REDUNDANT RESIDUE NUMBER SYSTEM BASED ERROR CORRECTION CODES

Lie-Liang Yang and Lajos Hanzo

Dept. of ECS, University of Southampton, SO17 1BJ, UK.
Tel: +44-703-593 125, Fax: +44-703-594 508
Email: lh@ecs.soton.ac.uk, http://www-mobile.ecs.soton.ac.uk

ABSTRACT

In this paper residue number system (RNS) arithmetic and redundant residue number system (RRNS) based codes as well as their properties are reviewed. We propose a number of applications for RRNS codes and demonstrate how RRNS codes can be employed in global communication systems, in order to simplify the associated systems by unifying the entire encoding and decoding procedure across global communication systems.

1. INTRODUCTION

RNS-based arithmetics exhibit a modular structure that leads naturally to parallelism in digital hardware implementations. The RNS has two inherent features that are attractive in comparison to conventional weighted number systems, such as for example the binary weighted number system representation. These two features are [1–4]: 1) the carry-free arithmetic and 2) the lack of ordered significance amongst the residue digits. The first property implies that the operations related to the different residue digits are mutually independent and hence the errors occurring during addition, subtraction and multiplication operations, or due to the noise induced by transmission and processing, remain confined to their original residue digits [1, 5–7]. In other words, these errors do not propagate and hence do not contaminate other residue digits due to the absence of a carry forward. The above-mentioned second property of the RNS arithmetic implies that redundant residue digits can be discarded without affecting the result, provided that a sufficiently high dynamic range is retained by the resultant reduced-range RNS system, in order to unambiguously describe the nonredundant information symbols.

As it is well known in VLSI design, usually so-called systolic architectures [1] are invoked to divide a processing task into several simple tasks performed by small, (ideally) identical, easily designed processors. Each processor communicates only with its nearest neighbour, simplifying the interconnection design and test, while reducing signal delays and hence increasing the processing speed. Due to its carry-free property, the RNS arithmetic further simplifies the computations by decomposing a problem into a set of parallel, independent residue computations.

The properties of the RNS arithmetic suggest that a redundant residue number system (RRNS) can be used for self-checking, error-detection and error-correction in digital processors. The RRNS technique provides a useful approach to the design of general-purpose systems, capable of sensing and rectifying their own processing and transmission errors. For example, if a digital receiver is implemented using the RRNS having sufficient redundancy, then errors in the processed signals can be detected and corrected by the RRNS-based decoding. Furthermore, the RRNS approach [1] is the only one, where it is possible to use the very same arithmetic module in the very same way for the generation of both the information part and the parity part of a RRNS codeword. Moreover, due to the inherent properties of the RNS, the residue arithmetic offers a variety of new approaches to the realization of digital signal processing algorithms, such as digital modulation and demodulation as well as the fault-tolerant design of arithmetic units [2–4, 7–9]. It also offers new approaches to the design of error-detection and error-correction codes. Let us first review the basic mathematical model required for the conversion of operands from the weighted number system to the RNS, or conversely, from the RNS to the weighted number system.

2. RESIDUE NUMBER SYSTEM TRANSFORM AND ITS INVERSE

For convenience, we define the residue number system transform (RNST) as the algorithm that transforms any conventional weighted number system, such as for example the natural binary coded decimal (NBCD) number system to the RNS [3, 7]. The inverse RNST (IRNST) is defined as the algorithm, which transforms the RNS to the weighted number system, an issue to be elaborated on below.

Let X be a non-negative integer number in the range of $0 \leq X < \varrho^n$, where ϱ is a base and X is represented in the weighted number system as $X = \{b_{n-1}\ b_{n-2}\ \ldots\ b_2\ b_1\ b_0\} = \sum_{j=0}^{n-1} b_j \varrho^j$, where $b_j \in \{0, 1, \ldots, \varrho - 1\}$. Specifically, a weighted binary representation is considered, when $\varrho = 2$.

Let $\{m_1, m_2, \ldots, m_u\}$, the set of so-called moduli involved in the RNS, be a set of pairwise relative prime numbers, where no two moduli have a common integer divisor greater than 1. If $M = \prod_{i=1}^{u} m_i \geq \varrho^n$, then the RNST is realized by uniquely representing X - which was defined in

This work has been performed in the framework of the Pan-Euroepan IST project IST-1999-12070 (TRUST), which is partly funded by the European Union. The authors would like to acknowledge the contributions of their colleagues, although the views expressed are those of the authors.

The financial support of the EPSRC, Swindon, UK is also gratefully acknowledged.

the range of $0 \leq X < \varrho^n$ - as a u-tuple residue digit sequence $(r_1, r_2, \ldots, r_u)$, where $r_i = X \pmod{m_i} = X - \lfloor \frac{X}{m_i} \rfloor m_i$ for $i = 1, 2, \ldots, u$, $\lfloor z \rfloor$ denotes the largest integer not exceeding z, and $0 \leq r_i \leq m_i - 1$. Furthermore, $[0, M-1]$ is defined as the dynamic range of the RNS and any integer X in this range is uniquely represented by a u-tuple residue sequence, which is expressed as:

$$X \Leftrightarrow (r_1, r_2, \ldots, r_u), \quad 0 \leq X < M,$$
$$r_i = X \pmod{m_i}, \quad i = 1, 2, \ldots, u. \tag{1}$$

Assuming that the integers X_1 and X_2 have RNS representations of $X_1 = (r_{11}, r_{12}, \ldots, r_{1u})$ and $X_2 = (r_{21}, r_{22}, \ldots, r_{2u})$, respectively, then $X_1 \bullet X_2$, where $\bullet$ denotes addition, subtraction or multiplication, yields another unique residue sequence X_3, provided that X_3 remains within the dynamic range of $[0, M)$. The arithmetic operations over the RNS can be carried out on a residue-by-residue basis, which can be expressed as:

$$X_1 \bullet X_2 \Leftrightarrow [(r_{1i} \bullet r_{2i}) \pmod{m_i}]_{i=1}^{u}, \tag{2}$$

where on the left the operation $\bullet$ represents ordinary addition, subtraction or multiplication of two integers, and on the right it represents the same operations performed on the basis of the appropriate residue digits r_{1i} and r_{2i}, with respect to their corresponding modulus m_i. This allows us to convert binary arithmetic operations carried out with large integers to residue arithmetic operations involving a higher number of smaller residue digits, where the operations can be executed in parallel fashion and there is no carry forward between the residue digits. The lack of carry forward between the residue digits prevents the propagation of errors between residue digits.

In contrast to the RNST, the IRNST is defined as the algorithm that transforms the operands from the RNS domain to the conventional weighted number system representation. A well-known implementation of the IRNST is the so-called Chinese remainder theorem (CRT) [5]. According to the CRT, for any given u-tuple $(r_1, r_2, \ldots, r_u)$, where $0 \leq r_i < m_i$ for $i = 1, 2, \ldots, u$ there exists one and only one integer X such that $0 \leq X < M$ and $r_i = X \pmod{m_i}$. It can be shown that the numerical value of X can be computed by [5]:

$$X = \sum_{i=1}^{u} r_i T_i M_i \pmod{M}, \tag{3}$$

where $M_i = M/m_i$ and the integers T_i are computed a *priori* by solving the so-called congruence $T_i M_i = 1 \pmod{m_i}$. The coefficients T_i are also often referred to as the multiplicative inverses of M_i.

The CRT is a classical algorithm for the implementation of the IRNST. However, the real-time implementation of the CRT is not practical, since it requires modular operations with respect to a large integer M. In order to avoid processing large integers, especially in the context of error control invoking the RRNS, a frequently used approach is the so-called base extension (BEX) operation in conjunction with the mixed radix conversion approach [7].

3. REDUNDANT RESIDUE NUMBER SYSTEM AND RRNS ENCODING

If a residue number system is designed not only for the representation of data, but also for the protection of data, usually we design the RNS using so-called redundant moduli, in order that the system has the capability of self-checking, error-detection and error-correction [1]. In this case the operand X is limited to the so-called information dynamic range of $\left[0,\ M = \prod_{i=1}^{v} m_i\right)$, where $v \leq u$, and $m_1, m_2, \ldots, m_v$ are referred to as the information moduli, while $m_{v+1}, m_{v+2}, \ldots, m_u$ are the so-called redundant moduli. We express the product of the redundant moduli as $M_R = \prod_{j=1}^{u-v} m_{v+j}$. The interval $[0, M)$ constitutes the legitimate range of the operand X, and the interval $[M, MM_R)$ is the associated so-called illegitimate range. Any integer belonging to the legitimate range will be labelled as legitimate and those belonging to the illegitimate range as illegitimate, since it does not represent a legitimate number or operand, directly accruing from the nonredundant information-bearing residue digits.

An important property of the RRNS is that an integer represented by the residues of the RRNS can be recovered by any group of v number of moduli and their corresponding residue digits. This property constitutes the basis in the design of RRNS based error-detection and error-correction schemes.

A RRNS having v number of information moduli and $u-v$ number of redundant moduli is denoted as a RRNS(u, v) code. The operation associated with generating the redundant residue digits of an integer operand X can also be viewed or interpreted, as an encoding operation generating the parity residue digits of a RRNS(u, v) code. The RRNS encoding operation can be implemented based on a group of codewords having typical integer values, by invoking exclusively addition operations. In order to augment this statement, below we provide an example showing the associated groups of codewords for the RRNS(7,3) code having information moduli of $m_1 = 4,\ m_2 = 5,\ m_3 = 7$ and redundant moduli of $m_4 = 9,\ m_5 = 11,\ m_6 = 13,\ m_7 = 17$. The legitimate range of this code is [0,139], since $4 \times 5 \times 7 = 140$. Table 1 portrays a range of typical decimal integer messages, which are based on the integers '1, 2, 5, 10, 20, 50, 100' often used in monetary systems. It is well known that on the average any integer in the legitimate range of [0,139] can be expressed by the sum of a subset of the lowest possible number of the specific integers given above. In Table 2 we summarised a range of binary integer numbers and their corresponding RRNS codewords.

Example 1 *For the encoding of the decimal integer* $X = 123$, *since* X *can be expressed as* $123 = 100 + 20 + 2 + 1$, *the RRNS codeword of* X *can be simply obtained from the codewords corresponding to* X_7, X_5, X_2 *in Table 1 and* X_1 *on the basis of modulo addition. It can be readily shown that the codeword of* $X = 123$ *is* $(3, 3, 4, 6, 2, 6, 4)$. *Similarly, for a binary integer* $X = 1011011$, *the associated codeword is simply constituted by the corresponding modulo addition of* X_7, X_5, X_4, X_2 *and* X_1 *of Table 2, where the subscripts* $7, 5, 4, 2, 1$ *are simply the indices of the binary* 1 *positions in* X. *The associated codeword of* $X = 1011011$ *is hence* $(3, 1, 0, 1, 3, 0, 6)$.

Decimal message X	Nonredundant residue digits			Redundant residue digits			
	r_1	r_2	r_3	r_4	r_5	r_6	r_7
$X_0 = 0$	0	0	0	0	0	0	0
$X_1 = 1$	1	1	1	1	1	1	1
$X_2 = 2$	2	2	2	2	2	2	2
$X_3 = 5$	1	0	5	5	5	5	5
$X_4 = 10$	2	0	3	1	10	10	10
$X_5 = 20$	0	0	6	2	9	7	3
$X_6 = 50$	2	0	1	5	6	11	16
$X_7 = 100$	0	0	2	1	1	9	15

Table 1: RRNS codewords of some typical decimal integer messages X in the RRNS with moduli $m_1 = 4$, $m_2 = 5$, $m_3 = 7$, $m_4 = 9$, $m_5 = 11$, $m_6 = 13$ and $m_7 = 17$, where $r_i = X \ (mod\ m_i)$ and $M = 4 \times 5 \times 7 = 140$.

Binary message X	Nonredundant residue digits			Redundant residue digits			
	r_1	r_2	r_3	r_4	r_5	r_6	r_7
$X_0 = 0000000$	0	0	0	0	0	0	0
$X_1 = 0000001$	1	1	1	1	1	1	1
$X_2 = 0000010$	2	2	2	2	2	2	2
$X_3 = 0000100$	0	4	4	4	4	4	4
$X_4 = 0001000$	0	3	1	8	8	8	8
$X_5 = 0010000$	0	1	2	7	5	3	16
$X_6 = 0100000$	0	2	3	5	10	6	15
$X_7 = 1000000$	0	4	1	1	9	12	13

Table 2: RRNS codewords of some typical binary messages X in the RRNS with moduli $m_1 = 4$, $m_2 = 5$, $m_3 = 7$, $m_4 = 9$, $m_5 = 11$, $m_6 = 13$ and $m_7 = 17$, where $r_i = X \ (mod\ m_i)$ and $M = 4 \times 5 \times 7 = 140$.

4. ERROR-DETECTION AND ERROR-CORRECTION IN RRNS

In this section, we briefly summarise some properties of the associated error-detection and error-correction procedures in the context of the RRNS. Error-detection/correction decoding of RRNS(u, v) codes has been discussed in depth in [3, 5–7]. Let us invoke two simple examples, in order to gain insight into the error-detection and error-correction mechanism of the RRNS.

Example 2 *Let us consider the moduli* $3, 4, 5, 7$*, where* $3, 4$ *and* 5 *are the information moduli and* 7 *is the redundant modulus. The information dynamic range is* $[0, M = 3 \cdot 4 \cdot 5) = [0, 60)$*. Upon considering an integer decimal message of* $X = 21$*, the corresponding residue values are* $X = (0, 1, 1, 0)$*. If there is an error in the RNS representation due to transmission or processing, for example* r_3 *is changed from* 1 *to* 3*, then the received RNS representation becomes* $(0, 1, \tilde{3}, 0)$*. Upon following the general approach of the CRT in Eq.(3) and using the first three residue digits and their moduli, we obtain:*

$M_1 = 4 \times 5 = 20,\ M_2 = 3 \times 5 = 15,\ M_3 = 3 \times 4 = 12,$
$T_1 = 2,\ T_2 = 3,\ T_3 = 3,$
$X = [0 \times 2 \times 20 + 1 \times 3 \times 15 + 3 \times 3 \times 12] \bmod 60$
$= 33.$

However, where $X = 33\ (mod\ 7) = 5 \neq r_4 = 0$*, and we can conclude that there were errors in the RNS representation. Therefore, upon designing the RNS using one redundant modulus, the residue digit error of* r_3 *can be detected.*

Example 3 *Let us now invoke an additional redundant modulus, namely* 11 *in the above example, which results in a total of two redundant moduli, namely* 7 *and* 11 *in the RRNS. Let us also consider the integer message* $X = 21$*, now having corresponding residue digits of* $X = (0, 1, 1, 0, 10)$ *and that* r_3 *is in error and it was changed from* 1 *to* 3*, i.e the received RNS representation becomes* $(0, 1, \tilde{3}, 0, 10)$*. According to the CRT's approach, the integer* X *in the range* $[0, 60)$ *can be recovered by invoking any three moduli and their corresponding residue digits, if no errors occurred in the received RNS representation. Let us now consider all possible cases and attempt to recover the integer* X *represented by* $(0, 1, \tilde{3}, 0, 10)$*, upon retaining all possible combinations of three out of five residue digits, which results in:*

$$
\begin{aligned}
(r_1, r_2, r_3) = (0, 1, 3) &\quad \Leftrightarrow \quad X_{123} = 33\ (mod\ 60),\\
(r_1, r_2, r_4) = (0, 1, 0) &\quad \Leftrightarrow \quad X_{124} = 21\ (mod\ 84),\\
(r_1, r_2, r_5) = (0, 1, 10) &\quad \Leftrightarrow \quad X_{125} = 21\ (mod\ 132),\\
(r_1, r_3, r_4) = (0, 3, 0) &\quad \Leftrightarrow \quad X_{134} = 63\ (mod\ 105),\\
(r_1, r_3, r_5) = (0, 3, 10) &\quad \Leftrightarrow \quad X_{135} = 153\ (mod\ 165),\\
(r_1, r_4, r_5) = (0, 0, 10) &\quad \Leftrightarrow \quad X_{145} = 21\ (mod\ 231),\\
(r_2, r_3, r_4) = (1, 3, 0) &\quad \Leftrightarrow \quad X_{234} = 133\ (mod\ 140),\\
(r_2, r_3, r_5) = (1, 3, 10) &\quad \Leftrightarrow \quad X_{235} = 153\ (mod\ 220),\\
(r_2, r_4, r_5) = (1, 0, 10) &\quad \Leftrightarrow \quad X_{245} = 21\ (mod\ 308),\\
(r_3, r_4, r_5) = (3, 0, 10) &\quad \Leftrightarrow \quad X_{345} = 98\ (mod\ 385),
\end{aligned}
$$

where X_{ijk} *represents the recovered result by using moduli* m_i, m_j *and* m_k *as well as their corresponding residue digits* r_i, r_j *and* r_k*. From these results we observe that* $X_{134}, X_{135}, X_{234}, X_{235}$ *and* X_{345} *are all illegitimate numbers, since their values are out of the legitimate range* $[0, 60)$*. In the remaining five cases, except for* X_{123}*, all the results are the same and equal to* 21*. Moreover, all these results were recovered from three moduli without including* m_3*, i.e. from* $X_{124}, X_{125}, X_{145}$ *and* X_{245}*, which are equal to* 21*. Hence, we might conclude that the correct result is* 21 *and that there was an error in* r_3*, which can be corrected by computing* $\hat{r}_3 = 21 \pmod 5 = 1$.

RRNS codes exhibit similar coding properties to the well-known Reed-Solomon (RS) codes [5, 7]. RRNS codes constitute a class of maximum minimum-distance separable codes. An RRNS(u, v) code - where the information dynamic range represented by the RRNS(u, v) is $[0, \prod_{i=1}^{v} m_i)$ and the RRNS code's dynamic range is $[0, \prod_{i=1}^{u} m_i)$ - has a minimum distance of $(u - v + 1)$ and hence it is capable of detecting $(u - v)$ or less residue digit errors and correct up to $\lfloor (u - v)/2 \rfloor$ random residue digit errors. An RRNS(u, v) code is capable of correcting t random residue digit errors and simultaneously detecting β $(\beta > t)$ residue digit errors, if and only if $t + \beta \leq (u - v)$. Moreover, an RRNS(u, v) code is capable of correcting t random residue digit errors and simultaneously correcting β residue erasures, provided that $2t + \beta \leq (u - v)$.

According to the carry-free property and due to the lack of weighted significance of the residue digits in the RNS arithmetic, in RRNS codes some of the channel-impaired residue digits can be discarded as an error correction measure, provided that a sufficiently high dynamic range is re-

tained by the reduced-range system, in order to unambiguously decode the result. The above statement can be augmented as follows. Let $\{m_1, m_2, \ldots, m_u\}$ be a set of u moduli of an RRNS(u, v) code, where $m_1 < m_2 < \ldots < m_u$. Furthermore, let X be the integer message, which is expressed with the aid of the residue digits as $(r_1, r_2, \ldots, r_u)$ with respect to the above moduli. If the dynamic range of X is $[0, \prod_{i=1}^{v} m_i)$, where $v \leq u$, then X can be recovered from any v out of the u number of residue digits and their relevant moduli. This property implies that d $(d \leq u - v)$ number of residue digits can be dropped before the IRNST stage, while still recovering the message X using the retained residue digits, provided that the retained residue digits are the correct, ie uncorrupted residue digits.

Moreover, if we let d be the number of discarded residue digits, where $d \leq u - v$, then, a RRNS(u, v) code is converted to a RRNS$(u-d, v)$ after d out of the u residue digits are discarded. Hence, the reduced RRNS$(u-d, v)$ code can detect up to $(u - v - d)$ residue digit errors and correct up to $\lfloor (u - v - d)/2 \rfloor$ residue digit errors. This property suggests that the RRNS(u, v) decoding can be designed by first discarding d $(d \leq u - v)$ out of the u residue digits, which is followed by RRNS$(u - d, v)$ decoding. Since the discarded residue digits and their corresponding moduli do not have to be considered in the RRNS$(u - d, v)$ decoding, the decoding procedure is therefore simplified.

5. APPLICATIONS OF RRNS CODES

In addition to the above mentioned applications of RRNS codes for self-checking, error-detection and error-correction, a useful property of the RRNS is that an RRNS codeword does not change its error-detection and error-correction characteristics after arithmetic operations, such as addition, subtraction and multiplication [7]. This property can be employed in order to support fault-tolerant signal processing. Let us invoke an example, in order to augment this property.

Example 4 *Let us consider the RRNS codes based on a RRNS using nonredundant moduli of* $m_1 = 4$, $m_2 = 5$ *and* $m_3 = 7$ *as well as redundant moduli of* $m_4 = 9$, $m_5 = 11$, $m_6 = 13$ *and* $m_7 = 17$. *Table 1 summarised a range of typical decimal integers and their corresponding RRNS codewords. Let us now consider an operation in the decimal integer field:*

$$Y = 5 \times Y_1 + Y_2 \times Y_3, \tag{4}$$

where Y_1, Y_2 *and* Y_3 *are possibly from different sources and may include the transmission and processing errors. However, due to the inherent properties of the RRNS codes, the operations in Eq.(4) can be carried out as follows. Firstly, before we evaluate Eq.(4),* Y_1, Y_2 *and* Y_3 *are first error-correction decoded. Since four redundant moduli are included, up to two residue digit errors in* Y_1, Y_2 *or* Y_3 *can be corrected. Secondly, Table 1 is used in order to evaluate Eq.(4). Thirdly, since an RRNS codeword does not change its error-detection/correction capability, when the codewords are subjected to the arithmetic operations of addition and multiplication, the resultant RRNS codeword corresponding to* Y *in Eq.(4) can be further error-correction decoded, in order to remove the errors that may have happened in the evaluation of Eq.(4). Note that no conventional codes, such as Hamming codes, BCH codes and convolutional codes have this property, since all these codes will change their structure when the associated codewords are subjected to the multiplication operation.*

Using the operand values of Table 1, let $Y_1 = X_1 = 1$, $Y_2 = X_2 = 2$ *and* $Y_3 = X_6 = 50$. *For example, the RRNS codeword corresponding to* 5 *is* $(1, 0, 5, 5, 5, 5, 5)$ *according to Table 1. It can be readily shown that the codeword corresponding to the operation in Eq.(4) is given by* $Y = 5 \times 1 + 2 \times 50 = 105$, *yielding the residue-sequence of* $(1, 0, 0, 6, 6, 1, 3)$ *- provided that no more than two residue digit errors occurred in* Y_1, Y_2 *or* Y_3. *Moreover, even if two residue digit errors occurred during the last operation of Eq.(4), and hence the above codeword changed to* $(1, \tilde{1}, 0, 6, \tilde{3}, 1, 3)$, *i.e.,* r_2 *and* r_5 *are in error, by using RRNS error-correction decoding, the value of* $Y = 105$ *can still be correctly recovered, since four redundant moduli were invoked.*

Let us now demonstrate another potential application of the RRNS codes in the context of the generic communication system depicted in Fig.1. In this framework, redundancy has to be added for the implementation of fault-tolerant computing (or signal-processing), protection of information over both wired and wireless channels, in order to achieve reliable processing and communications. Conventionally, the encoding and decoding at different stages has been treated separately. The encoded codewords of the wired channels have been historically decoded in order to remove the related redundancy, before forwarding the information to the encoder of the air interface of Fig.1. The information at the air interface is then often re-encoded using a different channel code, in order to implement reliable transmission over the associated wireless channels. Further additional encoding/decoding operations may take place at other interfaces of a global communications system, as illustrated in Fig.1.

However, with the advent of RRNS codes, the above mentioned encoding/decoding procedures can be substantially simplified, since the required redundancy of the entire global system can be jointly designed. Let us invoke an example to support this argument.

Example 5 *With reference to Table 1, let us assume that the fault-tolerant terminals use only one redundant modulus for the self-checking of the associated arithmetic units, employing for example the RRNS*$(4, 3)$ *code. Let us assume furthermore that two redundant moduli are required for the protection of information over the wired channel section of Fig.1, employing the one-residue digit error-correcting RRNS* $(5, 3)$ *code. Finally, four redundant moduli have to be employed for the protection of information over the low-reliability wireless channels, using the two-residue digit error-correction RRNS*$(7, 3)$ *code. Let* m_1, m_2, m_3, m_4, m_5, m_6 *and* m_7 *be the moduli invoked in the required RRNS, where* m_1, m_2 *and* m_3 *are the nonredundant moduli, while* m_4, m_5, m_6 *and* m_7 *are the redundant moduli. The RRNS codes protecting the entire system of Fig.1 can be designed as follows.*

Step 1*: The fault-tolerant terminal No.1 uses moduli* m_1, m_2, m_3 *and* m_4, *in order to form the RRNS*$(4, 3)$ *codewords, which are expressed as* (r_1, r_2, r_3, r_4), *where* r_1, r_2 *and* r_3 *are the nonredundant residue digits, while* r_4 *is the redundant residue digit. Following all the associated signal processing operations to be carried out by the terminal - which may involve additions, subtractions and multiplications on the basis of the RNS - and invoking self-checking,*

the fault-tolerant terminal forwards the RRNS$(4,3)$ codewords to the encoder of the wired channels. For simplicity the output codewords of the fault-tolerant terminal No.1 are still expressed as $(r_1,\ r_2,\ r_3,\ r_4)$, which will also be used throughout our further discourse.

Step 2*: With the aid of the Base Extension algorithm [2], the wired channel's encoder - namely Encoder 1 in Fig.1 - computes an additional redundant residue digit r_5 based on $(r_1,\ r_2,\ r_3,\ r_4)$ and $(m_1, m_2, m_3, m_4, m_5)$, in order to form the RRNS$(5,3)$ codewords $(r_1,\ r_2,\ r_3,\ r_4,\ r_5)$. Then, these codewords are transmitted over the wired channel in Fig.1.*

Step 3*: The wired channel's decoder - namely Decoder 1 in Fig.1 - receives the codewords $(r_1,\ r_2,\ r_3,\ r_4,\ r_5)$, which are error-correction decoded. The corrected codewords are expressed as $(r_1,\ r_2,\ r_3,\ r_4,\ r_5)$, which are forwarded to the encoder of the wireless channel.*

Step 4*: Similarly to Step 2, with the aid of the base extension algorithm [2], the wireless channel's encoder computes the additional redundant residue digit r_6 and r_7, based on $(r_1, r_2, r_3, r_4, r_5)$ and $(m_1, m_2, m_3, m_4, m_5, m_6, m_7)$, in order to form the RRNS$(7,3)$ codewords $(r_1,\ r_2,\ r_3,\ r_4,\ r_5,\ r_6,\ r_7)$. Then, these codewords are transmitted over the wireless channel of Fig.1.*

Step 5*: After the wireless channel's decoder received the codewords $(r_1,\ r_2,\ r_3,\ r_4,\ r_5,\ r_6,\ r_7)$, the codewords are error-correction decoded and the redundant residue digits r_6 and r_7 are removed from the corrected codewords. The wireless channel decoder then passes the RRNS$(5,3)$ codewords to the wired channel's encoder, namely to Encoder 3 in Fig.1. These RRNS$(5,3)$ codewords are expressed as $(r_1,\ r_2,\ r_3,\ r_4,\ r_5)$.*

Step 6*: The RRNS$(5,3)$ codewords are transmitted over the wired channel to the wired channel's decoder, namely to Decoder 3 in Fig.1.*

Step 7*: After the wired channel's decoder received the RRNS$(5,3)$ codewords $(r_1,\ r_2,\ r_3,\ r_4,\ r_5)$, error-correction decoding is invoked, in order to correct the residue digit errors and then the redundant residue digit r_5 is removed. The resulting RRNS$(4,3)$ codewords $(r_1,\ r_2,\ r_3,\ r_4)$ are then forwarded to the fault-tolerant terminal No.2.*

Step 8*: The fault-tolerant terminal No.2 finally carries out its tasks using the RNS representation of its operands in the form of the RRNS$(4,3)$ codewords, invokes self-checking, and finally outputs the recovered information.*

In the upper branch of Fig.1 the wired channel's encoder only had to compute one additional redundant residue digit, in contrast to invoking an independent encoder, as in a conventional system, where each section of the communications links is protected independently. Similarly, the wireless channel's encoder only had to compute two additional redundant residue digits, in contrast to complete reencoding in independently protected conventional systems. In the bottom branch of Fig.1, the encoder of the wired channel, and the encoder of the fault-tolerant terminal No. 2 - which is not shown in the figure - can be totally removed, while in conventional independently protected systems all these encoders would have to be included. Based on these facts, we can conclude that in conjunction with appropriate joint system design using RRNS codes, the complexity of the error-correction/detection sub-systems in global telecommunication systems can be decreased. Our future work in this field will be based on designing systems invoking the principles proposed in this contribution.

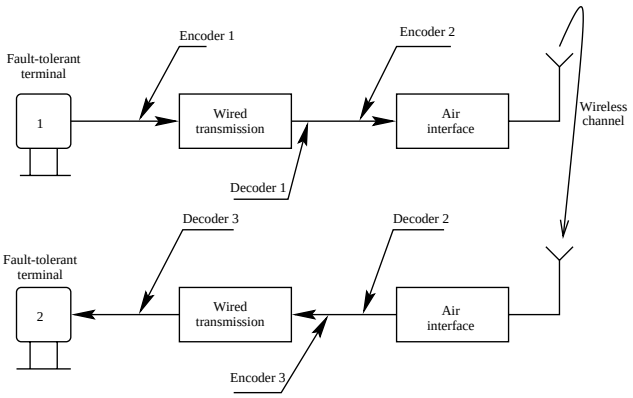

Figure 1: A transmission and computing system framework including fault-tolerant terminals, wired transmissions and wireless transmissions.

6. REFERENCES

[1] R. W. Watson and C. W. Hastings, "Self-checked computation using residue arithmetic," *Proceedings of the IEEE*, vol. 54, pp. 1920–1931, December 1966.

[2] W. K. Jenkins and E. J. Altman, "Self-checking properties of residue number error checkers based on mixed radix conversion," *IEEE Transactions on Circuit and Systems*, vol. 35, pp. 159–167, February 1988.

[3] L.-L. Yang and L. Hanzo, "Performance of residue number system based DS-CDMA over multipath fading channels using orthogonal sequences," *European Trans. on Telecommunications*, vol. 9, pp. 525–536, November - December 1998.

[4] L.-L. Yang and L. Hanzo, "Ratio statistic test assisted residue number system based parallel communication systems," in *Proc. of IEEE VTC'99*, (Houston, USA), pp. 894–898, May 1999.

[5] H. Krishna, K.-Y. Lin, and J.-D. Sun, "A coding theory approach to error control in redundant residue number systems - Part I: theory and single error correction," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 8–17, January 1992.

[6] J.-D. Sun and H. Krishna, "A coding theory approach to error control in redundant residue number systems - Part II: multiple error detection and correction," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 18–34, January 1992.

[7] L.-L. Yang and L. Hanzo, "Coding theory and performance of redundant residue number system codes." submitted to IEEE Transactions on Information Theory, 1999.

[8] L.-L. Yang and L. Hanzo, "Residue number system arithmetic assisted M-ary modulation," *IEEE Communications Letters*, vol. 3, pp. 28–30, February 1999.

[9] L.-L. Yang and L. Hanzo, "Residue number system based multiple code DS-CDMA systems," in *Proc. of IEEE VTC'99*, (Houston, USA), pp. 1450–1454, May 1999.

[1]

[1] [3, 4, 7–9]: Also see http://www-mobile.ecs.soton.ac.uk/lly.