A DHT-BASED FFT/IFFT PROCESSOR FOR VDSL TRANSCEIVERS

Chin-Liang Wang† ‡ *and Ching-Hsien Chang*‡

†Institute of Communications Engineering, National Tsing Hua University
‡Department of Electrical Engineering, National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
E-mail: clwang@ee.nthu.edu.tw

ABSTRACT

This paper presents a new VLSI architecture for computing the N-point discrete Fourier transform (DFT) of real data and the corresponding inverse (IDFT) based on the discrete Hartley transform, where N is a power of two. The architecture includes two real multipliers, three real adders, six memory-based buffers, two ROM's, and some simple logic circuits, making itself suitable for single-chip implementation. It is capable of evaluating one DFT sample or one IDFT sample every $(\log_2 N+1)/2$ clock cycles on average. Under 0.35 *m*m CMOS technology, the proposed design can operate at a clock rate of 100 MHz to reach a throughput of 20M transform samples per second for N=512. The processing speed will be higher if more advanced CMOS technology is adopted to implement the same circuit. Such low-complexity and high-throughput feature supports that the proposed design is well suited for use in discrete multitone based very high-speed digital subscriber line transceivers.

1. INTRODUCTION

The discrete multitone modulation (DMT) is a famous form of digital implementation of multicarrier modulation [1]. This technology has been widely investigated for high-speed data transmission on copper wires. For example, it has been selected by the American National Standards Institute and the European Telecommunications Standards Institute for asymmetric digital subscriber line (ADSL) and very high-speed digital subscriber line (VDSL) services [2]-[5] over ordinary phone lines. ADSL service can provide a date rate of several megabits per second, while VDSL service can provide a data rate up to 52 megabits per second. For a DMT-based VDSL transceiver, the modulator and demodulator need to respectively compute very long-length inverse discrete Fourier transforms (IDFT) and DFT, where the transform length may be as high as 4096 and the sampling rate may be up to 44.16 MHz. Obviously, the DFT/IDFT computation involved in VDSL applications is pretty complicated and there is a great need to develop fast processors for it to meet the real-time requirements.

The discrete Hartley transform (DHT) involves only real-valued arithmetic and has an identical inverse [6]. It is closely related to the DFT and has become an effective tool for computing the DFT of real data; we can evaluate the corresponding DHT first and then convert the result into the DFT. There have been a number of fast algorithms and architectures proposed earlier for the DHT computation (see, for example, [7]-[10]). However, they are either too slow to meet the speed requirement of VDSL applications or consume too many multipliers to be realized in a single chip.

In this paper, we propose a new DHT-based FFT/IFFT (fast Fourier transform and its inverse) processor for DMT-based VDSL applications. The proposed design involves two real multipliers, three real adders, two ROM's, six memory-based buffers, and some simple logic circuits to achieve the throughput performance required. This structure can be regarded as an improved version of the single-chip FFT design for ADSL applications described in [11], where the throughput is doubled (under the same technology) with an increase of one real multiplier and two $N/2$-word RAM's. The low-complexity feature makes it well suited to single-chip implementation.

This work was supported by the National Science Council of the Republic of China under Grant NSC 89-2213-E-007-128.

2. DHT-BASED FFT/IFFT COMPUTATION

2.1 A Fast DHT (FHT) ALGORITHM [10]

Define the N-point DHT of a real sequence by

$$y_k = \sum_{n=0}^{N-1} x_n H_N^{kn},\ k = 0,1,\ldots,N-1, \tag{1}$$

where $H_N^{kn}=\sin(2\pi kn/N)+\cos(2\pi kn/N)$. Also let $\mathbf{X}=[x_0\ x_1\ \ldots\ x_{N-1}]^T$ denote the transform input vector and $\mathbf{Y}=[y_0\ y_1\ \ldots\ y_{N-1}]^T$ represent the transform output vector. Then we can express (1) as

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

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

With the assumptions that N is a power of two and $\mathbf{Z}=[z_0\ z_1\ \ldots\ z_{N-1}]^T$ is the bit-reversed version of vector $\mathbf{Y}$, the following matrix formulation can be derived for the FHT computation [10]:

$$\mathbf{Z} = \mathbf{P}_{N/2}(2)\cdots\mathbf{P}_2(N/2)\mathbf{P}_1(N)\mathbf{X} \tag{4}$$

where $\mathbf{P}_{N/M}(M)$ represents the direct sum [12] of N/M $\mathbf{P}(M)$'s of size $M\times M$ given by

$$\mathbf{P}_{N/M}(M) = \mathbf{P}(M)\oplus\mathbf{P}(M)\oplus\cdots\oplus\mathbf{P}(M) = \begin{bmatrix} \mathbf{P}(M) & 0 & \cdots & 0 \\ 0 & \mathbf{P}(M) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{P}(M) \end{bmatrix}, \tag{5}$$

$$\mathbf{P}(M) = \begin{bmatrix} \mathbf{I}_{M/2} & 0 \\ 0 & \mathbf{K}(M/2) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{I}_{M/2} \\ \mathbf{I}_{M/2} & -\mathbf{I}_{M/2} \end{bmatrix}, \tag{6}$$

$$\mathbf{K}(M/2) = \mathbf{C}(M/2) + \mathbf{S}(M/2) = \begin{bmatrix} C_M^0 & & & & \\ & C_M^1 & & & \\ & & C_M^2 & & \\ & & & \ddots & \\ & & & & C_M^{M/2-1} \end{bmatrix} + \begin{bmatrix} S_M^0 & & & & \\ & & & & S_M^1 \\ & & & S_M^2 & \\ & & \iddots & & \\ & S_M^{M/2-1} & & & \end{bmatrix} \tag{7}$$

with $C_M^k=\cos(2\pi k/M)$ and $S_M^k=\sin(2\pi k/M)$.

2.2 FFT/IFFT Computation via the FHT Algorithm [11]

Define the N-point DFT and IDFT by

$$f_k = \sum_{n=0}^{N-1} x_n W_N^{nk},\ k = 0,1,2,\ \ldots,\ N\text{-}1, \tag{8}$$

$$x_n = (1/N)\sum_{k=0}^{N-1} f_k W_N^{-kn},\ n = 0,1,2,\ \ldots, N\text{-}1, \tag{9}$$

where $W_N=\exp(-j2\pi/N)$. With the properties of DHT given in [6], we can compute the DFT samples as follows:

Step 1: Forming the even and odd parts of the DHT of **X**

Even Part: $H_k^e=(y_k+y_{N-k})/2$, for $k=1, 2, \ldots, N/2$ (10)

Odd Part: $H_k^o=(y_k - y_{N-k})/2$, for $k=1, 2, \ldots, N/2$ (11)

Step 2: Generating the DFT samples from the even and odd parts

$f_0=y_0$ (12)

$f_k=H_k^e-jH_k^o$, for $k=1, 2, \ldots, N/2$ (13)

$f_{N-k}=H_k^e+jH_k^o$, for $k=1, 2, \ldots, N/2$ (14)

It should be noted that the input vector for the IDFT is symmetric for DMT-based VDSL applications, i.e., the input vector is in a form as $\mathbf{F}=[f_0\ f_1\ \ldots\ f_{N-1}]^T=[c_0\ \ c_1+jd_1\ \ c_2+jd_2\ \ldots\ c_{N/2-1}+jd_{N/2-1}\ \ c_{N/2}\ \ c_{N/2-1}-jd_{N/2-1}\ \ldots\ c_2-jd_2\ \ c_1-jd_1]^T$, where c_i and d_i are real. Thus, the IDFT definition given in (9) can be rewritten in matrix-vector form as follows:

$$\mathbf{X}=(1/N)\mathbf{W}(N)\mathbf{F}=(1/N)\mathbf{H}(N)\mathbf{LF} \tag{15}$$

where

$$\mathbf{W}(N)=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N^{-1} & \cdots & W_N^{-(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{-(N-1)} & \cdots & W_N^{-(N-1)^2} \end{bmatrix}, \tag{16}$$

$$\mathbf{L}=\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & (1+j)/2 & 0 & \cdots & 0 & \cdots & 0 & (1-j)/2 \\ \vdots & 0 & \ddots & 0 & \vdots & 0 & \iddots & 0 \\ 0 & \vdots & 0 & (1+j)/2 & 0 & (1-j)/2 & 0 & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & \vdots & 0 & (1-j)/2 & 0 & (1+j)/2 & 0 & \vdots \\ \vdots & 0 & \iddots & 0 & \vdots & 0 & \ddots & 0 \\ 0 & (1-j)/2 & 0 & \cdots & 0 & \cdots & 0 & (1+j)/2 \end{bmatrix}. \tag{17}$$

Neglecting the scaling factor $1/N$, we can evaluate the N-point IDFT by computing the N-point DHT of the input vector $\mathbf{LF}=[c_0\ \ c_1-d_1\ \ c_2-d_2\ \ldots\ c_{N/2-1}-d_{N/2-1}\ \ c_{N/2}\ \ c_{N/2-1}+d_{N/2-1}\ \ldots\ c_2+d_2\ \ c_1+d_1]^T$.

3. REALIZATION OF THE PROPOSED ALGORITHMS

Based on the FHT algorithm described above, we can construct a memory-based architecture for computing the N-point DFT/IDFT as shown in Fig. 1. The structure is mainly composed of six two-port RAM's, two ROM's, two real-valued multipliers, three real-valued adders, and eight multiplexers. Each two-port RAM contains $N/2$ addresses and five types of addressing/control signals, which are **RA** ("read" address), **WA** ("write" address), **DO** (data output), **DI** (data input), and **CK** (clock signal for triggering the "read/write" actions). Such a special RAM is able to read-and-then-write at the same address or to read and write independently at different addresses in one clock cycle. The two ROM's are used to store all the cosine and sine coefficients for the FHT algorithm.

Initially, the first input data vector $\mathbf{X}^0$ of N samples enters the system via the multiplexer MUX-1 on a sample-by-sample basis; the first $N/2$ samples are loaded into RAM-1 during the first $N/2$ cycle periods, and the other $N/2$ samples are loaded into RAM-2 during the second $N/2$ cycle periods. Once this initial data loading is completed, the N samples of $\mathbf{X}^0$ will be read out from RAM-1 and RAM-2 to perform the first-stage matrix-vector multiplication (i.e., $\mathbf{P}_1(N)\mathbf{X}^0$) of the FHT algorithm on the three adders, RAM-5, RAM-6, and two multipliers in the subsequent N cycles. The temporary results computed at this stage are then sent back to RAM-1 and RAM-2 via MUX-1 for use in the second-stage matrix-vector multiplication with coefficient matrix $\mathbf{P}_2(N/2)$. The second-stage matrix-vector multiplication et al. will be realized in a similar manner to the first-stage matrix-vector multiplication, where the main difference is in the arrangement of addressing/control sequences. Note that RAM-5 and RAM-6 respectively acts as a first-in-first-out (FIFO) buffer and a first-in-last-out-like (FILO-like) buffer with size of $N/2$ each. RAM-5 can delay an $N/2$-point data sequence by $N/2$ cycle periods and RAM-6 is used to perform the FILO-like permutation operations of those multiplication results with sine coefficients defined by the matrix $\mathbf{S}(M/2)$ in (7). If the input sequence of the FILO-like buffer is b_0, b_1, ..., $b_{M/2-2}$, $b_{M/2-1}$, then the required output from the FILO-like buffer is b_0, $b_{M/2-1}$, $b_{M/2-2}$, ..., b_1.

During the second N cycle periods when $\mathbf{P}_1(N)\mathbf{X}^0$ is computed, the second data vector $\mathbf{X}^1$ of N samples enters the system sample by sample; the first $N/2$ samples are loaded into RAM-3 via the multiplexer MUX-2 during the first half of this time interval; the other $N/2$ samples are moved directly to the adders via the multiplexer MUX-4 during the second half of this time interval. Once the loading operation of the first $N/2$ samples of $\mathbf{X}^1$ is completed, the first-stage matrix-vector multiplication of the FHT algorithm for computing the transform of $\mathbf{X}^1$ will be performed on the three adders, RAM-5, RAM-6, and two multipliers in the subsequent N cycles. The transform operations of data vector $\mathbf{X}^1$ are the same as those of data vector $\mathbf{X}^0$ described above, except that for $\mathbf{X}^1$ the temporary results computed are sent back to RAM-3 and RAM-4 via MUX-2 (instead of RAM-1 and RAM-2 via MUX-1) for use in the second-stage matrix-vector multiplication.

It should be noted that the transform computations of $\mathbf{X}^0$ and $\mathbf{X}^1$ are overlapped in the proposed architecture; those of $\mathbf{X}^0$ are $N/2$ cycles ahead of those of $\mathbf{X}^1$, and the three adders, RAM-5, RAM-6, and two multipliers are alternately employed for these two transform computations. After the transform computations of $\mathbf{X}^0$ and $\mathbf{X}^1$ are finished, those of $\mathbf{X}^2$ and $\mathbf{X}^3$ are overlapped in a similar manner, and so on.

To have better understanding of the proposed architecture, let us consider how to compute an 8-point DFT of real data and the corresponding IDFT. For the 8-point case, (4) becomes

$$\mathbf{Z}^i = \mathbf{P}_4(2)\cdot\mathbf{P}_2(4)\cdot\mathbf{P}_1(8)\cdot\mathbf{X}^i = \mathbf{P}_4(2)\cdot\mathbf{P}_2(4)\cdot\mathbf{A}^i = \mathbf{P}_4(2)\cdot\mathbf{B}^i \tag{18}$$

where $\mathbf{X}^i=[x_0^i\ x_1^i \cdots x_7^i]^T$, $\mathbf{Z}^i=[z_0^i\ z_1^i \cdots z_7^i]^T$, and i=0, 1, 2, …. For each RAM module shown in Fig. 1, the **WA** and **RA** signals are both of three bits. The most significant bit (MSB) of **WA** will enable or disable the "write" operation, and the two least significant bits will be used for addressing the RAM. The three bits of **RA** have similar functions to those of **WA**. For correct execution, we have the following arrangements:

- When the MSB of **RA** of RAM-1 or RAM-2 is 0, RAM-1 or RAM-2 is enabled for the "read" operation.
- When the MSB of **RA** of RAM-3 or RAM-4 is 1, RAM-3 or RAM-4 is enabled for the "read" operation.
- When the MSB of **WA** of RAM-1 or RAM-3 is 0, RAM-1 or RAM-3 is enabled for the "write" operation.
- When the MSB of **WA** of RAM-2 or RAM-4 is 1, RAM-2 or RAM-4 is enabled for the "write" operation.

The operations and addressing/control sequences of this example are shown in Table I, and they are briefly described as follows:

1) Transforms of $\mathbf{X}^0$, $\mathbf{X}^2$, $\mathbf{X}^4$, ..., etc.: In cycles 0 ~ 7, the input data vector $\mathbf{X}^0$ is loaded into RAM-1 and RAM-2 on a sample-by-

sample basis. After this loading operation, the first-stage matrix-vector multiplications of the FHT algorithm, i.e., $\mathbf{A}^0=[a_0^0\ a_1^0\ ...\ a_7^0]^T=\mathbf{P}_1(8)\mathbf{X}^0$, is performed during cycles 8 ~ 15, and the second-stage matrix-vector multiplication, i.e., $\mathbf{B}^0=[b_0^0\ b_1^0\ ...\ b_7^0]^T=\mathbf{P}_2(4)\mathbf{A}^0$, is performed during cycles 16 ~ 23, where the generating order of $\mathbf{A}^0$ and $\mathbf{B}^0$ are $\{a_0^0, a_1^0, a_2^0, a_3^0, a_4^0, a_5^0, a_6^0, a_7^0\}$ and $\{b_0^0, b_1^0, b_4^0, b_5^0, b_2^0, b_3^0, b_6^0, b_7^0\}$ respectively. In cycles 24 ~ 31, the third-stage matrix-vector multiplication, i.e., $\mathbf{Z}^0=[z_0^0\ z_1^0\ ...\ z_7^0]^T= \mathbf{P}_4(2)\mathbf{B}^0$, is computed to yield the transform results of $\mathbf{X}^0$ in the order $\{z_0^0, z_2^0, z_4^0, z_6^0, z_1^0, z_3^0, z_5^0, z_7^0\}$. During cycles 32 ~ 35, the processor carries out the transform results of $\mathbf{X}^0$ at ***Output*1** and ***Output*2**. The input data vector $\mathbf{X}^2$ begins to be loaded into RAM-1 and RAM-2 at cycle 32. Note that the addressing/control sequences for computing the transforms of $\mathbf{X}^2$, $\mathbf{X}^4$, ... are the same as those for computing the transform of $\mathbf{X}^0$.

2) Transforms of $\mathbf{X}^1$, $\mathbf{X}^3$, $\mathbf{X}^5$, … , etc.: Except the loading operations, the transform computations of $\mathbf{X}^1$, $\mathbf{X}^3$, ... are almost the same as those described above. For example, during cycles 8 ~ 11, the first four samples of the input data vector $\mathbf{X}^1$, i.e., $\{x_0^1, x_1^1, x_2^1, x_3^1\}$, are loaded into RAM-3; then, the other four samples of $\mathbf{X}^1$, i.e., $\{x_4^1, x_5^1, x_6^1, x_7^1\}$, and $\{x_0^1, x_1^1, x_2^1, x_3^1\}$ just stored on RAM-3 are used to generate the first four results of $\mathbf{A}^1=[a_0^1\ a_1^1\ ...\ a_7^1]^T=\mathbf{P}_1(8)\mathbf{X}^1$.
3) Table I shows the DHT computation of an 8-point real sequence. Note that the DHT output sequence is actually the IDFT of the 8-point complex sequence $\{c_0^i, c_1^i+jd_1^i, c_2^i+jd_2^i, c_3^i+jd_3^i, c_4^i, c_3^i-jd_3^i, c_2^i-jd_2^i, c_1^i-jd_1^i\}$ when the input sequence is $\{c_0^i, c_1^i-d_1^i, c_2^i-d_2^i, c_3^i-d_3^i, c_4^i, c_3^i+d_3^i, c_2^i+d_2^i, c_1^i+d_1^i\}$.
4) If the selection signal I_5 of MUX-5 is {1, 0, 0, 0, 2, 0, 0, 0} from cycle 32 to cycle 39, the processor will evaluate the DFT samples of $\mathbf{X}^i$, i.e., $f_0^i=y_0^i=z_0^i$, $f_1^{i*}=f_7^i=(y_1^i+y_7^i)/2+j(y_1^i-y_7^i)/2=(z_4^i+z_7^i)/2+j(z_4^i-z_7^i)/2$, $f_2^{i*}=f_6^i=(y_2^i+y_6^i)/2+j(y_2^i-y_6^i)/2=(z_2^i+z_3^i)/2+j(z_2^i-z_3^i)/2$, $f_3^{i*}= f_5^i=(y_3^i+y_5^i)/2+j(y_3^i-y_5^i)/2=(z_6^i+z_5^i)/2+j(z_6^i-z_5^i)/2$, and $f_4^i=y_4^i=z_1^i$ will be computed. Table II presents the required addressing/control sequences for this case.

We can see from Table I that the proposed architecture realizes the N-point FFT/IFFT algorithm in $N(\log_2 N+1)/2$ cycle periods. Moreover, the addressing/control sequences can easily be derived from a $\log_2 N$-bit counter. For N=8, the output bits of a 3-bit counter can be used as the **RA** sequences of RAM-1 to RAM-4. The control sequences of multiplexers and the **WA** sequences of RAM-1 to RAM-4 can also be derived from the output bits of a 3-bit counter.

4. CONCLUSION

A new VLSI architecture has been proposed for the N-point DFT/IDFT computation of real data based on the FHT, where N is a power of two. It consumes only two real multipliers, six N/2-word two-port RAM's, three real adders, two ROM's, and some simple logic circuits, and is pretty suitable for single-chip implementation. As compared to a similar approach presented in [11] with throughput performance of one transform sample per $\log_2 N$+1 cycles, the proposed one provides a double throughput with an increase of one multiplier and two N/2-word RAM's. Based on 0.35 *m*m CMOS technology, the estimated gate count of the proposed design is around 60 000 for the case of N=512, and the allowable clock rate is as high as 100 MHz, meaning a throughput of 20M transform samples per second. The processing speed will be higher if more advanced CMOS technology is adopted to implement the same circuit. The low-complexity and high-throughput feature makes the proposed architecture attractive for use in DMT-based VDSL applications.

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] ANSI Standard T1.413-1995, Asymmetric Digital Subscriber Line (ADSL) Metallic Interface, New York, NY, 1995.

[3] J. M. Cioffi, "Asymmetric digital subscriber lines," Chapter 34, The Communications Handbook, J. D. Gibson Ed., CRC Press, 1997.

[4] Very-high-speed Digital Subscriber Lines: System Requirements (T1E1.4/98-043R2), Draft Technical Document – Revision 14A, ANSI T1E1.4, May 1998.

[5] VDSL Alliance SDMT VDSL Draft Standard Proposal, ETSI STC/TM6, April 1998.

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

[7] H. V. Sorensen, D. L. Jones, C. S. Burrus, and M. T. Heideman, "On computing the discrete Hartley transform," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-33, pp. 1231-1238, Oct. 1985.

[8] H. S. Hou, "The fast Hartley transform algorithm," *IEEE Trans. Comput.*, vol. C-36, pp. 147-156, Feb. 1987.

[9] L. -W. Chang and S. -W. Lee, "Systolic arrays for the discrete Hartley transform," *IEEE Trans. Signal Processing*, vol. 39, pp. 2411-2418, Nov. 1991.

[10] C. -L. Wang, C. -T. Ho, and Y. -T. Chang, "A novel systolic design for fast computation of the discrete Hartley transform," in *Proc. IEEE Thirtieth Asilomar Conf. Signals, Systems & Computers*, pp. 1067-1071, Nov. 1996.

[11] C. -L. Wang and C. -H Chang, "A novel DHT-based FFT/ IFFT processor for ADSL transceivers," in *Proc. 1999 IEEE Int. Symp. Circuits Syst.*, Orlando, FL, May 30 - June 2, 1999, pp. 51-55.

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

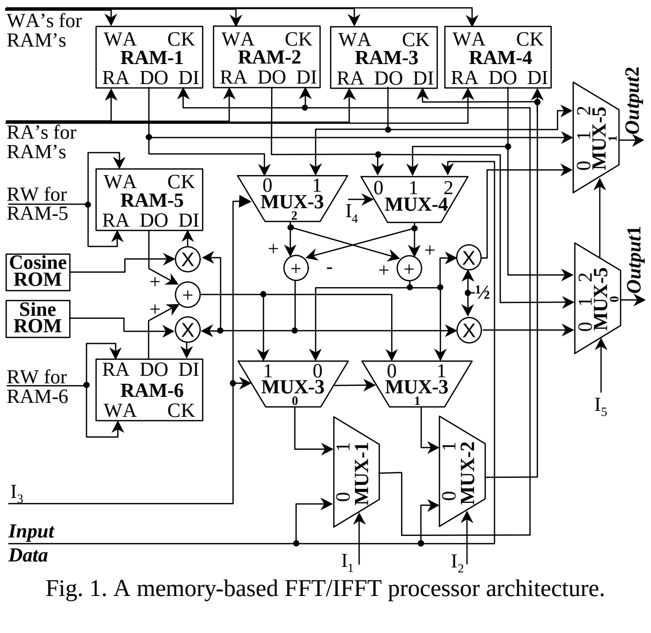

Fig. 1. A memory-based FFT/IFFT processor architecture.

TABLE I Arrangement of the Control/Addressing Sequences for the 8-Point FHT Algorithm (- : means “neglected”)

Cy.	I_1 I_2 I_3 I_4 I_5	RAM-1 or -3		RAM-2 or -4		RAM-1 or -2		RAM-3 or -4		RAM-5			RAM-6			Output	
		RA	DO	RA	DO	WA	DI	WA	DI	RW	DI	DO	RW	DI	DO	*Output* 2	*Output* 1
0	0 1 0 0 1	000	-	000	-	000	x_0^0	100	-	00	-	-	00	-	-	-	-
1	0 1 0 0 1	001	-	001	-	001	x_1^0	101	-	01	-	-	01	-	-	-	-
2	0 1 0 0 1	010	-	010	-	010	x_2^0	110	-	10	-	-	10	-	-	-	-
3	0 1 0 0 1	011	-	011	-	011	x_3^0	111	-	11	-	-	11	-	-	-	-
4	0 1 1 1 2	100	-	100	-	100	x_4^0	000	-	00	-	-	00	-	-	-	-
5	0 1 1 1 2	101	-	101	-	101	x_5^0	001	-	01	-	-	01	-	-	-	-
6	0 1 1 1 2	110	-	110	-	110	x_6^0	010	-	10	-	-	10	-	-	-	-
7	0 1 1 1 2	111	-	111	-	111	x_7^0	011	-	11	-	-	11	-	-	-	-
8	1 0 0 0 -	000	x_0^0	000	x_4^0	000	$x_0^0+x_4^0=a_0^0$	000	x_0^1	00	$(x_0^0-x_4^0)\cdot C_8^0$	-	00	$(x_0^0-x_4^0)\cdot S_8^0$	-	-	-
9	1 0 0 0 -	001	x_1^0	001	x_5^0	001	$x_1^0+x_5^0=a_1^0$	001	x_1^1	01	$(x_1^0-x_5^0)\cdot C_8^1$	-	01	$(x_1^0-x_5^0)\cdot S_8^3$	-	-	-
10	1 0 0 0 -	010	x_2^0	010	x_6^0	100	$x_2^0+x_6^0=a_2^0$	010	x_2^1	10	$(x_2^0-x_6^0)\cdot C_8^2$	-	10	$(x_2^0-x_6^0)\cdot S_8^2$	-	-	-
11	1 0 0 0 -	011	x_3^0	011	x_7^0	101	$x_3^0+x_7^0=a_3^0$	011	x_3^1	11	$(x_3^0-x_7^0)\cdot C_8^3$	-	11	$(x_3^0-x_7^0)\cdot S_8^1$	-	-	-
12	1 1 1 2 -	100	x_0^1	100	-	010	$(x_0^0-x_4^0)\cdot C_8^0+(x_0^0-x_4^0)\cdot S_8^0=a_4^0$	000	$x_0^1+x_4^1=a_0^1$	00	$(x_0^1-x_4^1)\cdot C_8^0$	$(x_0^0-x_4^0)\cdot C_8^0$	00	$(x_0^1-x_4^1)\cdot S_8^0$	$(x_0^0-x_4^0)\cdot S_8^0$	-	-
13	1 1 1 2 -	101	x_1^1	101	-	011	$(x_1^0-x_5^0)\cdot C_8^1+(x_3^0-x_7^0)\cdot S_8^1=a_5^0$	001	$x_1^1+x_5^1=a_1^1$	01	$(x_1^1-x_5^1)\cdot C_8^1$	$(x_1^0-x_5^0)\cdot C_8^1$	11	$(x_1^1-x_5^1)\cdot S_8^3$	$(x_3^0-x_7^0)\cdot S_8^1$	-	-
14	1 1 1 2 -	110	x_2^1	110	-	110	$(x_2^0-x_6^0)\cdot C_8^2+(x_2^0-x_6^0)\cdot S_8^2=a_6^0$	100	$x_2^1+x_6^1=a_2^1$	10	$(x_2^1-x_6^1)\cdot C_8^2$	$(x_2^0-x_6^0)\cdot C_8^2$	10	$(x_2^1-x_6^1)\cdot S_8^2$	$(x_2^0-x_6^0)\cdot S_8^2$	-	-
15	1 1 1 2 -	111	x_3^1	111	-	111	$(x_3^0-x_7^0)\cdot C_8^3+(x_1^0-x_5^0)\cdot S_8^3=a_7^0$	101	$x_3^1+x_7^1=a_3^1$	11	$(x_3^1-x_7^1)\cdot C_8^3$	$(x_3^0-x_7^0)\cdot C_8^3$	01	$(x_3^1-x_7^1)\cdot S_8^1$	$(x_1^0-x_5^0)\cdot S_8^3$	-	-
16	1 1 0 0 -	000	a_0^0	000	a_2^0	000	$a_0^0+a_2^0=b_0^0$	010	$(x_0^1-x_4^1)\cdot C_8^0+(x_0^1-x_4^1)\cdot S_8^0=a_4^1$	00	$(a_0^0-a_2^0)\cdot C_4^0$	$(x_0^1-x_4^1)\cdot C_8^0$	00	$(a_0^0-a_2^0)\cdot S_4^0$	$(x_0^1-x_4^1)\cdot S_8^0$	-	-
17	1 1 0 0 -	001	a_1^0	001	a_3^0	100	$a_1^0+a_3^0=b_1^0$	011	$(x_1^1-x_5^1)\cdot C_8^1+(x_3^1-x_7^1)\cdot S_8^1=a_5^1$	01	$(a_1^0-a_3^0)\cdot C_4^1$	$(x_1^1-x_5^1)\cdot C_8^1$	01	$(a_1^0-a_3^0)\cdot S_4^1$	$(x_3^1-x_7^1)\cdot S_8^1$	-	-
18	1 1 0 0 -	010	a_4^0	010	a_6^0	010	$a_4^0+a_6^0=b_4^0$	110	$(x_2^1-x_6^1)\cdot C_8^2+(x_2^1-x_6^1)\cdot S_8^2=a_6^1$	10	$(a_4^0-a_6^0)\cdot C_4^0$	$(x_2^1-x_6^1)\cdot C_8^2$	10	$(a_4^0-a_6^0)\cdot S_4^0$	$(x_2^1-x_6^1)\cdot S_8^2$	-	-
19	1 1 0 0 -	011	a_5^0	011	a_7^0	110	$a_5^0+a_7^0=b_5^0$	111	$(x_3^1-x_7^1)\cdot C_8^3+(x_1^1-x_5^1)\cdot S_8^3=a_7^1$	11	$(a_5^0-a_7^0)\cdot C_4^1$	$(x_3^1-x_7^1)\cdot C_8^3$	11	$(a_5^0-a_7^0)\cdot S_4^1$	$(x_1^1-x_5^1)\cdot S_8^3$	-	-
20	1 1 1 1 -	100	a_0^1	100	a_2^1	001	$(a_0^0-a_2^0)\cdot C_4^0+(a_0^0-a_2^0)\cdot S_4^0=b_2^0$	000	$a_0^1+a_2^1=b_0^1$	00	$(a_0^1-a_2^1)\cdot C_4^0$	$(a_0^0-a_2^0)\cdot C_4^0$	00	$(a_0^1-a_2^1)\cdot S_4^0$	$(a_0^0-a_2^0)\cdot S_4^0$	-	-
21	1 1 1 1 -	101	a_1^1	101	a_3^1	101	$(a_1^0-a_3^0)\cdot C_4^1+(a_1^0-a_3^0)\cdot S_4^1=b_3^0$	100	$a_1^1+a_3^1=b_1^1$	01	$(a_1^1-a_3^1)\cdot C_4^1$	$(a_1^0-a_3^0)\cdot C_4^1$	01	$(a_1^1-a_3^1)\cdot S_4^1$	$(a_1^0-a_3^0)\cdot S_4^1$	-	-
22	1 1 1 1 -	110	a_4^1	110	a_6^1	011	$(a_4^0-a_6^0)\cdot C_4^0+(a_4^0-a_6^0)\cdot S_4^0=b_6^0$	010	$a_4^1+a_6^1=b_4^1$	10	$(a_4^1-a_6^1)\cdot C_4^0$	$(a_4^0-a_6^0)\cdot C_4^0$	10	$(a_4^1-a_6^1)\cdot S_4^0$	$(a_4^0-a_6^0)\cdot S_4^0$	-	-
23	1 1 1 1 -	111	a_5^1	111	a_7^1	111	$(a_5^0-a_7^0)\cdot C_4^1+(a_5^0-a_7^0)\cdot S_4^1=b_7^0$	110	$a_5^1+a_7^1=b_5^1$	11	$(a_5^1-a_7^1)\cdot C_4^1$	$(a_5^0-a_7^0)\cdot C_4^1$	11	$(a_5^1-a_7^1)\cdot S_4^1$	$(a_5^0-a_7^0)\cdot S_4^1$	-	-
24	1 1 0 0 -	000	b_0^0	000	b_1^0	000	$b_0^0+b_1^0=z_0^0$	001	$(a_0^1-a_2^1)\cdot C_4^0+(a_0^1-a_2^1)\cdot S_4^0=b_2^1$	00	$(b_0^0-b_1^0)\cdot C_2^0$	$(a_0^1-a_2^1)\cdot C_4^0$	00	$(b_0^0-b_1^0)\cdot S_2^0$	$(a_0^1-a_2^1)\cdot S_4^0$	-	-
26	1 1 0 0 -	001	b_2^0	001	b_3^0	001	$b_2^0+b_3^0=z_2^0$	101	$(a_1^1-a_3^1)\cdot C_4^1+(a_1^1-a_3^1)\cdot S_4^1=b_3^1$	01	$(b_2^0-b_3^0)\cdot C_2^0$	$(a_1^1-a_3^1)\cdot C_4^1$	01	$(b_2^0-b_3^0)\cdot S_2^0$	$(a_1^1-a_3^1)\cdot S_4^1$	-	-
28	1 1 0 0 -	010	b_4^0	010	b_5^0	010	$b_4^0+b_5^0=z_4^0$	011	$(a_4^1-a_6^1)\cdot C_4^0+(a_4^1-a_6^1)\cdot S_4^0=b_6^1$	10	$(b_4^0-b_5^0)\cdot C_2^0$	$(a_4^1-a_6^1)\cdot C_4^0$	10	$(b_4^0-b_5^0)\cdot S_2^0$	$(a_4^1-a_6^1)\cdot S_4^0$	-	-
30	1 1 0 0 -	011	b_6^0	011	b_7^0	011	$b_6^0+b_7^0=z_6^0$	111	$(a_5^1-a_7^1)\cdot C_4^1+(a_5^1-a_7^1)\cdot S_4^1=b_7^1$	11	$(b_6^0-b_7^0)\cdot C_2^0$	$(a_5^1-a_7^1)\cdot C_4^1$	11	$(b_6^0-b_7^0)\cdot S_2^0$	$(a_5^1-a_7^1)\cdot S_4^1$	-	-
25	1 1 1 1 -	100	b_0^1	100	b_1^1	100	$(b_0^0-b_1^0)\cdot C_2^0+(b_0^0-b_1^0)\cdot S_2^0=z_1^0$	000	$b_0^1+b_1^1=z_0^1$	00	$(b_0^1-b_1^1)\cdot C_2^0$	$(b_0^0-b_1^0)\cdot C_2^0$	00	$(b_0^1-b_1^1)\cdot S_2^0$	$(b_0^0-b_1^0)\cdot S_2^0$	-	-
27	1 1 1 1 -	101	b_2^1	101	b_3^1	101	$(b_2^0-b_3^0)\cdot C_2^0+(b_2^0-b_3^0)\cdot S_2^0=z_3^0$	001	$b_2^1+b_3^1=z_2^1$	01	$(b_2^1-b_3^1)\cdot C_2^0$	$(b_2^0-b_3^0)\cdot C_2^0$	01	$(b_2^1-b_3^1)\cdot S_2^0$	$(b_2^0-b_3^0)\cdot S_2^0$	-	-
29	1 1 1 1 -	110	b_4^1	110	b_5^1	110	$(b_4^0-b_5^0)\cdot C_2^0+(b_4^0-b_5^0)\cdot S_2^0=z_5^0$	010	$b_4^1+b_5^1=z_4^1$	10	$(b_4^1-b_5^1)\cdot C_2^0$	$(b_4^0-b_5^0)\cdot C_2^0$	10	$(b_4^1-b_5^1)\cdot S_2^0$	$(b_4^0-b_5^0)\cdot S_2^0$	-	-
31	1 1 1 1 -	111	b_6^1	111	b_7^1	111	$(b_6^0-b_7^0)\cdot C_2^0+(b_6^0-b_7^0)\cdot S_2^0=z_7^0$	011	$b_6^1+b_7^1=z_6^1$	11	$(b_6^1-b_7^1)\cdot C_2^0$	$(b_6^0-b_7^0)\cdot C_2^0$	11	$(b_6^1-b_7^1)\cdot S_2^0$	$(b_6^0-b_7^0)\cdot S_2^0$	-	-
32	0 1 0 0 1	000	z_0^0	000	z_1^0	000	x_0^2	100	$(b_0^1-b_1^1)\cdot C_2^0+(b_0^1-b_1^1)\cdot S_2^0=z_1^1$	00	-	$(b_0^1-b_1^1)\cdot C_2^0$	00	-	$(b_0^1-b_1^1)\cdot S_2^0$	z_0^0	z_1^0
34	0 1 0 0 1	001	z_2^0	001	z_3^0	001	x_1^2	101	$(b_2^1-b_3^1)\cdot C_2^0+(b_2^1-b_3^1)\cdot S_2^0=z_3^1$	01	-	$(b_2^1-b_3^1)\cdot C_2^0$	01	-	$(b_2^1-b_3^1)\cdot S_2^0$	z_2^0	z_3^0
33	0 1 0 0 1	010	z_4^0	010	z_5^0	010	x_2^2	110	$(b_4^1-b_5^1)\cdot C_2^0+(b_4^1-b_5^1)\cdot S_2^0=z_5^1$	10	-	$(b_4^1-b_5^1)\cdot C_2^0$	10	-	$(b_4^1-b_5^1)\cdot S_2^0$	z_4^0	z_5^0
35	0 1 0 0 1	011	z_6^0	011	z_7^0	011	x_3^2	111	$(b_6^1-b_7^1)\cdot C_2^0+(b_6^1-b_7^1)\cdot S_2^0=z_7^1$	11	-	$(b_6^1-b_7^1)\cdot C_2^0$	11	-	$(b_6^1-b_7^1)\cdot S_2^0$	z_6^0	z_7^0
36	0 1 1 1 2	100	z_0^1	100	z_1^1	100	x_4^2	000	-	00	-	-	00	-	-	z_0^1	z_1^1
37	0 1 1 1 2	101	z_2^1	101	z_3^1	101	x_5^2	001	-	01	-	-	01	-	-	z_2^1	z_3^1
38	0 1 1 1 2	110	z_4^1	110	z_5^1	110	x_6^2	010	-	10	-	-	10	-	-	z_4^1	z_5^1
39	0 1 1 1 2	111	z_6^1	111	z_7^1	111	x_7^2	011	-	11	-	-	11	-	-	z_6^1	z_7^1
40	1 0 0 0 -	000	x_0^2	000	x_4^2	000	$x_0^2+x_4^2=a_0^2$	000	x_0^3	00	$(x_0^2-x_4^2)\cdot C_8^0$	-	00	$(x_0^2-x_4^2)\cdot S_8^0$	-	-	-
41	1 0 0 0 -	001	x_1^2	001	x_5^2	001	$x_1^2+x_5^2=a_1^2$	001	x_1^3	01	$(x_1^2-x_5^2)\cdot C_8^1$	-	01	$(x_1^2-x_5^2)\cdot S_8^3$	-	-	-
42	1 0 0 0 -	010	x_2^2	010	x_6^2	100	$x_2^2+x_6^2=a_2^2$	010	x_2^3	10	$(x_2^2-x_6^2)\cdot C_8^2$	-	10	$(x_2^2-x_6^2)\cdot S_8^2$	-	-	-
43	1 0 0 0 -	011	x_3^2	011	x_7^2	101	$x_3^2+x_7^2=a_3^2$	011	x_3^3	11	$(x_3^2-x_7^2)\cdot C_8^3$	-	11	$(x_3^2-x_7^2)\cdot S_8^1$	-	-	-
44	1 1 1 2 -	100	x_0^3	100	-	010	$(x_0^2-x_4^2)\cdot C_8^0+(x_0^2-x_4^2)\cdot S_8^0=a_4^2$	000	$x_0^3+x_4^3=a_0^3$	00	$(x_0^3-x_4^3)\cdot C_8^0$	$(x_0^2-x_4^2)\cdot C_8^0$	00	$(x_0^3-x_4^3)\cdot S_8^0$	$(x_0^2-x_4^2)\cdot S_8^0$	-	-
45	1 1 1 2 -	101	x_1^3	101	-	011	$(x_1^2-x_5^2)\cdot C_8^1+(x_3^2-x_7^2)\cdot S_8^1=a_5^2$	001	$x_1^3+x_5^3=a_1^3$	01	$(x_1^3-x_5^3)\cdot C_8^1$	$(x_1^2-x_5^2)\cdot C_8^1$	11	$(x_1^3-x_5^3)\cdot S_8^3$	$(x_3^2-x_7^2)\cdot S_8^1$	-	-
46	1 1 1 2 -	110	x_2^3	110	-	110	$(x_2^2-x_6^2)\cdot C_8^2+(x_2^2-x_6^2)\cdot S_8^2=a_6^2$	100	$x_2^3+x_6^3=a_2^3$	10	$(x_2^3-x_6^3)\cdot C_8^2$	$(x_2^2-x_6^2)\cdot C_8^2$	10	$(x_2^3-x_6^3)\cdot S_8^2$	$(x_2^2-x_6^2)\cdot S_8^2$	-	-
47	1 1 1 2 -	111	x_3^3	111	-	111	$(x_3^2-x_7^2)\cdot C_8^3+(x_1^2-x_5^2)\cdot S_8^3=a_7^2$	101	$x_3^3+x_7^3=a_3^3$	11	$(x_3^3-x_7^3)\cdot C_8^3$	$(x_3^2-x_7^2)\cdot C_8^3$	01	$(x_3^3-x_7^3)\cdot S_8^1$	$(x_1^2-x_5^2)\cdot S_8^3$	-	-
:	: : : : :	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:

TABLE II Arrangement of the Control/Addressing Sequences in Cycles 32 ~ 39 of Table I to Generate the FFT’s of X^0 and X^1

Cy.	I_1 I_2 I_3 I_4 I_5	RAM-1 or -3		RAM-2 or -4		RAM-1 or -2		RAM-3 or -4		RAM-5			RAM-6			Output	
		RA	DO	RA	DO	WA	DI	WA	DI	RW	DI	DO	RW	DI	DO	*Output* 2	*Output* 1
32	0 1 0 0 1	000	z_0^0	000	z_1^0	000	x_0^2	100	$(b_0^1-b_1^1)\cdot C_2^0+(b_0^1-b_1^1)\cdot S_2^0=z_1^1$	00	-	$(b_0^0-b_1^0)\cdot C_2^0$	00	-	$(b_0^0-b_1^0)\cdot S_2^0$	z_0^0	z_1^0
34	0 1 0 0 0	001	z_2^0	001	z_3^0	001	x_1^2	101	$(b_2^1-b_3^1)\cdot C_2^0+(b_2^1-b_3^1)\cdot S_2^0=z_3^1$	01	-	$(b_2^0-b_3^0)\cdot C_2^0$	01	-	$(b_2^0-b_3^0)\cdot S_2^0$	$(z_2^0+z_3^0)/2$	$(z_2^0-z_3^0)/2$
33	0 1 0 0 0	010	z_4^0	011	z_7^0	010	x_2^2	110	$(b_4^1-b_5^1)\cdot C_2^0+(b_4^1-b_5^1)\cdot S_2^0=z_5^1$	10	-	$(b_4^0-b_5^0)\cdot C_2^0$	10	-	$(b_4^0-b_5^0)\cdot S_2^0$	$(z_4^0+z_7^0)/2$	$(z_4^0-z_7^0)/2$
35	0 1 0 0 0	011	z_6^0	010	z_5^0	011	x_3^2	111	$(b_6^1-b_7^1)\cdot C_2^0+(b_6^1-b_7^1)\cdot S_2^0=z_7^1$	11	-	$(b_6^0-b_7^0)\cdot C_2^0$	11	-	$(b_6^0-b_7^0)\cdot S_2^0$	$(z_6^0+z_5^0)/2$	$(z_6^0-z_5^0)/2$
36	0 1 1 1 2	100	z_0^1	100	z_1^1	100	x_4^2	000	-	00	-	-	00	-	-	z_0^1	z_1^1
37	0 1 1 1 0	101	z_2^1	101	z_3^1	101	x_5^2	001	-	01	-	-	01	-	-	$(z_2^1+z_3^1)/2$	$(z_2^1-z_3^1)/2$
38	0 1 1 1 0	110	z_4^1	111	z_7^1	110	x_6^2	010	-	10	-	-	10	-	-	$(z_4^1+z_7^1)/2$	$(z_4^1-z_7^1)/2$
39	0 1 1 1 0	111	z_6^1	110	z_5^1	111	x_7^2	011	-	11	-	-	11	-	-	$(z_6^1+z_5^1)/2$	$(z_6^1-z_5^1)/2$