Comment on the Sequential and Indeterminate Behavior of an End-Around-Carry Adder

JOHN J. SHEDLETSKY

***Abstract*—The sequential and indeterminate behavior of an end-around-carry (EAC) adder is examined. This behavior is commonly overlooked in the literature. Design modifications to impose determinism are provided. These modifications also eliminate the troublesome negative zero found in the one's complement number system.**

***Index Terms*—End-around-carry (EAC) adder, negative zero, one's complement arithmetic.**

COMMENT

Subtraction by addition of the complement is a common technique in computer design. The diminished radix system for the representation of binary numbers is well known and referred to as the one's complement system [1], [2]. The one's complement of a binary number A is easily formed by complementing each bit of A. Zero has two representations, both $00\cdots0$ (positive zero) and $11\cdots1$ (negative zero). The one's complement system is particularly suited for use with some arithmetic error codes [3], [4].

A one's complement adder is constructed from a conventional binary adder by connecting the most significant carry-out line to the least significant carry-in line, as illustrated in Fig. 1.

The end-around-carry (EAC) adder is commonly assumed to form a unique sum for every possible pair of addends. In particular, the result of adding a number and its one's complement is popularly thought to be negative zero [1], [2]. Indeed, this conviction led to the design of a "subtractive" adder (a subtractor that complements the subtrahend) for the fixed add unit in the CDC 6600 [5]. It was thought that such an adder would always sum a number and its one's complement to positive zero instead of the troublesome negative zero. In fact, for both the adder in Fig. 1 and an iterative cell "subtractive" adder, the sum may be positive *or* negative zero depending on the preceding addends and the relative propagation delays in the adder.

For each cell in the adder in Fig. 1, the carry-out is equal to the carry-in when the addend input lines A and B assume carry-propagating values (A,B = 0,1 or 1,0). When a number and its one's complement are added, the addend input lines of every cell have carry-propagating values and the carry lines may assume one of two stable states. All the carry lines may be 0, resulting in a negative zero sum, or all the carry lines may be 1, resulting in a positive zero sum. Fig. 2 shows that the choice between these two possible states can depend either on the preceding values of the carry lines or on the relative propagation delays in the adder.

In Fig. 2(a) (Fig. 2(b)) the all 0 (1) state of the carry lines in the first addition is preserved in the second addition, leading to a negative (positive) zero sum. The addends in the second addition are the same in both Fig. 2(a) and (b), yet the sums differ. In this case, the values of the addends in the first addition affect the sum in the second addition; the adder exhibits sequential behavior.

In Fig. 2(c) the carry lines assume mixed values in the first addition. The stable state finally reached by the carry lines in the second addition depends on the relative propagation delays in the adder. In this case, the behavior of the adder is indeterminate. If the propagation delays are exactly equal, then neither state is finally reached; the values assumed by the carry lines in the first addition rotate to the left in a perpetual race.

The existence of two stable states in the EAC adder and the EAC "subtractive" adder (with or without carry lookahead) is only possible when a number and its one's complement are added. The adder is deterministic with one stable state for all other pairs of addends. No theoretical problem is posed in the indeterminate case, since both stable states lead to recognized forms of zero. A practical timing problem is presented though, by the potential for a long race between the two stable states. For example, the 50-bit EAC adder in the Rice University computer experienced 50 μs races, compared to a normal add time of 1 μs [6].

A simple solution is to drive the carry lines to either stable state when a number and its one's complement are added, thereby eliminating the indeterminism. Furthermore, by driving the carry lines to the all 1 state, the sum of a number and its one's complement is always positive zero. Since there is no other way to produce a negative zero from addends that are not negative zero,[1] the troublesome negative zero can thus be eliminated from the one's complement number system.

The carry lines can be driven to the all 1 state by holding every carry line at 1 for a short time at the beginning of an addition. The hold time

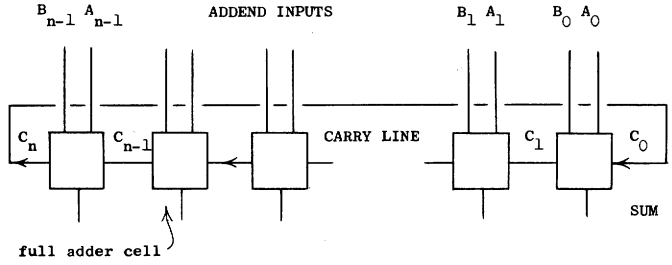

Fig. 1. End-around-carry (EAC) adder for the one's complement number system.

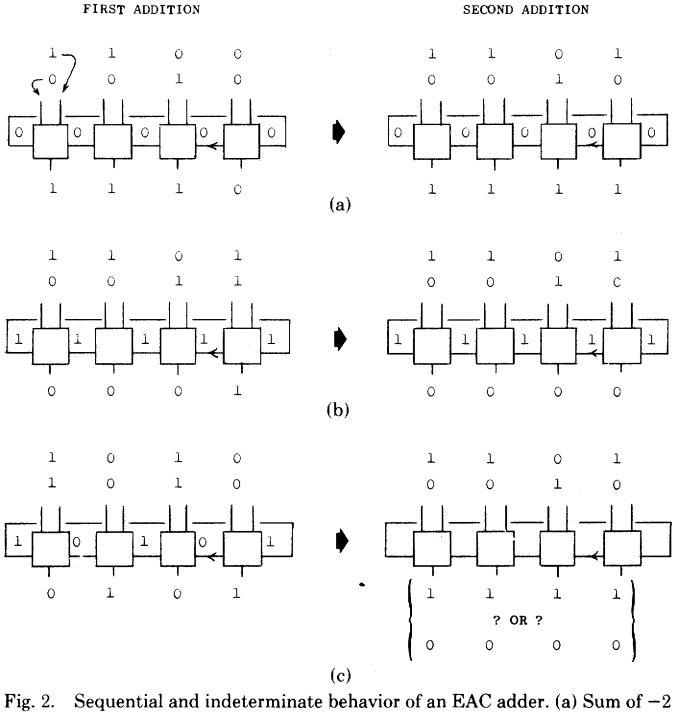

Fig. 2. Sequential and indeterminate behavior of an EAC adder. (a) Sum of −2 and 2 in the second addition is negative zero due to the all 0 state of the carry lines. (b) Sum of −2 and 2 in the second addition is positive zero due to the all 1 state of the carry lines. (c) Sum of −2 and 2 in the second addition is indeterminate due to a race in the carry lines.

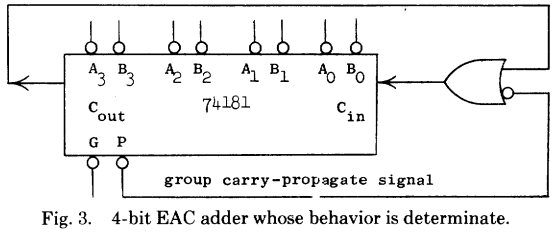

Fig. 3. 4-bit EAC adder whose behavior is determinate.

Manuscript received February 3, 1976. This work was supported in part by the National Science Foundation under Grant GJ-40286.

The author was with the Digital Systems Laboratory, Department of Electrical Engineering and Computer Science, Stanford University, Stanford, CA. He is now with the IBM Watson Research Center, Yorktown Heights, NY 10598.

[1] The sum of two negative zero addends is negative zero.

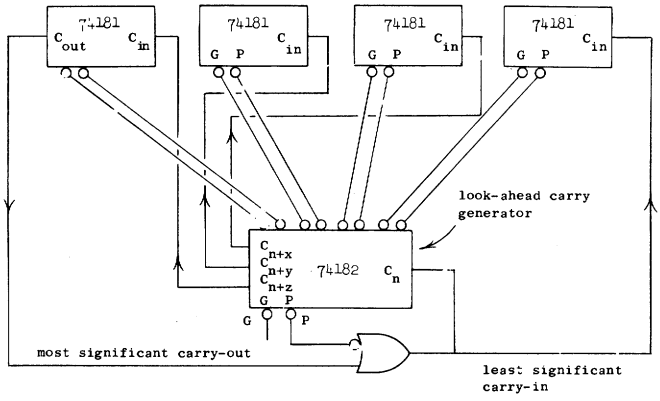

Fig. 4. 16-bit EAC adder whose behavior is determinate.

must be greater than the carry-propagation delay for a single adder cell.

An alternative scheme requiring less control logic to drive the carry lines to the all 1 state is illustrated in Figs. 3 and 4. The group carry-propagate signal P found in many medium- and large-scale integration (MSI and LSI) adders (Signetics 74181 for example) is 1 when the addend inputs to each bit position in the group are propagating inputs. When the group consists of all bit positions, the group carry-propagate signal is 1 only when a number and its one's complement are added. In this case, the output of the OR gate in Fig. 3 is forced to 1 and the external carry lines are driven to the all 1 state.

ACKNOWLEDGMENT

The author thanks Dr. A. Verdillon for pointing out that the indeterminism of an EAC adder is neatly described by a state model, where the stable states correspond to the all 0 and all 1 carry states mentioned in the text.

REFERENCES

[1] T. C. Bartee, *Digital Computer Fundamentals.* New York: McGraw-Hill, 1972.
[2] H. S. Stone, Ed., *Introduction to Computer Architecture.* Palo Alto, CA: Science Research Associates, 1975.
[3] A. Avizienis, "Arithmetic error codes: Cost and effectiveness studies for application in digital system design," *IEEE Trans. Comput.*, vol. C-20, Nov. 1971.
[4] J. F. Wakerly, "Low-cost error detection techniques for small computers," Digital Systems Lab., Stanford University, Stanford, CA., Tech. Rep. 51, Dec. 1973.
[5] J. E. Thornton, *Design of a Computer—The Control Data 6600.* Glenview, IL: Scott, Foresman, and Co., 1970.
[6] M. Graham, Dep. of Elec. Eng. and Comput. Sci., Univ. of California, Berkeley, CA, personal communication.

Textural Boundary Analysis

WILLIAM B. THOMPSON

***Abstract*—A procedure is demonstrated for locating textural boundaries in the digital image representation of a natural scene. The technique involves development of an edge operator capable of integrating multiple textural features into a single boundary determination. The process is designed to simulate actual perception of textural discontinuities. Success of the system is demonstrated on pictures with prominent perceived boundaries not detectable by methods based only on differences in average brightness.**

***Index Terms*—Pattern recognition, picture processing, scene segmentation, textural edge detection, texture.**

Manuscript received August 1, 1975; revised April 16, 1976. This research was sponsored by the Advanced Research Projects Agency of the Department of Defense monitored by the Air Force Eastern Test Range under Contract F08606-72-C-0008.

The author was with the Image Processing Institute, University of Southern California, Los Angeles, CA 90007. He is currently with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

I. INTRODUCTION

Texture is being increasingly recognized as an important cue for the analysis of natural imagery [1], [2]. The analysis of textural properties is particularly valuable for scene segmentation systems. In fact, readily perceived textural boundaries may be apparent in a scene where no obvious discontinuities in average brightness exist.

A number of authors have developed successful procedures for using image texture in the scene segmentation process. Bajcsy incorporates Fourier based measures into a region merger system [2]. Rosenfeld and Thurston describe an edge oriented approach capable of incorporating textural properties [3]. In the edge based system, a local operator sensitive to some property such as orientation or coarseness is applied at multiple points in a scene. Spatial discontinuities in the output of a given operator are assumed to correspond to textural boundaries. This approach has been employed in a number of subsequent papers which investigate different local operators and different criteria for making boundary determinations [4]–[6]. No clear mechanism has yet been suggested, however, for integrating the results from multiple operators. Thus, the approach must be limited to specific classes of imagery.

This correspondence describes a technique for incorporating multiple textural cues into a boundary analysis system. Furthermore, the procedure which is developed is designed to simulate actual human perception of textural discontinuities. The system is demonstrated on pictures with prominent perceived boundaries which could not be found by conventional techniques based on differences in average brightness.

II. SIMILARITY MEASURES

A central feature of any scene segmentation system using textural properties must be a meaningful measure of textural similarity. Textural edges may be defined as contiguous image regions of perceptually differing texture. Region oriented systems must merge or split regions based on measures of visual similarity. Unfortunately, few of the existing systems for scene segmentation make use of a textural similarity measure with any psychophysical foundation.

A previous paper described the construction of a textural distance function [7]. This function can numerically quantify the perceived degree of dissimilarity between two image regions. A prominent feature of the distance function is that it has been developed to accurately simulate human perception of textural differences. This is important in a system designed to describe a scene in a manner comparable to what a human observer would "see" in that scene.

Textural similarity is usually estimated by comparing specific image statistics in the two regions of interest. For example, one of the many numerical characterizations of texture [2], [8] could be evaluated in both regions. An absolute value difference of the two measures might be used as an indication of similarity (the smaller the value, the greater the similarity). Experience has shown, however, that none of the commonly used statistics, taken alone, is adequately correlated with perceptual response. The distance function model is able to integrate a large number of simple, statistical measures into a value which more closely corresponds with actual perception. Specifically, it was shown that in certain applications, a particular linear combination of simple difference measures was quite successful in simulating the perception of textural differences.

As an example, let $a_i(n)$ be the ith textural property of region n. Then, we can define the difference between regions l and m based on property i as

$$d_i(l,m) = |a_i(l) - a_i(m)|.$$

Each d_i represents an elementary difference function. A single estimate of region dissimilarity may be found by examining a collection of elementary measures. In particular, it is usually possible for an appropriate set of measures to find a set of coefficients $\{c_i\}$ such that the value