Brief Contributions

Signed Binary Addition Circuitry with Inherent Even Parity Outputs

M.A. Thornton, *Member, IEEE Computer Society*

Abstract—A signed binary (SB) addition circuit is presented that always produces an even parity representation of the sum word. The novelty of this design is that no extra check bits are generated or used. The redundancy inherent in a SB representation is further exploited to contain parity information.

Index Terms—Signed binary addition, computer arithmetic, parity, redundant binary, fast addition circuit.

1 Introduction

THE binary addition circuit is a basic building block for many arithmetic logic units. One technique for implementing fast adders is to utilize signed digit number systems where the digit set has cardinality larger than the radix value [1], [13]. When such number systems are used, particular quantities may be represented by more than one unique digit string. This redundancy is used to produce addition circuits that do not require carry ripples to traverse the entire set of digits comprising the addend and augend. Examples of circuits using redundant signed binary arithmetic may be found in [4], [8], [7], [10], [16].

In particular, the work described in [4] and [10] utilizes the signed binary digit (SBD) set comprised of the digits $\{\overline{1}, 0, 1\}$. Different addition tables were used in these two approaches. By examining the two digits in the augend and addend to the immediate right (one radix power less), intermediate sum and carry digits of the current addend and augend digits could be chosen such that no carry ripple will occur. As is shown in this paper, there are other possible signed binary addition tables that exhibit this property. An entire class of new addition tables is discussed with some specific examples given.

While this class of addition tables is useful for building high speed addition circuitry, the redundancy inherent in the data words can also be used for error detection purposes. It is shown that an addition circuit can be constructed that utilizes SBD redundancy for both the purpose of constructing a high speed addition circuit and to always generate sums that have even parity. This characteristic incorporates a fault detection characteristic without adversely affecting the parallelism needed for high speed operation. This approach has the advantage that no extra parity bits are generated, the bits representing the sum value itself are all that are present.

In general, SBD adders pay the implementation price of an increased area requirement in order to achieve a decreased delay characteristic. Furthermore, the resulting sum is in SBD form requiring a conversion to twos complement or some other form incurring the delay of an extra addition stage since the sum is typically separated into a negative and positive value, then combined using a more traditional adder. For these reasons, SBD addition circuits are generally not considered to be useful as standalone adders. However, in the case of accumulation of more than two operands, such as partial product accumulation in a multiplier circuit, SBD circuits offer distinct advantages.

As described in [6], [17], trees of 3:2, 7:3, and 15:4 counters allow for the sum of several operands to be formulated while incurring the cost of a single carry-propagation stage (or an equivalent addition circuit) only in the final level of the addition tree. In the tree-based schemes for partial product accumulation, the overall delay is dependent upon the number of levels of constant delay adders. The number of levels can be decreased by using SBD adders with reduction rates of 2:1 instead of the 3:2 counters used in [17]. This fact is responsible for the use of SBD adders in commercial products such as the one described in [4].

The central justification for the pursuit of the results described in this paper is to refine the SBD addition circuit which is the basis of some commercially produced integrated circuits. In particular, it is shown that an entire family of addition tables may be used as a basis for these circuits. Furthermore, exploitation of the redundancy inherent in the encoding of the operands such that they are represented using the digit set $\{\overline{1}, 0, 1\}$ may be used to incorporate an error detection mechanism with no additional check bits required. This can be a useful result for designers wishing to incorporate error checking in a SBD based circuit by enforcing a parity representation without generating and including extra parity check bits in the sum data word. The error detection circuitry, (a tree of XOR gates) can be used to perform the "error detection" function of intermediate sum words at the same time the operands are propagating through the addition tree structure. In the past, parity check codes have been mainly used for memory and communications circuit error detection. The basic reason for the use of codes other than those using parity check bits is that parity is not preserved under arithmetic operations [14].

The contribution of this paper is to show the existence of alternative SBD addition tables and to show that sufficient redundancy is present such that all results can be restricted to even parity words. The redundancy can be a beneficial characteristic for incorporating an error detection mechanism. The benefit lies in the fact that the mechanism is inherent, requiring no additional parity check bits to be generated, and it also does not cause significant decreases in the speed of the circuit. As an example, the costly parity checker circuit could operate in the background in a checkpointing scheme, allowing computations to proceed at a more rapid pace. In this scenario, the detection of an error by the parity checker would only result in the recomputation of all past operations that have an elapsed time equivalent to the delay of the checker circuit itself. With modest addition circuit reliability characteristics, this recomputation would only arise rarely.

The remainder of this paper is organized as follows. Section 2 will provide a brief review of the properties of signed binary number systems and will describe their use in past addition circuits. In Section 3, a new class of addition tables is presented with a description of the theory of their construction included. Section 4 will provide the details of the implementation of the even parity addition circuit. Finally, the paper will conclude with a discussion of the results obtained and their usefulness.

2 Signed Binary Arithmetic

An SBD system allows each digit to carry its own arithmetic sign information rather than assigning a single arithmetic sign to an

• *The author is with the Department of Computer Systems Engineering, University of Arkansas, Fayetteville, AR 72701. E-mail: mat1@engr.uark.edu.*

Manuscript received 18 Aug. 1995; revised 15 Mar. 1996.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number 104707.0.

TABLE 1
SIGNED BINARY ADDITION TABLE USED TO PREVENT LONG CARRY PROPAGATION CHAINS

Addend + Augend Digits in Position i	Sign Information of of Digits in Position $i-1$	Intermediate Carry Digit, c_{i+1}	Intermediate Sum Digit, s_i
$\overline{1}+\overline{1}$	Not Used	$\overline{1}$	0
$\overline{1}+0$	Either is Negative	$\overline{1}$	1
$\overline{1}+0$	Neither is Negative	0	$\overline{1}$
$0+0$	Not Used	0	0
$1+\overline{1}$	Not Used	0	0
$1+0$	Either is Negative	0	1
$1+0$	Neither is Negative	1	$\overline{1}$
$1+1$	Not Used	1	0

TABLE 2
SIGNED BINARY ADDITION TABLE USING INTERMEDIATE BORROWS AND CARRIES

Addend + Augend Digits in Position i	Intermediate Borrow Out, b_{i+1}	Intermediate Borrow In, b_i	Intermediate Carry Digit, c_{i+1}	Intermediate Sum Digit, s_i
$\overline{1}+\overline{1}$	$\overline{1}$	$\overline{1}$	0	$\overline{1}$
$\overline{1}+\overline{1}$	$\overline{1}$	0	0	0
$\overline{1}+0$	$\overline{1}$	$\overline{1}$	0	0
$\overline{1}+0$	$\overline{1}$	0	1	$\overline{1}$
$0+0$	0	$\overline{1}$	0	$\overline{1}$
$0+0$	0	0	0	0
$1+\overline{1}$	$\overline{1}$	$\overline{1}$	1	$\overline{1}$
$1+\overline{1}$	$\overline{1}$	0	1	0
$1+0$	0	$\overline{1}$	0	0
$1+0$	0	0	1	$\overline{1}$
$1+1$	0	$\overline{1}$	1	$\overline{1}$
$1+1$	0	0	1	0

entire quantity. Therefore, the signed binary digit set is $\{\overline{1}, 0, 1\}$, where $\overline{1}$ indicates a negative value with unity magnitude. Number systems that have a digit set cardinality greater than the radix value are said to be redundant since a single quantity can be represented with more than one digit string (or radix polynomial).

2.1 Signed Binary Addition

One of the first signed binary addition tables was given in [8]. In this table, each final sum digit is represented as an intermediate sum digit and an intermediate carry digit. For two addend/augend pair cases, $1+0$ and $0+\overline{1}$, two possible intermediate sum and carry digits are allowed. The selection of which two intermediate values are used is determined by the arithmetic signs of the next lower ordered augend and addend. The strategy is to ensure that an $x_i = \overline{1}$ and $y_i = \overline{1}$ (or, an $x_i = 1$ and $y_i = 1$) never occur in the same radix place in the intermediate carry and sum words. Therefore, the final step of combining the intermediate carry and sum digits to form a final sum can be accomplished with a guarantee that no carry ripples will occur. This type of addition is much faster than ripple carry forms and does not suffer from large fanout requirements that carry-lookahead forms can require. It should also be noted that some recent modifications have been accomplished resulting in modified carry-lookahead adders with reduced fanout requirements [12], [9]. Table 1 is a reproduction of the addition table that originally appeared in [8].

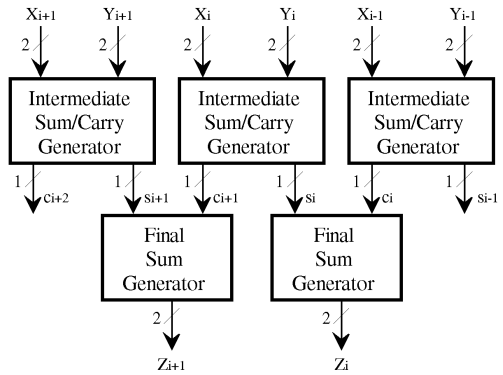

Fig. 1. Parallelism of signed binary adder.

As an example, consider the addition of the following two signed digit numbers:

$1\ 0\ \overline{1}\ \overline{1}\ 1\ 0\ 0$	addend
$0\ \overline{1}\ \overline{1}\ 0\ 1\ 0\ \overline{1}$	augend
$1\ 1\ 0\ \overline{1}\ 0\ 0\ \overline{1}$	intermediate sum
$0\ \overline{1}\ \overline{1}\ 0\ 1\ 0\ 0\ 0$	intermediate carry
$0\ 0\ 0\ 0\ 0\ 0\ 0\ \overline{1}$	final sum

As is apparent from the example, the formation of each final sum digit in the ith position depends only upon the addend and augend digits in the ith and $i-1$th positions. Using this method, a fast adder of arbitrary length can be built and is illustrated by the block diagram in Fig. 1.

This is not the only possible signed binary addition table that exhibits the behavior of limiting the carry ripple to one digit only. In the work in [4], an SBD addition table is described where all of the intermediate sum digits are restricted to the set $\{0, \overline{1}\}$ and all of the intermediate carry bits are restricted to the set $\{0, 1\}$. This ensures that during the formation of the final sum no carry will occur since the two possibilities $1+1$ and $\overline{1}+\overline{1}$ never exist. In

order to always form an intermediate sum word with all nonzero values being negative valued and an intermediate carry word with all nonzero digits being positive valued, a borrow digit is utilized. Addition using this type of table requires two steps; first, the borrow digits are formed and, second, the intermediate carry and sum digits are chosen based upon the value of the borrow digit. Like the intermediate carry digit, the borrow digit in the ith place depends only upon the the addend and augend digits in the $i-1$th position. The addition table for this type of adder is given in Table 2. As an example, the addition rules given in Table 2 will be used to compute the sum of the same signed binary numbers as used in the previous example.

$1\ 0\ \bar{1}\ \bar{1}\ 1\ 0\ 0$	addend
$0\ \bar{1}\ \bar{1}\ 0\ 1\ 0\ \bar{1}$	augend
$0\ \bar{1}\ \bar{1}\ \bar{1}\ 0\ 0\ \bar{1}\ 0$	borrow
$0\ 0\ \bar{1}\ \bar{1}\ 0\ \bar{1}\ \bar{1}$	intermediate sum
$0\ 0\ 0\ 1\ 1\ 0\ 1\ 0$	intermediate carry
$0\ 0\ 0\ 0\ 0\ 0\ 0\ \bar{1}$	final sum

In this method, a borrow digit is used to ensure that all intermediate sum digits are members of the set $\{0, \bar{1}\}$ and all intermediate carry digits are members of the set $\{0, 1\}$. In contrast, the previous method simply chooses a form of representation for the intermediate carry and sum digits that will never cause a carry ripple. An intuitive explanation of why the borrow digit is needed in the latter method is to view the augend and addend words as being "mapped" to two other SB values whose sum is equivalent to that of the addend and augend. Furthermore, the "mapped" values (which are actually the intermediate sum and intermediate carry words) must be composed with additional allowable digit constraints to ensure the prevention of carry ripple. These constraints are that the intermediate carry can only contain the digits, $\{0, 1\}$ and must be an even quantity. Likewise, the intermediate sum value can only contain $\{\bar{1}, 0\}$ and may be odd or even. This mapping is not possible unless a "borrow" is used to shift quantities from one radix place to another within the addend and augend digits.

A comparison of the block diagram for the previous method as shown in Fig. 1 and that of the this method as shown in Fig. 2 illustrates that both formulate the final sum digit based only upon the information contained in two consecutive addend and augend digits. However, the latter method requires a borrow generator to be included in the circuitry versus a digit selection portion as is used in the first method.

The second method requires an additional internal stage in the addition circuit (the borrow generator). This stage can be implemented as a single two-input logic gate with an appropriate digit encoding. The advantage gained by paying the price of the borrow generator circuitry is that the internal borrow and intermediate sum and carry digits only require a single bit for their representation. The previous method does not need the internal borrow generation stage, but requires two bits to represent the intermediate sum and carry digits since they both may have values from the set $\{\bar{1}, 0, 1\}$.

3 Alternative Signed Binary Addition Tables

Many other possibilities exist for suitable addition tables for constructing circuits using the ideas in [5]. It is worthwhile to derive the alternate addition tables since their implementation can result in circuits with more desirable implementation properties for various technologies. SBD addition circuits are known for requiring a relatively large amount of circuit area. However, the definition of circuit area differs from technology to technology. For example, if the addition circuit is to be implemented using a PLA structure, a

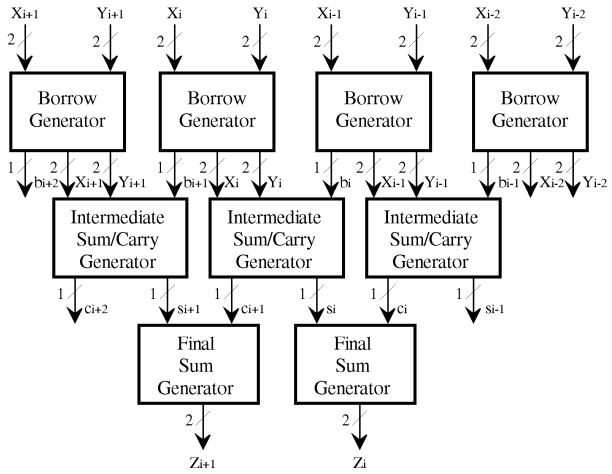

Fig. 2. Parallelism of signed binary adder using intermediate borrow and carry.

circuit with a minimal number of product terms (and fewer literals per product) may be desirable. On the other hand, the amount of internal connections, and, hence, routing constraints may limit the overall area in a standard cell ASIC implementation. The importance of deriving alternative addition tables is to offer the designer a choice of tables from which to realize a corresponding circuit that is the most suitable for his/her fabrication technology.

From a theoretical point of view, the alternative formulations arise since it is possible to choose the intermediate sum, carry, and borrow digits with values other than those in Table 2. One possibility is to let the intermediate sum and borrow digits be members of the set $\{0, 1\}$ and the intermediate carry digits be members of the set $\{\bar{1}, 0\}$. Another possibility is to allow the intermediate borrow digits to be members of the set $\{\bar{1}, 0, 1\}$, which is the entire set of SBDs.

In constructing alternative addition tables based upon this paradigm, it is useful to refer to an algebraic equation that describes the addend and augend digits as a function of the intermediate values. Before this equation is given, the notation used for each of the values is defined in Table 3. Note that the subscripts for each symbol indicate the power of the radix when the value is written in radix polynomial form.

TABLE 3
Notation Used to Represent Various Intermediate Values for Signed Binary Addition

SYMBOL	DESCRIPTION
x_i	SBD in the ith Position of the Addend Word
y_i	SBD in the ith Position of the Augend Word
b_i	SBD in the ith Position of the Borrow Word
s_i	SBD in the ith Position of the Intermediate Sum Word
c_i	SBD in the ith Position of the Intermediate Carry Word
z_i	SBD in the ith Position of the Final Sum Word

Any single addend/augend digit pair in the ith radix place contributes to the final overall sum by the intermediate carry and sum values, $2c_{i+1} + s_i$, the value borrowed from the $i+1$th digits, $2b_{i+1}$, and, by accounting for the value borrowed by the $i-1$th digits, b_i. Using these facts, an algebraic equation can be written using the notation in Table 3 and is given in (1).

$$x_i + y_i = 2(c_{i+1} + b_{i+1}) + s_i - b_i \tag{1}$$

TABLE 4
ALTERNATIVE SIGNED BINARY ADDITION TABLE USING INTERMEDIATE BORROWS AND CARRIES

Addend + Augend Digits in Position *i*	Intermediate Borrow Out	Intermediate Borrow In	Intermediate Carry Digit	Intermediate Sum Digit
$\overline{1}+\overline{1}$	0	0	$\overline{1}$	0
$\overline{1}+\overline{1}$	0	1	$\overline{1}$	1
$\overline{1}+0$	0	0	$\overline{1}$	1
$\overline{1}+0$	0	1	0	0
$0+0$	0	0	0	0
$0+0$	0	1	0	1
$1+\overline{1}$	1	0	$\overline{1}$	0
$1+\overline{1}$	1	1	$\overline{1}$	1
$1+0$	1	0	$\overline{1}$	1
$1+0$	1	1	0	0
$1+1$	1	0	0	0
$1+1$	1	1	0	1

TABLE 5
SIGNED BINARY ADDITION TABLE WITH NO RESTRICTIONS ON THE BORROW DIGITS

Addend + Augend Digits in Position *i*	Intermediate Borrow Out	Intermediate Borrow In	Intermediate Carry Digit	Intermediate Sum Digit
$\overline{1}+\overline{1}$	$\overline{1}$	$\overline{1}$	0	$\overline{1}$
		0	0	0
		1	1	$\overline{1}$
$\overline{1}+0$	$\overline{1}$	$\overline{1}$	0	0
		0	1	$\overline{1}$
		1	1	0
$0+0$	0	$\overline{1}$	0	$\overline{1}$
		0	0	0
		1	1	$\overline{1}$
$1+\overline{1}$	0	$\overline{1}$	0	$\overline{1}$
		0	0	0
		1	1	$\overline{1}$
$1+0$	0	$\overline{1}$	0	0
		0	1	$\overline{1}$
		1	1	0
$1+1$	1	$\overline{1}$	0	$\overline{1}$
		0	0	0
		1	1	$\overline{1}$

TABLE 6
ALTERNATIVE SIGNED BINARY ADDITION TABLE WITH NO RESTRICTIONS ON THE BORROW DIGITS

Addend + Augend Digits in Position *i*	Intermediate Borrow Out	Intermediate Borrow In	Intermediate Carry Digit	Intermediate Sum Digit
$\overline{1}+\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$	1
		0	0	0
		1	0	1
$\overline{1}+0$	0	$\overline{1}$	$\overline{1}$	0
		0	$\overline{1}$	1
		1	0	0
$0+0$	0	$\overline{1}$	$\overline{1}$	1
		0	0	0
		1	0	1
$1+\overline{1}$	0	$\overline{1}$	$\overline{1}$	1
		0	0	0
		1	0	1
$1+0$	1	$\overline{1}$	$\overline{1}$	0
		0	$\overline{1}$	1
		1	0	0
$1+1$	1	$\overline{1}$	$\overline{1}$	1
		0	0	0
		1	0	1

It should be noted that the sum $x_i + y_i$ can contain values from the set $\{\overline{2}, \overline{1}, 0, 1, 2\}$, which is not the signed binary digit set. However, (1) is extremely useful for building alternative addition tables since it may be used to find values for b_i, b_{i+1}, s_i, and c_{i+1} from the set $\{\overline{1}, 0, 1\}$ that satisfy $x_i + y_i$ regardless of whether its value is one of the members of the SBD set.

Since restrictions are placed upon the digit sets for the intermediate carry, intermediate sum, and sometimes, the borrow digits, the possible solutions to (1) may be enumerated and alternative addition tables may be created by choosing the solutions that satisfy the digit set constraints. Note that it is not necessary to restrict the borrow digits to a set containing only two digits from a theoretical standpoint. Using the procedure outlined above, several alternative addition tables were formed. In the first example, an addition table is given that requires the borrow and intermediate sum digits to be restricted to the values of $\{0, 1\}$ and the intermediate carry to be restricted to $\{\overline{1}, 0\}$. The result is given in Table 4.

Since the only requirement for this class of addition tables is that the nonzero digits in the intermediate sum and carry words have dissimilar arithmetic signs, it is not necessary to restrict the borrow digits to a subset of the SBDs. The following two addition tables, illustrated in Tables 5 and 6, are examples where the intermediate borrow digits are members of the set $\{\overline{1}, 0, 1\}$.

As an example, the addition performed previously will be performed again using Table 6.

$1\ 0\ \overline{1}\ \overline{1}\ 1\ 0\ 0$	addend
$0\ \overline{1}\ \overline{1}\ 0\ 1\ 0\ \overline{1}$	augend
$1\ 0\ \overline{1}\ 0\ 1\ 0\ 0\ 0$	borrow
$1\ 0\ 0\ 0\ 0\ 0\ 1$	intermediate sum
$\overline{1}\ \overline{1}\ 0\ 0\ 0\ 0\ \overline{1}\ 0$	intermediate carry
$0\ 0\ 0\ 0\ 0\ 0\ \overline{1}\ 1$	final sum

4 Implementation

This section will discuss some of the issues encountered in the implementation of the addition circuit. In particular, the SBD encoding and resulting logic equations are presented. Also, a brief discussion of the amount of circuit resources required to realize the circuit will be included. The even parity adder circuit (EPAC) described here is based upon the addition table given in Table 4. However, any of the SBD addition tables could have been used.

The basic difference between the circuit described here is that the two "Final Sum Generator" blocks in Fig. 2 are combined into a single module as indicated by the dotted lines. Since the inputs to the "Final Sum Generator" may be represented in terms of a single bit each, this part of the circuit transforms four binary inputs to four other binary values.

The output stage of the EPAC depends upon four adjacent addend and augend pairs instead of two as required by other implementations. This is a drawback in terms of area when compared to the SBD addition circuits in [5] and [10] which do not guarantee even parity outputs; however, when the EPAC and the addition circuit in [5] were both mapped to a common cell library, the gate count only increased by nine.

4.1 Signed Digit Encoding for Even Parity Generation

Even parity generation is achieved by carefully choosing the encoding of the signed binary digits and by allowing the signed binary digit 0 to have two different encodings. An important caveat in the design of this class of circuits is that care must be taken to avoid internal fanouts that could propagate errors to two individual bits in the sum word. This is necessary since only single bit errors can be detected as is the case with all single bit parity schemes.

The encoding scheme used for this implementation is given in Table 7. By using this encoding, each successive pair of signed binary digits when grouped together can be written such that they have even parity. In essence, the problem is transformed from a radix-2 system to a redundant radix-4 system with a digit set of $\{\overline{3}, \overline{2}, \overline{1}, 0, 1, 2, 3\}$, since successive pairs of signed digits can represent any value in this digit set. This is easy to see by examining all possible strings of bits of length 4 with even parity and their corresponding numeric value and noting that all possible values in the radix-4 digit set are represented.

TABLE 7
Signed Binary Digit Encoding for the Even Parity Adder

Signed Binary Digit, X_i	Binary Encoding, $x_{1i}x_{2i}$
$\overline{1}$	00
0	01
1	10
0	11

TABLE 8
Numeric Value of Even Parity Bit Strings Using New Encoding

Bit String	Signed Binary Value	Radix-4 Value
0000	$\overline{1}\ \overline{1}$	$\overline{3}$
0011	$\overline{1}\ 0$	$\overline{2}$
0101	00	0
0110	01	1
1001	10	2
1010	11	3
1100	$0\ \overline{1}$	$\overline{1}$
1111	00	0

4.2 Logic Circuit Implementation

Each block in Fig. 2 represents a logic function. This partitioning allows the overall circuit to be easily described but does not necessarily imply that the best design is an interconnection of these basic functions. Using the signed digit encodings given in Table 7, the following logic equations can be derived. These equations are expressed more compactly in terms of the intermediate borrow, sum, and carry values. Note that the restriction of the intermediate values to one of two SBDs allows them to be expressed with a single bit. In this case, the bit 0 is used to indicate the SBD, 0, and the bit 1 is used to indicate the SBD with unity magnitude.

$$b_{i+1} = x_{1i}\overline{x}_{2i} + y_{1i}\overline{y}_{2i} \tag{2}$$

$$s_i = b_i \oplus x_{2i} \oplus y_{2i} \tag{3}$$

$$c_{i+1} = \overline{x}_{2i}\overline{y}_{2i}\left(\overline{y}_{1i} + x_{i1}\right) + \overline{b}_i\left(x_{2i} \oplus y_{2i}\right) \tag{4}$$

Using these values as intermediate variables the equations representing the final sum values, Z_{i+1} and Z_i, can be expressed as follows.

$$z_{1i+1} = \overline{c}_{i+1}c_i s_i + c_{i+1}s_{i+1}\overline{s}_i + \overline{c}_{i+1}\overline{s}_{i+1}\overline{s}_i + c_i s_{i+1}s_i + \overline{c}_{i+1}\overline{c}_i s_{i+1} + c_{i+1}\overline{c}_i\overline{s}_{i+1}s_i \tag{5}$$

$$z_{2i+1} = c_i s_{i+1}\overline{s}_i + \overline{c}_i\overline{s}_{i+1}s_i + \overline{c}_{i+1}\overline{s}_{i+1}s_{i+1} + c_{i+1}s_{i+1} \tag{6}$$

$$z_{1i} = c_{i+1}\overline{c}_i\overline{s}_i + c_i\overline{s}_{i+1}s_i + \overline{c}_i\overline{s}_{i+1}\overline{s}_i + c_{i+1}s_{i+1}s_i + \overline{c}_{i+1}\overline{c}_i s_i + \overline{c}_{i+1}c_i s_{i+1}\overline{s}_i \tag{7}$$

$$z_{2i} = \overline{c_i \oplus s_i} \tag{8}$$

These equations describe the Boolean relationships within the EPAC. The actual circuit consists of 16 inputs (four addend and

TABLE 9
SYNTHESIS RESULTS USING misII

Circuit Block	Inputs	Outputs	Number of Logic Cells	Critical Path Length
Borrow Generator	4	1	3	2
Carry/Sum Generator	5	2	11	5
Final Sum Generator	4	4	18	6

augend signed digits) and four outputs (two final sum digits). Using the above equations, a description of the circuit was written using the *Verilog* hardware description language [15] and verified for all possible 2^{16} input combinations.

In addition to verifying the functionality of the EPAC, the *Verilog* code was also used to generate an *espresso* PLA file. The *espresso* program [2] minimized the file with the resulting PLA requiring 506 product terms. Also, each partitioned block in Fig. 2 was synthesized using the *misII* tool [3]. *misII* was invoked using the standard rugged script and the minimized logic was technology mapped to the smallest mcnc benchmark library [11]. The synthesis results are given in Table 9. The overall critical path using this synthesis approach is 13 logic cells including inverters. One EPAC would require a total of 60 logic cells including inverters. When the circuit was treated as a single 16 input, four output combinational logic block, *misII* was able to synthesize it using 53 logic cells from the mcnc library.

4.3 Conversion Methods for Binary and Signed Binary Values

Assuming the operands are in unsigned binary form initially, encoding simply requires replacing each 0 bit with the corresponding SBD zero value represented by 11 or 01. Likewise, each 1 bit is representing by the signed binary positive unity magnitude value encoded as 10. This simple encoding results in operands that may be directly used as inputs to the EPAC described above, however this encoding does not necessarily yield even parity input words. If a signed binary value is to be encoded, the sign bit is used to determine whether each 1 will be encoded as a 00 (a signed binary negative unity magnitude value) or a 10 (a signed binary positive unity magnitude value).

After performing this encoding, an additional step may be taken to transform the initial SBD encoded value into an equivalent even parity form. This is accomplished by using the portion of the signed binary addition circuit referred to as the "final sum generator." The initially encoded operand is used as input to a "final sum generator" module and the resulting output is an even parity representation.

5 CONCLUSION

A fast addition circuit with an even parity output, EPAC, has been developed and described in this paper. By exploiting the redundancy that is inherent in a signed binary adder, it was shown that a parity based fault detection mechanism can be incorporated while maintaining minimal adder delay through the avoidance of long carry ripples. This adder utilizes the same number of bits for the sum word as would be required in any implementation using the redundant SBD system. Also, a new class of addition tables for SBD representations has been developed. This new set of tables allows circuit designers to choose between various forms of SBD addition circuits.

REFERENCES

[1] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Trans. Electronic Computers*, vol. 10, pp. 389-400, 1961.
[2] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincintelli, *Logic Minimization Algorithms for VLSI Synthesis*. Boston: Kluwer Academic, 1984.
[3] R.K. Brayton, R. Rudell, A.L. Sanagiovanni-Vincintelli, and A.R. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Trans. Computer-Aided Design*, vol. 6, no. 6, pp. 1,062-1,081, Nov. 1987.
[4] W.S. Briggs and D.W. Matula, "A 17 × 69 Bit Multiply and Add Unit with Redundant Binary Feedback and Single Cycle Latency," *Proc. 11th Symp. Computer Arithmetic*, pp. 163-170, 1993.
[5] W.S. Briggs and D.W. Matula, "Signed Digit Multiplier," U.S. Patent no. 5,144,576, Sept. 1, 1992.
[6] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, vol. 34, pp. 349-356, Mar. 1965.
[7] H.M. Darley, "Signed Digit Adder Circuit," U.S. Patent No. 4,979,140, Dec. 18, 1990.
[8] Y. Harata, Y. Nakamura, H. Nagese, M. Takigawa, and N. Takagi, "A High-Speed Multiplier Using a Redundant Binary Adder Tree," *IEEE J. Solid-State Circuits*, vol. 22, pp. 28-34, Feb. 1987.
[9] V. Kantabutra, "A Recursive Carry-Lookahead/Carry-Select Hybrid Adder," *IEEE Trans. Computers*, vol. 42, no. 12, pp. 1,495-1,499, Dec. 1993.
[10] S. Kuninobu, T. Nishiyama, T. Edamatsu, T. Taniguchi, and N. Takagi, "Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation," *IEEE Trans. Computers*, vol. 36, pp. 80-85, 1987.
[11] R. Lisanke, "Logic Synthesis and Optimization Benchmarks User Guide," Microelectronics Center of North Carolina, version 2.0, pp. 1-62, Dec. 1988.
[12] T. Lynch and E.E. Swartzlander, "A Spanning Tree Carry Lookahead Adder," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 931-939, Aug. 1992.
[13] B. Parhami, "Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations," *IEEE Trans. Computers*, vol. 39, no. 1, pp. 89-98, Jan. 1990.
[14] T.R.N. Rao and Fujiwara, *Error-Coding for Computer Systems*. Englewood Cliffs, N.J.: Prentice Hall, 1989.
[15] E. Sternheim, R. Singh, R. Madhavan, and Y. Trivedi, *Digital Design and Synthesis with Verilog HDL*. San Jose, Calif.: Automata Publishing, 1993.
[16] N. Takagi and S. Yajima, "On-Line Error-Detectable High-Speed Multiplier Using Redundant Binary Representation and Three-Rail Logic," *IEEE Trans. Computers*, vol. 36, no. 11, pp. 1,310-1,317, Nov. 1987.
[17] C.S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Trans. Computers*, vol. 13, pp. 14-17, Feb. 1964.