A. Günther et al. (eds.), International Computing Symposium 1973

FAULT-TOLERANT COMPUTING BY USING RESIDUE NUMBER SYSTEMS

Ferruccio Barsi and Piero Maestrini

Istituto di Elaborazione dell'Informazione del CNR
Pisa, Italy

The error detecting and correcting properties of Redundant Residue Number Systems (RRNS) are discussed, taking into consideration single residue digit and single bit errors. Error detecting and correcting procedures are presented, also yielding additive overflow detection as a byproduct. The fault-effectiveness of the RRNS is analized and an approach to introduce reconfiguration and standby spare organization in processors using RRNS is presented.

1. INTRODUCTION

Given a set of pairwise prime, positive integers $m_1, m_2, \ldots, m_n$, called moduli, and an arbitrary integer X, the n-tuple $\{x_1, x_2, \ldots, x_n\}$, where $x_i = |X|_{m_i}$, $1 \leq i \leq n$, is defined the residue representation of X with the moduli $m_1, m_2, \ldots, m_n$. The residue x_i is called the i.th residue digit of X. As discussed by Szabo [1], the non-negative integers in the range $[0, M)$, where $M = \prod_{i=1}^{n} m_i$, are uniquely represented with the given moduli. The set N of the n-tuples of the residues modulo $m_1, m_2, \ldots, m_n$, and the invertible mapping between the integers modulo M and the set N define the nonredundant Residue Number System (RNS) of moduli $m_1, m_2, \ldots, m_n$.

A straightforward mean to introduce redundancy in an RNS consists in adding the residues $x_{n+1} = |X|_{m_{n+1}}$, $x_{n+2} = |X|_{m_{n+2}}, \ldots, x_{m+r} = |X|_{m_{n+r}}$ to the nonredundant representation of any number X, where $r \geq 1$ and the greatest common divisor of m_i and m_j, denoted (m_i, m_j), is one for each pair $i \neq j$, $1 \leq i \leq n+r$, $1 \leq j \leq n+r$. The moduli $m_{n+1}, m_{n+2}, \ldots, m_{n+r}$ and the residues $x_{n+1}, x_{n+2}, \ldots, x_{n+r}$ are called redundant moduli and redundant residue digits, respectively. The product $m_R = \prod_{j=1}^{r} m_{n+j}$ will be referred to as redundant product. Given a Redundant Residue Number System (RRNS) of moduli $m_1, m_2, \ldots, m_n, m_{n+1}, \ldots, m_{n+r}$, the (n+r)-tuples representing integers in the range $[0, M)$, or in the range $[M, Mm_R)$, are called legitimate, or illegitimate, respectively. The range $[0, M)$ is called legitimate range, while $[M, Mm_R)$ is called illegitimate range. If two legitimate numbers, X and Y, are added, the sum S=X+Y falls in the range $[0, 2M-1)$; if $S \geq M$ an overflow occurs. Then, the range $[M, 2M-1)$ will be referred to as overflow subrange. The (n+r)-tuples representing numbers in $[2M-1, Mm_R)$ cannot be generated when transmitting or adding legitimate numbers, unless an error occurs. Then, the range $[2M-1, Mm_R)$ will be referred to as error subrange.

Relative numbers are most naturally represented as $|X|_M$; i.e., by the "complement to the range" notation for negative numbers. Such a representation is unique for integers in $[-M/2, M/2)$; however, the sign is dependent on the value of all the residue digits, as shown by Szabo [2]. Since the nonnegative numbers are represented in the range $[0, M/2)$, while negative numbers are represented in $[M/2, M)$, the sign detection involves a magnitude comparison; i.e., a relatively complicated and time consuming operation. If the RNS is redundant, the sign is easily made explicit, while preserving the properties of the complement representations. In fact, assume that the numbers X in the range $[-M/2, M/2)$ are first encoded as $(X) = 2X$, and then represented as $|(X)|_M$. If the legitimate range M is odd, $|(X)|_M$ is even if X is nonnegative, while $|(X)|_M$ is odd if X is negative. Then, the sign is immediately detected from the residue $||(X)|_M|_2$. If one of the redundant moduli, say m_{n+j}, is even, then $||(X)|_M|_2 = |||(X)|_M|_{m_{n+j}}|_2$; i.e., the number is recognized to be nonnegative if and only if the residue $||(X)|_M|_{m_{n+j}}$ is even. Observe also that such a representation of relative numbers is closed under addition and subtraction.

Example 1. Given the RRNS of moduli $m_1=5$, $m_2=7$, $m_3=13$, $m_4=8$, where $m_4=8$ is the redundant modulus and the legitimate range is M=455, the positive number 38 is represented as $76=\{1,6,11,4\}$. The negative number -38 is represented as $|-76|_{455}=$ $= 379 =\{4,1,2,3\}$. Here, examining the residue digit modulo $m_4=8$ is sufficient for sign detection.

2. RESIDUE ARITHMETIC

In Residue Number Systems the residue digit modulo m_i of the sum, difference or product is obtained by adding, subtracting or multiplying the i.th residue digits of the operands; i.e., carries and borrows are suppressed and single precision integer multiplication is carried out in a single step. Because of this property, addition, subtraction and multiplication are called modular operations. Of course, the modularity of the fundamental operations influences the logical design of the arithmetic processor, where the different residue digits are separately processed by dedicated modules, the modules being most naturally realized by table look-up techniques. This choice enhances the speed and determines a regular structure, well suitable for large scale integration. The number of entries of each table (m_i^2 for two-operand operation) is compatible with the present ROM technology for any reasonable value of the modulus. Detecting an additive overflow is a lengthy operation in nonredundant RNS (Banerji[3]). This severe drawback is easily overcome in RRNS, provided that every addition is checked for possible errors, since the overflow detection is a byproduct of the error detecting or correcting procedures discussed in the next section.

Arithmetic processors using RNS are also requested to perform a number of module-interacting, or intermodular, operations. The basic intermodular operation is the mixed-radix conversion, allowing magnitude evaluation of numbers represented in an RNS. An elementary procedure for mixed-radix conversion, requiring 2(s-1) modular operations in the worst case, where s is the number of moduli of the given RNS, is presented by Szabo[1]. Here, interaction among moduli consists in the fact that one of the operands of some modular operations coincide with the residue representation, in the given RNS, of a residue digit, x_i, of the other operand. More advanced procedure is found in Hastings[4]. Other fundamental module-interacting operations are fractional multiplication and division. A relatively efficient method for fractional multiplication has been discussed by Watson[5] and Hastings [4]. In this method, the operands X and Y are first expressed as $X=\left[X/m_p\right]m_p+|X|_{m_p}$ and $Y=\left[Y/m_q\right]m_q+|Y|_{m_q}$, where m_p is the product of an appropriate subset of nonredundant moduli and $m_q = M/m_p$ is the product of the remaining nonredundant moduli. Then, the coefficients $\left[X/m_p\right]$, $|X|_{m_p}$, $\left[Y/m_q\right]$, and $|Y|_{m_q}$ are multiplied termwise and the partial products are added, controlling the additive overflows. Division may be reduced to a sequence of multiplications and comparisons, through the use of iterative trial-and-error methods, described by Szabo[1] and Hastings[4]. A module-interacting operation of great importance in the error detecting and correcting procedures presented in this paper is the m_i-projection computation. Given an integer X, represented in the RNS (either redundant or nonredundant) of moduli $m_1, m_2, \ldots, m_s$, where $M=\prod_{i=1}^{s} m_i$, by the s-tuple $\{x_1, x_2, \ldots, x_s\}$, the number $X_i=|X|_{M/m_i}$ is called the m_i-projection of X. Since X_i is an integer in the range $[0, M/m_i)$, it is completely defined by the (s-1)-tuple $\{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_s\}$. Thus, representing X_i in the given RNS implies a base extension; i.e., a procedure determining the residue digit x_i' of X_i as a function of the (s-1)-tuple $\{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_s\}$. An elementary method for base extension (Szabo [1]) requires 2s-1 modular operations in the worst case.

3. ERROR DETECTION AND CORRECTION IN RRNS

RRNS provide an effective mean to detect or correct errors occurring in data transmission and arithmetic processing. Different approach to error detection and correction in RRNS have been published by Watson[5], Rao[6], Mandelbaum[7], Sik-Song [8], and Barsi[9]. The purpose of this section is to present an unified approach to additive overflow detection and error detection or correction.

The error classes being considered for detection or correction include single residue digit or single bit errors. The procedures for detecting or correcting multiple residue digit errors are a straightforward extension of the theory being presented and will be omitted for the sake of brevity.

3.1 - Single residue digit error detection

The conditions under which additive overflow and single residue digit errors are concurrently detectable are stated by the following theorem (Barsi [10]), whose proof is omitted.

Theorem 1.

Given an RRNS of moduli $m_1, m_2, \ldots, m_n, m_{n+1}, \ldots, m_{n+r}$, any single error affecting an arbitrary number X, either legitimate or in the overflow subrange, is detectable if and only if the following inequalities hold for every i, $1 \leqslant i \leqslant n$:

$$r \geqslant 2 \quad ; \quad m_R \geqslant (2 - 1/M)\, m_i \qquad (1)$$

If inequalities [1] hold, additive overflow and single residue digit errors are detected by the following procedure, requiring a single module-interacting operation.

Procedure 1. Given the number $\bar{X}$ to be tested, its magnitude is evaluated by a mixed-radix conversion. Three cases are possible:
a) $\bar{X}$ is in the range [0, M). Then $\bar{X}$ is recognized to be legitimate; i.e., neither a single residue digit error nor an additive overflow has occurred;
b) $\bar{X}$ is in the range [M, 2M-1). Then $\bar{X}$ is recognized to be a number in overflow;
c) $\bar{X}$ is in the range [2M-1, Mm_R). Then $\bar{X}$ is recognized to be a number in error, originating either from a legitimate number or from a number in overflow.

Example 2. Consider the RRNS of moduli 11, 13, 17, 5, 7, where 5 and 7 are the redundant moduli. Since M=2341 and m_R=35, inequalities [1] hold. Given the legitimate number X=137={5, 7, 1, 2, 4} assume that the first residue digit is altered by an error, thus generating the number $\bar{X}$ = ={3, 7, 1, 2, 4}. Through a mixed-radix conversion, the magnitude of $\bar{X}$ is evaluated as 7735, and the error is detected since $\bar{X} > 2M-1$.

3.2 - Single residue digit error correction

The theory leading to single residue digit error correction, while concurrently detecting additive overflow, is derived as an extension of the matter discussed in a paper by Barsi [9]. The proof of the following theorem is again omitted for the sake of brevity.

Theorem 2.

Given an RRNS of moduli $m_1, m_2, \ldots, m_n, m_{n+1}, \ldots, m_{n+r}$, any single error affecting an arbitrary number X, either legitimate or in the overflow subrange, is correctable if and only if the following inequalities hold for each $i \neq j$, $1 \leqslant i \leqslant n+r$, $1 \leqslant j \leqslant n+r$:

$$r \geqslant 3 \quad ; \quad m_R \geqslant (2 - 1/M)\, m_i m_j \qquad (2)$$

If conditions [2] hold, the following procedure, based upon m_i-projection evaluation, allows single residue digit error correction and additive overflow detection to be concurrently performed.

Procedure 2. Given a number $\bar{X}$, and taken i=1, the m_i-projection $\bar{X}_i$ is computed and its magnitude is compared with 2M-1. As shown by Barsi [9], an elementary procedure for the combined computation and magnitude evaluation of $\bar{X}_i$ requires 2(n+2r) modular operations in the worst case. Two cases are possible:
a) if $\bar{X}_i \geqslant 2M-1$ and $i < n+r$, the procedure is iterated for i=i+1. If $\bar{X}_i \geqslant 2M-1$ and i=n+r, $\bar{X}$ is recognized to be a number in error, generated by a multiple error, and error correction is impossible;
b) if $\bar{X}_i < 2M-1$, the procedure stops and the i.th residue digit is recognized to be in error; $X=\bar{X}_i$ is the correct number. Further, if $\bar{X}_i$ is illegitimate (i.e., $M \leqslant \bar{X}_i < 2M-1$), the number X is recognized to be in overflow. Observe that the procedure keeps its validity if $\bar{X}$ is correct, since in this case necessarily $0 \leqslant \bar{X} < 2M-1$ and $\bar{X}_i < 2M-1$ for each i. Thus, the procedure stops after the first iteration. If $\bar{X}$ is a number in overflow, detection is guaranteed by b).

Example 3. Given the RRNS of moduli m_1 = =11, m_2=13, m_3=19, m_4=7, m_5=8, m_6=9, where m_4, m_5, m_6 are the redundant moduli, inequalities [2] are satisfied and $M=m_1m_2m_3=2717$, $m_R=m_4m_5m_6=504$. Let us consider the number X={3,4,17,2,0,4}=4216, in the overflow subrange, and suppose that an error affects the third residue digit which becomes 0. Then, the number $\bar{X}$={3,4,0,2,0,4}= 580 792 is generated. Systematic application of procedure [2] shows that:
$\bar{X}_1$= 82 840 > 2M-1,
$\bar{X}_2$= 54 112 > 2M-1,
$\bar{X}_3$= 4 216 < 2M-1.
Since $\bar{X}_3 < 2M-1$, the procedure stops and $X=\bar{X}_3$ is the correct number (in overflow).

3.3 - Single bit error detection and correction

Although error detecting and correcting properties of RRNS are most naturally derived for error classes related to residue digit multiplicity, an approach based on appropriate choice of the binary encoding of residue digits, first introduced by Barsi [9], allows definition of RRNS of relatively small redundancy, being able to detect or correct single bit errors.

In fact, assume that c_i^j and c_i^k are two words in the binary code for the i.th residue digit, and let x_i^j and x_i^k be the residues encoded by c_i^j and c_i^k. The error altering x_i^j in x_i^k is defined as $e_i^{j,k}=|x_i^k - x_i^j|_{m_i}$. The integer $p_i^{j,k}$ in the range $[0,m_i)$, uniquely defined by the congruence $e_i^{j,k} \equiv p_i^{j,k} \cdot M m_R/m_i \pmod{m_i}$, is called error parameter corresponding to $e_i^{j,k}$. If the Hamming distance of c_i^j and c_i^k is one, the error $e_i^{j,k}$ is a single bit error and the error parameter $p_i^{j,k}$ is called a single bit error parameter of the residue digit modulo m_i. The capability of detecting a given error, whose parameter is p_i is stated by the following theorem (Barsi [10]).

Theorem 3.

Given an RRNS of moduli $m_1, m_2, \ldots, m_n, m_{n+1}, \ldots, m_{n+r}$, the error whose parameter is p_i is detectable when affecting the i.th residue digit of any number X, either legitimate or in the overflow subrange, if and only if the following inequality holds:

$$m_R \geq (2 - 1/M) \frac{m_i}{|\pm p_i|_{m_i}} \qquad (3)$$

Similarly, the condition under which a given error is correctable, when affecting a number either legitimate or in overflow, is derived as an extension of the results published by Barsi [9], as stated by the following theorem, whose proof is omitted for the sake of brevity.

Theorem 4.

Given an RRNS of moduli $m_1, m_2, \ldots, m_n, m_{n+1}, \ldots, m_{n+r}$, the error whose parameter is p_i is correctable when affecting the i.th residue digit ($1 \leq i \leq n+r$) of any number X, either legitimate or in the overflow subrange, if and only if the following inequality

$$m_R \geq (2 - 1/M) \frac{m_i m_j}{|\pm p_i m_j|_{m_i}} \qquad (4)$$

holds for every $j \neq i$, $1 \leq j \leq n+r$.

Inequality [3], or [4], defines, for a given RRNS and for each m_i ($1 \leq i \leq n+r$), a set P_i of error parameters corresponding to detectable, or correctable, errors. If a binary code is found for each residue digit, such that the set P_i coincides with, or includes, the set of the single bit error parameters of the residue digit modulo m_i ($1 \leq i \leq n+r$), then the RRNS under consideration is able to detect, or correct, the single bit errors. Examples of such encodings are given by Barsi [9-10]. Once single bit error detecting, or correcting, RRNS have been determined, it is easily seen that procedure 1 , or procedure 2 , is suitable for error detection, or correction, and simultaneously provides additive overflow detection.

3.4 - Checking logical operations

Checking logical operations, such as bit-by-bit AND, OR and Exclusive OR, denoted $\wedge$, $\vee$, and $\oplus$ in the following, is not straightforward in RRNS, since residue codes are not closed under such operations. However, errors occurring in logical operations can be detected observing that the following relations hold for each residue digit, x_i and y_i, of the operands X and Y (Monteiro [11]):

$$x_i + y_i = (x_i \wedge y_i) + (x_i \vee y_i)$$

$$x_i + y_i = (x_i \oplus y_i) + 2(x_i \wedge y_i).$$

Given a single residue digit error detecting RRNS, and assumed, for example, that the logical operation to be executed is a bitwise AND or a bitwise OR, extended to all nonredundant residue digits, the test can be organized as follows.

- The functions $x_i \wedge y_i$ and $x_i \vee y_i$ are generated for each $i \leq n$ and interpreted as the nonredundant residue digits (since $x_i \vee y_i$ may exceed m_i, it need to be accepted as equivalent to $|x_i \vee y_i|_{m_i}$) of two integers, denoted $X \wedge Y$ and $X \vee Y$. The redundant digits of $X \wedge Y$ and $X \vee Y$ are computed by a base extension procedure.
- The integers $X \wedge Y$ and $X \vee Y$ are added, controlling the overflow, and the sum is compared with X+Y. Since each operation is checked except for the logical operations, any single residue digit error is detected.

4. FAULT EFFECTIVENESS

The error detecting or correcting properties of any error code provide a measure of the value effectiveness of the code; i.e., a listing of the error values whose detection or correction is guaranteed. For example, consider an RRNS satisfying the hypothesis of theorem 2 and assume that X is a correct number and $\bar{X}$ a number originating from X by effect of an error $e=|\bar{X}-X|_{Mm_R}$. Then, in an equivalent formulation, theorem 2 states that the error e is correctable if its magnitude satisfies to the relation $e= p_i\ Mm_R/m_i$, for some $1 \leqslant p_i < m_i$, $1 \leqslant i \leqslant n+r$. The value effectiveness of a code is, of course, independent of the logic structure of the processor using the code. However, the ultimate purpose of using an error code is detecting or masking a given class of logic faults, defined as the deviation of a binary variable from its "perfect" value. Evaluating the fault-effectiveness (Avizienis [12]) of a given error code; i.e., listing the logic faults being detected or masked by the code, requires the general organization and the logic diagram of the processor to be known and the fault-error relationship to be analized.

Determining the fault effectiveness of RRNS is a relatively easy task, because of the modular organization of the processor, as dicussed in Section 2. An analysis of the fault coverage guaranteed by the single residue digit error detecting or correcting RRNS shows that value effectiveness and fault effectiveness are well matched; i.e., that the error classes being either detected or corrected correspond to important classes of logic faults being either detected or masked. This essentially depends from the fact that any fault, or set of faults, occurring in a single memory or processing module has a local effect; i.e., determines a single residue digit error. The fault, or the set of faults, will be detected or masked by the previously discussed error detecting or correcting procedures, assumed that each elementary operation (data transmission, memory read out or modular operation) is individually checked. Checking a sequence of elementary operations will also allow detection or masking of any set of permanent faults, occurring in one or more modules dedicated to a some residue digit, provided that any other module is fault-free during the sequence. Since the module-interacting operations can be decomposed in a sequence of modular operations, coverage of any set of faults occurring in a single module is also guaranteed, if the modular operations are individually tested. Observe also that, in the hypothesis that table look-up techniques are used throughout in the processor, the same fault-error relationship holds in memory readout and in arithmetic operations, and most logic faults determine single bit errors, provided that the address decoding circuitry is appropriately protected. This explains the interest of the low redundancy single bit error detecting or correcting RRNS discussed in the previous section.

5. RECONFIGURATION

In order to briefly analize the reconfiguration capabilities of the processors using RRNS, assume, for example, that numbers are encoded in an RRNS whose redundancy is sufficient to guarantee single residue digit error correction. Then, error correcting procedure 2 ensure recovery from any error in this class, either originating from a transient or a permanent failure. If a transient fault has occurred, it has no further effect after correction and the capability of masking any other failure is unchanged. If the error originates from a permanent fault, or set of faults, localized in a memory or processing module dedicated to the i.th residue digit ($1 \leqslant i \leqslant n+r$), it will be effective during some successive elementary operation involving the use of the module. Since the error correcting capability of the code is limited to single residue digit errors, this fact will generally prevent masking of any subsequent fault localized in any module dedicated to the j.th residue digit ($j \neq i$). The obstacle is removed by assuming that any error being corrected through the use of procedure [2] is monitored and, whenever errors repeatedly affecting the i.th residue digit ($1 \leqslant i \leqslant n+r$) are recorded, the permanent failure of some module dedicated to the i.th residue digit is conjectured. Then, human intervention for repairing or replacing the faulty module is solicited. However, if the processor is a part of an on-line system, it may be required that the computation continues while maintenance is going on (although some degradation in performance can be tolerated). A simple approach to achieve reconfiguration, with degraded error control, of a processor using RRNS is suggested by the observation that in RRNS the residue digits are not hyerarchically ordered and are independently accessed and processed at the level of elementary operation. Disconnecting the modules storing or processing the i.th res=

idue digit does not prevent continuing the operation of the system, provided that an arbitrary output, e. g., zero, is assured from any disconnected module. If the active hardware is fault-free, the suppressed residue digit is recovered by the use of the error correcting procedure [2]. If a fault occurs in any active module, dedicated to the j.th residue digit, ($j \neq i$), error detection is guaranteed. In fact, if the product $m_i m_j$ is assumed to be a composite modulus, it is seen, by comparison of inequalities [1] and [2], that any pair of errors simultaneously affecting the i.th and the j.th residue digits are detectable in an RRNS satisfying the hypothesis of theorem 2. Moreover, it is easily seen that error detection is still yielded by procedure 2, case b). If the processor is unaccessible to human intervention, a standby-spare organization is necessary to replace the faulty modules. The modular organization of the processors using RRNS determines important savings in the number and the size of the spares to be used, provided that general purpose memory and processing spares are available, being able to replace any faulty memory or processing module. A complication arises in processing modules, since different residue digits require different arithmetic tables to be stored. The obstacle can be overcomed by assuming the use of "Read Mostly" memories for storing the tables, where the memory area being provided is sufficient to store the tables relative to the largest modulus. When a processing module fails, the appropriate tables are first copied in a spare; then, the specialized module is switched to replace the faulty module.

REFERENCES

[1] N.S.Szabo, R.I.Tanaka; Residue Arithmetic and its Applications to Computer Technology; 1967; Mc Graw-Hill; New York

[2] N.S.Szabo; Sign Detection in Non Redundant Residue Systems; IRE Trans. Electon. Comp.; EC-11; August 1962; 494

[3] D.K.Banerji, J.A.Brzozowski; Sign Detection in Residue Number Systems; IEEE Trans. on Comp.; C-18; April 1969; 313

[4] C.W.Hastings; Automatic Detection and Correction of errors in digital computers using residue arithmetic; Proc. 1966 IEEE-Region Six-Annual Conference; 429

[5] R.W.Watson; Error Detection and Correction and Other Residue Interacting Operations in a Redundant Residue Number System; 1965; Ph.D. Dissertation--University of California, Berkeley

[6] T.R.N.Rao; An (n,k) code for Residue arithmetic; Proc. Second Annual Princeton Conf. on Information Sciences and Systems; March 1968; 154

[7] D.Mandelbaum; Error Correction in Residue Arithmetic; IEEE Trans. on Comp. C-21; June 1972; 538

[8] S.Sik-Song Yan, Yu-Cheng Lin; Error Correction in Redundant Residue Number Systems; IEEE Trans. on Comp.; C-22; January 1973; 5

[9] F.Barsi, P.Maestrini; Error Correcting Properties of Redundant Residue Number Systems; IEEE Trans. on Comp.; C-22; March 1973; 307

[10] F.Barsi, P.Maestrini; Concurrent Detection of Additive Overflow and Arithmetic Errors in Residue Codes; to be published on "Calcolo"

[11] P.Monteiro, T.R.N.Rao; A Residue Checker for Arithmetic and Logical Operations; 1972 Intern. Symposium on Fault Tolerant Computing - Digest; Newton (Massachusetts); June 1972; 8

[12] A.Avizienis; Arithmetic Error Codes: cost and effectiveness studies in digital System Design; IEEE Trans. on Comp.; C-20; November 1971; 1322