Low-Cost Residue and Inverse Residue Error-detecting Codes for Signed-Digit Arithmetic*

Algirdas Avizienis

UCLA Computer Science Department
University of California, Los Angeles
Los Angeles, California 90024

Abstract -- Low-cost residue and inverse residue codes for error detection in signed-digit arithmetic are defined in this paper. Checking algorithms are presented for both digit-serial and parallel computation of the residues. Residue digit operations are defined for two-operand addition, multi-operand addition, multiplication, conversion, and reconversion algorithms. All algorithms employ the same Arithmetic Building Element "ABE" that has been previously developed for signed-digit arithmetic.

1. Introduction: Background and Scope

An immediate detection of the appearance of faults is an essential prerequisite for the introduction of effective fault-tolerance into computing systems. Array processors and distributed systems in which several communication and computation functions take place concurrently depend on the presence of multiple local fault-detection algorithms that continuously check the messages and the results for fault symptoms. Recovery must be initiated before the effect of faults can spread through the system and cause extensive mutilation of stored information.

A second constraint to be observed is the **cost** of fault detection. Duplication (or even triplication) for fault detection is an obvious, but very costly solution that may be unacceptable in large array processors or extensive distributed systems. Fault detection by means of **error codes** is a much less costly alternative. A judicious choice of an error-detecting code may detect 95%-99% of the most likely faults at a cost of 10%-30% increase in complexity. Arithmetic error codes are an especially attractive choice because they can be used to check storage, transmission, and computing functions using the same low-cost checking algorithms: the computing of modulo A residues of messages, operands and results.

A general approach to the cost and effectiveness study of arithmetic error codes has been presented in [AVIZ 71a]. This paper summarized the results of a long-term study (1963-1971) of the application and effectiveness of arithmetic error codes and introduced the concepts of **inverse residue** codes and of **multiple** arithmetic error codes. The concept of **repeated-use** faults was presented and the effectiveness of various arithmetic codes with respect to both **determinate** and **indeterminate** repeated-use faults was established. An especially important result is the proof that inverse residue codes can detect the "compensating" determinate repeated-use faults that are not detected by ordinary residue codes. The modulo 15 inverse residue code was applied in the JPL-STAR experimental computer [AVIZ 71b]. Further results on determinate faults were presented in [PARH 73], [PARH 78], [AVIZ 79].

Signed-digit (S-D) arithmetic is of interest because of an addition/subtraction algorithm that is carry-free and requires the same time for operands of any length [AVIZ 61]. It also allows variable-precision operations with the most significant digits of the results being generated first [AVIZ 62]. S-D arithmetic operations are performed by arrays of one-digit radix 2^n (or radix 10) Arithmetic Building Elements "ABE" suitable for LSI and VLSI implementation [AVIZ 70], [TUNG 70].

The present paper defines a signed-digit form of residue and inverse-residue error-detecting codes suitable for the checking of S-D arithmetic operations as well as of the transmission of S-D operands. ABE-based algorithms are defined for serial and parallel execution of the checking operation that generates the residues of operands and results. Residue operations using ABE units are defined for the sign change, two-operand addition, multi-operand addition, multiplication, reconversion, and conventional input algorithms previously derived in [AVIZ 70].

2. Residue Encodings and Checking Algorithms

A variable-precision radix 2^b ($b \geq 2$) or radix 10^c ($c \geq 1$) S-D number X consists of k digits. Each digit X_i assumes the values

$$\{\bar{a},...,\bar{1},0,1,...,a\}$$

with
$(2^b-1) \geq a \geq (2^{b-1}+1)$ and
$(10^c-1) \geq a \geq (1+10^c/2)$

* This research has been supported by ONR Contract No. N00014-79-C-0866 (Research in Distributed Processing)

In the subsequent discussion the **minimal redundancy** representations with $a = 1+2^{b-1}$ and $a = 1+10^c/2$ will be used unless explicitly otherwise stated. The Arithmetic Building Element "ABE" and the algorithms of the ABE were described in detail in [AVIZ 70]. The input-output diagram of the ABE is reproduced here as Figure 1. All examples of checking will be applied to the radix 16 (a=9) illustrations of the same reference.

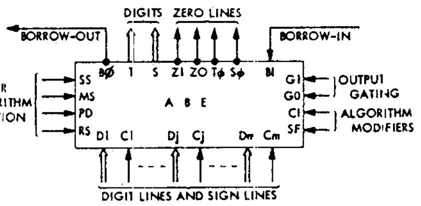

Figure 1. Arithmetic Building Element

The **Signed-Digit Residue Code** (S-D RC) is formed by attaching the modulo 2^b-1 (or 10^c-1) **residue digit** X^* ($\bar{a} \leqslant X^* \leqslant a$) next to the least significant digit X_0 of the radix r = 2^b (or r = 10^c) S-D operand X:

$$X^* = (2^b-1)|X \quad or\ X^* = (10^c-1)|X$$

The **Signed-digit Inverse Residue Code** (S-D IRC) is formed by attaching the additive inverse $\bar{X}^*$ of X^* to X, where:

$$\bar{X}^* = 0-X^*, \quad or\ \bar{X}^*+X^*=0$$

It should be noted that the residue digit X^* (or $\bar{X}^*$) itself is a redundant representation of the modulo 2^b-1 (or 10^c -1) residue. For example, with r=16 and a=9, the modulo 15 residue values {9, 8, 7, 6} have the second valid representations {$\bar{6}, \bar{7}, \bar{8}, \bar{9}$} respectively, while the values {5, 4, 3, 2, 1, 0, $\bar{1}, \bar{2}, \bar{3}, \bar{4}, \bar{5}$} are represented uniquely. With r=10 and a=6, the modulo 9 residue values {6, 5, 4, 3} have the second representations {$\bar{3}, \bar{4}, \bar{5}, \bar{6}$} respectively, with {2, 1, 0, $\bar{1}, \bar{2}$} remaining uniquely represented.

The advantage of this redundancy is that the ABE algorithms are directly applicable to perform residue digit operations and the checking algorithm. Any disadvantages that may occur are avoided when tests for equivalence of two residue digits X^* and Y^* are carried out as the zero test of the modulo 2^b-1 (or 10^c-1) sum of X^* and $-(Y^*)$. An equivalence is indicated by the result

$$(2^b-1)|(X^*-Y^*) = 0$$

For example, given r=16 and a=9, the four results $15|(9-9) = 0$; $15|(9-\bar{6}) = 0$; $15|(\bar{6}-9) = 0$ and $15|(\bar{6}-\bar{6}) = 0$ will always indicate the equivalence of residue digit values 9 and $\bar{6}$.

The **Serial Checking algorithm** that computes the value $(2^b$-1)|X uses one ABE (Fig. 1) set for the SS (Simple Sum) algorithm [AVIZ 70, p. 735]. One digit X_i is entered at a time (beginning with the most significant digit X_{k-1}) on the inputs D1 and D2. The inputs D3 and D4 are originally set to zero. The sum digit on output S is returned to the inputs D3 and D4 for the subsequent addition of X_{k-2}. After the last digit X_0 has been added, the output on S represents the residue $(2^b$-1)|X.

The algorithm can be stated (for j=0,1,...,k-1) as:

$$S(j+1) = [S(j) + X_{k-1-j}]-(r-1)g_j$$

where S(0) = 0, S(k) = X^* and the value of g_j is determined from $K_j = [S(j) + X_{k-1-j}]$ as follows:

$$g_j = \begin{cases} 1 & \text{if } K_j \geqslant a \\ \bar{1} & \text{if } K_j \leqslant \bar{a} \\ 0 & \text{for } \bar{a} < K_j < a \end{cases}$$

Example 1: The Modulo 15 residue of the radix 16 s-d operand $X = \bar{9}076$ is obtained as follows:

$$\begin{aligned} S(1) &= (0+\bar{9})-15(\bar{1}) = 6 \\ S(2) &= (6+0)-15(0) = 6 \\ S(3) &= (6+7)-15(1) = \bar{2} \\ S(4) &= (\bar{2}+6)-15(0) = 4 = 15|X \end{aligned}$$

The **Parallel Checking algorithm** uses one ABE unit set for the MS (Multiple Sum) algorithm [AVIZ 70, p. 736]. All digits X_i ($0 \leqslant i \leqslant k-1$) are summed at once to produce the two-digit result

$$\sum_{i=0}^{k-1} X_i = rt_{i-1} + S_i$$

The two output digits t_{i-1} and S_i are summed modulo r-1 in an ABE/SS unit as described for the Serial Checking algorithm:

$$(2^b-1)|X = [t_{i-1} + S_i] - (r-1)g_i$$

where the value of g_i is determined from $[t_{i-1}+S_i]$. If the number of digits exceeds the capacity of one ABE/MS unit, two or more ABE/MS units are used at once; their output digits are summed in another ABE/MS unit.

Example 2: The residue computation of Example 1 is performed in parallel as follows:

$$\sum X_i = (\bar{9}+0+7+6) = 16(0)+4 = rt_{i-1} + S_i$$

$$15|X = (0+4)-15(0) = 4$$

The checking algorithms for the radix 10^c ($c \geqslant 1$) S-D representations that generate the modulo 10^c-1 residues are obtained by an obvious modification of the radix 2^b algorithms.

The **effectiveness** of S-D residue codes can be assessed by noting that undetectable errors are caused only by faults that change the value of the S-D number by a multiple of 2^b -1 (or of 10^c-1). Such changes are generally unlikely; for example, given r=16 and a=9, a single digit must be changed as follows: $9 \leftrightarrow \bar{6}$, $8 \leftrightarrow \bar{7}$, $7 \leftrightarrow \bar{8}$, and $6 \leftrightarrow \bar{9}$.

A detailed study of effectiveness requires the full knowledge of the internal (binary or other) representation of digit values and an analysis of the effects of repeated-use faults when they may affect the operand or the result. A detailed study, following the methodology applied to conventional arithmetic [AVIZ 71a], [AVIZ 79] will be published separately.

3. Residue Digit Operations for S-D Arithmetic Algorithms

The modulo 2^b-1 (or 10^c-1) residue arithmetic operations that are performed on operand residue digits to generate residue digits for the results are defined here. Note that the same ABE units (without any modifications) are suitable for the residue arithmetic.

Additive Inverse (one operand): The change of sign of an operand X requires that the sign of the residue (or inverse residue) digit also should be changed. $(X;X^*) = (\bar{9}076;4)$ will be transformed to $-(X;X^*) = (90\bar{7}\bar{6};\bar{4})$ to get the additive inverse of X.

Simple Sum "SS" (two summands): The residue (or inverse residue) digit for the SS algorithm (addition of two operands X,Y) is formed by executing the ABE/SS Serial Checking algorithm (described previously) letting $S(0) = X^*$ and $X_{k-1} = Y^*$. The result S(1) is the residue digit for the SS result X+Y. For example, adding $(X, X^*) = (\bar{9}076;4)$ and $(Y,Y^*) = (\bar{8}95\bar{3};3)$ will yield $(\bar{1}0\bar{6}43;7)$.

Multiple Sum "MS" ($m \leqslant r+1$ summands): The residue digit for the result of the MS algorithm is formed by executing the Parallel Checking algorithm with the residue digits of all summands serving as the inputs.

Product "PD" (one digit X_i times an operand Y): The PD algorithm is specified in [AVIZ 70, p. 736]. The residue digit for this product is obtained from X_i and the residue digit Y^* (of Y) as follows: Using one ABE (Fig. 1) in the PD mode, enter X_i on D1, and apply the residue digit Y^* to the three inputs D2, D3 and D4. The output (on S) is the modulo 2^b-1 residue of the S-D result X_iY. The residue digit for the result of a complete multiplication XY is obtained by summing the PD residues the same way as for the MS algorithm. An independent check may be obtained by also directly computing the product residue digit from the residue digits Y^* and X^* in the same manner as the PD residue digit for X_i and Y^*, replacing X_i *with* X^*.

Reconversion "RS" (one operand): The algorithm reconverts S-D digits to conventional digits of the same radix, using digit ("one's" or "nine's") complement for negative operands [AVIZ 70, p. 737]. The residue digit is converted separately, using a single ABE in the RS mode and an "end-around borrow" or "eob" (connecting Bϕ to BI). For example (given r=16 and a=9), the residue digit $\bar{1}(=\bar{16}+15)$ is converted to 15-1 = 14, since the $\bar{16}$ causes the "eob". Similarly, the residue digit $\bar{9}(=\bar{16}+7)$ is converted to 7-1 = 6, etc.

Conventional-Input Modification "CI" [AVIZ 70, p. 738]: This requires that the conventional operand should be encoded with a modulo 2^b-1 (or 10^c-1) conventional residue. The residue is accepted and the specified operation is performed within the ABE that performs the operation on the residue digits.

4. Alternate Approaches to Residue Encoding and Checking

The encodings defined previously employ only one residue digit for an entire k-digit S-D operand. While this minimizes the cost of encoding, it may be inconvenient in variable-precision operations that generate the most significant digits of the results first [AVIZ 62] and that are "chained", executing further operations on high-significance digits of an intermediate result X even before the lower-significance digits become available. The Serial Checking algorithm that computes $(2^b-1)|X$ is completed only after all digits of X have obtained and the residue digit X^* is then computed and compared to $(2^b-1)|X$ to test for the presence of an error.

An error indication requires the cancellation of all results that have used at least one digit of X. The cancellation must reach k+3 digit levels downstream in the chain and identify all potentially erroneous results. Two solutions may be applied to shorten the "span" of the cancellation that must follow an error indication : (a) the **segmentation** of operands into check segments, and (b) **single-digit encoding** that employs a checking element within each ABE unit that performs single-digit operations.

Segmentation divides the k-digit operand into check segments of p digits length each and attaches one residue digit to the right end of each check segment, rather than using one residue digit at the end of the entire operand. The cancellation span is reduced to p+3 digit levels downstream in the chain. Furthermore, error detection effectiveness in the case of repeated-use faults may be increased because of the shorter length of the segment being checked. The cost of segmentation consists of the extra time and storage required by the proliferation of residue digits. The ABE structure remains unchanged, and all algorithms described in sections 2 and 3 above remain applicable to each check segment and its residue digit.

Single-Digit Encoding appears most suitable for VLSI-implemented ABE units that can execute the ABE algorithms for digits of S-D representations with relatively large radices, such as $r=2^8$, $r=2^{12}$, $r=10^3$, or even greater values. Here each individual S-D representation digit carries its residue digit modulo (r^*-1), where:

$r^* = 2^q$ *when* $r = 2^b$, and $b = kq$ $(q \geqslant 2)$

$r^* = 10^q$ *when* $r = 10^c$, and $c = kq$ $(q \geqslant 1)$

The internal structure of the ABE is modified to include a **checking element.** The main body of the ABE carries out the radix r S-D algorithms on the input digits (radix r). The separate checking element computes the radix r^* residue digit result for the specified ABE algorithm, then executes the Parallel modulo (r^*-1) Checking algorithm on the radix r result, and finally performs the test for an error condition within the self-contained ABE. A test for errors within the input digits may also be performed as a preliminary operation, using the same

Parallel modulo (r^*-1) Checking algorithm. Self-checking or morphic logic design is needed for the critical parts of the checking logic within the ABE.

The single-digit encoding approach is an extension of the segmentation concept, treating each digit of the radix $r = 2^b$ (or 10^c) representation similarly to a k-digits long segment of the radix 2^q (or 10^q) representation that is checked by one modulo 2^q-1 (or 10^q-1) residue digit. For example, $r = 256$ and $r^* = 4$; $r = 2^{12}$ and $r^* = 16$; $r = 1000$ and $r^* = 10$ are potentially effective choices. The evident advantage of this approach is the pinpointing of the error to the single ABE unit.

5. Conclusion

The results of this paper demonstrate that error-detecting codes of the residue class can be effectively adapted for the checking of signed-digit arithmetic. The residue is represented as a single digit in the same signed-digit form as the digits of the operand and is fully compatible with the previously developed concepts of signed-digit number representations [AVIZ 61], [AVIZ 62]. Generally, two or more digits can be used to represent residues for larger moduli, such as two radix 16 digits for the modulo 255 residue.

The research has shown that the same universal Arithmetic Building Element ABE that was previously developed for signed-digit arithmetic [AVIZ 70] is fully suitable for the execution of residue-digit arithmetic and for the checking algorithm, i.e., the computing of the residues of signed-digit numbers. It is concluded that checked signed-digit arithmetic units of any desired complexity can be implemented using only one type of building block - the ABE or its more advanced successors.

6. REFERENCES

[AVIZ 61] Avizienis, A. "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. on Electronic Computers,* EC-10: 389-400, 1961.

[AVIZ 62] Avizienis, A. "On a Flexible Implementation of Digital Computer Arithmetic," *Information Processing 1962, (Proc. IFIP Congress 1962),* Popplewell, C.M., ed., North Holland Publishing Co., Amsterdam, 1963, p. 664-670.

[AVIZ 70] Avizienis, A., Tung, C. "A Universal Arithmetic Building Element (ABE) and Design Methods for Arithmetic Processors," *IEEE Trans. on Computers,* C-19: 733-745, August 1970.

[AVIZ 71a] Avizienis, A. "Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design," *IEEE Trans. on Computers,* C-20: 1322-1330, November 1971.

[AVIZ 71b] Avizienis, A., et al, "The STAR (Self-Testing-And-Repairing) Computer: An investigation of the Theory and Practice of Fault-Tolerant Computer Design," *IEEE Trans. on Computers,* C-20: 1312-1321, November 1971. Reprinted in *Best Computer Papers of 1971,* L. Petrocelli, ed., Auerbach Publishers, 1972, p. 165-185.

[AVIZ 79] Avizienis, A. "A Study of Techniques for Concurrent Arithmetic Error Detection in Array Processors", *Technical Report* to NASA Ames Research Center, January 15, 1979.

[PARH 73] Parhami, B., Avizienis, A. "Application of Arithmetic Error Codes for Checking of Mass Memories," *Digest of the 1973 Int. Symposium on Fault-Tolerant Computing,* p. 47-51, June 1973.

[PARH 78] Parhami, B., Avizienis, A. "Detection of Storage Errors in Mass Memories Using Low-Cost Arithmetic Codes," *IEEE Trans. on Computers,* C-27: 302-308, April 1978.

[TUNG 70] Tung, C., Avizienis, A., "Combinational Arithmetic Systems for the Approximation of Functions," *AFIPS Conf. Proc.,* (1970 Spring Joint Computer Conf., Atlantic City, NJ), 36: 95-107.