Brief Papers

A 200-MHz Complex Number Multiplier Using Redundant Binary Arithmetic

Kyung-Wook Shin, Bang-Sup Song, and Kantilal Bacrania

***Abstract*—Modern digital communication systems rely heavily on baseband signal processing for in-phase and quadrature (I–Q) channels, and complex number processing in low-voltage CMOS has become a necessity for channel equalization, timing recovery, modulation, and demodulation. In this work, redundant binary (RB) arithmetic is applied to complex number multiplication for the first time so that an N-bit parallel complex number multiplier can be reduced to two RB multiplications (i.e., an addition of N RB partial products) corresponding to real and imaginary parts, respectively. This efficient RB encoding scheme proposed can generate RB partial products with no additional hardware and delay overheads. A prototype 8-bit complex number multiplier containing 11.5K transistors is integrated on 1.05 $\times$ 1.34 mm^2 using 0.8-μm CMOS. The chip consumes 90 mW with 2.5-V supply when clocked at 200 MHz.**

***Index Terms*— Multiplication, multiplying circuits, redundant number systems.**

I. Introduction

In RECENT YEARS, digital signal processing has made many new applications of data communications possible through wire and wireless media. Modern digital communication systems using fast Fourier transform (FFT) and baseband I–Q signal processing inherently deal with data streams and coefficients represented by complex numbers. Because all major data transceiver functions such as channel equalization, timing recovery, modulation, and demodulation are done digitally, it is now increasingly important to implement high-speed vector arithmetic units such as complex number multipliers and dividers with low voltage and power for cost-effective digital processing and communication systems.

Complex number multiplication has been done using four real number multiplications and two additions. In real number processing, carry needs to propagate from the least significant bit (LSB) to the most significant bit (MSB) when binary partial products are added. Therefore, the addition and subtraction after binary multiplications limit the overall speed, although many techniques have been proposed to overcome the carry issue. A technique to reduce the arithmetic complexity of complex number multiplication is to use the algebraic transformation so that one real multiplication can be saved at the expense of three additions as compared to the direct method [1]. Many alternative solutions have also been proposed for complex number multiplication. Among them are the CORDIC (coordinate rotation digital computer) algorithm [2], the quadratic residue number system (QRNS) [3], bit-serial multiplication using offset binary and distributed arithmetic [4], and recently, the redundant complex number system (RCNS) [5]. They require either large overheads for pre- and postprocessing or long latency, and many design issues of speed, accuracy, design overhead, etc., should be addressed for fast multiplication [6].

In this work, we explore a new approach based on RB arithmetic to realize a compact and high-speed complex number multiplier with two clear goals in mind. One is the simplicity and modularity in multiplication for VLSI implementation, and the other is the elimination of the carry propagation for fast multiplication. First by employing the RB arithmetic, an N-bit complex number multiplication is transformed into two RB multiplications for real and imaginary terms of the final product. The RB multiplication is simply replaced by an addition of N RB partial products. When compared with existing methods such as the direct method or the strength reduction technique, the proposed approach results not only in simplified arithmetic operations, but also in a regular array-like structure. Second, the redundancy used in the RB representation allows the addition of partial products to be performed in parallel without carry propagation from the LSB through to the MSB. This carry-propagation-free (CPF) feature of the RB arithmetic has been used for many high-speed real number multiplier designs [7]–[12], but this is the first work that applies it to complex numbers.

An algorithm that represents complex number multiplication with the RB arithmetic is derived in Section II, and the design of an 8-bit complex number multiplier based on the proposed algorithm is covered in Section III. Experimental results of a prototype chip are presented in Section IV.

II. Complex Number Multiplication Based on RB Arithmetic

A. RB Number System

The RB representation is one of the signed-digit number representations (SDNR's) proposed by Avizienis [13] to reduce carry propagation in addition, subtraction, multiplication, and division. They differ from conventional numbers in that the

Manuscript received May 14, 1997; revised November 1, 1997. This work was supported in part by the Korea Science and Engineering Foundation (KOSEF).

K.-W. Shin is with the Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801 USA, on leave from the Kumoh National University of Technology, Korea.

B.-S. Song is with the Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801 USA.

K. Bacrania is with Harris Semiconductor, Melbourne, FL 32902 USA.

Publisher Item Identifier S 0018-9200(98)03580-X.

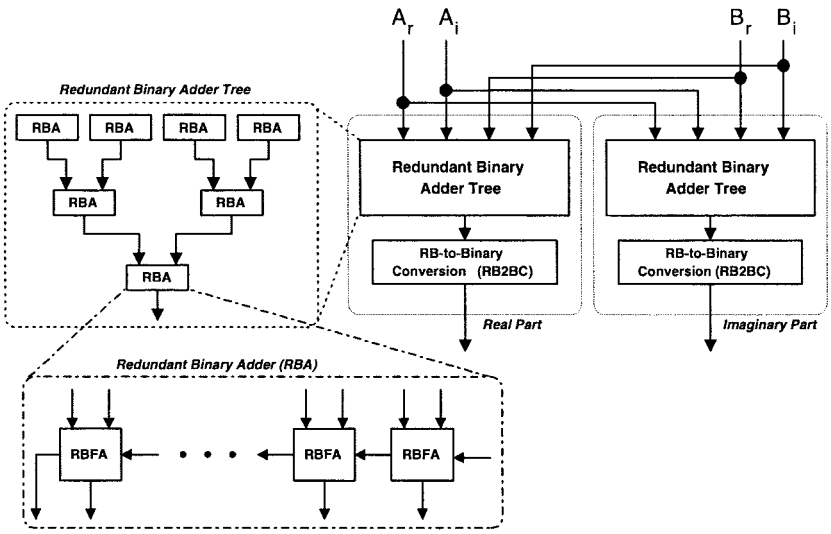


Fig. 1. Block diagram of the complex number multiplier.

individual digits comprising a number may have negative as well as positive values. For example, a radix-2 SDNR (i.e., RB representation) uses digits from set $\{-1, 0, 1\}$, and it allows the decimal value 5 to be represented as $[0101]_{\mathrm{RB}}$, $[011\bar{1}]_{\mathrm{RB}}$, $[1\bar{1}1\bar{1}]_{\mathrm{RB}}$, $[1\bar{1}01]_{\mathrm{RB}}$, or $[10\bar{1}\bar{1}]_{\mathrm{RB}}$. That is, five RB numbers are used to represent one number. This redundancy allows addition rules to be devised so that each digit of the result can be obtained using only two adjacent digits of the operands [7]. Therefore, arithmetic operations can be performed without carry propagation. The time required for parallel addition is fixed, and does not depend on the word length. Another feature of the RB representation is that there is no need for an explicit mechanism to handle the overall sign of a signed-digit number. The sign is determined by the sign of the most significant nonzero digit.

Signed-digit (SD) numbers are formally defined as follows. Given the maximum digit magnitude α, each digit of a SD number is in the range of $\{-\alpha, \cdots, -1, 0, 1, \cdots, \alpha\}$. When the radix r is 2, an SD number $D = [d_{n-1} \cdots d_1 d_0]$ where $d_i \in \{-1, 0, 1\}$ has an algebraic value of

$$D = \sum_{i=0}^{n-1} d_i 2^i. \tag{1}$$

Since d_i can take one of the three values, it should be encoded with two binary bits (x_i, y_i). This encoding can be defined in several ways, and one of the possible encoding schemes is

$$d_i \equiv (x_i, y_i) \equiv x_i - \overline{y}_i \tag{2}$$

where $\overline{y}_i$ denotes the inversion of y_i. Note that an SD d_i encoded by (x_i, y_i) has a value of $x_i - \overline{y}_i$.

Similarly, a useful relation between an N-digit RB number D and two N-bit binary numbers X and Y can be defined as

$$D \equiv (X, Y) \equiv X - \overline{Y} \tag{3}$$

where $\overline{Y}$ is obtained by inverting all bits of Y. If X and Y are 2's complement numbers, $X + Y$ can be expressed by using an RB number D as

$$X - (-Y) = X - (\overline{Y} + 1) = (X, Y) - 1. \tag{4}$$

Equation (4) implies that an RB number D can be regarded as the sum of two binary numbers. Therefore, RB can be converted into two binary numbers and vice versa. Based on this, a new RB representation of complex number multiplication is derived in the next subsection.

B. Proposed RB Complex Number Multiplication

Consider two complex numbers A and B represented in 2's complement form and their complex product Z expressed as follows:

$$\begin{aligned} A &= A_R + jA_I \\ &= \left(-a_{N-1}2^{N-1} + \sum_{k=0}^{N-2} a_k 2^k\right) \\ &\quad + j\left(-b_{N-1}2^{N-1} + \sum_{k=0}^{N-2} b_k 2^k\right) \\ B &= B_R + jB_I \\ &= \left(-c_{N-1}2^{N-1} + \sum_{k=0}^{N-2} c_k 2^k\right) \\ &\quad + j\left(-d_{N-1}2^{N-1} + \sum_{k=0}^{N-2} d_k 2^k\right) \end{aligned} \tag{5}$$

$$Z = (A_R + jA_I)(B_R + jB_I) = Z_R + jZ_I. \tag{6}$$

By substituting (5) into (6) and using 2's complement number characteristics, Z_R and Z_I become

$$\begin{aligned} Z_R &= A_R B_R - A_I B_I \\ &= (p_{N-1,N-1} - q_{N-1,N-1})2^{2N-2} \\ &\quad + \sum_{i=0}^{N-2}\sum_{j=0}^{N-2} (p_{i,j} - q_{i,j})2^{i+j} \\ &\quad + 2^{N-1}\sum_{i=0}^{N-2} (\overline{p}_{i,N-1} - \overline{q}_{i,N-1})2^i \\ &\quad + 2^{N-1}\sum_{j=0}^{N-2} (\overline{p}_{N-1,j} - \overline{q}_{N-1,j})2^j \end{aligned}$$

$$
\begin{aligned}
Z_I &= A_R B_I + A_I B_R \\
&= (u_{N-1,N-1} - \overline{v}_{N-1,N-1})2^{2N-2} \\
&\quad + \sum_{i=0}^{N-2}\sum_{j=0}^{N-2}(u_{i,j} - \overline{v}_{i,j})2^{i+j} \\
&\quad + 2^{N-1}\sum_{i=0}^{N-2}(\overline{u}_{i,N-1} - v_{i,N-1})2^{i} \\
&\quad + 2^{N-1}\sum_{j=0}^{N-2}(\overline{u}_{N-1,j} - v_{N-1,j})2^{j} + 1 \qquad (7)
\end{aligned}
$$

where $p_{i,j} = a_i c_j$, $q_{i,j} = b_i d_j$, $u_{i,j} = a_i d_j$, and $v_{i,j} = b_i c_j$ are bit-level partial products.

In order to convert (7) into RB, we define two RB numbers $\alpha_{i,j}$ and $\beta_{i,j}$, where $\alpha_{i,j}$, $\beta_{i,j} \in \{-1, 0, 1\}$, and their sign inversions $\alpha^*_{i,j}$ and $\beta^*_{i,j}$ (e.g., if $\alpha_{i,j} = 1$, then $\alpha^*_{i,j} = -1$) as

$$
\begin{aligned}
\alpha_{i,j} &\equiv p_{i,j} - q_{i,j} \\
\alpha^*_{i,j} &\equiv \overline{p}_{i,j} - \overline{q}_{i,j} \\
\beta_{i,j} &\equiv u_{i,j} - \overline{v}_{i,j} \\
\beta^*_{i,j} &\equiv \overline{u}_{i,j} - v_{i,j}. \qquad (8)
\end{aligned}
$$

By substituting (8) into (7), the RB representation of partial products is

$$
\begin{aligned}
P_R(i) &= \alpha^*_{i,N-1}2^{N-1+i} + \sum_{j=0}^{N-2}\alpha_{i,j}2^{i+j} \\
&\qquad (0 \le i \le N-2) \\
&= \alpha_{N-1,N-1}2^{2N-2} + \sum_{j=0}^{N-2}\alpha^*_{N-1,j}2^{N-1+j} \\
&\qquad (i = N-1) \\
P_I(i) &= \beta^*_{i,N-1}2^{N-1+i} + \sum_{j=0}^{N-2}\beta_{i,j}2^{i+j} \\
&\qquad (0 \le i \le N-2) \\
&= \beta_{N-1,N-1}2^{2N-2} + \sum_{j=0}^{N-2}\beta^*_{N-1,j}2^{N-1+j} \\
&\qquad (i = N-1). \qquad (9)
\end{aligned}
$$

Finally, the real and imaginary parts of the complex product are

$$
\begin{aligned}
Z_R &= \sum_{i=0}^{N-1} P_R(i) \\
Z_I &= \sum_{i=0}^{N-1} P_I(i) + 1. \qquad (10)
\end{aligned}
$$

That is, an N-bit complex number multiplication is reduced to two RB multiplications. One is for Z_R, and the other is for Z_I. It is not necessary to consider sign extension for partial product addition since the RB partial products already include sign-bit information.

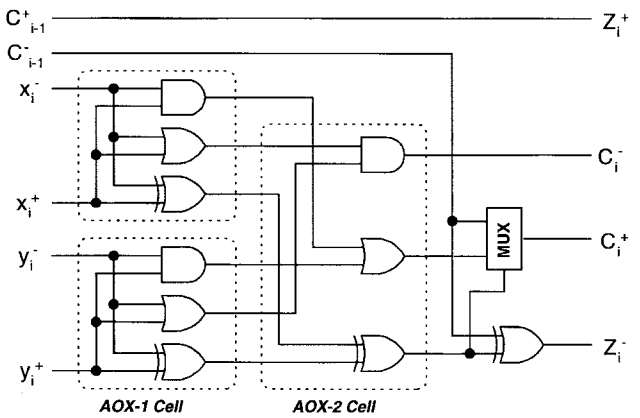


Fig. 2. Schematic diagram of RBFA circuit.

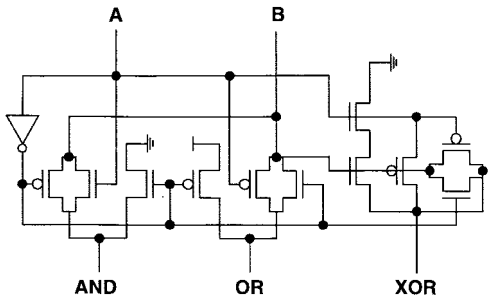


Fig. 3. TG-based implementation of AOX-1 cell.

TABLE I
GENERATION OF RB CODE BITS

$(0 \le i,j \le N-2)$ or $(i = j = N-1)$	$(i = N-1,\ 0 \le j \le N-2)$ or $(j = N-1,\ 0 \le i \le N-2)$
$s^-_{i,j} = p_{i,j} = a_i c_j$	$s^-_{i,j} = \overline{p_{i,j}} = \overline{a_i c_j}$
$s^+_{i,j} = \overline{q_{i,j}} = \overline{b_i d_j}$	$s^+_{i,j} = q_{i,j} = b_i d_j$
$t^-_{i,j} = u_{i,j} = a_i d_j$	$t^-_{i,j} = \overline{u_{i,j}} = \overline{a_i d_j}$
$t^+_{i,j} = v_{i,j} = b_i c_j$	$t^+_{i,j} = \overline{v_{i,j}} = \overline{b_i c_j}$

C. Proposed RB Coding Scheme

Since an RB digit has a digit set $\{-1, 0, 1\}$, it should be encoded with two binary bits. To simplify the RB encoding, two sets of binary codes $(s^-_{i,j}, s^+_{i,j})$ and $(t^-_{i,j}, t^+_{i,j})$ are defined to represent $\alpha_{i,j}$ and $\beta_{i,j}$, respectively

$$
\begin{aligned}
\alpha_{i,j} &\equiv \left(s^-_{i,j}, \overline{s}^+_{i,j}\right) \\
\alpha^*_{i,j} &\equiv \left(s^-_{i,j}, \overline{s}^+_{i,j}\right) \\
\beta_{i,j} &\equiv \left(t^-_{i,j}, \overline{t}^+_{i,j}\right) \\
\beta^*_{i,j} &\equiv \left(t^-_{i,j}, \overline{t}^+_{i,j}\right). \qquad (11)
\end{aligned}
$$

From (8) and (11), a mapping between RB code bits and binary partial products is obtained as summarized in Table I. Using this RB encoding scheme, the RB partial products are easily generated using simple two-input NAND or AND gates. No additional hardware or delay is needed to generate binary partial products.

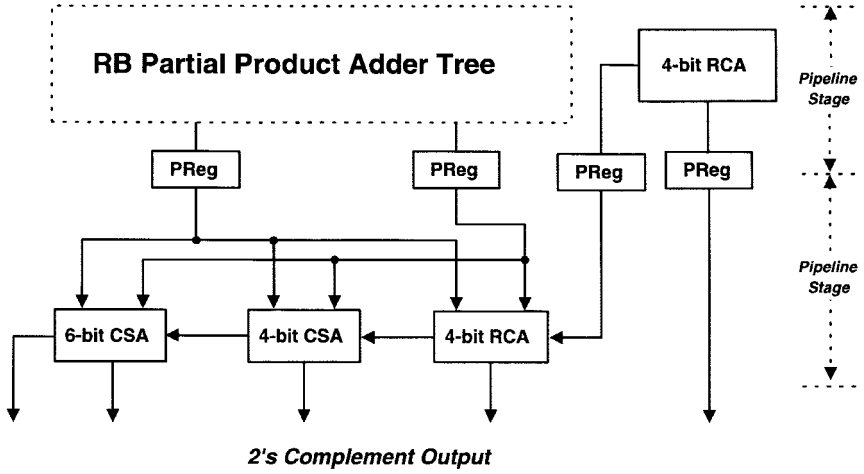


Fig. 4. Block diagram of RB2BC circuit.

TABLE II
LOGIC FUNCTIONS OF RBFA AND RBHA

RBFA Cell	RBHA Cell
$g_i = (x_i^- \oplus x_i^+) \oplus (y_i^- \oplus y_i^+)$	$g_i = (x_i^- \oplus x_i^+)$
$h_i = x_i^- x_i^+ + y_i^- y_i^+$	$h_i = x_i^- x_i^+$
$c_i^- = (x_i^- + x_i^+)(y_i^- + y_i^+)$	$c_i^- = (x_i^- + x_i^+)$
$c_i^+ = g_i c_{i-1}^- + \overline{g_i} h_i$	$c_i^+ = \overline{g_i} c_{i-1}^- + g_i h_i$
$z_i^- = g_i \oplus c_{i-1}^-$	$z_i^- = \overline{g_i} \oplus c_{i-1}^-$
$z_i^+ = c_{i-1}^+$	$z_i^+ = c_{i-1}^+$

III. COMPLEX NUMBER MULTIPLIER DESIGN

Fig. 1 shows the overall architecture of the complex number multiplier based on the algorithm proposed in Section II. It consists of two RB multipliers, which perform the operation given in (10), and an RB-to-binary conversion (RB2BC) block at the output stage. Each of the RB multipliers is implemented with a tree structure whose depth is $\log_2 N$ (where N is the word length of the multiplier). Each row in the tree consists of an array of RB full adder (RBFA) cells. The RB partial products are generated with simple NAND or AND gates, and they are incorporated into the RBFA cells at the first row of the tree. The RB2BC block converts the final result obtained from the RB adder tree into 2's complement form. The multiplier operation is pipelined with $(\log_2 N)+1$ pipeline stages including the RB2BC block.

An RBFA cell receives two RB digits (x_i^-, x_i^+) and (y_i^-, y_i^+), and a carry-in digit (c_{i-1}^-, c_{i-1}^+). It produces a sum digit (z_i^-, z_i^+) and a carry-out digit (c_i^-, c_i^+). Due to the redundancy in RB representation, a CPF addition can be defined so that carry propagation cannot occur from the $(i-1)$th digit to the $(i+1)$th digit through the ith digit. An addition of two RB partial products is performed in a fixed time interval independent of the word length. Table II describes the logic functions of RBFA and RB half adder (RBHA) circuits obtained from the CPF RB addition rule, and Fig. 2 shows the schematic diagram of the RBFA.

As shown in Fig. 1, the multiplier is constructed with regular arrays of RBFA cells. Therefore, it is important to design the RBFA cell compact and fast. To simplify this repeatedly used block, a circuit style based on transmission gates is used. Fig. 3 is the circuit diagram of an AOX-1 (AND/OR/XOR) cell to be used to construct RBFA and RBHA. The RBFA cell can be made with only 54 transistors, which is close to the simplest one reported to date [8]–[12].

The conversion of an N-digit RB number into a 2's complement number is carried out by an addition of two N-bit

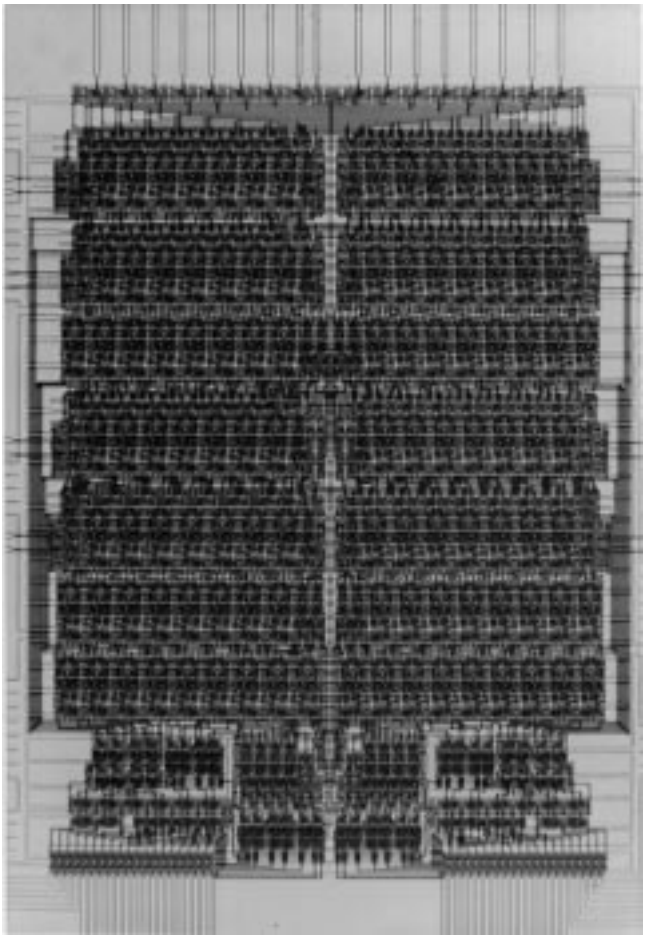
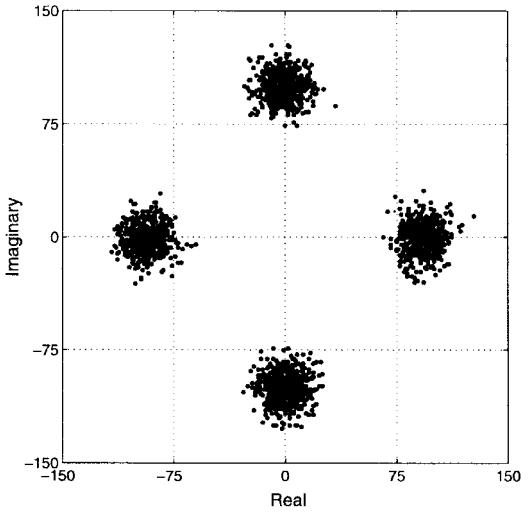
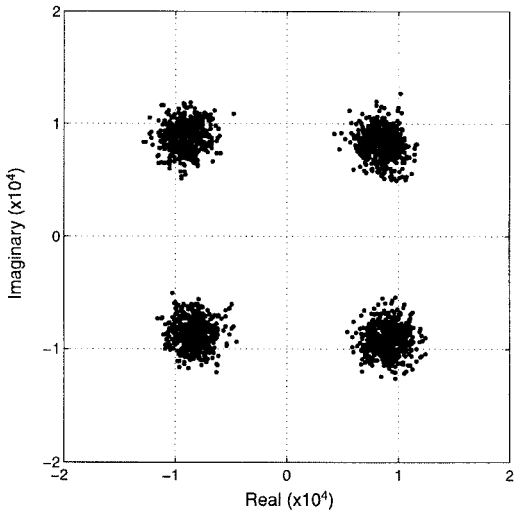


Fig. 5. Die picture of the complex number multiplier.



(a)



(b)

Fig. 6. Functional test results of the fabricated complex number multiplier: (a) constellation of test input vectors and (b) constellation of output vectors obtained from test chip, after rotating input vectors by 45°.

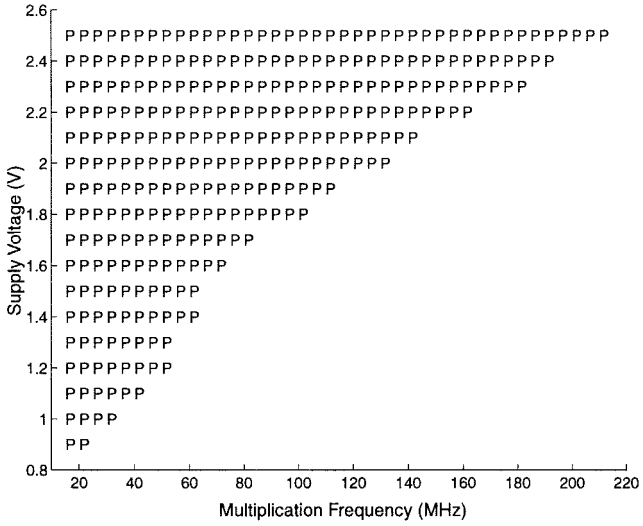


Fig. 7. Schmoo plot of supply voltage versus operating frequency.

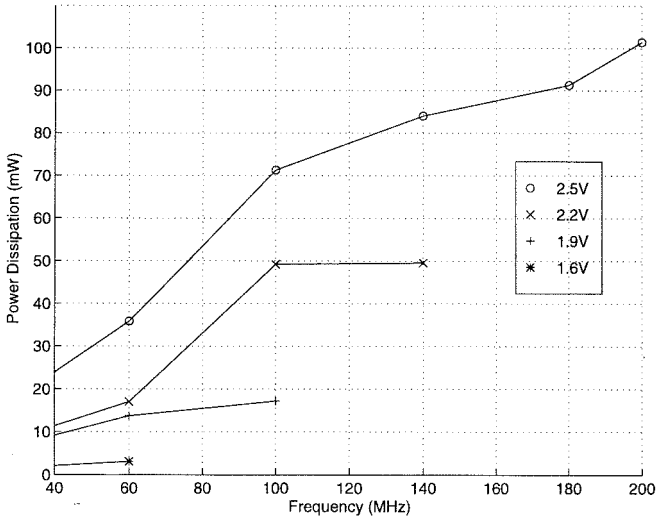


Fig. 8. Measured power dissipation.

numbers. However, the RB2BC block is a binary adder, and it has the carry propagation from LSB to MSB. That is, there is no carry propagation in the multiplication using RB, but the conversion back to binary needs a binary adder. This is the only binary adder needed in the entire multiplication process, and it can be designed using a hybrid structure for fast binary addition. The RB2BC block is made of a ripple-carry adder (RCA) and a carry-select adder (CSA) as shown in Fig. 4 so that high-speed conversion can be achieved with minimal hardware. An 8-bit RCA is used for the lower 8-bit addition, and a 10-bit CSA is used for the upper 10-bit addition. Therefore, an 18-bit addition can be completed within a single clock cycle, making the entire operation fully pipelined.

Careful consideration is given to the floorplan of the chip for compact layout. Key issues taken into account for compact layout are: 1) simple signal flow in the chip to minimize wiring area; 2) easy design by simply butting repetitive cells; 3) good extendibility for longer word lengths; and 4) proper aspect ratio of the final layout. On the basis of these considerations, a floorplan is configured by vertically stacking seven RBA blocks and an RB2BC block. The signal propagates in only one direction, resulting in a small wiring area. Since the multiplier operates in pipeline, the clock also follows pipelined stages as in the data flow path. The single-phase clocking scheme is adopted to minimize the overhead for clock signal distribution as well as to achieve a high clock rate. The clock is distributed using two separate clock buffer networks for the real and imaginary parts.

IV. Experimental Results

As a test vehicle, a prototype 8-bit complex number multiplier core is fabricated using 0.8-μm double-metal CMOS technology. Fig. 5 shows the die picture of the chip. It contains about 11 500 transistors on active area of 1.05×1.34 mm^2. The bottom one-fifth of the chip area is used as an overhead for the RB conversion back to binary.

TABLE III
SUMMARY OF MEASURED PERFROMANCE

Data format	8-b input and 16-b output complex numbers
Total transistor integrated	11,500
Active area	1.05×1.34 mm^2
Layout density	8,200 transistors/ mm^2
Operating clock frequency	200 MHz @ 2.5 V
Throughput rate	200 million vector multiplications/sec
Latency	4 clock cycles
Power dissipation	90 mW @2.5 V, 200 MHz
Technology	0.8 μm single-poly double-metal CMOS

The prototype chip was tested using digital I/O boards and a logic analyzer. To demonstrate the functionality of the chip for vector processing, a two-dimensional constellation of 2000 test input vectors as shown in Fig. 6(a) and a constant vector of $90 + j90$ are applied to the chip. This constellation closely resembles that of the quadrature phase-shift key (QPSK) modulation. The constellation of output vectors obtained from the chip is shown in Fig. 6(b), which shows that each of the input vectors is exactly rotated by 45°. Fig. 7 shows the Schmoo plot of the operating frequency versus supply voltage. The chip dissipates 90 mW with 200-MHz clock at 2.5-V supply. It means that its computational power is 200 million vector multiplications per second. The measured power dissipation as a function of operating frequency and supply voltage is shown in Fig. 8. Table III summarizes the measured performance of the prototype chip.

V. CONCLUSIONS

By applying RB arithmetic to complex number multiplication, we have the following. 1) An N-bit complex number multiplication is reduced to two RB multiplications. This is simpler than existing methods (i.e., the direct method or strength reduction technique) using real multipliers and adders. 2) The proposed approach is based on a highly parallel architecture with regularity and modularity. As a result, it leads to a more compact implementation than the conventional methods, making it very suitable for VLSI. 3) It operates at higher speed due to the carry propagation free addition of RB partial products. 4) Finally, there are no hardware and delay overheads in RB partial product compared with the binary partial product. The multiplier algorithm proposed in this paper enables us to achieve high-performance and compact implementation of complex number multipliers for high-speed DSP and digital communication systems.

ACKNOWLEDGMENT

The authors thank Y. H. Kim, M. J. Choe, and D. Wilson of University of Illinois, Urbana, for their help in testing.

REFERENCES

[1] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* Reading, MA: Addison-Wesley, 1987.
[2] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, pp. 330–334, Sept. 1959.
[3] R. Krishnan, G. A. Jullien, and W. C. Miller, "Complex digital signal processing using quadratic residue number systems," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, Feb. 1985.
[4] S. He and M. Torkelson, "A pipelined bit-serial complex multiplier using distributed arithmetic," in *Proc. ISCAS'95*, 1995, pp. 2313–2316.
[5] T. Aoki and T. Higuchi, "Redundant complex arithmetic," in *Proc. ITC-CSS'96*, July 1996, pp. 1230–1233.
[6] S. Y. Kung, H. J. Whitehouse, and T. Kailath, *VLSI and Modern Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1985.
[7] N. Takagi, H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," *IEEE Trans. Comput.*, vol. C-34, pp. 789–796, Sept. 1985.
[8] H. Edamatsu, T. Taniguchi, T. Nishiyama, and S. Kuninobu, "A 33 MFLOPS floating point processor using redundant binary representation," in *Dig. Tech. Papers, 1988 IEEE ISSCC*, Feb. 1988, pp. 152–153.
[9] X. Huang, W. J. Liu, and B. W. Y. Wei, "A high-performance CMOS redundant binary multiplication-and-accumulation (MAC) unit," *IEEE Trans. Circuits Syst. I*, vol. 41, pp. 33–39, Jan. 1994.
[10] H. Makino, Y. Nakase, H. Suzuki, *et al.*, "An 8.8-ns 54×54-bit multiplier with high speed redundant binary architecture," *IEEE J. Solid-State Circuits*, vol. 31, pp. 773–782, June 1996.
[11] W. Balakrishnan and N. Burgess, "Very-high-speed VLSI 2's complement multiplier using signed binary digits," *Proc. Inst. Elect. Eng.*, vol. 139, pt. E, pp. 29–34, Jan. 1992.
[12] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 28–33, Feb. 1987.
[13] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 389–400, Sept. 1961.