A VLSI Array Processor for 16-Point FFT

Moon-Key Lee, *Member, IEEE*, Kyung-Wook Shin, and Jang-Kyu Lee

Abstract—**This paper describes an implementation of a two-dimensional array processor for fast Fourier transform (FFT) using a 2-μm CMOS technology. The array processor, which is dedicated to 16-point FFT, implements a 4×4 mesh array of 16 processing elements (PE's) working in parallel. Design considerations both in chip level and in PE level have been examined. A layout design methodology based on bit-slice units (BSU's) results in a very simple design and easy debugging, as well as a regular interconnection scheme through abutment. It contains about 48 000 transistors on an area of 53.52 mm², excluding the 83-pad area, and operation is on a 15-MHz clock. The array processor performs 24.6 million complex multiplications per second, and computes a 16-point FFT in 3 μs.**

I. Introduction

RECENT advances in very large scale integration (VLSI) technology make the implementation of array processors technologically realizable and economically feasible. So, a new approach adopting algorithm-based VLSI array processors is becoming increasingly competitive for real-time digital signal processing (DSP) and image processing applications [1].

The fast Fourier transform (FFT) is probably one of the most important algorithms in DSP applications. There are two approaches for computing this transform: software implemented on a programmable DSP [2], [3], [23], [24], and dedicated FFT processor development [4]–[6]. Real-time DSP favors the use of the latter, which offers parallel processing capability.

In this paper, a dedicated array processor implementation is described. As known, the FFT algorithm requires global data exchanges for butterfly computation, thus the direct implementation of the FFT flow graph onto silicon is considered to be inefficient because of its inherent communication cost in terms of area and time. To achieve area-efficient implementation of the FFT with a high degree of regularity and parallelism, we utilize two-dimensional mesh array of processing elements (PE's).

This paper is organized as follows. Section II describes design considerations. Then an algorithm to compute FFT using half-butterfly arithmetic (HBA) will be presented in Section III. Section IV will describe the circuit and layout design. Finally, fabrication and some measurement results will be discussed in Section V.

Manuscript received May 21, 1990; revised April 22, 1991.

M.-K. Lee is with the VLSI and CAD Laboratory, Department of Electronic Engineering, Yonsei University, Seoul, 120-749, Korea.

K.-W. Shin was with the Department of Electronic Engineering, Yonsei University, Seoul, 120-749, Korea. He is now with the Department of Electronic Engineering, Kum Oh Institute of Technology, 188 Shinpyung-Dong, Kumi City, Kyungbuk 730-701, Korea.

J.-K. Lee was with the Department of Electronic Engineering, Yonsei University, Seoul, 120-749, Korea. He is now with the Semiconductor Division, Samsung Electronics, Inc., Kyungki Do, 449-900, Korea.

IEEE Log Number 9101784.

II. Design Considerations

This section discusses design considerations for our FFT array processor including chip architecture, the arithmetic function of a PE, and time complexity.

A. Chip Architecture

A number of approaches for FFT implementation are known including an FFT network [7], a perfect-shuffle network [8], a 2-D mesh array [9]–[12], pipelined architectures utilizing delay commutator [13] or systolic elevator [14], and a 1-D systolic DFT array [15]. Table I compares these architectures in terms of area, time, and regularity. From Table I, it is easily found that the FFT network and the perfect-shuffle network are not suitable for VLSI realization since irregular interconnections between PE's dominate the chip area. The pipelined architectures utilizing delay commutator or systolic elevator require additional hardware overhead of $O(N \log N)$ for the data shuffling of N-point FFT.

Our design goal in the architectural level is to determine a parallel architecture satisfying the following criteria: 1) highly parallel processing capability; 2) architectural regularity; 3) local data communications; and 4) no hardware overhead for data shuffling operation. Since the minimization of chip area with architectural regularity is considered much more important than any other factors in VLSI design, we choose the 2-D mesh array shown in Fig. 1(a) as the chip architecture.

B. Arithmetic Function of PE

In the implementation of a 2-D mesh array for FFT, careful examinations into the trade-offs between the number of PE's, the arithmetic function of PE, and control complexity should be made.

When N-point FFT is computed using a 2-D mesh array having M (where $2M \leqslant = N$) PE's whose arithmetic functions are the usual radix-2 butterfly [22], each PE requires $P = N/M$ data storages, and each datum should be identified using $O(\log_2 N)$ bits of address in order to control data shuffling operations through $\log_2 N$ butterfly stages. In this case, all PE's participate in butterfly computations. Note that a 2-D mesh array of $N/2$

TABLE I
COMPARISON OF VARIOUS PARALLEL ARCHITECTURES FOR FFT COMPUTATION

		FFT network [7]	Perfect shuffle network [8]	2-D mesh array [9]–[12]	Parallel arch. using DC's [13]	Parallel arch. using SE's [14]	1-D systolic DFT array [15]
Number of PE's		$(N/2)\log N$	$N/2$	N	$\log N$	$N \log N$	$2N-1$
Butterfly steps		pipelined	$\log N$	$\log N$	pipelined	pipelined	pipelined
Interconnection area		$O(N)$	$O(N^2/\log^2 N)$	No area	No area	No area	No area
Data shuffling time		$O(\log N)$	$O(\log N)$	$O(\sqrt{N})$	$O(N)$	$O(N)$	$O(N)$
Interconnection regularity		irregular	irregular	very regular	regular	regular	very regular
Additional hardware for data shuffling		No	No	No	$O(N \log N)$ DC's	$O(N \log N)$ SE's	No
Dominant factor	Time	Pipelined	Butterfly steps	Data shuffling	pipelined	pipelined	pipelined
	Area	interconnection	interconnection	PE's	PE's + DC's	PE's + SE's	PE's

PE's: Processing Elements, DC's: Delay Commutators, SE's: Systolic Elevators

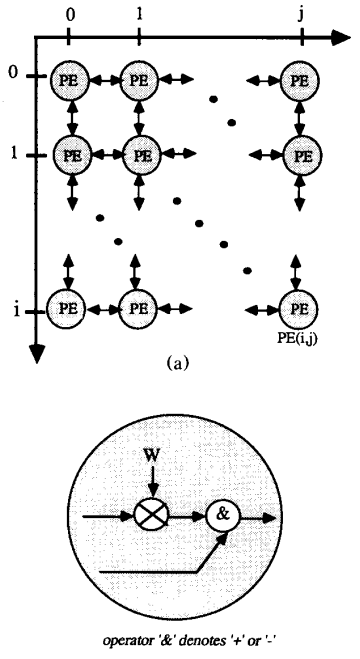


Fig. 1. (a) Two-dimensional mesh array for FFT computation. (b) Arithmetic function of a PE.

PE's for N-point FFT computation is considered to be a special case with $P = 2$.

On the other hand, a 2-D mesh array composed of N PE's whose arithmetic function is also the usual radix-2 butterfly [19] can perform data shuffling operation without $O(\log_2 N)$ bits of address. It, however, reveals some inefficiencies as follows: a) it requires data reshuffling operations after butterfly computation; and b) only half of the PE's in the array participate in butterfly computation.

In order to resolve these problems inherent in 2-D mesh array implementation for FFT, we make a compromise by adopting half-butterfly arithmetic (HBA) [16] as the arithmetic function of PE (see Fig. 1(b)), rather than the usual radix-2 butterfly. The HBA takes its name from the definition given in (1), which represents half of the conventional radix-2 butterfly:

$$\text{HBA\&} = f(q-1,k)\,\&\,f(q-1,k+D(q))\cdot W^{pt} \quad (1)$$

where

$$q = 1,2,3,\cdots,\log_2 N$$

$$W^{pt} = \exp(-j2\pi pt/N).$$

In (1), the operator "&" denotes either "+" (addition) or "−" (subtraction) and $D(q)$ represents the shuffling distance of the qth butterfly stage. An algorithm for computing FFT using HBA will be described in Section III.

The adoption of HBA as the arithmetic function of PE results in a simple control which does not require $O(\log_2 N)$ bits of address, as well as 100% PE utilization (i.e., no idle PE's during butterfly computation) and no data reshuffling. These facts reveal the merits of using the HBA concept.

C. Time Complexity

The time required to compute N-point FFT can be evaluated in terms of total time for data shuffling and butterfly arithmetic through $\log_2 N$ butterfly stages. A summation of the time contributions of all shuffling distances will give the total shuffling time of $2(\sqrt{N}-1)T_s$, if a unit shuffling distance takes time T_s. Also, the total time for butterfly arithmetic will be given by $T_{hb}\log_2 N$, where T_{hb} represents the times for HBA. Both T_{hb} and T_s can be considered to be a unit clock if the HBA is implemented with parallel arithmetic.

Asymptotically, however, $2(\sqrt{N}-1)T_s + T_{hb}\log_2 N$ can be reduced to $O(\sqrt{N})$, since T_s and T_{hb} are independent of N. This fact suggests that a bit-parallel communication scheme between PE's is preferable.

III. ALGORITHM FOR FFT COMPUTATION ON THE ARCHITECTURE

This section describes an algorithm for FFT computation using HBA. A procedure for computing a 16-point

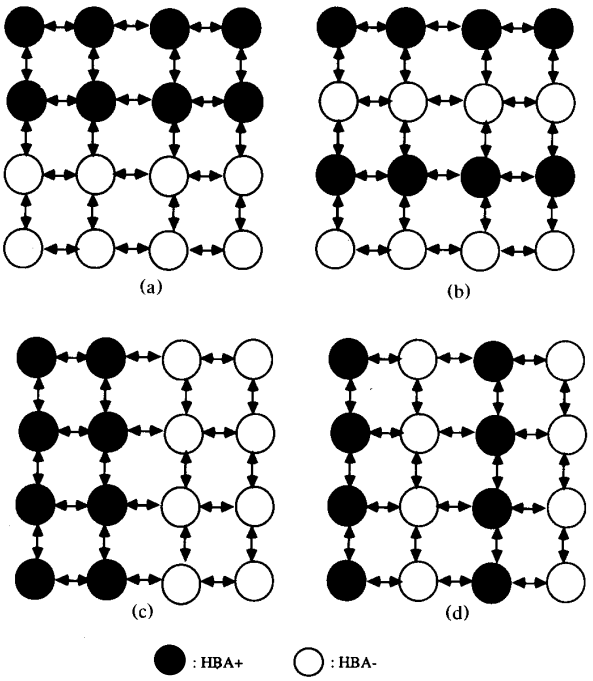


Fig. 2. Sixteen-point FFT computation using HBA: (a) first butterfly stage ($q = 1$), (b) second butterfly stage ($q = 2$), (c) third butterfly stage ($q = 3$), and (d) fourth butterfly stage ($q = 4$).

FFT is described as follows:

BEGIN
  FOR q = 1 TO 4 DO IN PARALLEL for all PE's
    BEGIN
      data shuffling (ALGORITHM I)
      HBA computation (ALGORITHM II)
    END
END.

Initially, 16 input data are loaded into the array in row-major order. Then, data shuffling and HBA computation are repeated through four (i.e., $\log_2 16$) butterfly stages according to the Algorithms I and II, respectively (see the Appendix). After the fourth butterfly stage has been completed, the array pipes out the results and accepts new input data in parallel and pipelined fashion through each of the four boundary PE's.

Fig. 2 visualizes these procedures. In the first butterfly stage (Fig. 2(a)), PE's in the first row send and receive data to and from the PE's in the third row; at the same time, the PE's in the second row send and receive data to and from the PE's in the fourth row. It takes two units of time to carry out this data shuffling operation because shuffling distance is 2. Then, the PE's in the first and second rows denoted as black circles compute HBA+, and the PE's in the third and fourth rows denoted as white circles compute HBA−. Similar operations are repeated in the second (Fig. 2(b)), third (Fig. 2(c)), and fourth (Fig. 2(d)) butterfly stages.

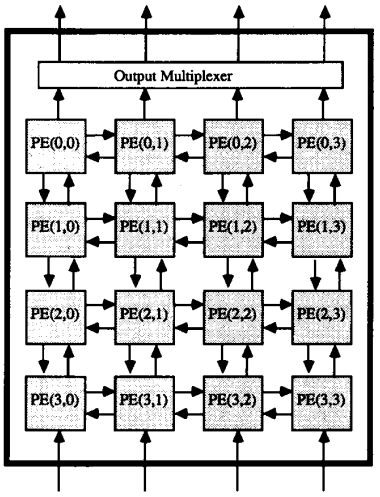


Fig. 3. Chip architecture.

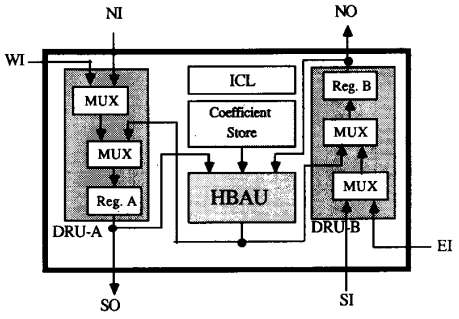


Fig. 4. Internal block diagram of PE.

IV. Circuit and Layout Design

A. Circuit Design

The chip architecture is a mesh array of 16 PE's as shown in Fig. 3. A PE consists of five blocks: two data routing units (DRU's), an HBA unit (HBAU), a coefficient store, and an internal control logic (ICL), as shown in Fig. 4.

The DRU's perform data communications with four nearest-neighbor PE's to perform data shuffling. It is composed of two multiplexers (MUX's) and a pipeline register. The MUX in the DRU controls data shuffling directions (i.e., vertical data shuffling or horizontal data shuffling). The pipeline register can be reconfigured depending on control inputs to serve three purposes. In data shuffling mode, it behaves as a parallel-in, parallel-out (PIPO) register to provide bit-parallel data communication with neighboring PE's. During the HBA computation, the pipeline register is reconfigured into a parallel-in, serial-out (PISO) register to carry out arithmetic shift right. Also, in self-testing mode, it is reconfigured into a linear feedback shift register (LFSR) to generate pseudo-random test patterns.

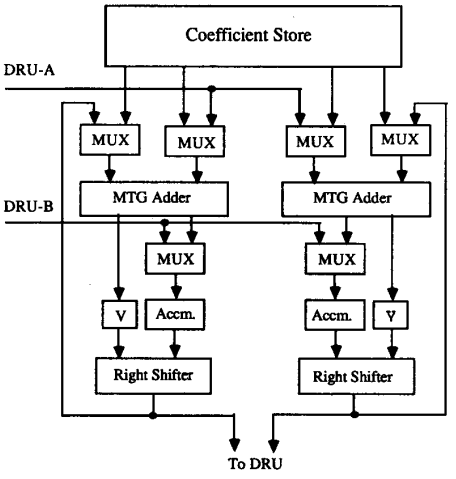


Fig. 5. Internal structure of the HBAU.

Fig. 5 shows the internal block diagram of the HBAU, which implements the HBA of (1) using the distributed arithmetic concept [17]. Using W-bit, fixed-point, fractional two's complement arithmetic, (1) is expressed by the distributed arithmetic as follows:

$$\mathrm{Re}\{\mathrm{HBA\&}\} = A_r \& 2^{-(W-1)} K_i + \sum_{n=0}^{W-1} Q_{rn} 2^{-n}$$

$$\mathrm{Im}\{\mathrm{HBA\&}\} = A_i \& 2^{-(W-1)} K_r + \sum_{n=0}^{W-1} Q_{in} 2^{-n} \quad (2)$$

where

$$A_r = \mathrm{Re}\{f(q-1,k)\}, \qquad A_i = \mathrm{Im}\{f(q-1,k)\}$$

$$B_r = \mathrm{Re}\{f(q-1,k+D(q))\},$$

$$B_i = \mathrm{Im}\{f(q-1,k+D(q))\}.$$

In (2), $f(q-1,k)$ and $f(q-1,k+D(q))$ represent the intermediate results obtained from the $(q-1)$th butterfly stage, and subscript n represents the nth bit of B_r and B_i. Also, the Q_{rn} and Q_{in} are coefficients whose values are determined by an EXCLUSIVE-OR (XOR) and EXCLUSIVE-NOR (XNOR) combination of B_{rn} and B_{in}. All the values for Q_{rn} and Q_{in} in (2), which vary depending on not only the position of PE but also the butterfly stage q, are precomputed and stored within each PE.

From (2) it is known that the HBA implementation based on the distributed arithmetic requires W addition steps, and the results obtained from the distributed arithmetic are combined with A_r and A_i to complete the HBA.

As shown in Fig. 5, the HBAU is implemented with two identical parts: one is responsible for the real part of HBA& and the other is for the imaginary part of HBA&. As a result, the HBAU is simply realized with only two adders. Among various adder structures, we choose a modified transmission gate adder (MTGA) [3] in order to minimize chip area. A 1-b MTGA requires only 16 transistors which have no connections to power or ground. Since it allows all drain and source regions within the adder to be shared between two devices, a dense layout of $90\times98\ \mu\mathrm{m}^2$ is achieved for a 1-b MTGA. Circuit simulation for an 8-b MTGA shows an 18-ns delay.

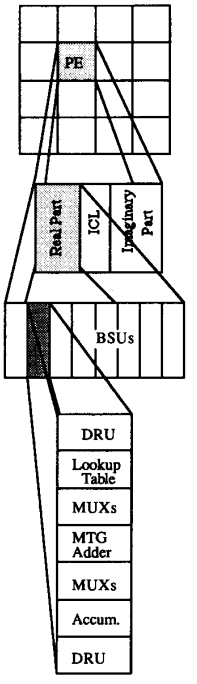


Fig. 6. Layout design hierarchy.

According to Parseval's theorem, the FFT output signal is larger than the input signal power. Thus, in implementing the FFT with fixed-point arithmetic, a scaling policy is usually necessary to prevent overflow. For this purpose, we adopt a step-by-step scaling method which scales down data by a factor of 2 at every butterfly stage. This scaling is simply performed by 1-b shift right.

In general, finite register length within an arithmetic unit causes some roundoff errors, and reveals trade-offs between chip area and arithmetic error. In our array processor, a 1-b increase in register length results in an increase in the chip area of about 8%, but reduces total mean-squared error by 12 dB. In this prototype design, we use fixed-point 8-b as the operation data word, which yields a total mean-squared error of about -83 dB.

B. Layout Design

The layout design is accomplished in a hierarchical approach based on bit-slice units (BSU's). The hierarchy adopted is depicted in Fig. 6, and is described as follows:

a) design basic cells in full custom,
b) build a BSU by vertically stacking the basic cells,
c) construct a PE by repeating the BSU's,
d) complete the chip layout by abutting the PE's in 4×4 mesh array.

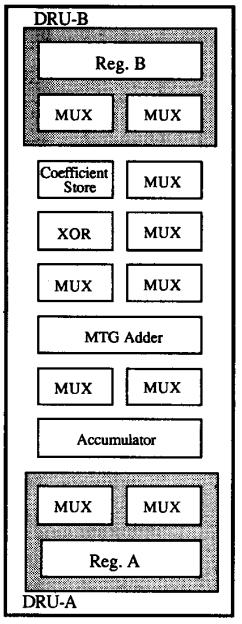


Fig. 7. Floor plan of a BSU.

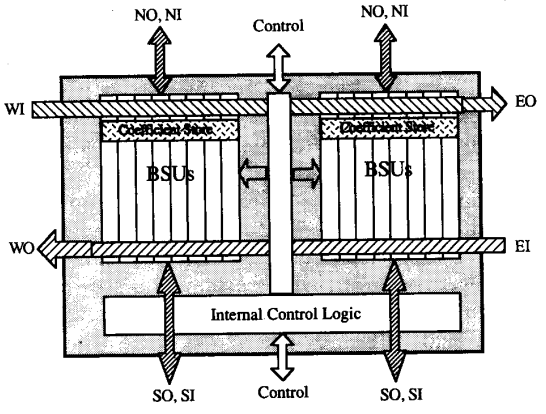


Fig. 8. Floor plan of a PE.

The physical layouts of the basic cells are tailored to satisfy the boundary connection requirements. To verify these basic cells, switch-level logic simulation is carried out using the netlist data extracted from layout.

Once all the basic cells have been physically designed and verified, a BSU is assembled from them. The BSU is constructed with seven basic cells stacked vertically, as shown in Fig. 7. The height and width of a BSU are 1468 and 90 μm, respectively.

Fig. 8 shows the floor plan of a PE, which is built up of two identical half-planes with a central control section. Each half-plane consists of eight BSU's. All interconnections between BSU's are achieved through abutment. Buffers are inserted between ICL and each half-plane to drive the control signals decoded by the ICL. The final

TABLE II
LAYOUT STATISTICS

Block	Area (mm^2)	Transistors
BSU	0.132	150
ICL	0.967	526
PE	3.345	2998
4×4 Array	53.52	48 000
Die size including 83 pads		76.56 mm^2

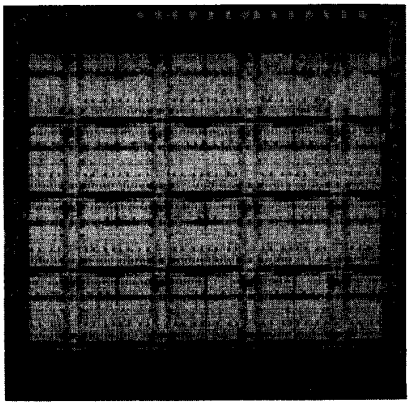


Fig. 9. Microphotograph of the fabricated array processor.

layout at the chip level is accomplished by a simple arrangement of the PE's in a 4×4 mesh array.

The chip contains about 48 000 transistors in an area of 53.52 mm^2, and die size including 83 pads is 76.56 mm^2. The layout area of a BSU containing 150 transistors is 0.132 mm^2, and a PE composed of 3000 transistors requires 3.345 mm^2. Some layout statistics are summarized in Table II.

Since all PE's in the chip are synchronized by a global clock signal, a clock skew problem due to different path lengths from clock pad to each PE was eliminated by H-tree clock distribution scheme [18].

The layout design methodology adopted in this paper features: a) regular layout structure; b) fast design and easy debugging; c) minimum design errors; and d) flexible extension in both array size and data bit width.

V. Fabrication and Measurement Results

The chip has been fabricated with a 2-μm p-well CMOS technology with two metal layers. The effective channel lengths of the PMOS and NMOS transistors are 1.6 and 1.44 μm, respectively. The microphotograph of the fabricated chip is shown in Fig. 9.

Fig. 10 shows the measured waveforms for load signal and data out for the DRU block. The measured delays from load signal to data out are 12, 8, and 6.2 ns at $V_{dd} = 4$, 5, and 6 V, respectively. This result says that each PE within the array can perform data communications with neighboring PE's at speeds up to 65 MHz.

From the measurement results, the HBA computation takes 0.65 μs, and the array processor computes a 16-point

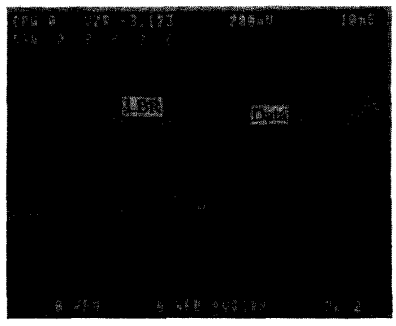


Fig. 10. Measured waveforms for the operation of DRU. Measured delay from load signal *LBR* to data out shows 8 ns at $V_{dd} = 5$ V.

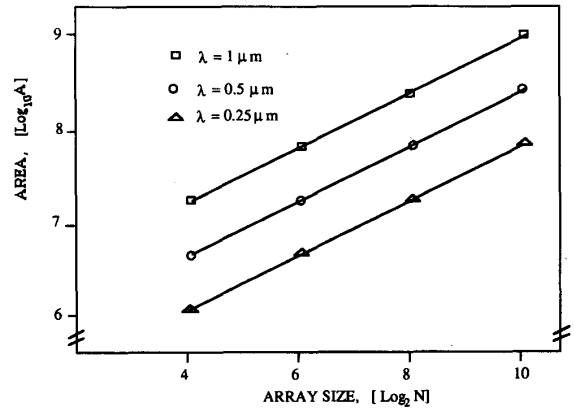


Fig. 11. Area estimation of the 2-D mesh array for FFT. Area is normalized by layout area of a PE A_0.

FFT every 3 μs, which is equivalently 24.6 million complex multiplications per second.

VI. Discussion

According to the measurement results of the fabricated chip, the throughput rate (i.e., the number of samples processed per unit time) of our array processor is estimated to be 5.3 MHz, which is much higher than the 40 kHz of the processor computing the discrete Fourier transform (DFT) [21]. It has been reported that the programmable DSP processors execute 512-point complex FFT within 4.5 ms [23] and 1024-point FFT in 1.0–1.33 ms [3], [24], achieving throughput rates in the range of 0.1–1.0 MHz, which are much lower than that of our array processor. Also, it is comparable to 8.8 MHz of the FFT processor based on radix-4 butterfly [4] and 10 MHz of the FFT system built with 27 IC's [5].

Although the array processor described in this paper implements only 16 PE's with a 2-μm CMOS process, it will be possible to integrate more than 16 PE's into a single chip if a submicrometer technology is used. The area $A(N, \lambda)$ required for an array of N PE's can be parameterized as follow:

$$A(N, \lambda) = A_0 N \lambda^2 \times 10^6. \tag{3}$$

In (3), λ denotes half of the minimum feature size of a given technology, and A_0 represents the physical layout area of a PE. The value of A_0 will depend on the physical realization of PE. In our design using double-metal layer, A_0 is 3.345 (see Table II).

Fig. 11 shows this area estimation using the λ parameter, which is normalized by A_0. Based on Fig. 11, it will be possible to implement an array processor for 32×32 2-D FFT in an area of about 1 cm^2 if a submicrometer CMOS technology is used.

VII. Conclusions

An array processor dedicated to compute 16-point FFT has been implemented using a 2-μm CMOS process. To achieve an area-efficient computation of the FFT, some design considerations have been examined both in array level and in PE level. The novel features in designing the array processor are the half butterfly arithmetic concept realized with the distributed arithmetic, and the hierarchical design methodology based on bit-slice units.

The prototype implementation of this paper leads us to conclude that the 2-D mesh array architecture and algorithm presented in this paper will be one of the most attractive candidates for the real-time computation of the FFT in VLSI.

Appendix

Algorithm I: Data Shuffling Procedure Based on HBA

```
BEGIN
  FOR q = 1 TO 4 DO IN PARALLEL for all i and j
  (i = j = 0, 1, 2, 3)
    BEGIN
      IF (q ≤ 2) THEN
        D(q) = 2^(2-q)
        IF (i MOD 2^(3-q) < 2^(2-q)) THEN
          move data in PE(i, j) to PE(i + D(q), j)
        ELSE
          move data in PE(i, j) to PE(i - D(q), j)
      ELSE
        D(q) = 2^(4-q)
        IF (j MOD 2^(5-q) < 2^(4-q)) THEN
          move data in PE(i, j) to PE(i, j + D(q))
        ELSE
          move data in PE(i, j) to PE(i, j - D(q))
    END
END.
```

Algorithm II: Determination of the Arithmetic Type of the HBA&

```
BEGIN
  FOR q = 1 TO 4 DO IN PARALLEL for all i and j
    (i = j = 0, 1, 2, 3)
    BEGIN
      IF (q ≤ 2) THEN
        IF (i MOD 2^(3-q) < 2^(2-q)) THEN
          PE(i, j) compute the HBA +
        ELSE
          PE(i, j) compute the HBA -
```

```
ELSE
    IF (j MOD 2^(5-q) < 2^(4-q)) THEN
        PE(i,j) compute the HBA +
    ELSE
        PE(i,j) compute the HBA -
    END
END.
```

References

[1] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[2] K. L. Kloker, B. Lindsley, N. Baron, and G. R. L. Sohie, "Efficient FFT implementation on an IEEE floating-point digital signal processor," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1988, pp. 1399–1402.

[3] R. W. Linderman, P. M. Ahau, W. H. Ku, and P. P. Peusens, "Cusp: A 2-μm CMOS digital signal processor," *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 761–769, June 1985.

[4] J. O'Brien, J. Mather, and B. Holland, "A 200 MIPS single-chip 1K FFT processor," in *ISSCC Dig. Tech. Papers*, Feb. 1989, pp. 166–167.

[5] J. Fox, G. Surace, and P. A. Thomas, "A self-testing 2 μm CMOS chip set for FFT applications," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 15–19, Feb. 1987.

[6] K. Yamashita *et al.*, "A wafer-scale 170 000-gate FFT processor with built-in test circuits," *IEEE J. Solid-State Circuits*, vol. 23, pp. 336–341, Apr. 1988.

[7] W. Cochran *et al.*, "What is the fast Fourier transform?," *IEEE Trans. Audio Electroacoust.*, vol. AU-15, pp. 45–55, 1967.

[8] H. Stone, "Parallel processing with the perfect shuffle network," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Dec. 1986.

[9] M. K. Lee, "Systolic array for FFT computation," in *'87 Multi Project Chip (MPC'87) Development*, Dept. Electronic Eng., Yonsei Univ., Seoul, Korea, Aug. 1988.

[10] K. W. Shin, "A VLSI architecture for the parallel computation of FFT," Ph.D. dissertation, Dept. Electronic Eng., Yonsei Univ., Seoul, Korea, Aug. 1990.

[11] K. W. Shin, B. Y. Choi, and M. K. Lee, "A VLSI architecture of systolic array for FFT computation," *J. KITE*, vol. 25, no. 9, pp. 97–106, 1988.

[12] B. Y. Choi, B. H. Kang, J. K. Lee, K. W. Shin, and M. K. Lee, "VLSI implementation of two-dimensional FFT algorithm on systolic array," in *Proc. TENCON87 IEEE Region-10 Conf.*, 1987, pp. 125–131.

[13] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph, "A radix 4 delay commutator for fast Fourier transform processor implementation," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 5, pp. 702–709, Oct. 1984.

[14] T. Wiley, T. S. Durrani, and R. Champinin, "An FFT systolic processor and its applications," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1984, vol. 2, pp. 4.1–4.4.

[15] L. W. Chang and M. Y. Chen, "A new systolic array for discrete Fourier transform," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 36, no. 10, pp. 1665–1666, 1988.

[16] K. W. Shin and M. K. Lee, "A VLSI architecture for parallel computation of FFT," *Systolic Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1989, pp. 116–125.

[17] S. A. White, "A simple FFT butterfly arithmetic unit," *IEEE Trans. Circuit Syst.*, vol. CAS-28, no. 4, pp. 352–355, Apr. 1981.

[18] A. L. Fisher and H. T. Kung, "Synchronizing large VLSI processor arrays," *IEEE Trans. Comput.*, vol. C-34, no. 8, pp. 734–740, Aug. 1985.

[19] C. D. Thomson, "A complexity theory for VLSI," Ph.D. dissertation, Comput. Sci. Dept., Carnegie-Mellon Univ., Pittsburgh, PA, Aug. 1980.

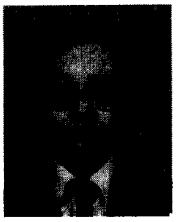
[20] I. R. Mactaggart and M. A. Jack, "A single chip radix-2 FFT butterfly architecture using parallel data distributed arithmetic," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 3, pp. 368–373, June 1984.

[21] J. L. V. Meerbergen and F. J. V. Wyk, "A 256-point discrete Fourier transform processor fabricated in 2 μm NMOS technology," *IEEE J. Solid-State Circuits*, vol. SC-18, no. 5, pp. 604–609, Oct. 1983.

[22] H. S. Lee, H. Mori, and H. Aiso, "Parallel processing FFT for VLSI implementation," *Trans. IECE Japan*, vol. E68, no. 5, pp. 284–291, May 1985.

[23] Y. Kawakami *et al.*, "A 32b floating point CMOS digital signal processor," in *ISSCC Dig. Tech. Papers*, Feb. 1986, pp. 86–87.

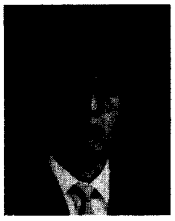
[24] A. Kanuma *et al.*, "A 20 MHz 32b pipelined CMOS image processor," in *ISSCC Dig. Tech. Papers*, Feb. 1986, pp. 102–103.



Moon-Key Lee (S'77–M'79) was born in Seoul, Korea, in 1941. He earned the B.S., M.S., and D.E. degrees in electrical engineering from Yonsei University, Seoul, Korea in 1965, 1967, and 1973, respectively. In 1980 he received the Ph.D. degree from the University of Oklahoma.

He was a Lecturer in the Department of Electrical Engineering, Yonsei University, from 1968 to 1970. From 1970 to 1976 he was with Kyunghee University, Seoul, Korea, where he held the positions of Assistant Professor, Associate Professor, and Chairman of the Electronic Engineering Department. He was Director of the IC design division at Korea Institute of Electronic Technology (ETRI, at present), Kumi, Korea, from 1980 to 1982. In 1982 he joined the faculty of Yonsei University, Seoul, Korea, where he is currently Professor and Chairman of the Electronic Engineering Department. He is a founder of the Research Institute of ASIC Design (RIAD), which was established in 1989 and located at Yonsei University. He has published five college textbooks on electronic engineering. He has authored and co-authored over 100 papers on intergrated circuit design and computer-aided design. He made pioneering contributions to IC design education by performing in the Multi Project Chip research, which was participated in by eight universities and supported by the Ministry of Science and Technology of Korea from 1985 through 1988, for the first time in Korea. His current research interests include VLSI design and computer-aided design and software.

Dr. Lee was the Chairman of the Chapter Promotion and educational activities in the IEEE Korea Section. He has served as Chairman of the IEEE Circuits and Systems chapter in Korea which was formed in 1989. In 1985, he organized the International Workshop on VLSI and CAD, Seoul, Korea and served as Chairman. He was the Program Chairman of the 1989 International Conference on VLSI and CAD (ICVC), Seoul, Korea. He was the Chairman of the financial committee of TENCON 87, IEEE Region 10 Conference, Seoul, Korea, in 1987. From 1982 through 1987 he was a member of the editorial committee and an Associate Editor of the *Journal of Korean Institute of Telematics and Electronics* (*KITE*). He received the distinguished honor award from KITE in 1984. In 1986, he was awarded the 40th anniversary prize for the outstanding paper published in the journal of KITE. Since 1988 he has served on the Board of Directors of the Korean Institute of Telematics and Electronics (KITE).



Kyung-Wook Shin was born in Chongju, Korea, in 1961. He received the M.S. and Ph.D. degrees in electronic engineering from Yonsei University, Seoul, Korea, in 1986 and 1990, respectively. During his Ph.D. course, he worked on the design of a VLSI array processor for parallel computation of fast Fourier transform (FFT).

In 1990 he joined the Integrated Circuits Technology Department of the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. In 1991 he joined the faculty of Kum Oh Institute of Technology, Kyungbuk, Korea. His research interests include VLSI signal processing, algorithm-oriented VLSI array architectures, and digital design.

Jang-Kyu Lee was born in Seoul, Korea, in 1963. He received the B.S. degree in electronic engineering from Sogang University, Seoul, Korea, in 1986, and the M.S. degree in electronic engineering from Yonsei University, Seoul, Korea, in 1988.

In 1988 he joined the Semiconductor Division of Samsung Electronics, Inc., Kyungki Do, Korea, where he works on memory IC design. His research interests include DRAM design and application-specific memory design.