Journal of VLSI Signal Processing 38, 13–24, 2004

Complex ±1 Multiplier Based on Signed-Binary Transformations

BORIS D. ANDREEV, EDWARD L. TITLEBAUM AND EBY G. FRIEDMAN
Department of Electrical and Computer Engineering, University of Rochester, Rochester, New York 14627, USA

Received October 21, 2002; Accepted January 22, 2003

Abstract. A complex ±1 multiplier is an integral element in modern CDMA communication systems, specifically as a pseudonoise code scrambler/descrambler. An efficient implementation is essential to reduce the critical path delay, power, and area of wireless receivers. A signed-binary architecture is proposed to achieve this complex multiplier function. Tradeoffs and design solutions are discussed. It is shown that the VLSI circuit implementation of the arithmetic operations may be significantly improved by using non-conventional number representations and transforming intermediate results from one format to another format. For a target function, the objective is to change the number representations of the input and output operands such that a minimum amount of logic circuitry is required to achieve a computation. An analytical framework is developed that expands the scope of the functions that can be efficiently implemented using signed-binary representation. Simulations exhibit a significant speed improvement as compared to alternative architectures.

Keywords: VLSI, CDMA, PN code scrambler, complex ±1 multiplier, redundant arithmetic, signed-binary number representation

1. Introduction

Modern CDMA cellular systems employ spread spectrum technology to provide multiuser access with enhanced capacity and quality characteristics. In addition to the spectrum spreading operation, an integral part of the transceiver is the scrambling operation, which involves the multiplication of the chip sequence with a pseudonoise (PN) code in order to distinguish signals from asynchronous users. In the Third Generation Partnership Project (3GPP) wireless standard [1], the scrambling code is complex, therefore, a corresponding complex multiplication operation is required in both the transmitter and the receiver. The standard transmission scheme is shown in Fig. 1, where after spreading and scaling operations, a complex signal is formed and multiplied by a complex PN code. Since the components of the complex PN code take binary values in the set $\{-1, +1\}$, the scrambling multiplier should be optimized to reduce the critical path delay, power, and area of the wireless transceivers. Since no such circuits have been reported to date, both conventional and novel architectural solutions are presented here.

The bit-width of the input and output operands is among the primary characteristics of any arithmetic circuit. A sufficient fixed-point number representation is dependent on both the parameters of the cellular system and the particular detection algorithms. As described in [2] and [3], the implementation of certain multiuser algorithms with 8-bit to 16-bit representation of the received signal suffers negligible performance degradation as compared to a system implemented with floating point precision. Therefore, arithmetic circuits on the order of 8 to 16 bits are discussed in this paper with particular attention focused on an 8-bit representation.

These results are not limited to the application of a complex ±1 multiplier as a scrambler in wireless transceivers. The ideas and relations may also be used as a basis for the efficient implementation of other arithmetic circuits.

Conventional architectural solutions to the complex multiplier problem are formally introduced in

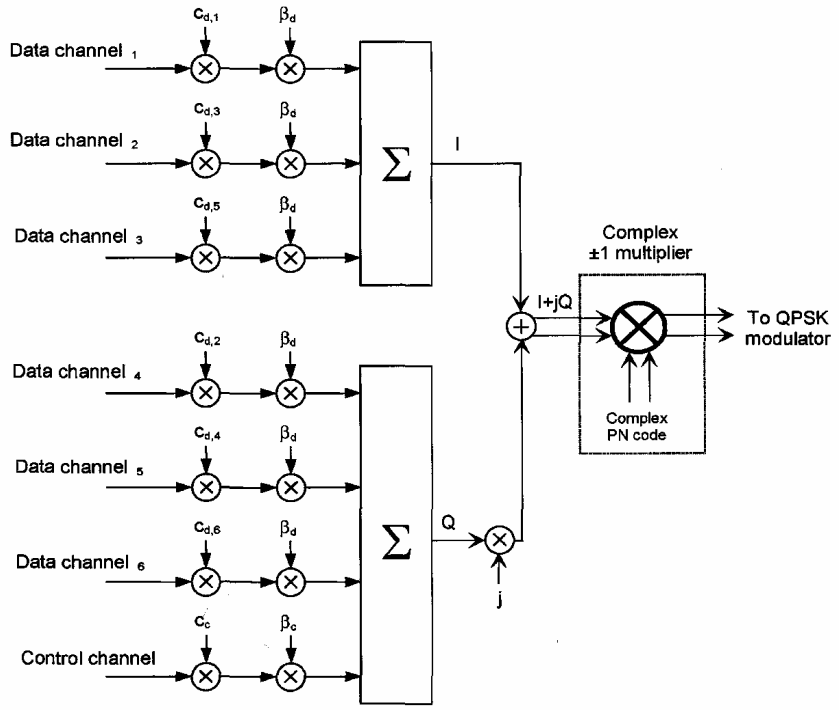


Figure 1. 3GPP standard transmission scheme [1].

Section 2. The core of this paper is Section 3, where an analytical framework for the efficient implementation of the required functions is developed and a new signed-binary architecture for the complex ±1 multiplier is proposed. Logic level design issues are discussed in Section 4 while simulation data and a comparison of the proposed implementation with standard alternatives are summarized in Section 5.

2. Conventional Architectural Solutions

A symbolic description of a ±1 complex multiplier is shown in Fig. 2 where each of the outputs can take on one of four possible values (as characterized in Table 1). The input signal is described by the complex number $a+jb$ and the PN code by $PN_{re} + jPN_{im}$ where PN_{re} and PN_{im} are in the binary set of $\{-1, +1\}$. The complex output signal is $A+jB = (a+jb)\cdot(PN_{re}+jPN_{im})$. All of the numbers, a, b, A, and B, are represented in two's-complement (TC) format with N-bit precision for the inputs and $N+1$-bit precision for the outputs.

The structure of a complex ±1 multiplier circuit is therefore different from that of a general purpose complex multiplier. Rather than considering two complex

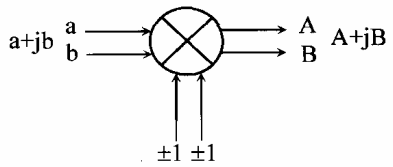


Figure 2. Schematic symbol of a complex ±1 multiplier.

input operands, there is only one complex input and a set of two binary control signals, PN_{re} and PN_{im}. Two attractive architectural solutions to achieve this function are shown in Fig. 3. In the Type I architecture, the two branches are completely independent. Both branches may produce any one of the four functions, whereas in the Type II architecture each branch is dedicated to providing either the $\pm(a+b)$ or the $\pm(a-b)$

Table 1. Input/output relations for a ±1 complex multiplier.

PN code		Outputs	
PN_{re}	PN_{im}	A	B
1	1	$a-b$	$a+b$
1	−1	$a+b$	$-(a-b)$
−1	1	$-(a+b)$	$a-b$
−1	−1	$-(a-b)$	$-(a+b)$

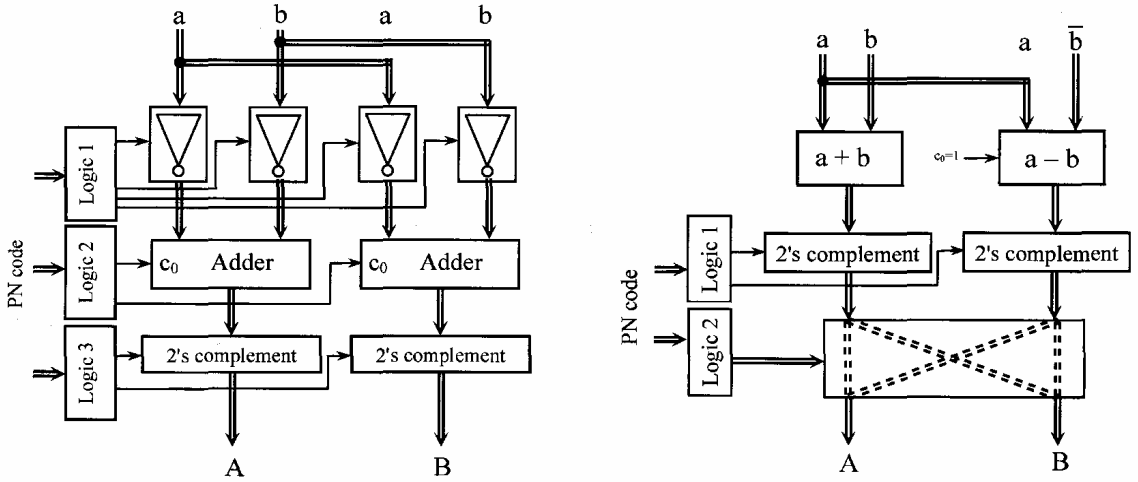


Figure 3. Conventional architectural solutions for a complex ±1 multiplier.

functions and a final switch is required to map these results to the correct complex output. Area and power improvements may be achieved by exploiting common features between the $a + b$ and $a - b$ operations. Circuit speed may be increased by reducing the overhead of the two's-complement circuits and the final switch in the Type II architecture. The conditional switches controlled by the PN logic produce additional delay along the critical path and, therefore, the number of these gates should be minimized. Note that the critical path delay of both branches must be equal, so that valid results appear almost simultaneously at the outputs.

3. Proposed Signed-Binary Architecture

3.1. *Signed-Binary Approach*

Generally, numbers in VLSI-based digital circuits are represented in two's-complement format to facilitate the implementation of arithmetic operations. During the past decade, redundant arithmetic has received increasing interest due to the attractive features for carry free addition [4–7], leading to significant benefits in the implementation of wide adders and more complex arithmetic functions. Parallel addition is performed by selecting an intermediate representation of the sum of two numbers $a + b$ such that the final result is obtained using simple logic without need for carry propagation. Although addition with redundant arithmetic techniques may offer significant improvements in computing speed, efficient circuit implementations have traditionally been difficult to achieve. Since input and output operands of the arithmetic circuits are often required in two's-complement format, conversion circuits to/from the intermediate representations are needed. These interface circuits degrade the overall improvement in speed—the conversion delay overhead must be smaller than the delay reduction achieved using parallel computation techniques. For these reasons, many systems for fast arithmetic, such as the residue number system (RNS) [8] and the logarithmic number system (LNS) [9], have not received widespread use because of the significant overhead of the conversion process. Alternatively, the number representations proposed in this paper may be transformed relatively easily to/from two's-complement format using the transformations and circuits described in the following sections.

The joint realization of four functions, $(a+b)$, $-(a+b)$, $(a-b)$, and $-(a-b)$, with minimum resources is discussed in this paper. For a particular arithmetic function, the objective is to select the number representations of the input and output numbers such that a minimum amount of logic circuitry is required to achieve the computation. Of all redundant sets, the *signed-binary* (SB) set $S_x = \{0, 1, \bar{1}\}$ and the *initial sum* set $S_y = \{0, 1, 2\}$, have received significant attention due to the small size, relative ease of representation with the binary number system, and low conversion overhead to/from the conventional two's-complement format. The addition of two numbers in

two's-complement format is essentially equivalent to the conversion of a signed-binary number representation (SBNR) into the two's-complement counterpart, as described by Blair [4]. The SBNR in sign-magnitude form is selected as an internal representation of the proposed complex ± 1 multiplier architecture. The attractiveness of the signed-binary (SB) approach lies in the parallel block implementation of an adder [5, 6] or in the utilization of this format in sequential arithmetic operations without the overhead of the final back conversion to TC [7]. In signed-binary format, the numbers b and $-b$ differ only in the sign bits. This feature may be exploited to identify common stages of the $a + b$ and $a - b$ operations. Inverting a number in sign-magnitude format is accomplished by inverting all of the sign bits. This operation is more efficient than two's-complement, which has a complexity on the order of an adder. The benefits of the signed-binary representation may potentially increase if several arithmetic stages are incorporated in an SBNR tree before the final conversion to two's-complement [7]. Considering the significant complexity of CDMA multiuser detection algorithms [1–3], a number of demanding operations may be implemented in this intermediate format without inefficiently transforming all results into two's-complement format. This strategy leads to considerable improvements in power, area, and delay as compared to conventional TC arithmetic.

Having chosen a number representation, the tradeoffs between the two proposed architectures are analyzed in more detail. The final switch of the Type II solution shown in Fig. 3 directs the results to the real and imaginary outputs. This switch may be eliminated if both the addition and subtraction operations over any operands ($\pm a$, $\pm b$) are produced in both branches as is the case of the Type I architecture. The switch may be implemented as $2(N + 1)$ multiplexers, while in the Type I circuit, $4N$ gates are required. This switch, however, becomes less efficient if the SBNR results are supplied to the following stage, doubling the number of output lines to $4N$.

3.2. *Transformations Between Signed-Binary and Two's Complement Number Representations*

In order to benefit from the advantages of redundant arithmetic, the SB number must be transformed into the two's-complement format of the target function. This transformation is relatively straightforward to achieve for a single function but becomes nontrivial when four functions are required for the real and imaginary outputs as specified in Table 1. One approach is to replace the addition and formation of the carry-lookahead adder (CLA) generate-propagate (G-P) signals with a preprocessing stage as suggested in [4]. This concept is further developed to achieve an efficient implementation of all four functions of interest: $(a + b)$, $-(a + b)$, $(a - b)$ and $-(a - b)$. The objective is to ensure that the two branches have close to equal delay times with a minimum number of conditional switches controlled by the PN code logic.

The sum of any two bits $a_i + b_i$ may be represented by a digit y_i in the *initial sum* set $S_y = \{0, 1, 2\}$. The sum of two N-bit numbers $a + b$ may therefore be expressed as an N-digit *initial sum* number y, with digits $y_i \in S_y$. The digits from the two sets, S_x and S_y, are related through the self-inverting transformation $tr_{xy}(z_i) = 1 - z_i$, $z_i \in \{S_x \cup S_y\}$ [4]. The relations among the input bits and the digits from the two sets S_x and S_y are shown in Table 2, where the first transformation performs the sum and the second transformation is represented by $\mathrm{tr}_{xy}(y_i)$, where the initial sum digit y_i is mapped to the signed-binary digit x_i.

The numerical values of some N-digit numbers from both sets may be expressed as

$$T_x(x) = \sum_{i=0}^{N-1} x_i 2^i \tag{1}$$

and

$$T_y(y) = \sum_{i=0}^{N-1} y_i 2^i. \tag{2}$$

In the following discussion, the notations $T_x(x)$ and $T_y(y)$ are used to denote both the numerical values of the corresponding number, x or y, as well as the $(N + 1)$-bit representation of these values in two's complement binary format. Note that the value $T_y(y)$ is equal to the sum of $a + b$; only the number format of

Table 2. Redundant arithmetic transformations at the prelogic stage.

Input bits			Initial sum digit $y_i \in S_y$				Signed-binary digit $x_i \in S_x$		
a_i	b_i	+	y_i	c_{i+1}	S_i	tr_{xy}	x_i	sign_i	magn_i
0	0		0	0	0		1	0	1
01, 10		→	1	0	1	→	0	0	0
1	1		2	1	0		$\bar{1}$	1	1

y is not binary. As shown in [4], if an initial sum y is transformed into a SB number x through the digit transformation $\mathrm{tr}_{xy} : y_i = 1 - x_i$, the relation between the two's-complement numbers is

$$a + b = T_y(y) = T_y[\mathrm{tr}_{xy}(x)] = \sum_{i=0}^{N-1} \mathrm{tr}_{xy}(x_i)2^i$$
$$= 2^N - 1 - T_x(x) = 2^N + \overline{T_x(x)}, \qquad (3)$$

where $-T_x(x) = \overline{T_x(x)} + 1$ and $\overline{T_x(x)}$ is the binary number $T_x(x)$with all bits inverted. Equation (3) and the relations in Table 2 reveal the redundancy of the intermediate representation in S_y and the direct relations for the sign-magnitude format,

$$\mathrm{sign}_i = a_i b_i = G_i \qquad (4)$$
$$\mathrm{magn}_i = \overline{a_i \oplus b_i} = \overline{P_i}. \qquad (5)$$

These functions are implemented in *prelogic* stages in both branches. Note that these intermediate signals are the same as the carry-generate G_i and the inverse of the carry-propagate P_i in the carry-lookahead adder [6, 10, 11]. It is conceptually convenient to distinguish between these signals such that a number is referred to as sign-magnitude when discussing the SBNR form. These signals are applied (with the magnitude inverted) to the corresponding *G-P* inputs of a CLA. An alternative VLSI implementation of an adder may therefore be achieved by mapping the bits of the two N-bit numbers to signed-binary digits x_i from S_x (skipping the initial summation to y_i), converting the redundant number into the two's-complement counterpart $T_x(x)$, and finally, transforming that result into the sum of the numbers $T_y(y)$.

The conversion of an SB number to TC is achieved through the reverse application of (3) (tr_{xy} is symmetric). This approach produces more flexible arithmetic circuits when certain manipulations of the intermediate results (or several consecutive operations) are applied to perform carry-free addition in the $S_x - S_y$ domain. Efficiency is achieved because expensive operations in the two's-complement domain are performed with less resources expended on the intermediate signed-binary number which is easily manipulated (inverted and/or added) without a carry propagation delay. The SB number is mapped onto the correct result in two's-complement. The most expensive component in a signed-binary architecture is the SB $\rightarrow$ TC conversion. This operation is essentially equivalent to a two's-complement addition. Existing efficient adder structures may be applied in this conversion process [4]. The implementation of an 8-bit circuit is possible through a carry-select architecture with 4-bit carry-generation blocks [5, 6] or via a standard 8-bit carry-lookahead adder (CLA). $T_x(x)$ becomes the sum of the numbers through an inversion of all but the last bit of $T_y(y)$ as described by (3).

Based on (3), the complimentary function $-(a + b)$ can be expressed as

$$-(a + b) = -T_y(y) = -(2^N - 1 - T_x(x))$$
$$= -2^N + 1 + T_x(x). \qquad (6)$$

As described by (3), the addition operation may be accomplished by inverting the bits of $T_x(x)$. Unfortunately, such a realization is not possible for the $-(a+b)$ function. In order to minimize the circuit differences between the $(a + b)$ and $-(a + b)$ functions, as realized in the left branch, the alternative realization of $a + b$ through the $-[T_x(x) + 1]$ operation is used. The left branch prelogic maps the input bits directly to $T_x(x) + 1$ while the rest of the circuit remains the same as the right branch. Conventional addition of 1 requires a carry propagation chain, making the $-(a + b)$ function difficult to implement, however, the "+1" operation is simple to perform in SB. Any addition of a signed-binary number with a two's-complement number may be completed in two gate delays, producing a signed-binary output [6]. An algorithm to implement the $-(a + b)$ function is:

1. Sum the two input operands bitwise, producing an initial sum in the set $S_y = \{0, 1, 2\}$.
2. Map this result to a signed-binary number using $x_i = (1 - y_i), x_i \in S_x = \{0, 1, \bar{1}\}$.
3. Perform the addition of +1 in SB format in two gate delays. The result is in borrow-save signed-binary form [5, 6].
4. Convert the result from SB to TC format using a conventional adder [4–6].

Note that the first three operations are simple logic functions over a limited number of input operands, making this algorithm amenable to optimization. It is assumed that the result of the +1 addition must be in sign-magnitude format. A number in signed-magnitude format can be efficiently transformed and passed to the carry generate-propagate (*G-P*) inputs of a carry lookahead adder (CLA), where the conversion to TC is performed [11, 12]. Applying conventional logic optimization, the $(N + 1)$-digit signed-binary result

corresponding to $T_x(x)+1$ is

$$d_0'' \begin{cases} S_0'' = \overline{M_0'} = a_0 \oplus b_0 \\ M_0'' = \overline{M_0'} = a_0 \oplus b_0 \end{cases} \tag{7}$$

$$d_1'' \begin{cases} S_1'' = M_1' \cdot S_0' = \overline{a_1 \oplus b_1} \cdot a_0 b_0 \\ M_1'' = \overline{M_1' \oplus S_0'} = \overline{a_1 \oplus b_1} \oplus a_0 b_0 \end{cases} \tag{8}$$

$$d_i'' \begin{cases} S_i'' = M_i' \cdot \overline{S_{i-1}' M_{i-1}'} \\ \quad = \overline{a_i \oplus b_i} \cdot (a_{i-1} + b_{i-1}) \\ M_i'' = M_i' \oplus \left(\overline{S_{i-1}' M_{i-1}'}\right) \\ \quad = \overline{a_i \oplus b_i} \cdot \overline{a_{i-1} + b_{i-1}} \end{cases} 2 \le i \le N-1 \tag{9}$$

$$d_N'' \begin{cases} S_N'' = 0 \\ M_N'' = \overline{S_{N-1}'} M_{N-1}' = \overline{a_{N-1} + b_{N-1}} \end{cases}. \tag{10}$$

Each digit at the output d_i'' is expressed in sign-magnitude form and is a function of the signed-binary input or the input two's-complement operands, a and b. With these expressions, the signed-binary number representation (SBNR) of $T_x(x)+1$ is achieved in two gate delays. This number is converted to two's-complement or, alternatively, may be inverted while in signed-binary form to alternate between the $(a+b)$ and $-(a+b)$ functions.

Similar expressions are considered for the difference $a-b$. In this case, the SB digits x_i are produced from the initial sum y_i, which is obtained from the bit-wise sum $a_i + \overline{b_i}$. The function is equal to

$$\begin{aligned}(a-b) &= T_y[y]+1 = T_y[\mathrm{tr}_{xy}(x)]+1 \\ &= 1 + \sum_{i=0}^{N} (1-x_i)2^i \\ &= 1 + [2^N - 1 - T_x(x)] = 2^N + T_x(x). \end{aligned} \tag{11}$$

The x number is in signed-binary representation. The inverse $-T_x(x)$, the same number with toggled sign bits, is converted to TC to achieve $a-b$. The corresponding inverse function $-(a-b)$ is

$$-(a-b) = -(2^N - T_x(x)) = -2^N + T_x(x). \tag{12}$$

The implementation of the two difference functions is similar to that of the summation functions, with the exception of the $+1$ addition. This addition is conveniently incorporated in a prelogic stage by (7)–(10). The transformations and operations required to compute the four functions, with two alternative implementations of the $a+b$ addition, are summarized in Table 3. Note that although the sign-magnitude combination "10" is forbidden, the sign inversion does not cause problems in the CLA performing the SB→TC conversion. This behavior occurs because in the case of this forbidden bit pair, the corresponding generate-propagate (G-P) signals of the CLA become "11" and the value of the generate bit does not affect the output result [11, 12].

This algorithm is only correct if the input operands are N-bit unsigned numbers or if the output result does not cause overflow in the N-bit precision. This issue is addressed in Appendix A where it is shown that for two's-complement numbers, the Nth (sign) bit of the general $(N+1)$-bit result is

$$\begin{aligned} r_N &= a_{N-1} b_{N-1} + (a_{N-1} \oplus b_{N-1}) \overline{c_{N-1}} \\ &= G_{N-1} + P_{N-1} \overline{c_{N-1}}, \end{aligned} \tag{13}$$

where r_i denotes the ith bit of the output result and c_i is the input carry of the ith full-adder cell. This function may be conveniently implemented by an inversion of the carry bit c_{N-1} passed to the circuit forming the final

Table 3. Summary of transform relations for the functions in a complex ±1 multiplier.

Arithmetic function	Bitwise processing	Relation to $T_Y(y)$	Relation to $T_x(x)$	Implementation description
$a+b$	a_i, b_i	$T_Y(y)$	$2^N - (1 + T_x(x))$	1. Obtain all x_i from a_i, b_i, with $+1$ and invert sign bits 2. Produce the 2's complement $-(T_x(x)+1)$, set the Nth bit
$-(a+b)$	a_i, b_i	$-T_{Y_y}(y)$	$-2^N + 1 + T_x(x)$	1. Obtain all x_i from a_i, b_i with $+1$ 2. Produce the 2's complement $T_x(x)+1$, set the Nth bit
$(a-b)$	$a_i, \overline{b_i}$	$T_Y(y)+1$	$2^N - T_x(x)$	1. Obtain all x_i from a_i and $\overline{b_i}$ 2. Invert all sign bits of x 3. Produce the 2's complement $-T_x(x)$, set the Nth bit
$-(a-b)$	$a_i, \overline{b_i}$	$-[T_Y(y)+1]$	$-2^N + T_x(x)$	1. Obtain all x_i from a_i and $\overline{b_i}$ 2. Produce the 2's complement $T_x(x)$, set the Nth bit

carry c_N, in order to account for the negative weight of the input sign bits. Since the sign bit is controlled according to (10), the computation of the Nth digit as in (9) is unnecessary (only r_N and r_{N+1} are controlled by d_N). An N-digit SB number is therefore computed.

Alternative implementations of all four functions may be obtained if a different mapping between the two sets $S_x \leftrightarrow S_y$ is applied. These transformations may be preferred in some applications and are discussed in Appendix B.

The realization of the $\pm(a - b)$ functions in the right branch is the same as that of the left branch with the exception of the "+1" addition. In the specific logic implementation, this additional operation produces a negligible delay overhead. Since the left branch prelogic must operate on a and $\bar{b}$ (see Table 3), it is preferable to combine the prelogic stages at either the gate or the layout level, considering the following relations,

$$\text{sign}_i(+) = a_i b_i \tag{14}$$

$$\text{sign}_i(-) = a_i \overline{b_i} \tag{15}$$

$$\text{magn}_i(+) = \overline{a_i \oplus b_i} \tag{16}$$

$$\text{magn}_i(-) = \overline{a_i \oplus \overline{b_i}} = \overline{\text{magn}_i(+)}. \tag{17}$$

An architecture that includes these design concepts is shown in Fig. 4.

4. Logic Level Design

Most of the operations along the critical path are implemented in NMOS CPL logic due to the speed, power efficiency [13], and complementary outputs. Complementary outputs are employed to achieve an efficient implementation of the conditional inverters by integrating these circuits with the previous stage. The complementary outputs also support resource sharing between the $(a + b)$ and $(a - b)$ branches [see (14)–(17)].

The adder performing the conversion to two's-complement is the key logic circuit of the architecture shown in Fig. 4. Adder circuits have been extensively discussed in the literature and are applicable to the conversion process [4]. A slow but area- and power-efficient adder is preferable as long as the circuit speed satisfies the target specification.

Several relations can be deduced from the full adder truth Table [11], (1)–(13), and the Karnaugh maps as-

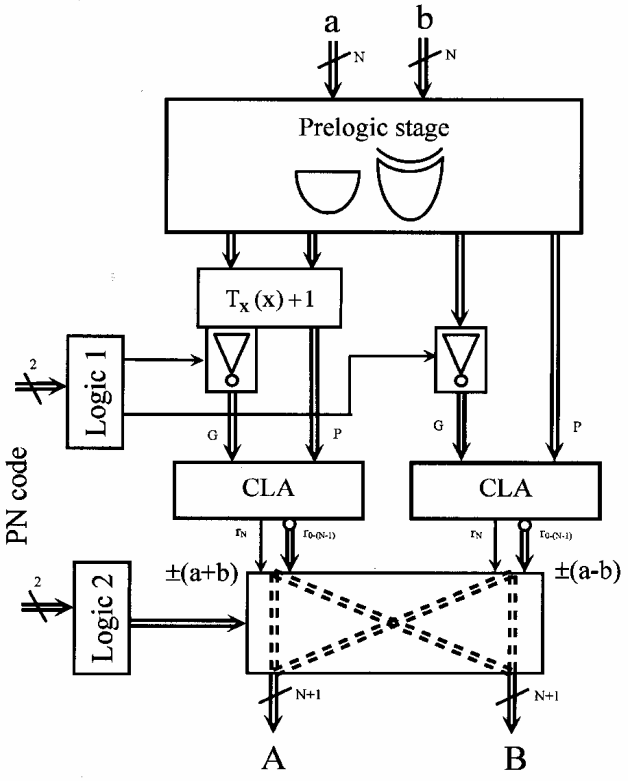


Figure 4. SBNR architecture of a complex ±1 multiplier.

sociated with the corresponding signals,

$$G_i = \overline{P_i} G_i, \quad P_i = P_i \overline{G_i}, \tag{18}$$

$$c_{i+1} = G_i + P_i c_i = \overline{P_i} G_i + P_i c_i, \quad 0 \le i \le (N-1) \tag{19}$$

$$\overline{c_{i+1}} = \overline{P_i}\,\overline{G_i} + P_i \overline{c_i}, \quad 0 \le i \le (N-1) \tag{20}$$

$$r_i = P_i \oplus c_i = \overline{P_i} \oplus \overline{c_i} = \overline{P_i \oplus \overline{c_i}}, \quad 0 \le i \le (N-1) \tag{21}$$

$$r_N = G_{N-1} + P_{N-1} \cdot \overline{c_{N-1}} = \overline{P_{N-1}} G_{N-1} + P_{N-1} \cdot \overline{c_{N-1}}, \tag{22}$$

$$\overline{r_N} = \overline{P_{N-1}} \cdot \overline{G_{N-1}} + P_{N-1} \cdot c_{N-1}. \tag{23}$$

As shown in (19)–(21), the inverted carry may also be propagated, permitting a single inverter to be inserted as a repeater to improve the delay characteristics of the carry propagation chain. The sign bit is controlled according to (22) and (23). An expression for the sign inverse is also implemented by an inverter to achieve higher output current and enhanced noise margins. Since r_N is a function of c_{N-1}, an even

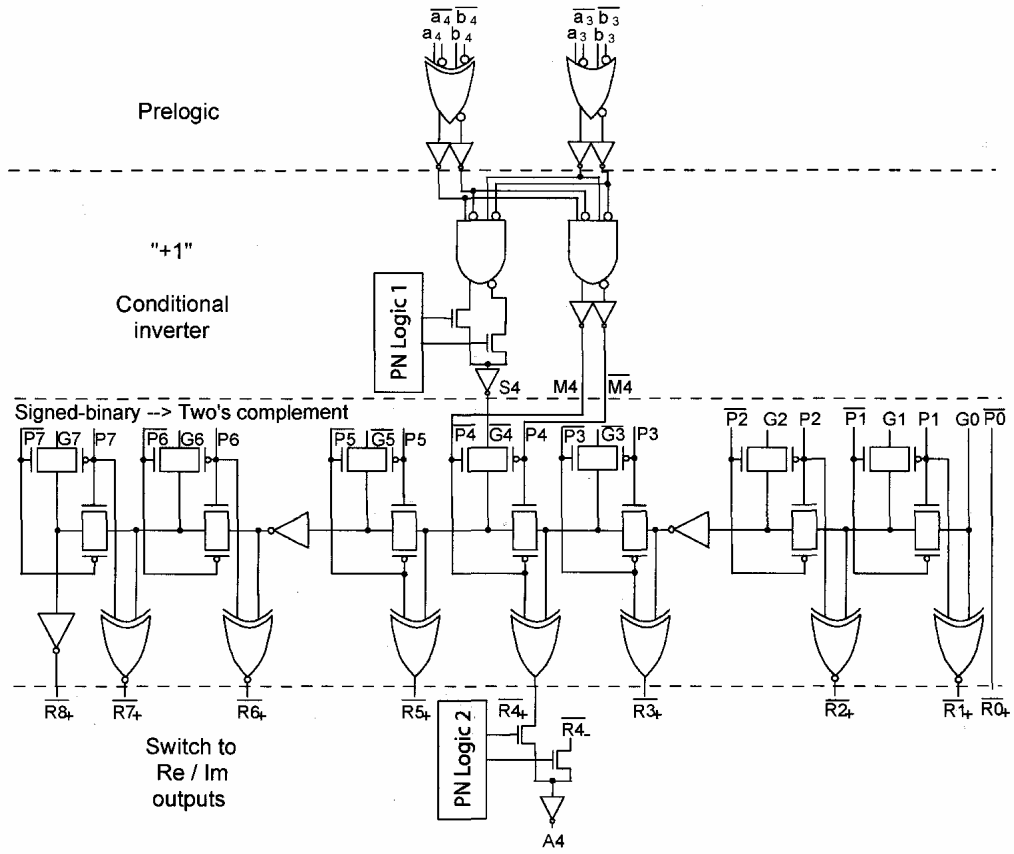


Figure 5. An 8-bit implementation of the left branch of the SBNR architecture as shown in Fig. 4.

number of inverters is required along the propagation chain.

A logic level circuit of an 8-bit implementation of the left branch of the proposed SBNR architecture is shown in Fig. 5. The critical path delay includes a carry propagation through $N-1$ transmission gates and one inverter [6] (note that $c_0 = 0$, so effectively G_0 is propagated). The carry propagation speed is significantly increased through circuit level optimization of the transmission gate chain. Inserting two additional inverters in a chain of seven transmission gates decreases the propagation time by approximately three times. In this case, either the carry signal or the inverse carry signal is propagated.

5. Simulation Results

To demonstrate the performance characteristics of the proposed architecture, an 8-bit circuit is analyzed in a TSMC 0.25 μm CMOS technology with $V_{DD} = 2.5$ volts. Delay and area estimates are presented in Table 4 along with a comparison of the alternative architectures under similar technology and bias conditions. The architectures shown in Fig. 3 have similar area-delay characteristics and, for approximately the same area, the delay of the critical path is 50% higher in these conventional architectures than the proposed SBNR realization. This increased speed is due to reducing the two carry propagation chains to a single chain in the proposed architecture. Adder techniques that trade off area, delay, and/or power may be applied to the three architectures to customize the circuit according to application-specific performance requirements.

The circuit is targeted for a base station receiver where power consumption is not a primary consideration. Since in CPL most of the PMOS transistors along the critical path are eliminated, the node capacitances are significantly reduced, thereby achieving a higher operational speed and lower power consumption. The signal path in CPL is from the source to the drain of each transistor rather than from the gate to the drain. Since the gate capacitance is usually much larger than

Table 4. Area and delay of an 8-bit complex ±1 multiplier in TSMC 0.25 μm technology [15].

	Proposed SBNR architecture			Type I: Independent-branches architecture			Type II: Switched-outputs architecture		
		Delay			Delay			Delay	
Functional unit	Area (μm^2)	Gates	(ps)	Area (μm^2)	Gates	(ps)	Area (μm^2)	Gates	(ps)
Prelogic	1500	1	200	1000	1	200	–	–	–
"+1" and inverter	1900	1 (+)	250	–	–	–	–	–	–
Converter or adder	3000	N	1350	3000	N	1350	3000	N	1350
Two's complement	–	–	–	3000	N	1350	3000	N	1350
Switch	900	1	200	–	–	–	900	1	200
PN logic	300	–	–	600	–	–	300	–	–
Total	7600	$N+3$	2000	7600	$2N+1$	2900	7200	$2N+1$	2900

'–' correspond to functional units, which are either not applicable to the specific architecture or are not along the critical path.

the junction capacitance, the delay is further reduced. Additional speed improvement is achieved by tapering the transmission gate carry propagation chain performing the SB→TC conversion as described in [14].

6. Conclusions

An efficient architecture of a complex ±1 multiplier circuit is proposed in this paper for insertion into the scrambler/descrambler portion of a wireless CDMA detector. Redundant signed-binary arithmetic is used to achieve a significant reduction in the critical path delay. A comparison of these results with standard architectures is provided. An analytical treatment of number representations for efficient VLSI arithmetic circuits is presented. It is shown that a variety of arithmetic functions, $(a+b)$, $-(a+b)$, $(a-b)$, and $-(a-b)$, may be realized for any two's-complement number, saving significant resources. A speed increase of more than 30% is observed in the proposed SBNR architecture as compared to conventional architectures.

Appendix A: Sign Bit Relations

The summation of two N-bit numbers is generally an $(N+1)$-bit number. However, the results presented in Section 3.2 are valid only if the input operands are N-bit positive numbers or if the output is N-bit with no overflow. Special care is required to set the Nth sign bit in order to achieve correct results for all input signs conditions. An alternative approach is to limit the input operands such that the output is constrained within the range of an N-bit TC number. In order to resolve this issue, the two's-complement addition of two N-bit numbers $a+b$ is presented as

a		a_{N-1}	a_{N-2}	...	a_2	a_1	a_0
	+						
b		b_{N-1}	b_{N-2}	...	b_2	b_1	b_0
Partial positive sum of bits $0:(N-2)$		c_{N-1}	c_{N-2}	...	c_2	c_1	c_0
Actual summation result	r_N	r_{N-1}	r_{N-2}	...	r_2	r_1	r_0

The sum of all positive bits $(N-2):0$ is denoted by the N-bit number c and the final addition by the $(N+1)$-bit number r. The lower $N-1$ bits of c and r are equal such that only the two most significant bits require specific attention. The input sign bits a_{N-1} and b_{N-1} are both weighted by -2^{N-1}, while the output sign bit r_N has a weight of -2^N. The relations between all sign bits are listed in Table 5, where the following expressions are considered,

$$\begin{aligned} r_{N-1} &= a_{N-1} \oplus b_{N-1} \oplus c_{N-1} \\ &= P_{N-1} \oplus c_{N-1}, \end{aligned} \tag{A.1}$$

$$\begin{aligned} r_N &= a_{N-1}b_{N-1} + (a_{N-1} \oplus b_{N-1})\overline{c_{N-1}} \\ &= G_{N-1} + P_{N-1}\overline{c_{N-1}}. \end{aligned} \tag{A.2}$$

G_i and P_i are the carry generate and carry propagate inputs of the CLA, which performs the SB → TC conversion. Note that the N-1st bit is the same bit

Table 5. Sign bit relations.

Inputs			Result		
a_{N-1}	b_{N-1}	c_{N-1}	r_N	r_{N-1}	Value
0	0	0	0	0	0
0	0	1	0	1	$+2^{N-1}$
0	1	0	1	1	-2^{N-1}
0	1	1	0	0	0
1	0	0	1	1	-2^{N-1}
1	0	1	0	0	0
1	1	0	1	0	-2^N
1	1	1	1	1	-2^{N-1}

Table 6. Comparison of transformations.

Arithmetic function	T^0: $y_i = 1 - x_i$	T^1: $y_i = 1 + x_i$
$a+b$	$2^N + \overline{T_x(x)}$	$2^N + \overline{-T_x(x)}$
	$2^N - (1 + T_x(x))$	$2^N - (1 - T_x(x))$
$-(a+b)$	$-2^N + 1 + T_x(x)$	$-2^N + 1 - T_x(x)$
$a-b$	$2^N - T_x(x)$	$2^N + T_x(x)$
$-(a-b)$	$-2^N + T_x(x)$	$-2^N - T_x(x)$

as computed by the carry-lookahead adder. Only the Nth bit is changed and is similar to the carry c_N, with c_{N-1} inverted to account for the negative weight of the sign bits [11, 12]. This function is achieved inside the adder by inverting the propagated carry c_{N-1} when c_N is computed. This result is correct for all four functions. Since the sign bit is controlled by (A.1) and (A.2), computing the Nth digit from (10) is unnecessary (only r_N and r_{N+1} are affected); therefore, only an N-digit SB number is required.

Appendix B: Alternative Transformations

Alternative relations between the sets $S_y(y)$ and $S_x(x)$ are discussed in this section. The transformation $\mathrm{tr}_{xy} = 1 - z_i$ is denoted as T^0 in the following discussion. Rather than a strict requirement for duality (the transform to be self-inverting), only a one-to-one mapping condition is imposed. One such transformation is

$$T^1 : y_i = x_i + 1 : -1 \to 0; 0 \to 1; 1 \to 2. \tag{B.1}$$

In this case, the two's-complement of the *initial sum* y is

$$\begin{aligned} a + b = T_y(y) = T_y[\mathrm{tr}_{xy}(x)] &= \sum_{i=0}^{N-1} \mathrm{tr}_{xy}(x_i) 2^i \\ &= \sum_{i=0}^{N-1} (1 + x_i) 2^i = 2^N - 1 + T_x(x) \\ &= 2^N + \overline{-T_x(x)}. \end{aligned} \tag{B.2}$$

As in the previous transformation, relations for the other functions are derived based on T^1 and are listed in Table 6.

$$-(a+b) = -2^N + 1 - T_x(x), \tag{B.3}$$

$$\begin{aligned} (a-b) &= T_y[y] + 1 = T_y\left[T^1_{xy}(x)\right] + 1 \\ &= 1 + \sum_{i=0}^{N-1} \left[T^1_{xy}(x_i)\right] \cdot 2^i \\ &= 1 + \sum_{i=0}^{N-1} [(1 + x_i)] 2^i = 1 + 2^N - 1 + T_x(x) \\ &= 2^N + T_x(x), \end{aligned} \tag{B.4}$$

$$-(a-b) = -2^N - T_x(x). \tag{B.5}$$

Note that either $(a-b)$ or $-(a-b)$ can be achieved with the same adder circuit, changing only the preprocessing step which maps a_i and b_i onto the x_i digits. The results are the same up to the Nth bit (which is the sign bit). The sign bit is controlled separately or ignored if overflow precautions are applied.

The remaining four of the six possible mappings from S_y to S_x are listed in Table 7. The relationships between S_y and the SB sets S_{x1} and S_{x2} provide a

Table 7. All possible mappings of intermediate results.

Input bits	Sum	$x_i = 1 - y_i$	$x_i = 1 + y_i$	$x_{1i} = x_{3i} + x_{4i}$		$x_{2i} = x_{5i} + x_{6i}$	
	S_y	S_{x1}	S_{x2}	S_{x3}	S_{x4}	S_{x5}	S_{x6}
00	0	1	$\bar{1}$	1	0	$\bar{1}$	0
01, 10	1	0	0	$\bar{1}$	1	1	$\bar{1}$
11	2	$\bar{1}$	1	0	$\bar{1}$	0	1

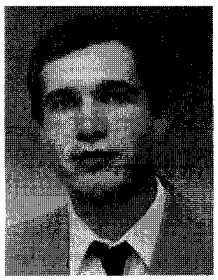
means for fast digit-processing and efficient computation. Alternatively, transforms 3 to 6 require a more elaborate transformation to/from S_y and can be useful for decomposing an initial sum number from S_y into two signed-binary numbers. Both transformations, T^0 and T^1, produce all of the functions listed in Table 6. These transformations may also be used interchangeably to switch between two functions by only changing the prelogic mapping stage.

Acknowledgments

This research is supported in part by the Semiconductor Research Corporation under Contract No. 99-TJ-687, the DARPA/ITO under AFRL Contract F29601-00-K-0182, grants from the New York State Office of Science, Technology & Academic Research to the Center for Advanced Technology—Electronic Imaging Systems and to the Microelectronics Design Center, and by grants from Xerox Corporation, IBM Corporation, Intel Corporation, Lucent Technologies Corporation, and Eastman Kodak Company.

References

1. 3G TS 25.213 V3.4.0 (2000-12) 3rd Generation Partnership Project; TSG Radio Access Network; Spreading and Modulation (FDD), Release 1999.
2. R. Cameron, *Fixed-point Implementation of a Multistage Receiver*, Ph.D. Thesis, Virginia Polytechnic Institute and State University, January 1997.
3. N. Zhang et al., "Architectural Implementation Issues in a Wideband Receiver Using Multiuser Detection," in *Proceedings of the Annual Allerton Conference on Communication, Control and Computing*, Sept. 1998, pp. 765–771,
4. G.M. Blair, "The Equivalence of Twos-Complement Addition and the Conversion of Redundant-Binary to Twos-Complement Numbers," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, no. 6, 1998, pp. 669–671.
5. J. Dobson and G.M. Blair, "Fast Two's Complement VLSI Adder Design," *Electronics Letters*, vol. 31, no. 20, 1995, pp. 1721–1722.
6. H. Srinivas and K. Parhi, "A Fast VLSI Adder Architecture," *IEEE Journal on Solid-State Circuits*, vol. 27, no. 5, 1992, pp. 761–767.
7. T. Kim and J. Um, "A Practical Approach to the Synthesis of Arithmetic Circuits Using Carry-save Adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 5, 2000, pp. 615–624.
8. M. Soderstrand, W. Jenkins, G. Jullien, and F. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, 1986.
9. V. Paliouras and T. Stouraitis, "Logarithmic Number System for Low-Power Arithmetic," in *Proceedings of the International Workshop on Integrated Circuit Design Power and Timing Modeling, Optimization and Simulation*, Sept. 2000, pp. 285–290.
10. B.D. Andreev, E.G. Friedman, and E.L. Titlebaum, "Efficient Implementation of a Complex ±1 Multiplier," in *Proceedings of the ACM Great Lakes Symposium on VLSI*, April 2002, pp. 83–88.
11. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd Edition, Addison-Wesley, 1993.
12. R. Brent and H. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers*, vol. C-31, no. 3, 1982, pp. 260–264.
13. K. Yano et al., "A 3.8 Ns CMOS 16 × 16-b Multiplier Using Complementary Pass-Transistor Logic," *IEEE Journal on Solid-State Circuits*, vol. 25, no. 2, 1990, pp. 388–395.
14. B.D. Andreev, E. Titlebaum and E.G. Friedman, "Tapered Transmission Gate Chains for Improved Carry Propagation," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, vol. 3, Aug. 2002, pp. 449–452.
15. TSMC 0.25 μm Logic 1P5M Salicide 2.5V/3.3V technology, Documents TA-1099-4003 and TA-1099-6001, Taiwan Semiconductor Manufacturing Co., Ltd.



Boris D. Andreev received a M.Sc. degree in Electrical Engineering from the Technical University of Varna, Bulgaria, in 1998. He is currently a Ph.D. candidate and research assistant at the High Performance VLSI/IC Design and Analysis Laboratory at the University of Rochester.

His professional interests are primarily in efficient implementation of VLSI signal processing systems at the algorithmic, architectural, and circuit design levels. Much of this work is related to the optimization of cellular CDMA receivers and communication systems. The effects of physical limitations of deep submicron technologies on system level design are currently being investigated. Since August 2000, Boris Andreev has also been working with the Digital Technology Center at Eastman Kodak Co. on substrate coupling noise issues in mixed-signal image processing integrated circuits. During the summer of 2002, he held an internship position at the Central Research and Development division of ST Microelectronics Inc. developing a novel methodology for ASIC synthesis based on dynamic logic circuits.

Boris Andreev was awarded the 1998 Best Graduate of the Year award of the Bulgarian Federation of Science and Technology Unions. He is a University of Rochester Sproull Fellow and held a graduate research fellowship from Lucent Corporation.

Edward L. Titlebaum was born in Boston Mass. on March 23, 1937. He was educated in the public schools of Boston. He received a B.S.E.E. degree in 1959 from Northeastern University, Boston, Mass., an M.S. degree in 1962 and a Ph.D. degree in 1964 both in electrical engineering from Cornell University, Ithaca, New York.

He has been in the Department of Electrical Engineering at the University of Rochester from 1964 to the present, currently holding the position of Professor of Electrical Engineering. On July 1, 1996 he was appointed Vice Provost for Computing in charge of Telecommunications. His research interests lie in the general areas of radar, sonar and communications signal design, natural echolocation systems including those used by bats and whales. Currently he is studying frequency hop codes based upon congruential coding methods for use in multiple access spread spectrum communications and multiuser radar and sonar systems. He is investigating the role of these signals in CDMA spread spectrum communications systems, computer networks, and is a founding member of the Music Research Laboratory at the University of Rochester.

Eby G. Friedman received the B.S. degree from Lafayette College in 1979, and the M.S. and Ph.D. degrees from the University of California, Irvine, in 1981 and 1989, respectively, all in electrical engineering.

From 1979 to 1991, he was with Hughes Aircraft Company, rising to the position of manager of the Signal Processing Design and Test Department, responsible for the design and test of high performance digital and analog IC's. He has been with the Department of Electrical and Computer Engineering at the University of Rochester since 1991, where he is a Distinguished Professor, the Director of the High Performance VLSI/IC Design and Analysis Laboratory, and the Director of the Center for Electronic Imaging Systems. He also enjoyed a sabbatical at the Technion—Israel Institute of Technology during the 2000/2001 academic year. His current research and teaching interests are in high performance synchronous digital and mixed-signal microelectronic design and analysis with application to high speed portable processors and low power wireless communications.

He is the author of about 250 papers and book chapters, several patents, and the author or editor of seven books in the fields of high speed and low power CMOS design techniques, high speed interconnect, and the theory and application of synchronous clock distribution networks. Dr. Friedman is the Chair of the *IEEE Transactions on VLSI Systems* steering committee, Regional Editor of the *Journal of Circuits, Systems, and Computers*, a Member of the editorial boards of the *Proceedings of the IEEE, Analog Integrated Circuits and Signal Processing*, *Journal of VLSI Signal Processing*, and *Microelectronics Journal*, a Member of the Circuits and Systems (CAS) Society Board of Governors, and a Member of the technical program committee of a number of conferences. He previously was the Editor-in-Chief of the *IEEE Transactions on VLSI Systems*, a Member of the editorial board of the *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, CAS liaison to the Solid-State Circuits Society, Chair of the VLSI Systems and Applications CAS Technical Committee, Chair of the Electron Devices Chapter of the IEEE Rochester Section, Program or Technical chair of several IEEE conferences, Guest Editor of several special issues in a variety of journals, and a recipient of the Howard Hughes Masters and Doctoral Fellowships, an IBM University Research Award, an Outstanding IEEE Chapter Chairman Award, and a University of Rochester College of Engineering Teaching Excellence Award. Dr. Friedman is a Senior Fulbright Fellow and an IEEE Fellow.