RESIDUE NUMBER SYSTEM AND ITS APPLICATIONS

DHANYA NOBLE, AMEENUDEEN PE, JILU JAMES, DONA JOSEPH

Abstract: Residue Number System (RNS) is a non-weighted number system. In RNS, the arithmetic operations are split into smaller parallel operations which are independent of each other. There is no carry propagation between these operations. Hence devices operating in this principle inherit property of high speed and low power consumption. But this property makes overflow detection very difficult. Hence the moduli set is chosen such that there is no carry generated. In this paper, the use of residue number system (RNS) is portrayed in designing solution to various applications of Communication and Signal Processing. RNS finds its application where integer arithmetic is authoritative process, since residue arithmetic operates efficiently on integers. New moduli set selection process, magnitude comparison routine and sign detection methods were limed on the onset of this article.

Keywords*:* RNS, RRNS, CRT, MRC, Error Correction

Introduction: Residue number system (RNS) is a non-weighted, non - positional number system which can be represented by specifying its base. RNS helps to increase the speed of arithmetic operations like addition, subtraction and multiplication when compared to traditional number systems like binary, decimal, hexadecimal etc., which are linear and are highly dependent on the order of the digits [1]. In all these systems, each bit position is associated with a weight and these weights are derived from the same base (radix). Unlike these systems, it does not have a single fixed radix. The bases of RNS are represented by a u-tuple of integers (m_1, $m2$,..., m_u) where each of these bases is called a modulus. Thus, in RNS any given integer is represented by a set of residues which are obtained by modulo dividing the integer with a moduli set. The moduli are usually relatively prime to each other.

The arithmetic operations like addition, subtraction and multiplication are carry free in RNS. This implies that the error due to these operations or due to noise does not propagate from one residue digit to other. They remain confined to the original residues [2]. Due to the absence of carry between residues, operation is said to be closed within each residue position. So large calculations can be decomposed into a series of smaller parallel calculations and thus RNS finds application in the field of digital computer arithmetic. Most significant research work in the field of RNS was done in forward and reverse conversion. Reverse converters uses Chinese Remainder Theorem (CRT) proposed by Y. Wang and Mixed Radix Conversions (MRC) [3][4].

RNS can be used for error detection and correction by adding some redundancy to the RNS. This forms Redundant Residue Number System (RRNS). RRNS can be used for self- checking in digital computers. Thus it helps in the design of general purpose systems, which are capable of sensing and rectifying their own transmission and processing errors. An important property of RRNS is lack of ordered significance among residue digits. Therefore, an integer can be recovered from its residues even after discarding some of the redundant residues, provided that the retained residue digits should be correct [5].

RNS is mostly used in VLSI implementation of DSP architecture for achieving low power and high speed. Among the applications of RNS, implementation of FIR filters, IIR filters, adaptive filters, digital frequency synthesis, two dimensional filters, image encryption and coding are most significant [6]. One dimensional and two dimensional Discrete Wavelet Transform architecture based on residue arithmetic made a crucial impact. RNS had been effectively used in encryption and coding. In field of communication, RNS were introduced in field of CDMA by many researchers. Frequency hopping techniques, DS-CDMA and PN sequence co-realtors based on RNS were proposed and designed [7]. There was much programmable architecture designed based on residue arithmetic.

This article describes the single and multiple error correction algorithms using RRNS. The algorithms are explained using representative examples. Also various applications of RNS in the field of communication and signal processing are presented in this article.

Rest of the paper is organised as follows: section 2 describes the basics of Residue Number System. Section 3 describes the two decoding techniques CRT and MRC. Redundant Residue Number System is explained in section 4. Error detection and correction algorithms based on RRNS along with the examples are given in section 5. Section 6 describes the various applications of RNS.

Residue Number System: Residue number systems are based on congruence relation. Two integers, a and b are said to be congruent modulo m if m divides exactly the difference of a and b, that is $a \equiv b(mod\ m_i)$. The number m is called modulus or base. If r is the

remainder of the integer division of a by m, then $a \equiv r(mod\ m_i)$[3]. The remainder r is said to be the residue of a with respect to m and is denoted by $r = |a|_m$. Consider a set of v moduli ($m_1, m_2,\ldots,m_v$), then the product of these v moduli is given as M and is called the dynamic range. Every integer within this range can be represented by a unique set of residues corresponding to these moduli. The dynamic range R for negative numbers is given as[8]

$$R = \begin{cases} (-\frac{M-1}{2}, \frac{M-1}{2}) & \text{if } M \text{ is odd} \\ (-\frac{M}{2}, \frac{M}{2}-1) & \text{if } M \text{ is even} \end{cases}$$

An important property of RNS operations are its arithmetic operation like addition, subtraction and multiplication are carry free[9]. Let X_1 and X_2 be two integers

$X_1 \bullet X_2 \Longleftrightarrow (r_{1i} \bullet r_{2i})\ mod\ m_i,\ i = 1, 2,\ldots, v$ (1)

Where $\bullet$ denotes arithmetic addition, subtraction or multiplication and r_{1i} and r_{2i} are residues of X_1 and X_2 with respect to moduli m_i.

Example 1: Addition in RNS with moduli (6, 2, 4)

Residues of 8 → [2, 0, 0]
Residues of 15 → [3, 1, 3]
Sum is 23 → [5, 1, 3]

In this way the operations can be carried out without any carry propagation between residues and thus errors do not propagate between residue digits. Thus with RNS larger operations can be simplified by dividing it into simpler parallel operations [2].

Decoding Of Residue Number System: For the purpose of converting back the received residues into the integers the decoding algorithms are used. Two decoding algorithms, Chinese Remainder Theorem and Mixed Radix Conversion are explained in this section.

1.Chinese Remainder theorem (CRT)

A classical algorithm for the decoding is Chinese Remainder theorem (CRT) [4]. Consider a set of moduli ($m_1, m_2,\ldots, m_k$) and a residue sequence ($r_1, r_2,\ldots,r_k$). Then the integer X can be recovered from these residues by using an algorithm called Chinese Remainder Theorem (CRT), which gives

$$X = (\sum_{i=1}^{k} r_i\ T_i\ M_i)\ mod\ M \quad (2)$$

Where $M_i = M/m_i$ and T_i is the multiplicative inverse of M_i which can be calculated as

$T_i M_i = 1\ mod\ m_i$ (3)

This can be simplified as

$$\frac{T_i\ M_i}{m_i} - \lfloor \frac{T_i\ M_i}{m_i} \rfloor \frac{1}{m_i} = 0 \quad (4)$$

Example 2: For a moduli set (3, 5, 7), the integer 23 can be represented in residue format as (2,3,2).

Applying CRT to recover the integer gives:

Step 1: Dynamic range can be computed using M = 3 X 5 X 7 = 105, Dynamic range is [0,104]

Step 2: M_i can be found out by $M_i = \frac{M}{m_i}$

i.e., $M_1 = 105/3 = 35$, $M_2 = 105/5 = 21$, $M_3 = 105/7 = 15$

Step 3: Multiplicative inverse of M_i, T_i, can be calculated using $T_i M_i = 1\ (mod\ m_i)$

i.e., $T_1 M_1 = 1\ (mod\ m_1) \Rightarrow 35\ T_1 = 1\ (mod\ 3)$, $T_2 M_2 = 1\ (mod\ m_2) \Rightarrow 21\ T_2 = 1\ (mod 5)$ $T_3 M_3 = 1\ (mod\ m_3) \Rightarrow 15\ T_3 = 1\ (mod\ 7)$

where T_1, $T2$, $T3$ are multiplicative inverses of 35, 21 and 15 over modulo 3, 5 and 7. By solving above equations,

$T_1 = 2$, $T2 = 1$ and $T_3 = 1$.

Step 4: $X = [(35 *2 *2) + (21 *1 *3) + (15 *1 *2)]\ (mod\ 105) = 233\ (mod\ 105) = 23$

The real time implementation of CRT is not possible because it involves a modular operation with large integer M which results in a complexity of $O(n^3)$. To avoid the computations with such larger M, an alternate technique called Mixed Radix Conversion (MRC) is used [4].

Mixed Radix Conversion (MRC)

Consider a set of moduli ($m_1, m_2,\ldots,m_k$) and let the residue sequence for X be ($u_1, u_2,\ldots,u_k$)[4][10]. Then the integer X can be recovered using mixed radix conversion as:

- Compute constants c_{ij} where $1 \le i,j \le k$ as $c_{ij} m_i = 1\ (mod\ m_j)$
- compute $v_1 = u_1\ (mod\ m_1)$ $v_2 = (u_2 - v_1)\ c_{12}\ (mod\ m_2)$

$v_3 = ((u_3 - v_1)\ c_{13} - v_2)\ c_{23}\ (mod\ m_3)$

.

.

$v_k = (..((u_k - v_1)\ c_{1k} - v_2)\ c_{2k} - \ldots - v_{k-1})\ c_{k-1,\ k}\ (mod\ m_k)$ is the mixed radix representation of X. Then X can be recovered as

$X = v_1 + v_2 m_1 + v_3 m_1 m_2 + \ldots. + v_k m_1 m_2 \ldots m_{k-1}$ (5)

Example 3: For a moduli set (3, 5, 7), the integer 23 can be represented in residue format as (2, 3, 2). Using Mixed Radix conversion, we can recover the integer 23 from these residues by: Step 1: Compute C_{ij} using $c_{ij} m_i = 1\ (mod\ m_j)$

$c_{12}\ m_1 = 1\ (mod\ m_2) \Rightarrow 3\ c_{12} = 1\ (mod\ 5) \Rightarrow 3\ (mod\ 5) = 3$ *and* $3.2\ (mod\ 5) = 1 \Rightarrow c_{12} = 2$ $c_{13}\ m_1 = 1\ (mod\ m_3) \Rightarrow 3\ c_{13} = 1\ (mod\ 7) \Rightarrow 3\ (mod\ 7) = 3$ *and* $3.5\ (mod\ 7) = 1 \Rightarrow c_{13} = 5$

$c_{23}\ m_2 = 1\ (mod\ m_3) \Rightarrow 5\ c_{23} = 1\ (mod\ 7) \Rightarrow 5\ (mod\ 7) = 5$ *and* $5.3\ (mod\ 7) = 1 \Rightarrow c_{23} = 3$

Step 2: $v_1 = u_1\ (mod\ m_1) = 2\ (mod\ 3) = 2$. $v_2 = (u_2 - v_1)\ c_{12}\ (mod\ m_2) = (3 - 2)2\ (mod\ 5) = 2\ (mod\ 5) = 2$

$v_3 = ((u_3 - v_1)\ c_{13} - v_2)\ c_{23}\ (mod\ m_3) = ((2 - 2) *5 - 2)3\ (mod\ 7) = -6\ (mod\ 7) = 1$

Step 3: $X = v_1 + v_2 m_1 + v_3 m_1 m_2 + \ldots. + v_k m_1 m_2 \ldots m_{k-1} = 2 + (2 *3) + (1 *3 *5) = 2 + 6 + 15 = 23$ MRC provides a good alternative for CRT because of the reduced complexity of $O(n)$.

Redundant Residue Number System: By adding some redundant information to RNS, we obtain Redundant Residue Number System (RRNS). The main use of RRNS is in error detection and

correction. An integer X is represented in the RRNS form as

$X \rightarrow [r_1, r_2, \ldots, r_v, r_{v+1}, \ldots, r_u]$ (6)

where $(m_1, m_2, \ldots, m_v)$ are called information moduli and $(m_{v+1}, m_{v+2}, \ldots, m_u)$ are called redundant moduli[10]. Similarly $(r_1, r_2, \ldots, r_v)$ are called information residues and $(r_{v+1}, r_{v+2}, \ldots, r_u)$ are called redundant residues. M_r denotes the product of redundant moduli. In RRNS, the legitimate range is defined as $[0, M]$ and illegitimate range, where residues are obtained using redundant moduli, is $[M, MM_r]$[2].

An important property of RRNS is lack of ordered significance among residue digits. Therefore, an integer can be recovered from its residues even after discarding some of the redundant residues, provided that the retained residue digits should be correct [8].

In a residue number system, the number of non-zero elements in a vector is defined as its hamming weight. Let x_i and x_j are two code vectors, then hamming distance $d(x_i, x_j)$ is defined as the number of bits in which two code vectors x_i and x_j differ. Minimum distance, d is the minimum of hamming distances [3][10].

$d = min\ (d\ (x_i, x_j); x_i 6= x_j)$ (7)

Theorem 1: The minimum hamming distance d of an RRNS (u, v)-code is defined as $d = u - v + 1$, provided $(m_1 < m_2 < \ldots < m_v < m_{v+1} < \ldots < m_u)$.

Theorem 2: For a redundant residue number system, the error detecting capability, $c = d - 1$ and the error correcting capability [4], $t = b(d-1)/2c$ where bac is the largest integer smaller than a.

Thus RRNS (u,v) code can detect up to $u - v$ residue digits and can correct up to $t = b(u-v)/2c$ residue digits. This means that single error and multiple error correction algorithms can be developed by suitably selecting u and v.

Application Of Rrns In Error Detection And Correction: RRNS codes can be used for error detection and correction in both information residues and redundant residues. The idea behind this is that, an integer can be recovered using any v from u of the residues. Therefore $d = u - v$ residues can be discarded [2].

a. Single Error Detection and Correction:

The redundant residue number system can correct any single residue error if it satisfies the following two conditions [1]: $R > m_v m_{v-1}$ and $\min (R/m_{v+i}) \geq m_v$ where $R = m_{v+1} m_{v+2} \ldots m_{v+r}$ and $r \geq 2$.

Consider that a set of residues $|X|_{mi}$, are transmitted and received sequence are $|X|'_{m_i}$, where $i = 1, 2, \ldots, u$. With the help of v received information residues, redundant residues $(|X|''_{m_{v+1}}, |X|''_{m_{v+2}}, \ldots, |X|''_{m_u})$ can be computed using base extension method [4]. Define

$$|\triangle|_{m_{v+i}} = ||X|''_{m_{v+i}} - |X|'_{m_{v+i}}|_{m_{v+i}} \quad (8)$$

This procedure is called as consistency checking which forms the basis of error correction algorithm described in this article [5].

Theorem 3: If any one of the residue is in error, then one of the following cases can occur [1]

Case 1: If all the elements in the set $(|4|_{mv+r}, \ldots, |4|_{mv+1})$ are zero, then all the residues are correct.

Case 2: If only one of the element in this set $(|4|_{mv+r}, \ldots, |4|_{mv+1})$ is non-zero, then $|X|'_{m_z}$ is wrong and this can be corrected by replacing with $|X|''_{m_z}$.

Case 3: If more than one element in this set $(|4|_{mv+r}, \ldots, |4|_{mv+1})$ is non-zero, then any one of the information residue is in error and all the redundant residues are correct.

The above theorem helps in the detection of a single residue error and helps to identify whether the detected error is in information residue or in redundant residue.

Then consider m_{v-2j-1} and m_{v-2j} as redundant moduli and $(m_1, \ldots, m_{v-2j-2}, m_{v-2j+1}, \ldots, m_{v+r})$ as information moduli and compute $(|4|_{mv-2j}, |4|_{mv-2j-1})$.

Theorem 4: If one of the residues in the set $(|X|'_{m_1}, |X|'_{m_2}, \ldots, |X|'_{m_{v-2j}})$ is in error, then one of the following cases can occur [1].

Case 1: If the residue $|X|^{o}_{mz}, 1 \leq z \leq v - 2j - 2$, is in error, then $|4|_{mv-2j}$ and $|4|_{mv-2j-1}$ are non-zero.

Case 2: If $|X|^{o}_{mv-2j-1}$ is in error, then $|4|_{mv-2j-1}$ is non-zero and $|4|_{mv-2j}$ is zero. Error can be corrected by replacing it with $|X|''_{m_{v-2j-1}}$.

Case 3: If $|X|^{o}_{mv-2j}$ is in error, then $|4|_{mv-2j}$ is non-zero and $|4|_{mv-2j-1}$ is zero. Error can be corrected by replacing it with $|X|''_{m_{v-2j}}$.

This theorem helps in finding the location of a single residue error. This paper describes a single error correction algorithm which is based on the above theorem [2].

Algorithm:

Step 1: Compute the sets $(|4|_{mv+r}, \ldots, |4|_{mv+1})$, $(|4|_{mv}, |4|_{mv-1}), \ldots, (|4|_{mv-2j}, |4|_{mv-2j-1}), \ldots, (|4|_{m2}, |4|_{m1})$.

Step 2: If all the elements in the set $(|4|_{mv+r}, \ldots, |4|_{mv+1})$ are zero, then all the residues are correct. If only one of the elements in this set is non-zero, then go to step 4. If more than one element in this set is non-zero then go to step 3 by setting j=0.

Step 3: Check $(|4|_{mv-2j}, |4|_{mv-2j-1})$. If only one element in this set is non-zero, go to step 4. If both of the elements are non-zero and if this is not last set, increment j by 1 and repeat this step. If both the elements are non-zero and this is the last set then there are multiple errors.

Step 4: if $|4|_{mz}$ is non-zero, then $|X|^{o}_{mz}$ is wrong and this can be corrected by replacing with $|X|''_{m_z}$.

The implementation of the algorithm is explained using 2 examples. First example shows the correction of error in the redundant residues and in the second case the error is in the information residues.

Example 4: Consider an integer X = 274 which is encoded into a RRNS code using a moduli set (7, 9, 11, 13, 16) where 7, 9 and 11 are information moduli and 13 and 16 are redundant moduli. The RRNS representation of 274 is (1, 4, 10, 1, 2). Consider that r_4 is changed from 1 to 5, and then the received codeword is (1, 4, 10, 5, 2). The decoding is done using MRC. Here first three residues are taken for decoding.

Step 1: Compute C_{ij} using $c_{ij}m_i = 1(mod m_j)$ c_{12} = 1, c_{13} = 8 and c_{23} = 5

Step 2: $v_1 = u_1 \ (mod\ m_1) = 1(mod 7) = 1$, $v_2 = (u_2 - v_1)\ c_{12}\ (mod\ m_2) = 3$, $v_3 = ((u_3 - v_1)\ c_{13} - v_2)\ c_{23}\ (mod\ m_3) = 4$

Step 3: $X = v_1 + v_2m_1 + v_3m_1m_2 + ... + v_nm_1m_2...m_{n-1} = 274$

So the decoded integer is 274. Now we can find redundant residues using base extension method. $|X|''_{13} = 274(mod 13) = 1$ and $|X|''_{13} = 274(mod_{16}) = 2$. So $|4|_{13} = |1 - 2|_{13} 6= 0$ and $|4|_{16} = |2 - 2|_{16} = 0$ Since $|4|_{13}$ is non-zero, $|X|'_{13}$ is error and $|X|''_{13}$ is correct. That is redundant residue $r_4 = 1$.

Example 5: Consider that an integer X = 274 which is encoded into a RRNS code using a moduli set (7, 9, 11, 13, 16) where 7, 9 and 11 are information moduli and 13 and 16 are redundant moduli. The RRNS representation of 274 is (1, 4, 10, 1, 2). If due to error information moduli r_2 is changed from 4 to 7, then the received codeword is (1, 7, 10, 1, 2).

Here first three residues are taken for decoding. The received codeword is decoded as 43. Now we can find redundant residues using base extension method. $|X|''_{13} = 43(mod_{13}) = 4$ and $|X|''_{16} = 43(mod_{16}) = 11$. So $|4|_{13} = |4 - 1|_{13} 6= 0$ and $|4|_{16} = |11 - 2|_{16} 6= 0$. Since both of them are non-zero, one of the information residues is in error.

Now take m_2 and m_3 as redundant moduli and rest as information moduli. Using MRC we can find $C_{14} = 2$, $C_{15} = 7$, $C_{45} = 5$. The mixed radix representation of X is $V_1 = 1$, $V_4 = 0$, $V_5 = 3$ and $X = 274$. Using base extension, $|X|''_9 = 4$, $|X|''_{11} = 10$ and $|4|_9 = |4 - 7|_9 = 0$, $|4|_{11} = |10 - 10| - 11\ 6= 0$. Therefore we can conclude that $|X|'_9$ is incorrect and $|X|''_9$ is correct. That is information residue $r_3 = 4$.

b. Multiple Error Correction Algorithm

The performance of multiple error correction algorithms is also analysed along with single error correction algorithm. The two algorithms analysed in this article are given below:

Algorithm 1

In the case of multiple residue errors, 2*l* redundant moduli will correct *l* residue errors if it satisfies the condition [1],

$$m_1 < m_2 < ... < m_v < m_{v+1} < ... < m_{v+2l}. \quad (9)$$

With a received $v + 2l$ residue sequence, using base extension compute $|X|''_{m_d}$ and $|4|_{md}$, where m_d is any 2l redundant moduli.

Algorithm:

Step 1: Compute the following sets: $S_1 = (|4|_{mv+1}, ..., |4|_{mv+2l})$, $S_2 = (|4|_{mv-b} |4|_{mv-l+1}, ..., |4|_{mv-+l-1})$, Sd $(k+l-1)/\ (l+1)$ e+1= $(|4|m_1, |4|m_2, ..., |4|m_{2l})$.

Step 2: If all the elements in the set S_1 are zero, then all the residues are correct. If the number of nonzero elements in S_1 are less than or equal to *l*, then go to Step 4. If more than *l* non-zero elements are present in set S_1, then go to Step 3 by setting i = 2.

Step 3: Check S_i. If all elements in S_i are zero, then an uncorrectable error is detected. If the number of nonzero elements in S_i are less than or equal to *l*, then go to Step 4. If more than *l* non-zero elements are present in set S_i, and *i*<d $(k + 1 - 1)/\ (1 + 1)$e + 1, then increment i by 1 and repeat Step 3. If more than *l* non-zero elements are present in set S_i, and *i*= d $(k+1-1)/\ (1+1)$ e+1, then an uncorrectable error is detected. *Step 4*: if $|4|_{mc}$ is non-zero, then $|X|'_{m_c}$ is wrong and this can be corrected by replacing with $|X|''_{m_c}$.

In this algorithm if we restrict the errors to be burst residue errors of length < *l*, then if at least one information residue is in error implies that the last *l* + 1 redundant residues should be correct.

Example 6: Consider an integer X = 25 which is encoded into a RRNS code using a moduli set (5, 7, 9, 11, 13, 16) where 5 and 7 are information moduli and 11, 13 and 16 are redundant moduli. The RRNS representation of 25 is (0, 4, 7, 3, 12, 9). Consider that then the received codeword is (1, 2, 7, 3, 12, 9). Here the number of errors that can be corrected is *l*=2.

With the received information residues, the integer is decoded as 34 using MRC. Now using base extension, $|X|'_3 = 7$, $|X|'_4 = 1$, $|X|'_5 = 8$ and $|X|'_6 = 2$. Therefore, in set S_1, $|4|_3 = 0$, $|4|_4 = 06$, $|4|_5 6= 0$ and $|4|_6) 6= 0$. The number of non-zero elements in S_1 are greater than 2.

Now check for set S_2 using m_5 and m_6 are non-redundant moduli. Using MRC, the decoded integer is 25.

$|X|'_1 = 0$, $|X|'_2 = 4$, $|X|'_3 = 7$ and $|X|'_4 = 3$. Therefore, in set S_2, $|4|_1 6= 0$, $|4|_2 6= 0$, $|4|_3 = 0$ and $|4|_4) = 0$. The number of non-zero elements is 2. So we can conclude that the received residues r_1 and r_2 are in error and we can correct it as r_1=0 and r_2=4.

Algorithm 2

This algorithm overcome the limitation of the above described algorithm 1 that in order to correct errors in the information residues, last *l* + 1 redundant residues should be correct. Also this algorithm can detect *l* errors and correct *l* –1 errors if *l* redundant residues are added [9].

Algorithm:

Step 1: Define legitimate range, $D = \prod_{j=1}^{v} m_j$

Step 2: Using the received residues $(r_1, r_2, ..., r_v)$, find the decoded message $X'_{12..v}$.

Step 3: If $X'_{12..v}$ is less than or equal to D, add $X'_{12..v}$ to the set S.

Step 4: Repeat the steps 3 and 4 using another combination of v number of residues.
Step 5: Check for repetitions of elements in set S. If any one of the elements is present more than once, then it is the correct decoded message X.
Step 6: Find which combinations of residues form the correct decoded message X. These residues are the correct ones and remaining residues are erroneous.
Step 7: Correct the incorrect residues by $r_j = X \bmod m_j$ (10)

The following two examples shows the error detection and correction capability of the algorithm. First example shows the error detection and second example shows error correction in RRNS.

Example 7: Consider an integer X = 21 which is encoded into a RRNS code using a moduli set (3, 4, 5, 7) where (3, 4, 5) are information moduli and 7 is redundant modulus. Legitimate range is [0, 60) The RRNS representation is (0, 1, 1, 0). Consider that r_3 is changed from 1 to 3, and then the received codeword is (0, 1, 3, 0). Using this received sequence the message can be decoded as X=33.

However, 33 (mod 7) =56= r_4, hence there were errors. Error in the residue digit r_3 can be detected by designing RRNS with one redundant modulus.

Example 8: For error correction consider another redundant modulus 11. Then the information moduli are (3, 4, 5) and redundant moduli (7, 11). Residue representation of integer 21 based on this moduli set is (0, 1, 1, 0, 10). Let r_3 is changed from 1 to 3 and received sequence is therefore (0, 1, 3, 0, 10). According to the properties of RRNS, if there is no error, then any integer can be recovered by using any set of 3 moduli. So we consider all possible combinations of 3 moduli:

$(r_1,r_2,r_3) = (0,1,3) \Longleftrightarrow X_{123} = 33(mod 60)$, $(r_1,r_2,r_4) = (0,1,0) \Longleftrightarrow X_{124} = 21(mod 60)$,

$(r_1,r_2,r_5) = (0,1,10) \Longleftrightarrow X_{125} = 21(mod 60)$, $(r_1,r_3,r_4) = (0,3,0) \Longleftrightarrow X_{134} = 63(mod 60)$,

$(r_1,r_3,r_5) = (0,3,10) \Longleftrightarrow X_{135} = 153(mod 60)$, $(r_1,r_4,r_5) = (0,0,10) \Longleftrightarrow X_{145} = 21(mod 60)$,

$(r_2,r_3,r_4) = (1,3,0) \Longleftrightarrow X_{234} = 133(mod 60)$, $(r_2,r_3,r_5) = (1,3,10) \Longleftrightarrow X_{235} = 153(mod 60)$,

$(r_2,r_4,r_5) = (1,0,10) \Longleftrightarrow X_{245} = 21(mod 60)$, $(r_3,r_4,r_5) = (3,0,10) \Longleftrightarrow X_{345} = 98(mod 60)$

The integers $X_{134}, X_{135}, X_{234}, X_{235}$ and X_{345} are in the illegitimate range. In the rest of the 5 cases, $X_{124}, X_{125}, X_{145}$ and X_{245} gives the integer 21 and therefore 21 is the correct result. It is observed that all the combinations of moduli resulted in 21 does not include residue r_3. Therefore we can conclude that residue r_3 is in error. It can be corrected by replacing it with 21(mod 5) =1.

Other Applications Of Rns: RNS finds application in the field of digital computer arithmetic. RNS is mostly used in VLSI implementation of DSP architecture for achieving low power and high speed. Among the application RNS, implementation of FIR filters, IIR filters, adaptive filters, digital frequency synthesis, two dimensional filters, image encryption and coding are most significant. The foremost canonical reason for implementation of filter in residue arithmetic is the inherent property of carry-free addition, subtraction and multiplication. As a result we add, subtract and multiply in unison regardless to the numbers. Hereby, devices operating in this principle are fast and have low power [6].

The major security goals for which image, needs to be encoded and encrypted before transmission through a channel are confidentiality, integrity and availability. These are serious issues with respect to data security and transmission and has been challenging for researchers to develop secured encoding techniques. There have been multiple proposed techniques. However there is no specific absolutely reliable technique. The concept of break and process is again expended to operate on the data in parallel such that the computational load is minimum as well as shared [6]. Moreover the number system works effectively with integer arithmetic. The proposed technique has very low. The robustness and performance of this scheme under adverse channel conditions is much effective. This technique was found to be robust.

Conclusion: This paper provides the basic concepts of Residue Number Systems based on the congruence relation. As Residue Number System (RNS) is a non-weighted, non - positional number system, the arithmetic operations are split into smaller parallel operations which are independent of each other. The most important aspect of RNS is break and process. In its application, focus was on exploring the potential of Residue Arithmetic in the field of Communication and Signal Processing. Various algorithms of error detection and correction using RNS is described in this article. The advantages of RNS encoding are fast computation, computational blocks are non-interacting, arithmetic operations are carry free and parallel operations are possible which helps in developing fast DSP processors. Efforts may be desirable to simplify the encoding as well as decoding, division, magnitude comparison and sign detection.

References:

1. S.-S. Yau, Y.-C. Liu, *Error correction in redundant residue number systems,* IEEE Trans. Comput. C-22 (1) (1973) 511. http://dx.doi.org/10.1109/T-C.1973.223594.
2. *A. Patra, S. Saha Ray,* Adomian Approximation Solution for Fractional Order Riccati ; Mathematical Sciences International Research Journal ISSN 2278 – 8697 Vol 2 Issue 2 (2013), Pg 245-248
3. L.L. Yang, L. Hanzo, *Redundant residue number system based error correction codes* , IEEE VTS 54th Vehicular Technology Conference, 2001, VTC 2001 Fall, Vol. 3, 2001, pp. 14721476. http://dx.doi.org/10.1109/VTC.2001.956442.
4. H. Krishna, J.-D. Sun, *On theory and fast algorithms for error correction in residue number system product codes,* IEEE Trans. Comput. 42 (7) (1993) 840853. http://dx.doi.org/10.1109/12.237724.
5. *Annamma Abraham,* Rayleigh-Benard-Marangoni instability in A Micropolar Dielectric ; Mathematical Sciences International Research Journal ISSN 2278 – 8697 Vol 2 Issue 2 (2013), Pg 254-258
6. Amusa, K. A.1, E. O. Nwoye, *Novel Algorithm For Decoding Redundant Residue Number Systems (RRNS) Codes,* IJRRAS 12, July 2012, Vol 12, Issue 1.
7. L.L. Yang, L. Hanzo, *Coding theory and performance of redundant residue number system codes,* [Online]. Available: http://wwwmobile.ecs.soton.ac.uk.
8. S. Elizabeth, L. Sujatha, on Searching Shortest Path in A Classical Network; Mathematical Sciences international Research Journal ISSN 2278 – 8697 Vol 3 Issue 2 (2014), Pg 568-574
9. P. Maji, G. S. Rath, *Bit Efficient Finite Impulse Response Filter based on Residue Arithmetic*, ICSSA-11, Gujarat, India, 2011.
10. R. Chithra, P. Maji, S. K. Patra, G. S. Rath, *A PN Sequence Generator based on Residue Arithmetic for Multi-User DS-CDMA Applications*, ICCCISE, Copenhagen, Denmark, June 11-12, 2012.
11. *Biswapati Jana,* Survey on Cheating and Prevention Techniques in Visual Cryptography; Mathematical Sciences international Research Journal ISSN 2278 – 8697 Vol 3 Spl Issue (2014), Pg 961-976
12. F. Barsi, P. Maestrini, *Error correcting properties of redundant residue number systems,* IEEE Trans. Comput. C-22 (3) (1973) 307315. http://dx.doi.org/10.1109/T-C.1973.223711.
13. A. Sengupta, D. Zhu, B. Natarajan, *On the performance of redundant residue number system codes assisted STBC design,* International Conference on Computing, Networking and Communications, Wireless Communications Symposium, ICNC12-WC, Maui, Hawaii, USA, 2012.
14. Ishfaq A. Ganaie, Jitender Rattan, Ajay Kumar Mittal, V.K. Kukreja, Simulation of Packed Bed of Porous Particles; Mathematical Sciences International Research Journal ISSN 2278 – 8697 Vol 3 Issue 1 (2014), Pg 134-141
15. J.-D. Sun, H. Krishna, *A Coding Theory Approach to Error Control in Redundant Residue Number Systems, II. Multiple Error Detection and Correction,* IEEE Trans. Circuits Syst. II 39 (1) (1992) 1834. http://dx.doi.org/10.1109/82.204107.

***

Dhanya Noble/Ameenudeen Pe/ Jilu James/ Dona Joseph/
Department of Electronics and Communication Engineering SJCET/Palai/ Kerala/India.