On-Line Error Detection in a Carry-Free Adder

Whitney J. Townsend and Mitchell A. Thornton

Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS
{wjt1, mitch}@ece.msstate.edu

Parag K. Lala

Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR
lala@engr.uark.edu

***Abstract*--This work presents an improved design for a carry-free adder featuring on-line error detection. The salient contribution of this work is an extremely quick and cost-effective method of conversion from either two's complement or signed magnitude format into the internal 1-out-of-3 code used within this adder.**

***Index Terms*--carry-free addition, on-line error detection, signed binary digits, 1-out-of-3 code**

I. Introduction

Addition is one of the most fundamental operations for digital computations. Thus much effort has been invested in research that has led to faster and more efficient ways to perform this operation. [4, 5, 11, 15]. Nonetheless carry lookahead, carry select, and carry skip adders are all still constrained by the carry propagation delay. Residue number systems provide one way to restrict the carry propagation delay, signed digit representations offer another solution.

Interest in on-line error detection continues to grow as VLSI circuits increase in complexity. On-line error detection is increasingly becoming a desirable characteristic due to its ability to detect transient faults that may occur in a circuit during normal operation. On-line error detection provides an opportunity for self-diagnosis and self-correction within a circuit design [10, 16].

This work presents an improved design for a carry-free adder featuring on-line error detection. The use of parity encoding to enable the detection of incorrect operation in a signed binary digit adder was first suggested in [19]. The carry-free features of this adder draw upon the ideas from signed digit representations, but represent these digits internally using a 1-out-of-3 code facilitating the on-line error detection. The use of this encoding in a carry-free adder was first described in [6].

The salient contribution of this work is an extremely quick and cost-effective method of conversion from either two's complement or signed magnitude representation into the 1-out-of-3 code used internally within this adder. Another feature of this design includes a reduction in the size of the internal representation when compared to the design described in [6] resulting in area savings for both the adder circuit itself and also for the accompanying self-testing checker circuit.

The organization of this paper is as follows. Part II provides the necessary background in the areas of carry-free arithmetic using signed digit representations, numerical encoding, and on-line error detection. Part III describes the methodology used in the design of this adder. Part IV presents conclusions and highlights a few of the many possibilities for implementation of this design.

II. Background

The design of this adder incorporates ideas from carry-free arithmetic using signed number representations, numerical encoding, and on-line error detection.

A. Carry-Free Arithmetic

One way to remove the penalty of carry propagation delay is through the use of carry-free addition. Carry-free addition permits addition to become a parallel operation in which completion time is no longer dependant upon the number of bits to be added. Carry-free addition is possible through the use of redundant number systems. Common number systems have digits sets that have the same cardinality as the value of the radix. In contrast, a redundant number system can be implemented by a digit set which has more digits in the set than the value of the radix. This allows a given number to have more than one representation [2].

1) Signed Digit Numbers

The first redundant number systems proposed used radices of greater than two. [2, 12]. Each digit within these digit sets with the exception of zero is present in both positive and negative polarities. A line drawn above the digit indicates a negative-valued digit. In these systems, the carry-free addition is implemented by restricting the possible intermediate sum and carry word digits such that a carry out can never occur when the final sum is computed.

2) Signed Binary Digits

Radix-2 or signed binary digit number representations are of particular interest. The signed binary digit number set

This work was supported in part by the National Science Foundation under grant CCR-0000891 and CCR-0098272.

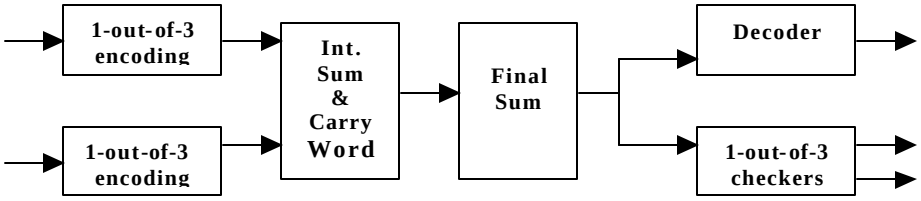

Figure 1. Block Diagram of Carry -Free Adder with Error Detection

consists of $\{\bar{1},0,1\}$. Due to these restricted possibilities for the digit set, the digits that generate the intermediate sum and carry word cannot be predetermined. Implementation of a radix-2 signed digit number representation requires the additional constraint that the next lower bit position also be considered during computation of the intermediate sum and carry word. Several different encodings of signed binary digits have been described in [3, 17, 19].

B. Numerical Encoding

Different ways to perform numerical encoding have long been of interest as a means of error detection and error correction [6].

1) *m-out-of-n Codes*

An *m-out-of-n* code is a particular type of encoding in which *m* and only *m* 1's occur in every valid code word of length *n*. These *m-out-of-n* codes have been studied with respect to error detection via totally self-checking checkers in [1, 18]. The parity encoding for error detection in carry-free addition proposed in [19] would result in an *k*-out-of-2*k* encoding, which are a subset of the *m*-out-of *n* codes. Totally self-checking checkers for these codes were also considered in [1].

2) *1-out-of-n Codes*

Much research effort has been invested in another subset of *m-out-of-n* codes, the 1-out-of-*n* encodings. The adder design presented in this work utilizes 1-out-of-3 encoded bit strings. A 1-out-of-3 encoding has three valid code words, 001, 010, and 100; all other possible encodings of 3 bits are invalid code words. Checkers for 1-out-of-3 codes have been studied in [20].

C. On-Line Error Detection

An on-line error detecting circuit consists of a circuit with two parts; one a functional part that performs the desired computational function of the circuit and the other a checker part that verifies the correct operation of the functional part of the circuit [7, 9, 14]. A *totally self-checking checker* (TSC) is used to check the output from the functional part of the circuit. The TSC is designed to output a two-rail code, which during normal operation is a 1-out-of-2 code. The TSC will output an error in the presence of either an internal fault within its own circuit or in the presence of an incorrect input from the functional circuitry. The two-rail checker needed to detect the output from the 1-out-of-3 checkers has been described in [8, 13].

III. Methodology

As an overview of the design, the block diagram of the adder design is shown in Figure 1. Two *n*-bit numbers are encoded into 1-out-of-3 bit strings in the leftmost blocks of the figure. In the next block the intermediate sum and carry word are calculated. In the following block the final sum is computed. This result is then passed to both the decoder for conversion from the 1-out-of-3 encoded bit strings and also to the checker for error detection. This circuit will detect all transient and static faults including stuck-at, stuck-open, and bridging faults that cause a unidirectional error. These faults will manifest themselves by violating the 1-out-of-3 code.

A. Encoding

The adder circuit described here uses the theory of signed binary digit arithmetic to create its carry-free addition. In contrast to the circuit described in [6] however, this method does not actually convert the addend and augend into signed binary digits prior to 1-out-of-3 encoding. Instead the 1-out-of-3 encoding is computed directly from either a two's complement or a signed magnitude representation of the addend and augend. Also in contrast to [6], no additional bit strings are required, only *n* 1-out-of-3 encoded bit strings are needed to represent an *n*-bit addend or an *n*-bit augend. This encoding is completed by a series of two XOR gates for each of the *n*-1 bits that represent the magnitude of the number and a differently configured set of two XOR gates for the *n*th bit that encodes the sign of the number.

For this encoding method, 100 represents a -1, 010 represents a 0, and 001 represents a +1. All the *n*-1 bits of an

TABLE I. INTERMEDIATE SUM AND CARRY WORD CALCULATION

Addend [i]	Augend [i]	Augend [i-1]	Carry Word	Intermediate Sum
100	100	100, 010, 001	100	010
	010	100	100	001
		010, 001	010	100
	001	100, 010, 001	010	010
010	100	100	100	001
		010, 001	010	100
	010	100, 010, 001	010	010
	001	100, 010	010	001
		001	001	100
001	100	100, 010, 001	010	010
	010	100, 010	010	001
		001	001	100
	001	100, 010, 001	001	010

operand will be converted into either 010 or 001 while the *n*th bit will become either 100 or 010. An example for an addend equal to +5 and an augend equal to -3 represented first astwo's complement 4-bit numbers and then converted into 1-out-of-3 encoded bit strings is shown.

Addend	+5	0101	010 001 010 001
Augend	-3	1101	100 001 010 001

This encoding does not alter the value of the numbers as can be shown by expressing them in radix polynomial form as illustrated below.

$$\begin{aligned}\text{Addend} &= 010 \times 2^3 + 001 \times 2^2 + 010 \times 2^1 + 001 \times 2^0 \\ &= 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 0 + 4 + 0 + 1 \\ &= +5\end{aligned}$$

$$\begin{aligned}\text{Augend} &= 100 \times 2^3 + 001 \times 2^2 + 010 \times 2^1 + 001 \times 2^0 \\ &= -1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= -8 + 4 + 0 + 1 \\ &= -3\end{aligned}$$

B. Partial Sum & Carry Word

Once the addend and the augend have been converted into the 1-out-of-3 encoded bit strings, the actual addition begins. Each of the *n* encoded digits of the operands are added together using Table I. This addition is shown for our example operands.

As shown in Table I, the *i* - 1 augend will have no effect on the choice made for the *i*th position for some combinations. The intermediate sum and the carry word generated in this block are then passed to the final sum block as shown in Figure 1.

010 001 010 001	Addend
100 001 010 001	Augend
100 010 010 010	Intermediate Sum
010 001 010 001	Carry Word

C. Final Sum

The intermediate sum and the carry word are combined within the final sum block to form the final sum. Due to the restrictions imposed by the addition table used for the previous block no carry will be generated. This is because the two additions that could generate a carry, 1 + 1 and -1 + -1, are guaranteed never to occur. The addition in this block is performed using Table II. The calculation of the final sum for the example operands is shown below.

100 010 010 010	Intermediate Sum
010 001 010 001	Carry Word
010 010 010 001 010	Final Sum

The leftmost bit pattern is the carry out from the addition and the resulting sum is present in the remaining four encoded bit strings.

D. Decoding

The last step in the calculation is to convert the encoded result back into either two's complement or signed magnitude form. This is accomplished by creating two *n*-bit words. The first word contains all of the positive one bit strings represented by 1's and all of the zero bit strings and all of the negative one bit strings represented as 0's.

TABLE II. FINAL SUM CALCULATION

Intermediate Sum	Carry Word	Final Sum
100	010	100
	001	010
010	100	100
	010	010
	001	001
001	100	010
	010	001

Conversely, in the second word all of the negative one bit strings are represented as 1's and all of the zero bit strings and all of the positive one bit strings are represented as 0's. The second word is then subtracted from the first word to produce the decoded result.

E. 1-out-of-3 Checkers

Concurrent with decoding, the output of the addition circuit is also checked for errors. If any bit string has more or less than one 1 an error has occurred. The output of these checkers is a two-rail code that should be either 01 or 10. If a 00 or a 11 bit string should occur then there has been an error either within the functional circuit or within the checkers themselves.

IV. CONCLUSIONS AND FUTURE WORK

This paper has presented the design of an efficient carry-free adder with on-line error detection. This adder has been designed to use the idea of signed binary digit representation but represents its digits internally using a 1-out-of-3 code. Next the adder must be implemented to enable further study. While a straight forward implementation is the next step, it is anticipated that much more interesting results will come not from the mere simplistic implementation of this adder, but will result from research into its implementation within more complex circuits such as multipliers, dividers, and other complex arithmetic circuits. These implementations will be able to take greater advantage of the adder's carry-free properties and will also be better able to bear the overhead of the on-line error detection as well as to benefit from its advantages.

REFERENCES

[1] D. A. Anderson and G. Metze, "Design of totally self-checking check circuits for *m*-out-of-*n* codes", *IEEE Trans. on Computers*, vol. 22, pp. 263-269, March 1973.

[2] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic", *IRE Trans. on Electronic Computers*, vol. 10, pp. 389-400, September 1961.

[3] H. Edamatsu, T. Taniguchi, T. Nishiyama, and S. Kuninobu, "A 33 mflops floating point processor using redundant binary representation", In *IEEE International Solid-State Circuits Conf.*, pp. 152-153 and 342-343, February 1988.

[4] M. J. Flynn and S. F. Oberman, *Advanced Computer Arithmetic Design*. New York, NY: John Wiley & Sons, 2001.

[5] I. Koren, *Computer Arithmetic Algorithms*. Englewood Cliffs, NJ: Prentice Hall Publishers, 1993.

[6] P. K. Lala, and A. Walker, "On-line error detectable carry-free adder design", in *International Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 66-71, October 2001.

[7] P. K. Lala, *Self-Checking and Fault-Tolerant Digital Design.* San Diego, CA: Academic Press, 2001.

[8] J. C. Lo, "Novel area-time efficient static cmos totally self-checking comparator", *IEEE Journal of Solid-State Circuits*, vol. 28, pp. 165-168, February 1993.

[9] T. Nanya and T. Kawamura, "On error indication for totally self-checking systems", in *IEEE Trans. on Computers*, vol. 36, pp. 1389-1392, November 1987.

[10] M. Nicolaidis, "Efficient implementations of self-checking adders and alus", in *23rd International Symp. on Fault-Tolerant Computing*, pp. 586-595, June 1993.

[11] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. New York, NY: Oxford University Press, 2000.

[12] B. Parhami, "Generalized signed-digit number systems: a unifying framework for redundant number representations", *IEEE Trans. on Computers*, vol. 39, pp. 89-98, January 1990.

[13] S. J. Piestrak, "Design method of a class of embedded combinational self-testing checkers for two-rail codes", *IEEE Trans. on Computers*, vol. 51, pp. 229-234, February 2002.

[14] S. J. Piestrak, "Self-checking design in eastern Europe", *IEEE Design & Test of Computers*, vol. 13, pp. 16-25, Spring 1996.

[15] E. E. Swartzlander, Jr. (Ed.), *Computer Arithmetic, Volume II*. Los Alamitos, CA: IEEE Computer Society Press, 1990.

[16] N. Takagi and S. Yajima, "On-line error-detectable high-speed multiplier using redundant binary representation and three-rail logic", *IEEE Trans. on Computers*, vol. 36, pp. 1310-1317, November 1987.

[17] N. Takagi, H. Yasura, and S. Yajima, "High-speed vlsi multiplication algorithm with a redundant binary adder tree", *IEEE Trans. on Computers*, vol. 34, pp. 789-796, September 1985.

[18] D. L. Tao, C. R. P. Hartmann, and P. K. Lala, "A general technique for designing totally self-checking checker for 1-out-of-n code with minimum gate delay", *IEEE Trans. on Computers*, vol. 41, pp. 881-886, July 1992.

[19] M. A. Thornton, "Signed binary addition circuitry with inherent even parity outputs", *IEEE Trans. on Computers*, vol. 46, pp. 811-816, July 1997.

[20] J. Q. Wang, and P. K. Lala, "Partially strongly fault secure and partially strongly code disjoint 1-out-of-3 code checker", *IEEE Trans. on Computers*, vol. 43, pp. 1238-1240, October 1994.