# A Carry-Free 54b × 54b Multiplier Using Equivalent Bit Conversion Algorithm

Yun Kim, *Member, IEEE*, Bang-Sup Song, *Fellow, IEEE*, John Grosspietsch, *Member, IEEE*, and Steven F. Gillig, *Member, IEEE*

***Abstract*—An equivalent bit conversion algorithm (EBCA) is proposed to eliminate the need for final carry propagation in the redundant binary (RB) to normal binary (NB) conversion step for RB multiplication. The multiplication process helps with the carry-free conversion step by eliminating certain combinations of RB product. When the EBCA is applied, conventional power-consuming carry-propagating adders are replaced by simple, minimum-sized carry-free converters, and the entire multiplication process can be made free of carry propagation from input to output. The method employed in this work reduces 40% of the total power and 30% of the total multiplication time in the final adder stage of traditional multipliers. The prototype fabricated in 0.35-$\mu$m CMOS demonstrates that the 54b × 54b multiplier consumes only 53.4 mW at 3.3 V for 74-MHz operation.**

***Index Terms*—Adders, algorithms, encoding, multiplication, redundant number systems.**

## I. Introduction

FOR MOBILE digital signal processing, broadband communications, and computer applications, conventional high-speed digital multiplier architectures are no longer capable of providing the high-speed computational needs at low-power requirements of the portable devices. In the last decade, most improvements and speed gains in multipliers have been achieved only by using extreme circuit optimization and advanced fabrication technology. However, the system architecture of the state-of-the-art multipliers has changed very little. The current predominant multiplier architecture uses $4-2$ compressors in a binary tree for high-speed parallel computing [1]. While this method is useful in eliminating the intermediate carry propagation, it still suffers from the possible carry propagation from the least significant bit (LSB) to the most significant bit (MSB) in the final adder stage, where a fast, complicated, and power-consuming adder is needed.

One alternative approach to multiplication is to use the redundant binary (RB) system, which can be configured to add any RB numbers free of carry propagation. The only drawback to the RB system is that often the final product is needed in normal binary (NB) number form, requiring a conversion step in the final stage of the multiplier. To date, the same slow carry-propagating adders used in the final stage of the state-of-the-art multipliers are employed for the RB-to-NB conversion. This paper describes a method to convert the RB product of the proposed multiplier to the NB product using the equivalent bit conversion algorithm (EBCA), which eliminates the need for carry propagation in the final conversion stage by taking maximum advantage of the RB multiplication process. Incorporated with the radix-4 Booth recoding technique, the proposed multiplier architecture exhibits greatly reduced power consumption. The method exploited in this work reduces 40% of the total power and 30% of the total conversion time consumed in the final adder stage of a traditional multiplier architecture.

A brief introduction to the radix-4 Booth recoding and RB numbers is given in Section II. Section III describes the EBCA as well as the error-correction method, which removes the errors intentionally introduced in the RB coding for speed enhancement while aiding in elimination of carry-propagating patterns of the RB final product. Detailed implementation and fabrication issues are covered in Section IV, followed by experimental results in Section V.

## II. Radix-4 Booth Recoding and RB Numbers

The block diagram of the proposed 54b × 54b multiplier is shown in Fig. 1. The architecture uses the radix-4 Booth recoding to reduce the total number of partial products by half. The NB input operands are then converted to RB operands for carry-free intermediate summation of the partial products. In a normal 54b × 54b linear-array multiplier architecture, 54 partial products are added to produce the final product. However, the radix-4 Booth recoding reduces this number by half, and the RB coding reduces it further by half again. Therefore, the total number of partial products in the prototype multiplier is 13 plus one error-correcting word. The addition of these partial products can be performed with $\lceil \log_2 14 \rceil$ or four adder delays using a parallel binary tree of the adder cells. After four stages of RB adders, the final RB product is converted back to the NB product using the proposed EBCA. Note that the total architectural latency of the proposed multiplier is the same as that of the standard architecture including this last conversion step.

The radix-4 Booth recoding reduces the partial products by half by placing a single partial product for every consecutive pair of bits in the multiplier. The idea of radix-4 Booth recoding is to select the partial products from the set $\{-2M, -M, 0, M, 2M\}$, where $M$ stands for the multiplicand, and $2M$ stands for 2 times the multiplicand. The advantage is that $-2M$, $-M$, 0, $M$, and $2M$ are multiples of the multiplicand which can be

Manuscript received December 10, 1999; revised June 27, 2001.

Y. Kim was with Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801 USA. He is now with Motorola Inc., Schaumburg, IL 60196 USA.

B.-S. Song was with Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801 USA. He is now with Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA.

J. Grosspietsch and S. F. Gillig are with Motorola Inc., Schaumburg, IL 60196 USA.

Publisher Item Identifier S 0018-9200(01)08426-8.

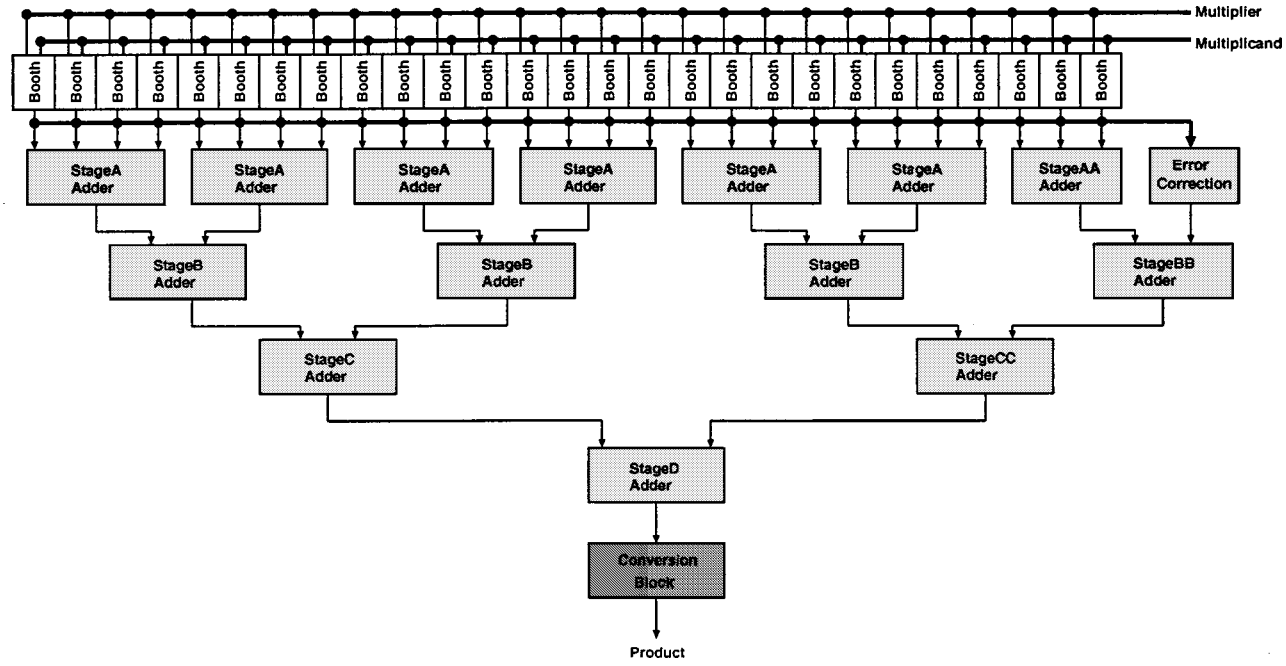


Fig. 1. Block diagram of the 54b × 54b RB multiplier.

easily generated by shifting and/or inverting the multiplicand bits. The complete radix-4 Booth algorithm is summarized in Table I. The multiplier is partitioned into overlapping groups of three adjacent bits, and each group is decoded to select a single partial product as indicated by the selection table. Each partial product is shifted by 2-bit positions with respect to its neighboring ones for the correct binary weight of the partial products. The radix-4 Booth recoding eliminates 13 of 54-bit adders and one adder delay in the total conversion time.

The other idea used in the prototype multiplier for speed enhancement is the use of RB adders for carry-free summation of the partial products. An RB number is a part of a more generalized set of numbers known as signed digit number representation [2]. An RB number consists of digits from the set $\{-1, 0, 1\}$. To represent the $-1_{\mathrm{RB}}$ digit, it will be notated as $\overline{1}$ throughout this paper. The reason that RB numbers are of great interest is their ability to represent a single number in many different ways. For example, the following are five different, yet equivalent, ways to represent the decimal value 5 as a 4-bit RB word.

$$[0101]_{\mathrm{RB}} = 4 + 1 = 5$$
$$[011\overline{1}]_{\mathrm{RB}} = 4 + 2 - 1 = 5$$
$$[1\overline{1}1\overline{1}]_{\mathrm{RB}} = 8 - 4 + 2 - 1 = 5$$
$$[1\overline{1}01]_{\mathrm{RB}} = 8 - 4 + 1 = 5$$
$$[10\overline{11}]_{\mathrm{RB}} = 8 - 2 - 1 = 5$$

When using the RB digit set, two bits are required to represent the three distinct digits in the RB domain. Although there are a few possible coding schemes to map three RB digits to four states represented by two binary digits, the one chosen for this work is shown in Table II. Symbolically, $\{x^-, x^+\}$ notation is used to represent the two digits of an RB digit. With the RB coding as described in Table II, the negation of the RB digit can be achieved simply by inverting the $x^-$ and $x^+$ bits individually. Furthermore, interchanging the $x^-$ and $x^+$ bits will have no effect on the actual RB digit that $\{x^-, x^+\}$ represents.

TABLE I
BOOTH'S PARTIAL PRODUCT SELECTION TABLE

| Multiplier Bits | Selection |
|---|---|
| 000 | 0 |
| 001 | + Multiplicand |
| 010 | + Multiplicand |
| 011 | + 2 x Multiplicand |
| 100 | − 2 x Multiplicand |
| 101 | − Multiplicand |
| 110 | − Multiplicand |
| 111 | 0 |

TABLE II
RB CODING USED IN THIS WORK

| $x^-$ | $x^+$ | RB digit |
|---|---|---|
| 0 | 0 | $\overline{1}$ |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

The real advantage of using the RB numbers is that a carry-free addition rule can be used by exploiting the flexibility in representing any numbers. The main goal of formulating a carry-free addition rule is to guarantee that the carry-out of an RB full adder (RBFA) is made independent of the carry-in. In other words, a parallel array of $N$ adders can add two $N$-bit words without carry propagation. A carry-free addition rule for an RBFA is shown in Table III. In the table, $x_k$ and $y_k$ notate the $k$th digits of the two input words to add. Similarly, $x_{k-1}$ and $y_{k-1}$ notate the $(k-1)$th digits of the same two words. The symbol $c_k$ notates the carry-out of the $k$th bit RBFA, and the symbol $s_k$ notates the intermediate sum of the $k$th bit obtained without relying on the incoming carry $c_{k-1}$. The final sum of

the $k$th bit is obtained by adding the intermediate sum $s_k$ and $c_{k-1}$.

Table III lists all nine possible combinations of different pairs of $x_k$ and $y_k$ and the resulting $c_k$ and $s_k$. There are three trivial cases where the resulting $c_k$ and $s_k$ values are predetermined by values of $x_k$ and $y_k$ regardless of $x_{k-1}$ and $y_{k-1}$ values. The first such case is when $x_k$ and $y_k$ are both 1. Regardless of the previous operands $x_{k-1}$ and $y_{k-1}$, the RBFA should generate a carry-out of 1 and an intermediate sum of 0 to be mathematically correct. The second case is when both $x_k$ and $y_k$ are $\overline{1}$. Once again, regardless of the previous operands, the RBFA should generate a carry-out of $\overline{1}$ and an intermediate sum of 0. The third case is when $x_k$ and $y_k$ add up to 0, such as in pairs $[1,\overline{1}]$, $[\overline{1},1]$, and $[0,0]$. In such cases, both the carry-out and the intermediate sum can be set to 0 immediately.

The remaining combinations are more difficult to analyze. When $x_k$ and $y_k$ add up to 1, such as in the case of $[1,0]$ and $[0,1]$, the previous bit operands are needed to decide which intermediate sum should be generated. If both of the previous operands are nonnegative (0 or 1), then the expected carry-in is either 0 or 1. In this case, the RBFA addition rule intentionally leaves an intermediate sum of $\overline{1}$ to eliminate the propagation of the possible incoming carry of 1. Therefore, a carry-out of 1 is generated to compensate for the required sum of 1. However, if any of the previous operands are negative, then a carry-in of either 0 or $\overline{1}$ is expected from the previous operands. In that case, the intermediate sum is intentionally set to 1 to stop the propagation of the possible incoming carry of $\overline{1}$, and the carry-out is set to 0, preserving the necessary sum of 1. Similarly, when $x_k$ and $y_k$ add up to $\overline{1}$ as in the cases of $[\overline{1},0]$ and $[0,\overline{1}]$, additional information is needed to determine the intermediate sum. If the previous operands are both nonnegative, then to stop the propagation of expected incoming carry of 0 or 1, the rule forces the intermediate sum to be $\overline{1}$. However, if the operands contain a negative digit, an intermediate sum of 1 and a carry-out of $\overline{1}$ are generated. Note that both cases produce the correct intermediate sum of $\overline{1}$ when the carry-out is taken into account.

The resulting logic equations for the RBFA which follows the carry-free addition rule of Table III are as follows.

$$
\begin{aligned}
g_k &= \left(x_k^- \oplus x_k^+\right) \oplus \left(y_k^- \oplus y_k^+\right) \\
h_k &= x_k^- x_k^+ + y_k^- y_k^+ \\
c_k^- &= \left(x_k^- + x_k^+\right)\left(y_k^- + y_k^+\right) \\
c_k^+ &= g_k c_{k-1}^- + \overline{g_k} h_k \\
s_k^- &= g_k \oplus c_{k-1}^- \\
s_k^+ &= c_{k-1}^+
\end{aligned}
$$

The above equations guarantee that the chain of carry propagation is limited to one adder. Using the logic equations presented, any two RB numbers can be added in a constant time, regardless of the word length.

## III. EBCA and Error Correction

Although the multiplication in the RB domain is promising, the challenging task is the conversion of the NB inputs to RB and back to NB for the final output. Fortunately, the NB-to-RB conversion can be accomplished without using extra hardware or causing extra delay in the multiplication process. For the RB coding scheme chosen for this work, the corresponding code to its RB digit representation can be summarized as, $x^- - \overline{x^+} = F$, where $\{x^-, x^+\}$ is the two digit representation of the RB digit $F$. This coding scheme can be used to add two NB numbers and to convert the sum of the NB numbers to an RB number in one simple step. When adding two NB words $A$ and $B$, this is equivalent to $A-(-B)$, which is equivalent to $A-(\overline{B}+1)$, which is then equivalent to $A-\overline{B}-1$. Since $\{A,B\}$ represents the RB number $A-\overline{B}$, this is equivalent to $A+B+1$. Therefore, simply by pairing up the NB operands $A$ and $B$ to $\{A,B\}$, one can convert the NB operands to an RB representation of the sum $A+B+1$ without any extra hardware. The extra addition of $-1$ needed for the correct sum of $A+B$ is taken care of in the error-correction stage. A final error-correcting word includes all these errors generated during the multiplication process. Therefore, the minimal NB-to-RB conversion process reduces the number of partial products by half by simultaneously performing an addition of the NB operands during the RB coding procedure. It should be noted, however, that RB coding of the MSB digit is slightly different from the coding shown in Table II. Because the input words are in two's complement form, the MSB behaves as a sign bit. MSB is 0 for positive words and 1 for negative words. To compensate for this effect, the equivalent RB for the sum of the MSBs is formed by inverting the MSBs of NB operands $A$ and $B$ first, before the individual bits are joined to form the RB sum. The inversion of the MSBs conserves the correct sum of the NB words in the newly formed RB.

TABLE III
Carry-Free Addition Rule for RBFA

| $x_k$ | $y_k$ | $x_{k-1}y_{k-1}$ | $c_k$ | $s_k$ |
|---|---|---|---|---|
| 1 | 1 | don't care | 1 | 0 |
| 1 | 0 | both are non-neg | 1 | $\overline{1}$ |
| 0 | 1 | otherwise | 0 | 1 |
| $\overline{1}$ | $\overline{1}$ | don't care | 0 | 0 |
| $\overline{1}$ | 1 | | 0 | 0 |
| 0 | 0 | | 0 | 0 |
| 0 | $\overline{1}$ | both are non-neg | 0 | $\overline{1}$ |
| $\overline{1}$ | 0 | otherwise | $\overline{1}$ | 1 |
| $\overline{1}$ | $\overline{1}$ | don't care | $\overline{1}$ | 0 |

In an effort to reduce the number of partial products by a factor of 4, radix-4 Booth recoding and RB coding are employed in the proposed multiplier. Unfortunately, there are a few instances when the coding techniques above produce erroneous results. However, these errors can be removed in a single error-correction step without increasing the overall latency of the multiplier. Fig. 2 illustrates the error-correction scheme and the proper bit alignment of the error-correcting word. One source of error is the extra $-1$ term needed at the LSB of every partial product from the RB coding, as mentioned previously. The other source of error is from the radix-4 Booth recoding. There are three cases when the radix-4 Booth recoding dictates that the multiplicand be multiplied by $-1$ or $-2$. Because negating an NB number requires a carry-propagating addition, only the inverse of the multiplicand is used. To compensate for this error, $+1$ needs to be added to the LSB of the partial product which is coded with $-1$ or $-2$ Booth multiplier. Combining the two

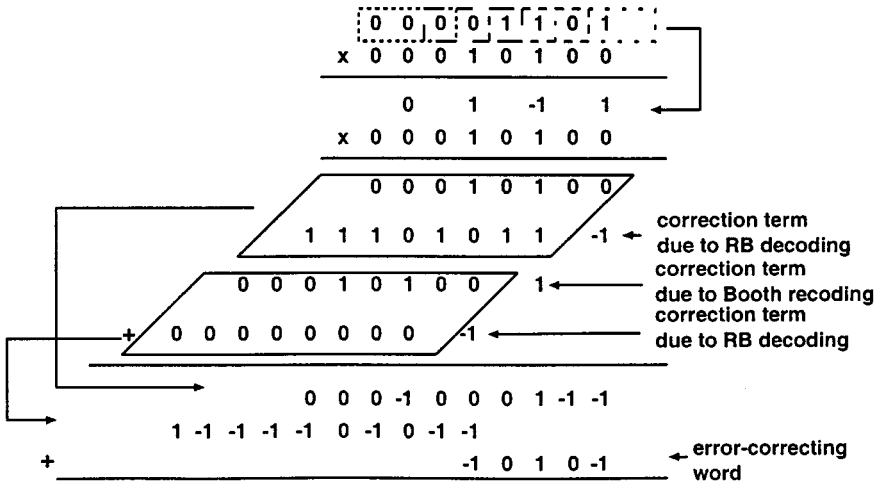


Fig. 2. Illustration of the error-correction procedure.

types of errors together, a single error-correcting word can compensate for all occurrences of erroneous results. The error-correcting word has the form of $\cdots 0\, X\, 0\, Y\, 0\, X\, 0\, Y \cdots$, where $X \in \{0, 1\}$ and $Y \in \{0, \overline{1}\}$. In $Y$ digits, $-1$ correction factor is always present from the RB coding. If $Y$ digit is also corrected for Booth recoding, then the error corrections cancel out to 0. Therefore, $Y$ digits are limited to 0 or $\overline{1}$, 0 when the error corrections nullify one another, $\overline{1}$ when only RB correction is present. For $X$ digits, there are no RB corrections at these digit locations. Thus, $X$ is determined solely by the Booth recoding correction, 0 when there are no corrections and 1 when Booth recoding correction is required. The inclusion of the error-correcting word to the partial product addition does not increase the number of adder stages needed in the case of 54 partial products. However, if the number of partial products were perfect powers of 2, then the error-correction word would require one additional stage of adders to complete the sum.

Unlike the NB-to-RB conversion process, the RB-to-NB conversion process is more complicated. In traditional RB multiplier architectures, this conversion process limits the speed of the multiplication process because a carry-propagating adder is used for the final conversion. While some efforts have been published in attempts to find a better RB-to-NB conversion method, these works speed up the already established conversion technique by employing a fast carry-rippling process of the NB adder in the RB domain [3], [4]. The EBCA proposed here is fundamentally different from any earlier works. It exploits the fact that the NB numbers are a subset of the RB numbers and attempts to find a suitable representation of the final RB product, which is also a valid representation in the NB domain. A logical solution to the problem described is to eliminate the RB digit $\overline{1}$, which is absent from the NB digit set. With the exception of the MSB, the RB digit $\overline{1}$ can be eliminated from other digits. The MSB is special because an RB digit of $\overline{1}$ is required to notate a negative number, which in NB form is notated with 1 in the MSB. The simple EBCA proposed systematically eliminates the $\overline{1}$'s from any RB representation of a number, except for the MSB. It eliminates the $\overline{1}$'s from the RB representation by replacing a block of RB digits by an equivalent RB representation block. Blocks of an RB word which contain $0\,0\,\cdots\,0\,\overline{1}$ are replaced by an equivalent form of $\overline{1}\,1\,\cdots\,1\,1$. Similarly, blocks of $1\,\overline{1}\,\cdots\,\overline{1}\,\overline{1}$ are replaced by an equivalent RB form of $0\,0\,\cdots\,0\,1$. Other than a possible $\overline{1}$ in the MSB position, the resulting RB representation after applying the two conversion steps in sequence can be directly interpreted as an NB number. If the MSB of the reformatted RB number is $\overline{1}$, notating that the overall word is negative, it is simply replaced by 1.

TABLE IV
ALL POSSIBLE RB REPRESENTATIONS OF THE NUMBER 7 IN RB

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 0 | 0 | $\overline{1}$ |
| 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 0 | $\overline{1}$ | 1 |
| 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 1 | 1 |
| 0 | 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 0 | 0 | $\overline{1}$ |
| 0 | 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 0 | $\overline{1}$ | 1 |
| 0 | 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 1 | 1 |
| 0 | 0 | 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 0 | 0 | $\overline{1}$ |
| 0 | 0 | 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 0 | $\overline{1}$ | 1 |
| 0 | 0 | 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 1 | 1 |
| 0 | 0 | 0 | 1 | $\overline{1}$ | $\overline{1}$ | 0 | 0 | $\overline{1}$ |
| 0 | 0 | 0 | 1 | $\overline{1}$ | $\overline{1}$ | 0 | $\overline{1}$ | 1 |
| 0 | 0 | 0 | 1 | $\overline{1}$ | $\overline{1}$ | $\overline{1}$ | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | $\overline{1}$ | 0 | 0 | $\overline{1}$ |
| 0 | 0 | 0 | 0 | 1 | $\overline{1}$ | 0 | $\overline{1}$ | 1 |
| 0 | 0 | 0 | 0 | 1 | $\overline{1}$ | $\overline{1}$ | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | $\overline{1}$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | $\overline{1}$ | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | $\overline{1}$ | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

Because a sequence of bits is being replaced by its equivalent counterpart using the EBCA, the overall value of the digital word does not change. For example, one possible RB representation of the number $-5$ is $0\overline{1}\overline{1}1$. After performing the first conversion step, the RB representation is $\overline{1}1\overline{1}1$. Continuing on to the second conversion step, the number becomes $\overline{1}011$. The resulting number is still $-5$, as easily verified from $-8+2+1$. The remaining question is, what is the original RB number $0\overline{1}\overline{1}1$ expressed in NB format? It is simply 1011, as predicted by the conversion steps after the MSB is replaced by 1.

The completeness of the conversion scheme proposed must be carefully studied to guarantee that all $\overline{1}$'s are removed by the two conversion steps. The first conversion step eliminates from

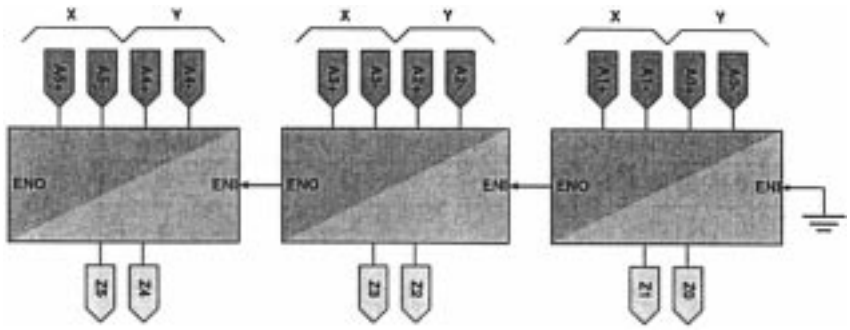


Fig. 3. Multibit converter cell.

TABLE V
TRUTH TABLE FOR CARRY-FREE RB-TO-NB CONVERSION

| | | ENI=0 | | | ENI=1 | | |
|---|---|---|---|---|---|---|---|
| X | Y | $Z_1$ | $Z_2$ | ENO | $Z_1$ | $Z_2$ | ENO |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | $\bar{1}$ | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| $\bar{1}$ | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| $\bar{1}$ | $\bar{1}$ | 0 | 1 | 1 | 0 | 0 | 1 |
| $\bar{1}$ | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | $\bar{1}$ | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

the RB word all instances of 0 being followed by $\bar{1}$. Hence, the resulting RB number is guaranteed to have all $\bar{1}$'s preceded only by 1 with possible exception of the MSB. The second conversion step eliminates all remaining $\bar{1}$'s by converting $1\,\bar{1}\cdots\bar{1}$ to $0\cdots01$. As a demonstration of the conversion process, Table IV lists all possible RB representations of the number 7 in a 9-bit RB word. Employing the two conversion steps sequentially, it is possible to convert from any representation in Table IV to the last entry on the table, which is both the NB and RB representation of the number 7.

In order to simplify the algorithm and reduce the complexity of the converter, the two-step conversion process described can be combined into a one-step multibit converter. The one-step multibit converter designed is illustrated in Fig. 3. Each box in the figure represents a conversion box of two RB digits into two NB digits. In the figure, $X$ and $Y$ are two consecutive RB digits to be converted. The ENI (for enable-in) and ENO (for enable-out) signals communicate information from previous converter block to the next converter block. However, it is important to note that ENO signal is generated independently of ENI values, and hence, the conversion box does not have any carry propagation. The $Z$ bits are the converted NB bits. The truth table of the multibit converter is shown in Table V. The carry-free nature of this conversion process can be easily confirmed by the fact that the ENO bit is independent of the ENI bit.

The error-correction block of the multiplier serves a dual purpose. One purpose is to compensate for the errors from the two speed enhancement techniques like RB coding and Booth recoding. The second purpose of the error-correction block is to eliminate the carry propagating patterns of the RB forms. As explained in the earlier section, the error-correction word has the

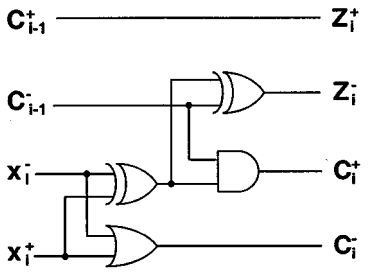


Fig. 4. Schematic diagram of an RBHA.

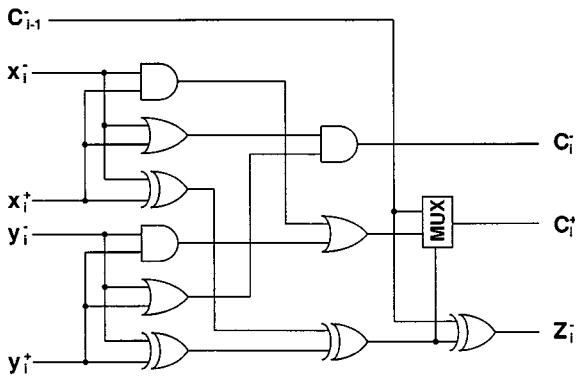


Fig. 5. Schematic diagram of an RBFA.

form of $\cdots\,0\,X\,0\,Y\,\cdots$, where $X$ and $Y$ are possible nonzero RB digits. The real benefit of the above form is that every odd digit location of the error-correcting word is guaranteed to be 0 regardless of the input operands. Furthermore, the $X$'s and $Y$'s are further constrained to $\{0,1\}$ and $\{0,\bar{1}\}$, respectively. As these 0's propagate through the redundant binary adder trees, they limit the possible RB form of the final product. It should be noted that the EBCA is dependent on the error-correction block to filter out the some of the redundant forms of the RB representation. Furthermore, the EBCA is not a general RB-to-NB converter, and depends heavily upon the multiplication architecture proposed here for the generation of correct product of the input operands.

The EBCA algorithm has been exhaustively tested in simulation for all pairs of 16-b input vector words. In every case, the algorithm produced the correct product. Due to the exponentially growing vector space, the authors have not exhaustively tested the EBCA algorithm for higher vector widths. However, 20 000 random 54-b input vectors were simulated for correctness.

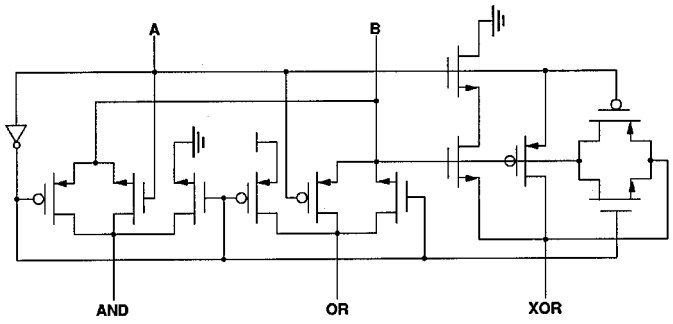


Fig. 6. Transmission gate based implementation of an AOX cell.

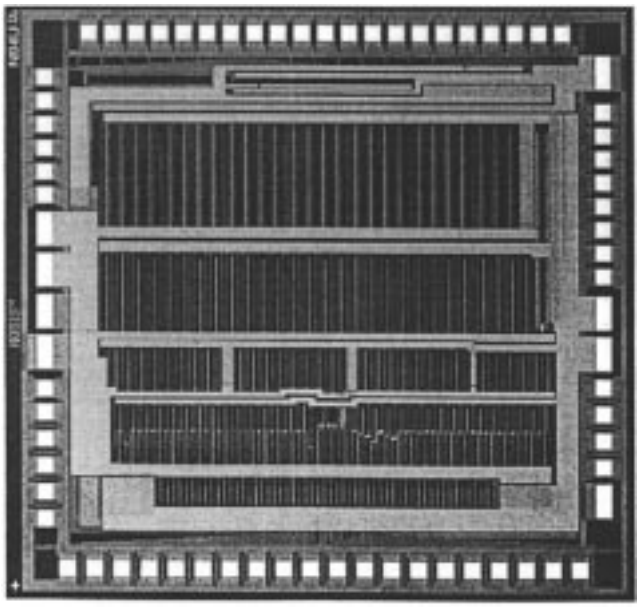

Fig. 7. Die photo of the prototype multiplier.

## IV. Logic Design and Fabrication

To demonstrate the architecture and to prove the validity of the proposed EBCA, a 54b × 54b multiplier is prototyped in 0.35-$\mu$m CMOS. To achieve the high-speed and low-power goals of the multiplier, all circuits for RBHA, RBFA, Booth recoding, and RB-to-NB conversion are designed using transmission gate logic. Figs. 4 and 5 are the gate-level diagrams of the RBHA and RBFA. The RBFA which is used repeatedly throughout the design is implemented with 54 transistors, and the RBHA is optimized to use only 20 transistors. Fig. 6 shows a more detailed transistor level diagram of the AND–OR–XOR (AOX) cell used in the RBFA.

The multibit RB-to-NB converter is implemented with 35.5 transistors per bit. As previously mentioned, the ENO signal is independent of the ENI signal. Thus, a parallel array of the RB-to-NB converters is completely free of carry propagation. Overall, there are over 86 000 transistors in the 54b × 54b multiplier design. A full custom layout of the multiplier using 0.35-$\mu$m 4-metal CMOS process occupies 3.2 mm × 3.3 mm of active area including 73 I/O pads and some testing circuitry. A chip photo of the prototype multiplier is shown in Fig. 7.

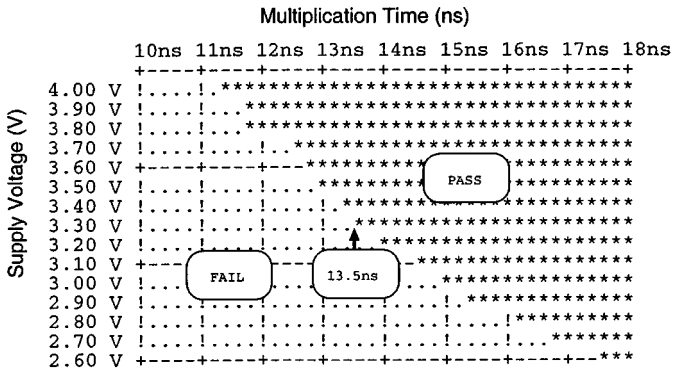


Fig. 8. Schmoo plot.

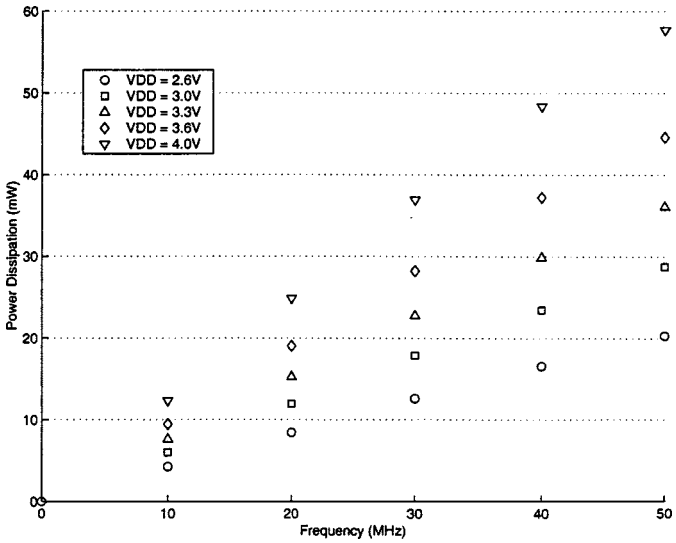


Fig. 9. Measured power dissipation.

## V. Experimental Results

The functionality of the prototype multiplier is verified by multiplexing the inputs and outputs. Two thousand random 54-b input vectors were multiplexed in via a pattern generator. The resulting product was then captured in a logic analyzer and later verified for correctness. For speed testing, an external clock and control signals are used to toggle two internal ROM vectors. The maximum operation speed of the multiplier is measured by progressively increasing the speed of the external clock until the multiplier produces an incorrect result. The testing of the prototype multiplier shows that the chip is operational at 74 MHz with 3.3-V power supply consuming 53.4 mW. Fig. 8 shows the Schmoo plot of the multiplication time versus the supply voltage. At 3.3-V power supply, the multiplier prototype can operate at 74-MHz operation speed. Fig. 9 shows the measured power for various power supply voltages and various operating frequencies.

Analysis of the chip performance shows that the prototype multiplier's speed performance is primarily limited by the internal ROM drivers providing vectors for the speed testing. Fig. 10 compares the normalized power consumption of the prototype multiplier to those of other 54b × 54b multipliers which have reported power consumption [5]–[9]. As it clearly

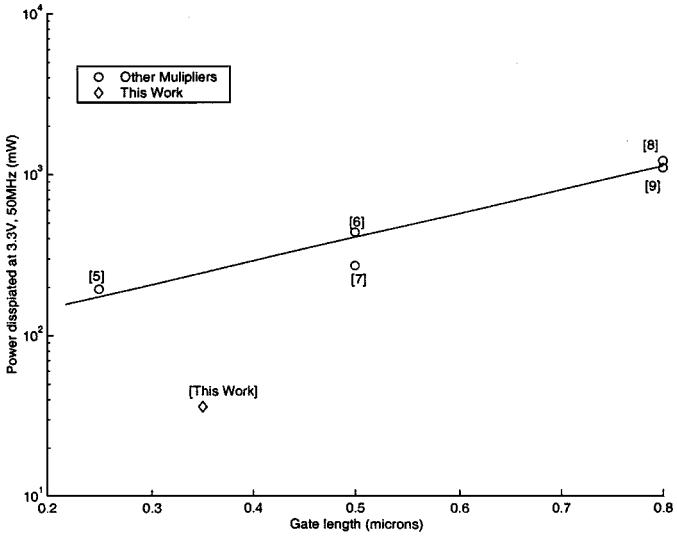


Fig. 10. Power dissipation of 54b × 54b multipliers normalized to 50 MHz at 3.3 V.

TABLE VI
SUMMARY OF THE MEASURED PERFORMANCE

| | |
|---|---|
| Technology | 0.35$\mu$m 4-metal CMOS |
| Transistor Count | 86,000+ |
| Area | 3.2 x 3.3 mm$^2$ |
| Latency | 1 clock |
| Operating Freq. | 74MHz @ 3.3V |
| Power | 53.4mW @ 3.3V, 74-MHz |
| I/O Pins | 73 |

demonstrates, the power consumption of the prototype multiplier is the lowest due to the carry-free RB-to-NB conversion stage, which does not require large devices or other power-consuming techniques to speedily propagate the carry information. A summary of the chip performance is given in Table VI.

## VI. CONCLUSION

In summary, a multiplier architecture which is completely carry-free from input to output is presented. Using the EBCA, the RB number, and the radix-4 Booth recoding, the 54b × 54b multiplication is possible with a 13.5-ns latency at 3.3 V consuming 53.4 mW. The proposed multiplier architecture with algorithmic enhancement eliminates the need for power-consuming fast carry-propagating adder in the final adder stage. As a result, the proposed multiplier is a high-speed low-power architectural solution for many mobile digital signal processing, broadband communications, and computer applications.

## ACKNOWLEDGMENT

The authors would like to thank G. Pratte of Motorola Inc. for packaging support of the prototype multiplier.

## REFERENCES

[1] Y. Hagihara, S. Inui, A. Yoshikawa, S. Nakazato, S. Iriki, R. Ikeda, Y. Shibue, T. Inaba, M. Kagmihara, and M. Yamashina, "A 2.7-ns 0.25-$\mu$m CMOS 54 × 54b multiplier," in *ISSCC Dig. Tech. Papers*, vol. 41, Feb. 1998, pp. 296–297.

[2] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 380–400, Sept. 1961.

[3] 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.

[4] M. D. Ercegovac and T. Lang, "On-the-fly conversion of redundant into conventional representations," *IEEE Trans. Comput.*, pp. 895–897, July 1987.

[5] C. Goto, A. Inoue, R. Ohe, S. Kashiwakura, S. Mitarai, T. Tsuru, and T. Izawa, "A 4.1-ns compact 54 × 54-b multiplier utilizing sign-select Booth encoders," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1676–1682, Nov. 1997.

[6] J. Mori, M. Nagamatsu, M. Hirano, S. Tanaka, M. Noda, Y. Toyoshima, K. Hashimoto, H. Hayashida, and K. Maeguchi, "A 10-ns 54 × 54-b parallel structured full array multiplier with 0.5-$\mu$m CMOS technology," *IEEE J. Solid-State Circuits*, vol. 26, pp. 600–606, Apr. 1991.

[7] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8-ns 54 × 54-bit multiplier with high speed redundant binary architecture," *IEEE J. Solid-State Circuits*, vol. 31, pp. 773–783, June 1996.

[8] T. Hanyu and M. Kameyama, "A 200 MHz pipelined multiplier using 1.5-V supply multiple-valued MOS current-mode circuits with dual-rail source-coupled logic," *IEEE J. Solid-State Circuits*, vol. 30, pp. 1239–1245, Nov. 1995.

[9] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 × 54-b regularly structured tree multiplier," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1229–1236, Sept. 1992.

**Yun Kim** (S'96–M'01) received the B.S.E.E. degree from the California Institute of Technology, Pasadena, CA, in 1995, and the M.S.E.E. degree from the University of Illinois at Urbana-Champaign in 1997.

From 1995 to 2000, he was with the circuits group of the University of Illinois at Urbana-Champaign. Currently, he is with Integrated Systems Research Labs within Motorola Labs. Since joining Motorola Labs in 2000, he has been involved in digital design of baseband processors for flexible low-power digital communications systems.

**Bang-Sup Song** (S'79–M'83–SM'88–F'99) received the B.S. degree from Seoul National University, Seoul, Korea, in 1973, the M.S. degree from the Korea Advanced Institute of Science, Taejon, Korea, in 1975, and the Ph.D. degree from the University of California, Berkeley, in 1983.

From 1975 to 1978, he was a Research Staff Member with the Agency for Defence Development, Korea, working on fire-control radars and spread-spectrum communications. From 1983 to 1986, he was a Member of Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ, and was also an Adjunct Professor in the Department of Electrical Engineering, Rutgers University, New Brunswick, NJ. From 1986 to 1999, he was a Professor in the Department of Electrical and Computer Engineering, University of Illinois, Urbana. He currently holds the Powell Endowed Chair in Wireless Communication in the Department of Electrical and Computer Engineering, University of California, San Diego.

Dr. Song received a Distinguished Technical Staff Award from AT&T Bell Laboratories in 1986, a Career Development Professor Award from Analog Devices in 1987, and a Xerox Senior Faculty Research Award in 1995. His IEEE activities have been in the capacities of Associate Editor and Guest Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS and the IEEE JOURNAL OF SOLID-STATE CIRCUITS, and a Program Committee Member for the IEEE International Solid-State Circuits Conference and the IEEE Symposium on Circuits And Systems.

**John Grosspietsch** (M'85) received the B.S.E.E. degree from the Illinois Institute of Technology, Chicago, in 1978. He received the M.S.E.E. degree in 1979 and the Ph.D. degree in 1984 also from the Illinois Institute of Technology.

Prior to joining Motorola, he designed low-power analog CMOS signal processing systems for hearing aids. He joined Motorola in 1989 in the IC Design Research Laboratory within the Chicago System and Technology Laboratory of Motorola Labs. He has worked on many mixed-signal CMOS IC design projects for analog, TDMA, and CDMA cellular telephones as well as LCD display drivers and wireless data modems. Currently, he is Manager of the Integrated Systems Research Labs within Motorola Labs, where he manages development of integrated signal processing systems for communications applications.

**Steven F. Gillig** (M'95) received the B.S. and M.S. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1974 and 1976, respectively, and the M.B.A. degree from the University of Chicago, Chicago, IL, in 1990.

Since joining Motorola in 1976, he has been involved in the research and design of enabling technologies for Motorola's communications equipment. He holds 27 issued patents and is currently Director of the Communication Technologies Research Labs of Motorola Labs.