cioffi, "Asymmetric digital subscriber lines," Chapter of the CRC Handbook of Communications, 1997.

[6] Digital Broadcasting Systems for Television, Sound, and Data Services, European Telecommunications Standard, prETS 300 744 (Draft, Version 0.0.3), Apr. 1996.

[7] S. Hara and R. Prasad, "Overview of multicarrier CDMA," IEEE Commun. Mag., vol. 35, pp. 126-133, Dec. 1997.

[8] L.-W. Chang and M.-Y. Wu, "A new systolic array for discrete Fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1165-1167, Oct. 1988.

[9] J. A. Beraldin, T. Aboulnasr, and W. Steenaart, "Efficient one-dimensional systolic array realization of discrete Fourier transform," *IEEE Trans. Circuits Syst*., vol. 36, pp. 95-100, Jan. 1989.

[10] J. Choi, and V. Boriakoff, "A new linear systolic array for FFT computation," *IEEE Trans. Circuits Syst.-II.*, vol. 39, pp. 236-239, Apr. 1992.

[11] V. Boriakoff, "FFT computation with systolic arrays, a new architecture," *IEEE Trans. Circuits Syst.-II.*, vol. 41, pp. 278-284, Apr. 1994.

[12] H. Frieddberg, A. J. Insel, and L. J. Spence, *Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1989, 2nd edition.

[13] A. V. Oppenhiem, and R. W. Schafer, Discrete-Time Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[14] M. C. Pease, "An adaptation of the fast Fourier transform for parallel processing," *J. Association for Computing Machinary*, vol-15, pp. 252-264, Feb. 1968.

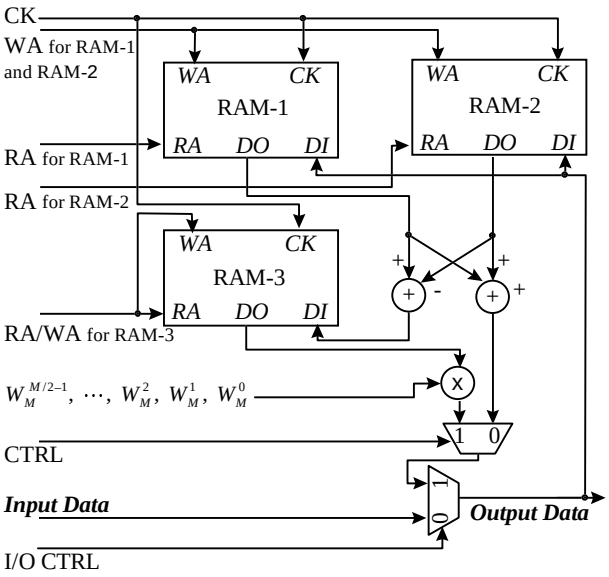

Fig. 1. A memory-based architecture for FFT computation.

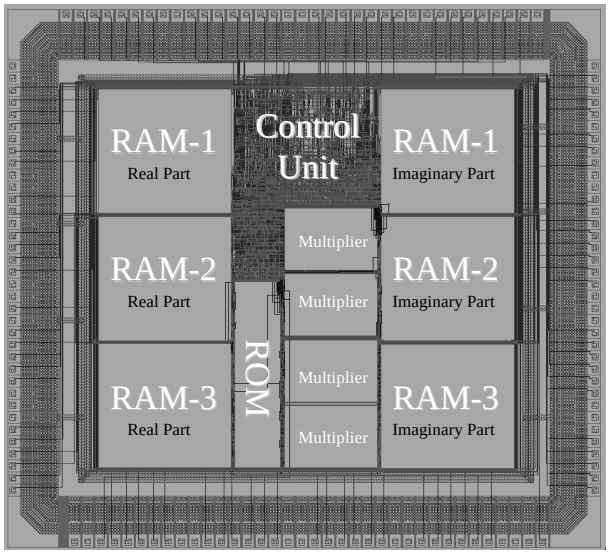

Fig. 2. A chip layout of the proposed 512-point FFT architecture.

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

(a)	RAM-1		RAM-2		RAM-3			RAM-1 or RAM-2	
CTRL	*RA*	*DO*	*RA*	*DO*	*RA/WA*	*DI*	*DO*	*WA*	*DI*
X	X	X	X	X	X	X	X	000	x_0
X	X	X	X	X	X	X	X	001	x_1
X	X	X	X	X	X	X	X	010	x_2
X	X	X	X	X	X	X	X	011	x_3
X	X	X	X	X	X	X	X	100	x_4
X	X	X	X	X	X	X	X	101	x_5
X	X	X	X	X	X	X	X	110	x_6
X	X	X	X	X	X	X	X	111	x_7
I/O CTRL=0									

Time
000
001
010
011
100
101
110
111

(b)	RAM-1		RAM-2		RAM-3			RAM-1 or RAM-2	
CTRL	*RA*	*DO*	*RA*	*DO*	*RA/WA*	*DI*	*DO*	*WA*	*DI*
0	000	x_0	100	x_4	00	x_0-x_4	X	000	$x_0+x_4= a_0$
0	001	x_1	101	x_5	01	x_1-x_5	X	001	$x_1+x_5= a_1$
0	010	x_2	110	x_6	10	x_2-x_6	X	100	$x_2+x_6= a_2$
0	011	x_3	111	x_7	11	x_3-x_7	X	101	$x_3+x_7= a_3$
1	000	X	100	X	00	X	x_0-x_4	010	$(x_0-x_4) \cdot W_8^0 = a_4$
1	001	X	101	X	01	X	x_1-x_5	011	$(x_1-x_5) \cdot W_8^1 = a_5$
1	010	X	110	X	10	X	x_2-x_6	110	$(x_2-x_6) \cdot W_8^2 = a_6$
1	011	X	111	X	11	X	x_3-x_7	111	$(x_3-x_7) \cdot W_8^3 = a_7$
I/O CTRL=1									

(c)	RAM-1		RAM-2		RAM-3			RAM-1 or RAM-2	
CTRL	*RA*	*DO*	*RA*	*DO*	*RA/WA*	*DI*	*DO*	*WA*	*DI*
0	000	a_0	100	a_2	00	a_0-a_2	X	000	$a_0+a_2= b_0$
0	001	a_1	101	a_3	01	a_1-a_3	X	100	$a_1+a_3= b_1$
1	000	X	100	X	00	X	a_0-a_2	001	$(a_0-a_2) \cdot W_4^0 = b_2$
1	001	X	101	X	01	X	a_1-a_3	101	$(a_1-a_3) \cdot W_4^1 = b_3$
0	010	a_4	110	a_6	00	a_4-a_6	X	010	$a_4+a_6= b_4$
0	011	a_5	111	a_7	01	a_5-a_7	X	110	$a_5+a_7= b_5$
1	010	X	110	X	00	X	a_4-a_6	011	$(a_4-a_6) \cdot W_4^0 = b_6$
1	011	X	111	X	01	X	a_5-a_7	111	$(a_5-a_7) \cdot W_4^1 = b_7$
I/O CTRL=1									

Time
000
001
010
011
100
101
110
111

(d)	RAM-1		RAM-2		RAM-3			RAM-1 or RAM-2	
CTRL	*RA*	*DO*	*RA*	*DO*	*RA/WA*	*DI*	*DO*	*WA*	***Output Data***
0	000	b_0	100	b_1	00	b_0-b_1	X	X	$b_0+b_1= z_0$
1	000	X	100	X	00	X	b_0-b_1	X	$(b_0-b_1) \cdot W_2^0 = z_1$
0	001	b_2	101	b_3	00	b_2-b_3	X	X	$b_2+b_3= z_2$
1	001	X	101	X	00	X	b_2-b_3	X	$(b_2-b_3) \cdot W_2^0 = z_3$
0	010	b_4	110	b_5	00	b_4-b_5	X	X	$b_4+b_5= z_4$
1	010	X	110	X	00	X	b_4-b_5	X	$(b_4-b_5) \cdot W_2^0 = z_5$
0	011	b_6	111	b_7	00	b_6-b_7	X	X	$b_6+b_7= z_6$
1	011	X	111	X	00	X	b_6-b_7	X	$(b_6-b_7) \cdot W_2^0 = z_7$
I/O CTRL=1									

X :don't care.

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

Up	CTRL for Phase				RA of RAM-1 for Phase				RA of RAM-2 for Phase				RA/WA of RAM-3 for Phase				WA of RAM-1 or RAM-2 for Phase			
Counter	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
000	X	0	0	0	X	000	000	000	X	100	100	100	X	00	00	00	000	000	000	X
001	X	0	0	1	X	001	001	000	X	101	101	100	X	01	01	00	001	001	100	X
010	X	0	1	0	X	010	000	001	X	110	100	101	X	10	00	00	010	100	001	X
011	X	0	1	1	X	011	001	001	X	111	101	101	X	11	01	00	011	101	101	X
100	X	1	0	0	X	000	010	010	X	100	110	110	X	00	00	00	100	010	010	X
101	X	1	0	1	X	001	011	010	X	101	111	110	X	01	01	00	101	011	110	X
110	X	1	1	0	X	010	010	011	X	110	110	111	X	10	00	00	110	110	011	X
111	X	1	1	1	X	011	011	011	X	111	111	111	X	11	01	00	111	111	111	X
$\omega_2\omega_1\omega_0$	X	ω_2	ω_1	ω_0	X	$0\omega_1\omega_0$	$0\omega_2\omega_0$	$0\omega_2\omega_1$	X	$1\omega_1\omega_0$	$1\omega_2\omega_0$	$1\omega_2\omega_1$	X	$\omega_1\omega_0$	$0\omega_0$	00	$\omega_2\omega_1\omega_0$	$\omega_1\omega_2\omega_0$	$\omega_0\omega_2\omega_1$	X

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

Phase Number	RA of RAM-1	RA of RAM-2	RA/WA of RAM-3	CTRL	WA of RAM-1 or RAM-2
0 (Initial loading)	X	X	X	X	$\omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$
1 (Computation with $\mathbf{P}_1(512)$)	$0\ \omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	$1\ \omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	$\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	ω_8	$\omega_7\omega_8\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$
2 (Computation with $\mathbf{P}_2(256)$)	$0\ \omega_8\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	$1\ \omega_8\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	$0\ \omega_6\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	ω_7	$\omega_6\omega_8\omega_7\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$
3 (Computation with $\mathbf{P}_4(128)$)	$0\ \omega_8\omega_7\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	$1\ \omega_8\omega_7\omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	$0\ 0\ \omega_5\omega_4\omega_3\omega_2\omega_1\omega_0$	ω_6	$\omega_5\omega_8\omega_7\omega_6\omega_4\omega_3\omega_2\omega_1\omega_0$
4 (Computation with $\mathbf{P}_8(64)$)	$0\ \omega_8\omega_7\omega_6\omega_4\omega_3\omega_2\omega_1\omega_0$	$1\ \omega_8\omega_7\omega_6\omega_4\omega_3\omega_2\omega_1\omega_0$	$0\ 0\ 0\ \omega_4\omega_3\omega_2\omega_1\omega_0$	ω_5	$\omega_4\omega_8\omega_7\omega_6\omega_5\omega_3\omega_2\omega_1\omega_0$
5 (Computation with $\mathbf{P}_{16}(32)$)	$0\ \omega_8\omega_7\omega_6\omega_5\omega_3\omega_2\omega_1\omega_0$	$1\ \omega_8\omega_7\omega_6\omega_5\omega_3\omega_2\omega_1\omega_0$	$0\ 0\ 0\ 0\ \omega_3\omega_2\omega_1\omega_0$	ω_4	$\omega_3\omega_8\omega_7\omega_6\omega_5\omega_4\omega_2\omega_1\omega_0$
6 (Computation with $\mathbf{P}_{32}(16)$)	$0\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_2\omega_1\omega_0$	$1\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_2\omega_1\omega_0$	$0\ 0\ 0\ 0\ 0\ \omega_2\omega_1\omega_0$	ω_3	$\omega_2\omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_1\omega_0$
7 (Computation with $\mathbf{P}_{64}(8)$)	$0\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_1\omega_0$	$1\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_1\omega_0$	$0\ 0\ 0\ 0\ 0\ 0\ \omega_1\omega_0$	ω_2	$\omega_1\omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_0$
8 (Computation with $\mathbf{P}_{128}(4)$)	$0\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_0$	$1\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_0$	$0\ 0\ 0\ 0\ 0\ 0\ 0\ \omega_0$	ω_1	$\omega_0\omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1$
9 (Computation with $\mathbf{P}_{256}(2)$)	$0\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1$	$1\ \omega_8\omega_7\omega_6\omega_5\omega_4\omega_3\omega_2\omega_1$	$0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$	ω_0	X