High-Speed Array Multipliers Based on On-the-Fly Conversion

Sang-Man Moh and Suk-Han Yoon

CONTENTS

ABSTRACT

A new on-the-fly conversion algorithm is proposed, and high-speed array multipliers with the on-the-fly conversion are presented. The new on-the-fly conversion logic is used to speed up carry-propagate addition at the last stage of multiplication, and provides constant delay independent of the number of input bits. In this paper, the multiplication architecture and the on-the-fly conversion algorithm are presented and discussed in detail. The proposed architecture has multiplication time of $(n+1)t_{FA}$, where n is the number of input bits and t_{FA} is the delay of a full adder. According to our comparative performance evaluation, the proposed architecture has shorter delay and requires less area than the conventional array multiplier with on-the-fly conversion.

I. INTRODUCTION

Parallel multipliers are classified mainly into array multipliers and tree multipliers. Basically the array multipliers include two-dimensional full adder arrays, and the full adder arrays are based on the algorithm proposed by Baugh and Wooley [1]. The tree multipliers consist of full adder trees, which were initially proposed by Wallace [2] and evolved by Dadda [3] and many other researchers [4].

Regardless of multiplication schemes, most multipliers require carry-propagate addition at the last stage of multiplication to generate a complete $2n$-bit product [5]. Such a carry-propagate addition is comparatively slow and requires significant area in VLSI implementation [6].

To speed up the carry-propagate addition for the last stage, on-the-fly conversion techniques have been studied. Ercegovac and Lang [7] proposed a multiplication scheme not requiring carry-propagate addition at the last stage of conventional multipliers, by carrying out the on-the-fly conversion based on the technique presented in [8]. And Montuschi and Ciminiera [9] proposed $n \times n$ multipliers without final carry-propagate addition, which produce $2n$-bit results with delay comparable to the multiplier proposed by Ercegovac and Lang in [7]. In the multiplication schemes proposed in [9], the determination of the most significant digits is carried out by using the on-the-fly conversion in parallel with the computation of the least significant bits.

Still, however, it is required to speed up the multiplication with minimum area overhead. Our new approach to fast array multiplication is presented in this paper. We propose high-speed array multipliers with new on-the-fly conversion logic. The newly proposed on-the-fly conversion logic provides the constant delay independent of the number of input bits. We evaluate our multiplication scheme and compare it with the conventional multiplier proposed in [9]. The proposed scheme provides shorter delay than the multiplication scheme, with less area than [9].

In Section II of this paper, the algorithm and architecture of the proposed on-the-fly conversion and multiplication scheme are presented in detail. In Section III, the performance evaluation results are discussed. And the conclusions are covered in Section IV.

II. ALGORITHM AND ARCHITECTURE

Two's complement numbers X and Y can be represented as

$$X = [x_{n-1}x_{n-2}\cdots x_0]_{(2)} = \sum_{i=0}^{n-2} x_i \cdot 2^i - x_{n-1}\cdot 2^{n-1} \quad (1)$$

and

$$Y = [y_{n-1}y_{n-2}\cdots y_0]_{(2)} = \sum_{i=0}^{n-2} y_i \cdot 2^i - y_{n-1}\cdot 2^{n-1}, \quad (2)$$

respectively. The $[x_{n-1}x_{n-2}\cdots x_0]_{(2)}$ denotes that the n-bit string $x_{n-1}x_{n-2}\cdots x_0$ is a binary number composed of 0s and 1s. If the product Z is given by multiplying the above X and Y, it can be calculated by

$$Z = X \cdot Y. \tag{3}$$

And the $2n$-bit product Z can be represented as

$$\begin{aligned} Z &= [z_{2n-1}z_{2n-2}\cdots z_0]_{(2)} \\ &= \sum_{i=0}^{2n-2} z_i \cdot 2^i - z_{2n-1}\cdot 2^{2n-1}. \end{aligned} \tag{4}$$

Baugh and Wooley [1] suggested a two's complement array multiplication algorithm, and it was extended by Blankenship [10] as shown in Fig. 1. The elementary products of 5-bit multiplication can be rearranged as shown in Fig. 2.

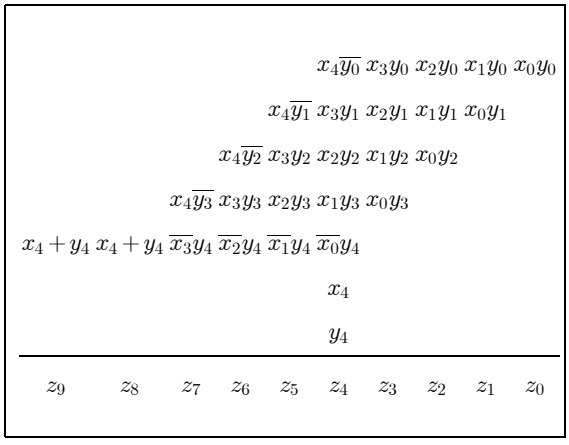

Fig. 1. Elementary products ($n = 5$).

Based on the rearranged elementary products, we propose an array multiplier which avoids the carry-propagate addition at the last step of multiplication. The proposed multiplier for $n = 5$ is shown in Fig. 3, where the least significant $n+1$ bits of array output are produced in carry-assimilated representation and the most significant $n-1$ bits are produced in carry-sum pair form.

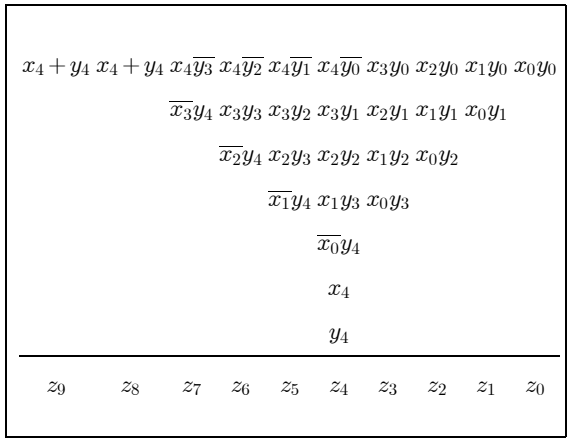

Fig. 2. Rearrangement of elementary products.

If we represent each carry-sum pair as $(a_i,\ b_i)$, the sum s_i and carry-out c_i of half adder outputs are given by

$$s_i = a_i \oplus b_i, \quad i = 1,2,\cdots,n-1 \tag{5}$$

and

$$c_i = a_i \cdot b_i, \quad i = 1,2,\cdots,n-1, \tag{6}$$

respectively, where $\oplus$ is the symbol of logical exclusive-or. And in the proposed multiplier, the most significant $n-1$ bits of the product Z are produced by an on-the-fly conversion algorithm. For example, the

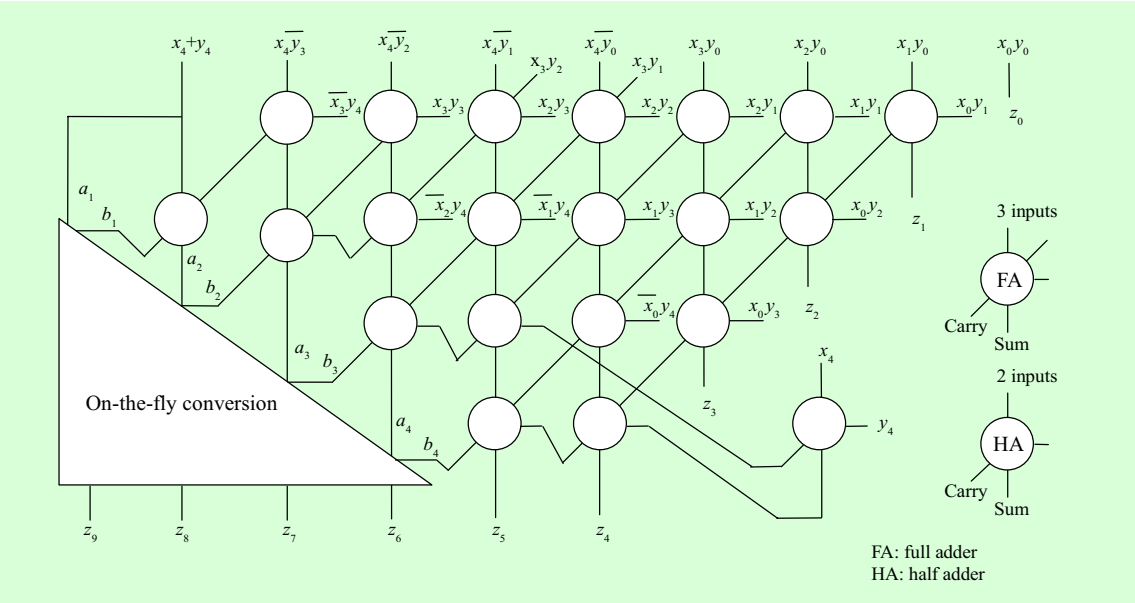

Fig. 3. Proposed array multiplier.

products z_{n+1}, z_{n+2}, z_{n+3}, z_{n+4}, $\cdots$, z_{2n-1} are given by

$$z_{n+1} = s_{n-1}, \tag{7}$$

$$z_{n+2} = s_{n-2} \oplus c_{n-1}, \tag{8}$$

$$z_{n+3} = s_{n-3} \oplus c_{n-2} \oplus s_{n-2}c_{n-1}, \tag{9}$$

$$z_{n+4} = s_{n-4} \oplus c_{n-3} \oplus s_{n-3}c_{n-2} \oplus s_{n-3}s_{n-2}c_{n-1}, \tag{10}$$

$$\vdots$$

$$z_{2n-1} = s_1 \oplus c_2 \oplus s_2c_3 \oplus s_2s_3c_4 \oplus s_2s_3s_4c_5 \oplus \cdots \oplus s_2s_3\cdots s_{n-2}c_{n-1}, \tag{11}$$

respectively. Therefore, the product z_{2n-i} for $i = 1,2,\cdots,n-1$ is given by

$$z_{2n-i} = s_i \oplus c_{i+1} \oplus s_{i+1}c_{i+2} \oplus s_{i+1}s_{i+2}c_{i+3} \oplus \cdots \oplus s_{i+1}s_{i+2}\cdots s_{n-2}c_{n-1} \tag{12}$$

in generalized notation.

Let us define $k_{i,j}$ and $t_{i,j}$ for $i = 1,2,\cdots, n-1$ as

$$k_{i,j} = \begin{cases} k_{i,j-1}s_{i+j-1} & \text{if } j = 2,3,\cdots,n-1-i \\ 1 & \text{if } j = 1 \end{cases} \tag{13}$$

and

$$t_{i,j} = \begin{cases} t_{i,j-1} \oplus k_{i,j}c_{i+j} & \text{if } j = 1,2,\cdots,n-1-i \\ s_i & \text{if } j = 0, \end{cases} \tag{14}$$

respectively, in recursive form.

The $k_{i,j}$ and $t_{i,j}$ can be effectively used to represent the most significant $n-1$ bits of the product Z. The products z_{n+1}, z_{n+2}, z_{n+3}, z_{n+4}, $\cdots$, z_{2n-1}, which are produced by the on-the-fly conversion algorithm, can be represented again by

$$z_{n+1} = s_{n-1}$$

$$= t_{n-1,0}, \tag{15}$$

$$\begin{aligned} z_{n+2} &= s_{n-2} \oplus c_{n-1} \\ &= t_{n-2,0} \oplus c_{(n-2)+1} \\ &= t_{n-2,0} \oplus k_{n-2,1} c_{(n-2)+1} \\ &= t_{n-2,1}, \end{aligned} \tag{16}$$

$$\begin{aligned} z_{n+3} &= s_{n-3} \oplus c_{n-2} \oplus s_{n-2} c_{n-1} \\ &= t_{n-3,1} \oplus s_{(n-3)+2-1} c_{(n-3)+2} \\ &= t_{n-3,1} \oplus k_{n-3,2} c_{(n-3)+2} \\ &= t_{n-3,2}, \end{aligned} \tag{17}$$

$$\begin{aligned} z_{n+4} &= s_{n-4} \oplus c_{n-3} \oplus s_{n-3} c_{n-2} \oplus s_{n-3} s_{n-2} c_{n-1} \\ &= t_{n-4,2} \oplus k_{n-4,2} s_{(n-4)+3-1} c_{(n-4)+3} \\ &= t_{n-4,2} \oplus k_{n-4,3} c_{(n-4)+3} \\ &= t_{n-4,3}, \end{aligned} \tag{18}$$

$$\vdots$$

$$\begin{aligned} z_{2n-1} &= s_1 \oplus c_2 \oplus s_2 c_3 \oplus s_2 s_3 c_4 \oplus s_2 s_3 s_4 c_5 \oplus \cdots \\ &\quad \oplus s_2 s_3 \cdots s_{n-2} c_{n-1} \\ &= t_{1,n-3} \oplus k_{1,n-3} s_{1+(n-2)-1} c_{1+(n-2)} \\ &= t_{1,n-3} \oplus k_{1,n-2} c_{1+(n-2)} \\ &= t_{1,n-2}, \end{aligned} \tag{19}$$

respectively. By making the above formulas be formal and generalized notation, each product z_{2n-i} of the most significant $n-1$ bits of the product Z for $i = 1, 2, \cdots, n-1$ can be represented by

$$\begin{aligned} z_{2n-i} &= s_i \oplus c_{i+1} \oplus s_{i+1} c_{i+2} \oplus s_{i+1} s_{i+2} c_{i+3} \oplus \cdots \\ &\quad \oplus s_{i+1} s_{i+2} \cdots s_{n-2} c_{n-1} \\ &= t_{i,n-i-2} \oplus k_{i,n-i-1} c_{i+(n-i-1)} \\ &= t_{i,n-i-1}. \end{aligned} \tag{20}$$

Figure 4 shows the on-the-fly conversion logic for $n = 5$. As shown in Fig. 4, as soon as each carry-sum pair (a_i, b_i) is generated out of the full adder array, the on-the-fly conversion is carried out for each pair.

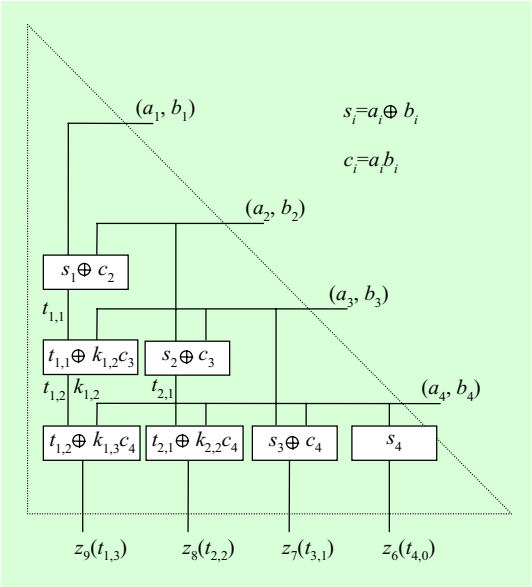

Fig. 4. On-the-fly conversion logic.

III. PERFORMANCE EVALUATION

The proposed multiplier can be evaluated by using two parameters of multiplication time and area. We evaluate the proposed multiplier on the basis of these two parameters, and then we compare it with the multiplier proposed by Montuschi and Ciminiera [9].

1. Speed

Let us denote the delay in generating the elementary product terms by $t_{NOT-AND}$, the delay of a full adder by t_{FA}, and the delay of the final step of the on-the-fly conversion logic by $t_{AND-XOR}$. The multiplication time of the proposed multiplier, t_{pro}, is represented as

$$t_{pro} = t_{NOT-AND} + n t_{FA} + t_{AND-XOR}. \tag{21}$$

The $t_{NOT-AND} + t_{AND-XOR}$ can be represented by

$$t_{NOT-AND} + t_{AND-XOR} \leq t_{FA} \tag{22}$$

since the longest path of a full adder is $sum(FA) = a \oplus b \oplus c_{in}$. Therefore, t_{pro} in equation (21) is represented again as

$$t_{pro} \approx (n+1)t_{FA}. \tag{23}$$

According to [9], the multiplication time of the multiplier proposed by Montuschi and Ciminiera is

$$t_{MC} \geq (n+2)t_{FA} + t_{NOT-AND}. \tag{24}$$

The comparison of t_{pro} and t_{MC} shows that the proposed multiplier architecture has shorter delay than the conventional multiplication scheme. That is, t_{pro} is reduced by the delay of $t_{FA} + t_{NOT-AND}$ over t_{MC}.

2. Area

To evaluate the hardware required to implement the proposed multiplier, we denote the hardware used to implement an AND gate, a full adder, a half adder and the logic function $t_{i,j}$ by K_{AND}, K_{FA}, K_{HA}, and K_t, respectively. The hardware required to build up the proposed multiplier, K_{pro}, can be represented as

$$K_{pro} = n^2 K_{AND} + (n-1)^2 K_{FA} + nK_{HA} + \frac{(n-1)(n-2)}{2} K_t. \tag{25}$$

As shown in Fig. 4, K_t can be expressed as $K_{AND} + K_{XOR}$, where K_{XOR} is the hardware used to inplement an XOR gate. Therefore, K_{pro} can be represented again as

$$K_{pro} = n^2 K_{AND} + (n-1)^2 K_{FA} + nK_{HA} + \frac{(n-1)(n-2)}{2}(K_{AND} + K_{XOR}). \tag{26}$$

According to [9], the hardware required to implement the multiplier proposed by Montuschi and Ciminiera, K_{MC}, is given by

$$K_{MC} = n^2 K_{AND} + (2n^2 - 3n + 1)K_{FA}. \tag{27}$$

For more practical evaluation, we use typical area for basic logic elements in terms of the area of a two-input NAND gate, which is shown in Table 1 [11], [12]. The entries in Table 1 have been normalized with respect to the area of a two-input NAND gate.

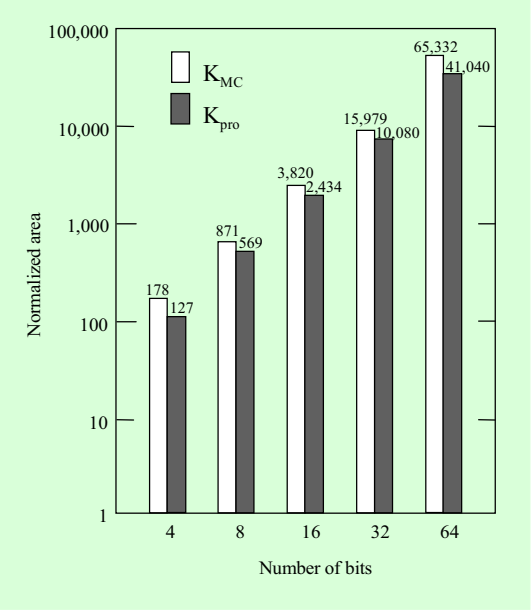

Fig. 5. Normalized area of two multiplication schemes.

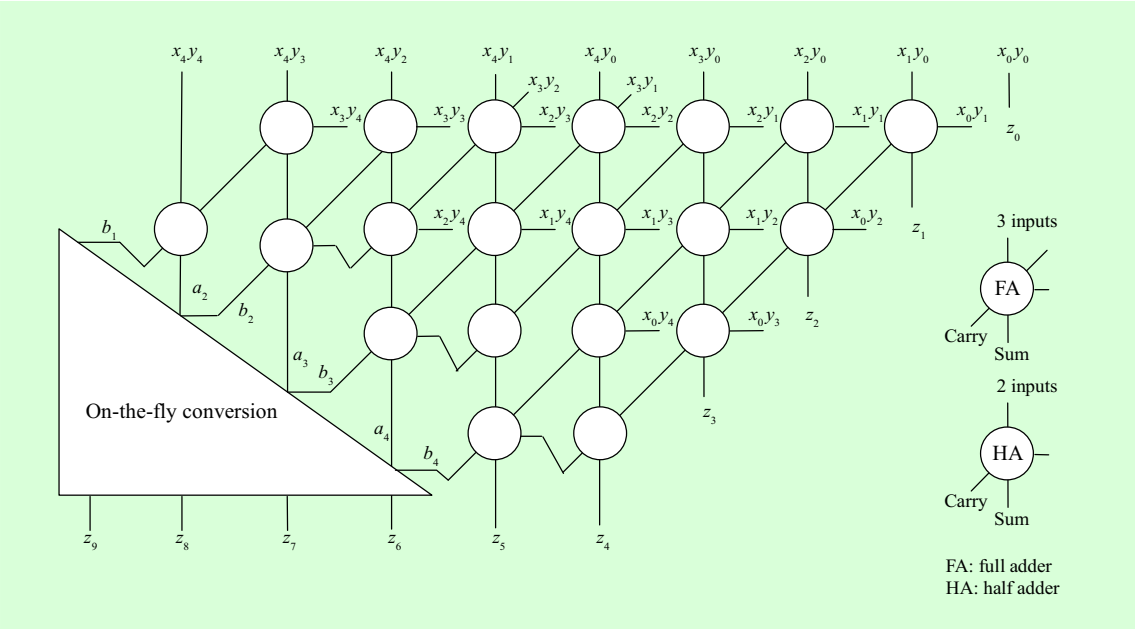

Fig. 6. The multiplier for unsigned numbers ($n = 5$).

Table 1. Area of basic logic elements.

Logic element	Normalized area
Two-input NAND	1.00
Two-input AND	1.30
Two-input XOR	1.50
Full adder	7.50
Half adder	5.00

For comparative evaluation, we have plotted the normalized area of the conventional multiplication scheme [9] and the proposed one in accordance with the number of input bits. Figure 5 shows the area of two multiplication schemes. As shown in Fig. 5, K_{pro} is 71.3% to 62.8% of K_{MC} for the multiplication of 4 to 64 bits. That is, the proposed multiplier requires less area than the conventional multiplier [9].

On the basis of the proposed two's complement multiplier described above, the unsigned scheme with the proposed on-the-fly conversion logic is easily designed as shown in Fig. 6. Since there is no sign bit in the unsigned multiplication, the multiplier for unsigned numbers is simpler and more modular than the two's complement scheme. Figure 6 shows such an unsigned multiplication scheme for $n = 5$.

IV. CONCLUSIONS

In this paper, we have presented the high-speed array multipliers with on-the-fly conversion. The newly proposed on-the-fly conversion logic provides constant delay independent of the number of input bits. According to our evaluation results, the multiplication time of the proposed scheme is reduced by the delay of $t_{FA}+t_{NOT-AND}$ over [9], and the area of the proposed one is 71.3% to 62.8% of [9] for the multiplication of 4 to 64 bits. That is, the proposed architecture has shorter delay and requires less area than the conventional array multiplication scheme with on-the-fly conversion. Also, we have designed the multiplier for unsigned numbers, which is simpler and more modular than the two's complement scheme.

REFERENCES

[1] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. on Computers*, vol. 22, no. 12, pp. 1045-1047, Dec. 1973.

[2] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. on Electronic Computers,* pp. 14-17, Feb. 1964.

[3] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza,* vol. 19, pp. 349-356, May 1965. Reprinted in *Computer Design Development,* E. E. Swartzlander Jr., Ed. Rochelle Park, New Jersey: Hayden Book, 1976.

[4] I. Koren, *Computer Arithmetic Algorithms.* Englewood Cliffs, New Jersey: Prenctice Hall, 1993. pp. 99-123.

[5] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design.* New York: John Wiley & Sons, 1979. pp. 129-210.

[6] M. Uya, K. Kaneko, and J. Yasui, "A CMOS floating-point multiplier," *IEEE J. Solid State Circuits,* vol. 19, no. 5, pp. 697-701, Oct. 1984.

[7] M. D. Ercegovac and T. Lang, "Fast multiplication without carry-propagate addition," *IEEE Trans. on Computers*, vol. 39, no. 11, pp. 1385-1390, Nov. 1990.

[8] M. D. Ercegovac and T. Lang, "On-the-fly conversion of redundant into conventional representations," *IEEE Trans. on Computers*, vol. 36, no. 7, pp. 895-897, July 1987.

[9] P. Montuschi and L. Ciminiera, "$n \times n$ carry-save multipliers without final addition," *Proc. 11th IEEE Symp. on Computer Arithmetic*, June 1993. pp. 54-61.

[10] P. E. Blankenship, "A comment on "A Two's complement Parallel Array Multiplication Algorithm"," *IEEE Trans. on Computers*, vol. 23, no. 12, pp. 1327, Dec. 1974.

[11] S. Sunder, F. El-Guibaly, and A. Antoniou, "Two's-complement fast serial-parallel multiplier," *IEE Proceedings - Circuits, Devices and Systems*, vol. 142, no. 1, pp. 41–44, Feb. 1995.

[12] N. Weste and K. Eshraghin, *Principles of CMOS VLSI Design: A Systematic Perspective.* Massachusetts: Addison Wesley, 1985. pp. 1-231.

Sang-Man Moh received the M.S. degree in computer science from Yonsei University, Seoul, Korea in 1991. Since 1991, he has been with Processor Section of Electronics and Telecommunications Research Institute, Taejon, Korea as a senior member of research staff. He received the national qualification for Professional Engineer in information technology from the Korean Goverment in 1993. His research interests include computer architecture, parallel processing systems, computer arithmetic, VLSI-oriented algorithms and ASIC design. He has published over 20 papers as first author in international and domestic journals and proceedings, and has held over 30 patents. He is a member of the IEEE Computer Society, Association for Computing Machinery, Korea Information Science Society, and Korea Institute of Telematics and Electronics.

Suk-Han Yoon received the B.E. degree in electronic engineering from Korea University, Seoul, Korea in 1977, the M.S. degree in computer science from KAIST, Taejon, Korea in 1986, and the Ph.D. degree in electronic engineering from Korea University, Seoul, Korea in 1995. He joined Electronics and Telecommunications Research Institute in 1977, where he is currently working as a Director of Computer Technology Division and in charge of High-Performance Computer Development Project. His current research interests include high-performance computer architecture, parallel computing systems and microprocessor architecture.