Example: Consider $n = 8$, $A = 150 + j230$ and $B = 102 + j218$. Here, $R_3(A) = (10)_2 = 2$, $R_2(A) = (01)_2 = 1$, $R_1(A) = (01)_2 = 1$, $R_o(A) = (10)_2 = 2$, $I_3(A) = (11)_2 = 3$, $I_2(A) = (10)_2 = 2$, $I_1(A) = (01)_2 = 1$, $I_o(A) = (10)_2 = 2$, $R_3(B) = (01)_2 = 1$, $R_2(B) = (10)_2 = 2$, $R_1(B) = (01)_2 = 1$, $R_o(B) = (10)_2 = 2$, $I_3(B) = (11)_2 = 3$, $I_2(B) = (01)_2 = 1$, $I_1(B) = (10)_2 = 2$, and $I_o(B) = (10)_2 = 2$, and according to (4),

$$(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (2, 3, 3, 1, -1, -4, -1, 0)$$

and

$$(b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7) = (1, 3, 3, -1, -1, -4, -2, 0).$$

Using (12) one gets: $(c_0, \frac{c_1}{2}, c_2, \frac{c_3}{2}, c_4, \frac{c_5}{2}, c_6, \frac{c_7}{2}) =$ (5904, 4592, 38 072, 8728, 36 128, −576, −3136, −1056), while (14) and (15) will finally give

$$P = -34840$$

and

$$Q = 56160.$$

IV. Conclusions

A way of decomposing large-word length complex multipliers into multipliers of smaller word length has been presented. The technique relies on encoding the complex numbers as polynomials of degree 7 with coefficients obtained from the byte decomposition of the numbers. This achieves a translation of complex multiplication to cyclic convolution which is suitable for efficient VLSI designs using systolic arrays. Any errors introduced during this translation, can be easily eliminated.

As compared to conventional modular array multipliers, complex multipliers based on the proposed scheme require the same number of operations. Modular array multiplication of two n-bit integers can be performed using 16 multiplications of size $\frac{n}{4} \times \frac{n}{4}$ and Wallace trees [6], [7]. Since a complex multiplication requires 4 real multiplications, 64 multiplications of size $\frac{n}{4} \times \frac{n}{4}$ will be required. The same count of 64 $\frac{n}{4} \times \frac{n}{4}$ multiplications for computing the complex product is required by the proposed scheme.

References

[1] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988, pp. 28–29.
[2] H. T. Kung, "Why systolic architectures?" *IEEE Comput. Mag.*, vol. 15, no. 1, pp. 37–46, Jan. 1982.
[3] K. Hwang, *Computer Arithmetic.* New York: Wiley, 1979, pp. 201–205.
[4] S. Waser and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers.* New York: Holt, Rinehart and Winston, 1982, pp. 137–139.
[5] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* Reading, MA: Addison-Wesley, 1987, pp. 66–68.
[6] K. Hwang, *Computer Arithmetic.* New York: Wiley, 1979, pp. 167–171.
[7] J. J. F. Cavanagh, *Digital Computer Arithmetic.* New York: McGraw-Hill, 1984, pp. 183–203.

Single-Precision Multiplier with Reduced Circuit Complexity for Signal Processing Applications

Y.C. Lim

***Abstract*— When two numbers are multiplied, a double-wordlength product is produced. In applications where only the single-precision product is required, the double-wordlength result is rounded to single-precision. Hence, in single-precision applications, it is not necessary to compute the least significant part of the product exactly. Instead, it is only necessary to estimate the carries generated in the computation of the least significant part that will ripple into the most significant part of the product. This will produce single-precision multiplier with significantly reduced circuit complexity. In this paper, we present three novel methods for realizing this class of reduced complexity single-precision multipliers. Their performances are also analyzed.**

***Index Terms*—Reduced complexity multiplier, signal processing VLSI, single-precision arithmetic.**

I. Introduction

Consider the multiplication of two numbers A and B producing the product $P_{(0)}$. Assume that A and B are each an N-bit positive number whose magnitude is less than unity. It follows that

$$A = \sum_{n=1}^{N} a_n 2^{-n}, \qquad a_n = 0, 1 \tag{1c}$$

$$B = \sum_{n=1}^{N} b_n 2^{-n}, \qquad b_n = 0, 1 \tag{1d}$$

$$P_{(0)} = \sum_{n=1}^{2N} p_n 2^{-n}, \qquad p_n = 0, 1. \tag{1e}$$

The multiplication process is shown diagrammatically in Fig. 1. Each p_n is computed by summing the partial product terms $a_j b_{n-j}$ and the carries generated from the computation of P_i, $i \geq n$ [1], [2]. In single-precision arithmetic, the $2N$-bit product $P_{(0)}$ is rounded to the N-bit product $P_{(r)}$. This rounding process may be modeled as the injection of a rounding noise $r_{(0)}$ as shown in Fig. 2.

$$r_{(0)} = P_{(r)} - P_{(0)}. \tag{2}$$

Although the value of $r_{(0)}$ is deterministic for each multiplication process, it is still possible to analyze the rounding error statistically by considering a large number of rounding processes [3], [4]. Statistically, the rounding noise power is the variance of $r_{(0)}$ and is given by $Q^2/12$ [3], [4] where $Q = 2^{-N}$ is the quantization stepsize.

In single-precision applications, it is not necessary to compute the least significant part of the product exactly. It is sufficient to compute the sum of the first $(N+k)$ columns (see Fig. 1) of the partial product terms. Let the sum of these partial product terms be $P_{(T)}$ where

$$P_{(T)} = \sum_{n=1}^{N+k} p_n^{(T)} 2^{-n}. \tag{3}$$

The partial product terms in the $(N + k + 1)$th column through the $2N$th column are not computed but their sum is estimated instead.

Manuscript received March 15, 1987; revised October 1, 1988.
The author is with the Electrical Engineering Department, National University of Singapore, Singapore 0511.
IEEE Log Number 9200318.

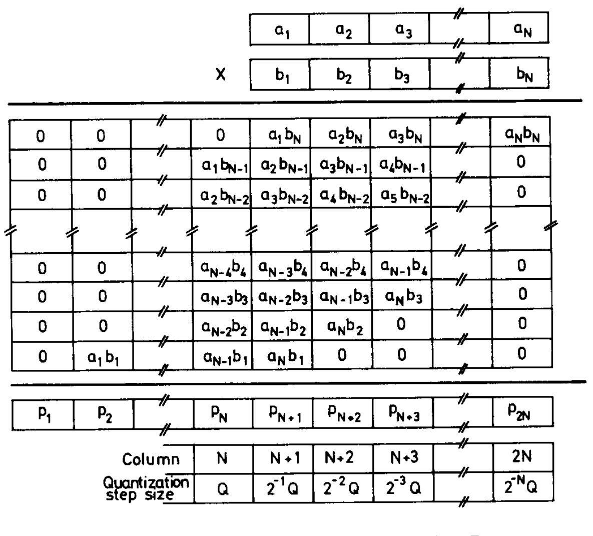


Fig. 1. Diagrammatic representation of $A \times B$.

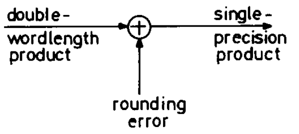


Fig. 2. A rounding process may be modeled as the addition of a rounding error to the double-wordlength product.

Let this estimate be C. The estimated product $P_{(E)}$ is obtained by summing $P_{(T)}$ and C.

$$P_{(E)} = P_{(T)} + C. \tag{4}$$

$P_{(E)}$ is then rounded to single-precision. There are two sources of error for the single-precision result obtained in this way. The rounding of $P_{(E)}$ into single-precision generates a rounding error. The difference between $P_{(E)}$ and $P_{(0)}$ is another error. Let

$$e = P_{(E)} - P_{(0)}. \tag{5}$$

For the multiplier to be useful, the magnitude of e must be small. We shall impose the statistical bound that the variance of e should be less than $Q^2/12$, the variance of $r_{(0)}$. In this paper, we show that it is possible to obtain a sufficiently accurate value of C which makes the variance of e less than that of $r_{(0)}$ by using structures with k as small as 3.

II. Method I: Simple Fixed Bias Correction

In this method, each of the partial product term in the $(N + k + 1)$th through the $2N$th column is regarded as an independent random variable whose value is either 0 or 1. In general, the probability of a_i (or b_i) having the value 1 is 1/2. Since each of the partial product term a_ib_j is 1 if and only if a_i and b_j are both 1, the probability of a_ib_j having the value 1 is 1/4, and the probability of it having the value 0 is 3/4. Hence, the expected value of a_ib_j, denoted by $\eta_{(1)}$ is 1/4.

$$\eta_{(1)} = E\{a_ib_j\} = \frac{1}{4}. \tag{6}$$

The variance $\sigma^2_{(1)}$ of a_ib_j is

$$\sigma^2_{(1)} = E\left\{\left(a_ib_j - \eta_{(1)}\right)^2\right\} = \frac{3}{16}. \tag{7}$$

For $m > N + k$, there are $(2N + 1 - m)$ partial product terms in the mth column of Fig. 1. Hence, the expected value of the sum of the partial product terms in the mth column is $(2N + 1 - m)/4$. Since the mth column has a weight of 2^{-m}, the expected value of the contribution due to the mth column is $2^{-m}(2N + 1 - m)/4$. Hence, the expected value of the contribution due to the sum of the partial product terms in the $(2N + k + 1)$th through the $2N$th column is $C_{(1)}$ where

$$C_{(1)} = \frac{1}{4}\sum_{m=N+k+1}^{2N}(2N+1-m)2^{-m} = \epsilon_{(1)}Q \tag{8a}$$

where

$$\epsilon_{(1)} = \frac{1}{4}\sum_{n=k}^{N}(N+1-n)2^{-n}. \tag{8b}$$

The values of $\epsilon_{(1)}$ for various N and k are tabulated in Table I. The estimated product $P_{(1)}$ (i.e., the $P_{(E)}$ of method I) is obtained by adding $C_{(1)}$ to $P_{(T)}$.

$$P_{(1)} = P_{(T)} + C_{(1)}. \tag{9}$$

Let the difference between $P_{(1)}$ and $P_{(0)}$ be $e_{(1)}$

$$e_{(1)} = P_{(1)} - P_{(0)}. \tag{10}$$

It is straightforward to show that $E\{e_{(1)}\}$ is zero. Since each of the partial product term a_ib_j is regarded as an independent random variable, the variance of the sum of a_ib_j is equal to the sum of the variance of a_ib_j. Hence, the variance of $e_{(1)}$ is

$$\sigma^2_{(1)} = \nu_{(1)}Q^2 \tag{11a}$$

where

$$\nu_{(1)} = \frac{3}{16}\sum_{n=k+1}^{N}(n+1-m)2^{-2n} \tag{11b}$$

$$Q^2 = 2^{-2N}. \tag{11c}$$

The values of $\nu_{(1)}$ for various values of N and k are tabulated in Table I. In order to ensure that the variance of $e_{(1)}$ is less than that of $r_{(0)}$, $\nu_{(1)}$ must be less than 1/12. (Note that $1/12 = 0.0833$). From Table I, we note that for $N = 8$ or 16, $k = 2$ is adequate; for $N = 24$ and 32, $k = 3$ is sufficient.

When implemented in silicon, this approach produces a saving in the silicon area used. The amount of saving depends on the implementation scheme. Consider the case where the multiplier is implemented as a parallel multiplier using a large number of single-bit full adders. In the conventional double wordlength implementation, $N(N - 1)$ single-bit full adders are needed whereas using the foregoing proposed scheme only $N(N-1) - ((N-k)(N-k-1))/2$ single-bit full adders are needed. The percentage savings in terms of the number of single-bit full adders used are tabulated in Table II; this is also the savings in the silicon area used. Note that summing the correction constant and the result does not require additional hardware since this can be achieved by summing through the carry-in input of the first row adders.

The foregoing proposed technique is a nonexact computational technique trading hardware complexity for precision. It can easily be incorporated into other exact computational techniques reported in the literatures [5]–[7]. The improvement in operation speed depends on the actual hardware handling the carry signals. Since the proposed scheme uses less adders, there is a possibility that the operation speed may be improved. However, this is not always the case. For example,

TABLE I
VALUES OF $\epsilon_{(1)}$ AND $\nu_{(1)}$ FOR VARIOUS N AND k

N	k	$\epsilon_{(1)}$	$\nu_{(1)}$	N	k	$\epsilon_{(1)}$	$\nu_{(1)}$
8	0	1.7510	0.4792	24	0	5.7500	1.4792
8	1	0.7510	0.1042	24	1	2.7500	0.3542
8	2	0.3135	0.0221	24	2	1.3125	0.0846
8	3	0.1260	0.0046	24	3	0.6250	0.0202
8	4	0.0479	0.0009	24	4	0.2969	0.0048
16	0	3.7500	0.9792	32	0	7.7500	1.9792
16	1	1.7500	0.2292	32	1	3.7500	0.4792
16	2	0.8125	0.0534	32	2	1.8125	0.1159
16	3	0.3750	0.0124	32	3	0.8750	0.0280
16	4	0.1719	0.0029	32	4	0.4219	0.0068

TABLE II
PERCENTAGE SAVINGS IN TERMS OF THE NUMBER OF SINGLE-BIT FULL ADDERS

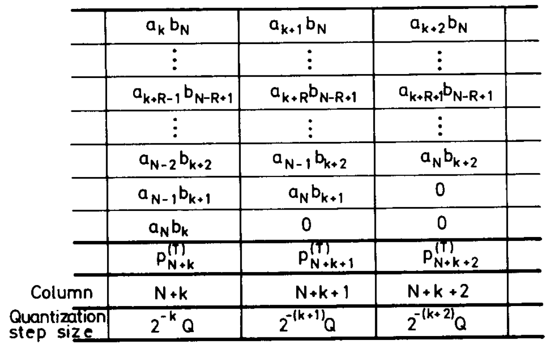
N	k	Savings	N	k	Savings
8	1	38%	24	1	46%
8	2	27%	24	2	42%
8	3	18%	24	3	38%
8	4	11%	24	4	24%
16	1	44%	32	1	47%
16	2	38%	32	2	44%
16	3	33%	32	3	41%
16	4	28%	32	4	38%

in a highly pipelined implementation such as that presented in [8], our proposed method does not produce any improvement in operation speed although it produces a reduction in hardware complexity.

III. METHOD II: PARTIAL PRODUCT CONDITIONAL CORRECTION

Assume that the value of the partial product term $a_m b_j$ is known and we wish to estimate the value of the partial product term $a_i b_j$ where $m \neq i$. If $a_m b_j = 1$, then b_j must also be 1; as a consequence, the probability of $a_i b_j$ having the value 1 is 1/2. Hence, $E\{a_i b_j\}|_{a_m b_j = 1}$ is 1/2 and the variance $E\{(a_i b_j - (1/2))^2\}|_{a_m b_j=1}$ is 1/4. If $a_m b_j = 0$, then either a_m or b_j (or both) must be zero. Thus, the probability of b_j having the value 1 is 1/3; as a consequence, $E\{a_i b_j\}|_{a_m b_j = 0}$ is 1/6 and the variance $E\{(a_i b_j - (1/6))^2\}|_{a_m b_j=0}$ is 5/36. The dependency between the values of $a_i b_j$ and $a_m b_j$ can be used to improve the estimate of the correction term. A full implementation of this technique requires considerable additional hardware but a partial implementation can be achieved easily with negligible additional cost.

In this method, the sum of the first R partial product terms, where R is a convenient integer, of the $(N + k)$th column (see Fig. 3) is used to estimate the sum of the first R partial product terms of the $(N + k + j)$th column. Suppose the sum of the first R partial product terms of the $(N + k)$th column is M, i.e. $a_k b_N + \cdots a_{k+R-1} b_{N-R+1} = M$. (This sum is readily available at no extra cost from the relevant wallace tree output.) This means that M of the partial product terms from $a_k b_N$ through $a_{k+R-1} b_{N-R+1}$ are 1 and R-M terms are 0. Hence, the expected value of the sum of the first R partial product terms in the $(N + k + 1)$th column, i.e., $E\{a_{k+1} b_N + \cdots a_{k+R} b_{N-R+1}\}$ is $(R/6) + (M/3)$. Although M is readily available, $M/3$ involves division by 3 which is a very complicated process and so is impractical for actual implementation. This difficulty is resolved by taking the $(N+k+2)$th column into account. When the sum of the first R partial product terms of the $(N + k + 1)$th and the $(N + k + 2)$th column is considered simultaneously (with the quantization step size taken



	$a_k b_N$	$a_{k+1} b_N$	$a_{k+2} b_N$
	$\vdots$	$\vdots$	$\vdots$
	$a_{k+R-1} b_{N-R+1}$	$a_{k+R} b_{N-R+1}$	$a_{k+R+1} b_{N-R+1}$
	$\vdots$	$\vdots$	$\vdots$
	$a_{N-2} b_{k+2}$	$a_{N-1} b_{k+2}$	$a_N b_{k+2}$
	$a_{N-1} b_{k+1}$	$a_N b_{k+1}$	0
	$a_N b_k$	0	0
	$p_{N+k}^{(T)}$	$p_{N+k+1}^{(T)}$	$p_{N+k+2}^{(T)}$
Column	N+k	N+k+1	N+k+2
Quantization step size	$2^{-k} Q$	$2^{-(k+1)} Q$	$2^{-(k+2)} Q$

Fig. 3. Computation of $P_{(T)}$.

into consideration), the expected value of the sum of the first R partial product terms of the $(N + k + 1)$th column and those of the $(N + k + 2)$th column is $2^{(k+1)}\ Q((R/4) + (M/2))$. Since divide-by-two is only a matter of wiring in hardware implementation, $M/2$ is readily available. Hence, this method is suitable for realization in hardware.

In this method, only the sum of the first R partial product terms of the $(N + k + 1)$th column and those of the $(N + k + 2)$th column are estimated using conditional probability based on the sum of the first R partial product terms of the $(N + k)$th column. The sum of the remaining partial product terms are estimated by treating each of them as an independent random variable with expected value equal to 1/4 in the same way as method I. The improvement over method I depends, of course, on R.

IV. MEHTOD III: TIME INVARIANT MULTIPLICAND

In signal processing applications such as in the realization of a digital filter, the multiplication process is mainly of the type of multiplying a signal data by a coefficient data. The signal data is a variable, i.e., every multiplication involves a different value, but the coefficient data is a constant, i.e., every multiplication involves the same value. In such applications, it is possible to count the number of 1's in the bit pattern of the coefficient data. This information is stored and recalled whenever that coefficient is used.

Let A be the signal data and let B be the coefficient data. Let $\sum_{j=k+1}^{N} b_j = L_{k+1}$ (see column $N + k + 1$ of Fig. 3). Since $E\{a_i b_j\}|_{b_j=1} = 1/2$ and since $E\{a_i b_j\}|_{b_j=0} = 0$, the sum $E\{\sum_{i=0}^{N-k-1} a_{N+1} b_{k+1+i}\} = L_{k+1}/2$. Its variance is $L_{k+1}/4$. Applying the same argument to column $N + k + j$ of Fig. 3, $E\{\sum_{i=0}^{N-k-j} a_{N+1} b_{N+j+i}\} = L_{k+j}/4$. Hence, using this scheme, the correction factor is

$$C_{(3)} = \frac{2^{-k}Q}{2} \sum_{j=1}^{N} 2^{-j} L_{k+j}. \tag{12}$$

The variance is

$$\sigma_{(3)}^2 = \frac{2^{-2k}Q^2}{4} \sum_{j=1}^{N} 2^{-2j} L_{k+j}. \tag{13}$$

Note that $C_{(3)}$ is a function of L_{k+j} and so depends on the bit pattern of the coefficient data. It is a constant for a given coefficient data but, in general, is different for different coefficient data. Hence, a computation overhead is incurred to compute the values of $C_{(3)}$ for all the coefficient data. This computation overhead is trivial if the coefficient data is used for a large number of times such as in the case of digital filtering.

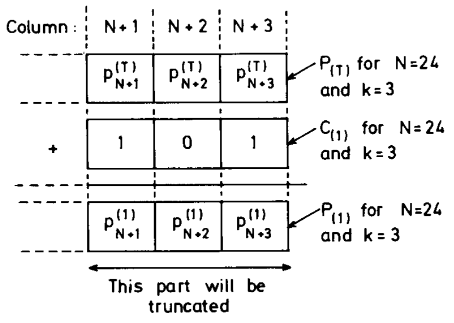

Column : N+1 N+2 N+3
$p^{(T)}_{N+1}$ $p^{(T)}_{N+2}$ $p^{(T)}_{N+3}$
$P_{(T)}$ for N=24 and k=3
+ 1 0 1
$C_{(1)}$ for N=24 and k=3
$p^{(1)}_{N+1}$ $p^{(1)}_{N+2}$ $p^{(1)}_{N+3}$
$P_{(1)}$ for N=24 and k=3
This part will be truncated

Fig. 4. Addition of $P_{(T)}$ and $C_{(1)}$ for $N = 24$ and $k = 3$.

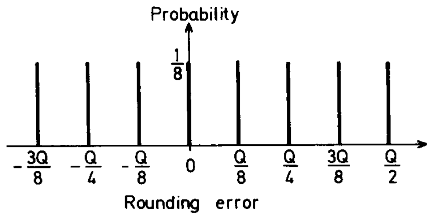


Fig. 5. Probability function of the rounding error of the result of Fig. 4.

V. Rounding Error Analysis

In this section, we shall examine the error due to rounding $P_{(E)}$ into the single-precision result. The standard practice in rounding error analysis is to model the rounding error as a zero mean noise source with variance $Q^2/12$. This analysis is true if a large number of bits is truncated during the rounding process and that the probability of any of the truncated bit having the value 1 is 1/2, i.e., a zero mean rectangular distribution for the rounding error is assumed.

In our schemes, since k is small, only a small number of bits is truncated during the rounding process. Furthermore, if the wordlength of C is larger than k, some of the truncated bits are constants. We shall examine these effects on the rounding error statistics.

We shall illustrate the rounding error analysis using two examples. The first example is that of method I with $N = 24$ and $k = 3$. The addition of $P_{(T)}$ and $C_{(1)}$ to obtain $P_{(1)}$ is shown in Fig. 4. When $P_{(1)}$ is rounded to single-precision, a rounding error is produced. The probability function of this rounding error is shown in Fig. 5. The mean of the rounding error is $Q/16$ and its variance is $21Q^2/256$ which is slightly less than $Q^2/12$.

The second example is that of method I with $N = 24$ and $k = 1$. In this example, the wordlength of $C_{(1)}$ is larger than k. The correction process is shown in Fig. 6. Notice that, in this case, $p^{(1)}_{N+2}$ is always 1. The probability function of the rounding error due to rounding $P_{(1)}$ into single precision is shown in Fig. 7. This rounding error has zero mean with variance $Q^2/16$.

Extending the above analysis to the analysis of other cases is straightforward.

VI. Conclusions

When only the single-precision product is desired, the circuit complexity of a multiplier can be significantly reduced if the least significant part of a double-wordlength product is not computed accurately. A good estimate of the least significant part is that all that is needed. In this paper, we propose and analyze three new methods for producing sufficiently accurate estimate of the least significant part.

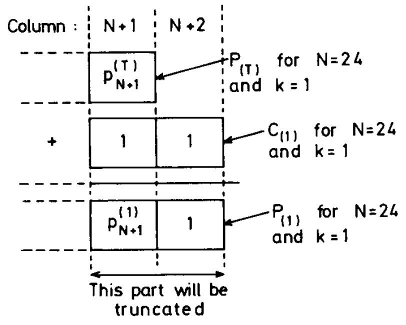

Column : N+1 N+2
$p^{(T)}_{N+1}$
$P_{(T)}$ for N=24 and k=1
+ 1 1
$C_{(1)}$ for N=24 and k=1
$p^{(1)}_{N+1}$ 1
$P_{(1)}$ for N=24 and k=1
This part will be truncated

Fig. 6. Addition of $P_{(T)}$ and $C_{(1)}$ for $N = 24$ and $k = 1$.

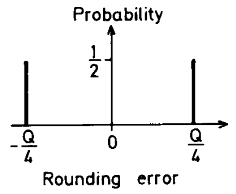


Fig. 7. Probability function of the rounding error of the result of Fig. 6.

References

[1] K. Hwang, *Computer Arithmetic: Principles. Architecture, and Design.* New York: Wiley, 1979.
[2] J. J. F. Cavanagh, *Digital Computer Arithmetic: Design and Implementation.* New York: McGraw-Hill, 1985.
[3] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Englewood Cliffs, N J: Prentice-Hall, 1975.
[4] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
[5] S. Nakamura and K. Y. Chu, "A single chip parallel multiplier by MOS technology," *IEEE Trans. Comput.,* vol. C-37, pp. 274–282, Mar. 1988.
[6] R. Gnanasekaran, "A fast serial-parallel binary multiplier," *IEEE Trans. Comput.,* vol. C-34, pp. 741–744, Aug. 1985.
[7] J. P. Shen and F. J. Ferguson, "The design of easily testable VLSI array multipliers," *IEEE Trans. Comput.,* vol. C-33, pp. 554–560, June 1984.
[8] T. G. Noll, D. S. Landsiedel, H. Klar, and G. Enders, "A pipelined 330-MHz multiplier," *IEEE J. Solid-State Circuits,* vol. SC-21, pp. 411–416, June 1986.