Unfairness Correction in P2P Grids Based on Residue Number System of a Special Form

Mikhail Babenko, Nikolay Chervyakov
North-Caucasus Federal University
Stavropol, Russia
mgbabenko@ncfu.ru, k-fmf-primath@stavsu.ru

Andrei Tchernykh
CICESE Research Center
Ensenada, Mexico
chernykh@cicese.mx

Nikolay Kucherov, Maria Shabalina, Irina Vashchenko
North-Caucasus Federal University
Stavropol, Russia
{nkucherov, ikrisina}@ncfu.ru, mari.qwertyqwerty@mail.ru

Gleb Radchenko
South Ural State University,
Chelyabinsk, Russia
gleb.radchenko@susu.ru

Daniil Murga
King's College London
London, United Kingdom
daniilmurga@gmail.com

***Abstract*—This paper addresses error correction codes approach in order to improve the performance of BOINC under uncertainty of users' behavior. Redundant Residue Number System (RRNS) moduli set of the special form provides correction of user unfairness, reliability, decreased redundancy and load of the computing network. Error correction code in RRNS is improved by using error syndrome. It decreases the amount of computations required for data decoding up to 20/7 times compared to the projection methods. The proposed modification of the error syndrome allow to omit the assumption that clients are honest and reliable. The proposed approach decreases the execution time of the client programs asymptotically by 4 times. On the other hand, encoding with RRNS places some restrictions on the range of performed operations.**

Keywords— Uncertainty, BOINC, residue number system, error correction codes, replication, redundancy.

I. INTRODUCTION

A large class of problems in science and technology requires high-performance distributed computing to handle large amounts of data. One of the possible solutions to this problem is the high-performance grid computing based on the Peer-to-Peer (P2P) data distribution technique. Berkeley Open Infrastructure for Network Computing (BOINC) is volunteer computing platforms currently being successfully used for solving scientific problems in various fields such as: astrophysics, physics, cryptography, medicine, chemistry, etc.

However, there is a number of technical problems with this technology due to the following uncertainties [1]

1. *Uncertainty of running time of the client program*. It occurs due to internal and external system factors such as load of the computer, where the client program is executed, and communication bandwidth. To minimize the total running time of the tasks BOINC assigns the deadline for obtaining the results.
2. *Loss of interest* of the user to the project.
3. *Uncertainty of getting the correct results* returned by the client program. This uncertainty is either because of the dishonesty of the participants, when they can send faked result, or technical failures, attacks of hackers, or viruses.

An error in one of the subtasks can lead to a wrong result of the entire project. BOINC makes replications of the subtasks to solve this problem. By default, the number of replicas is five, and the result is correct, if it matches three or more results.

The classical replication technique has a number of shortcomings.

1. Redundancy leads to losses of computing power, reducing communication bandwidth, etc. For example, if the default replication value is used, the system performance drops by a factor of five with respect to the system without failures.
2. Delay in obtaining earned credits is increased, since it is provided according to the last client program that returns the result.

An alternative solution to this problem is to use error correction codes in the Residue Number System (RNS). Due to arithmetic properties, they allow results verification, error detection, localization and correction as well as parallel data processing.

To achieve the efficiency of error correction codes, we use numbers of a special form, for which the data encoding is performed in linear time.

Let us consider the following seven RRNS moduli set

$$\left\{2^m - 2^{\frac{m+1}{2}} + 1,\quad 2^m - 3,\quad 2^m - 1,\ 2^m,\ 2^m + 3,\ 2^m + 2^{\frac{m+1}{2}} + 1\right\}, \tag{1}$$

where m is an odd number. Four moduli define the dynamic range, and three ones are control moduli. Therefore, three errors in computation results can be detected.

The proposed scheme reduces data redundancy asymptotically by a factor of three. It significantly improves the system performance. However, only narrow class of problems can be solved efficiently. It is necessary to develop algorithms for error detection and localization with linear running time.

II. UNCERTAINTY IN BOINC

An extensive research has led to a general understanding of uncertainty issues in different fields ranging from physics, and computational biology to decision making in the economy and

2378-3915/17 $31.00 © 2017 IEEE
DOI 10.1109/DEXA.2017.46

social sciences. However, an impact of uncertainty on P2P large scale computing systems is not studied.

The nature of uncertainties in BOINC is close to the uncertainties in cloud computing.

Tchernykh et al 2016 [2] study the impact of various types of uncertainties on the cloud computing resource provisioning. Particular attention is paid to the security and reliability of the distributed data processing.

Follow the classification of Tychinsky, 2006 [3], and Tchernykh et al., 2014 [9], we highlight the following uncertainties that significantly affect the data processing in BOINC: Retrospective uncertainty, Technical uncertainty, Uncertainty of restrictions, and Uncertainty of participants.

Retrospective uncertainty. Since the number of participants in the data processing projects in BOINC changes over time, there is no statistically valuable information on the past behavior of new users;

Technical uncertainty. BOINC has a probability do get results from one or several project participants.

Uncertainty of restrictions. It is caused by partial or complete ignorance of the operating conditions of the BOINC client programs. Since during the time interval, in which the calculations are performed, participants demands in computing capacity and load unpredictable varies;

Uncertainty of participants. It occurs in a situation of conflict between users and administrators of BOINC, when each one has his own preferences, incomplete, inaccurate information about the motives and behavior of the opposite party.

To minimize the influence of uncertainty on performance, a reliable synthesis based on the use of either a probabilistic approach or the worst-case approach are used. In this case, the uncertainty is perceived as a random variable or an interval variable.

The Monte Carlo simulation method allows obtaining reliability evaluation in the framework of optimization. Ali et al., 2003 [4] show that parallel and distributed systems need robustness to guarantee limited performance degradation despite fluctuations in the behavior of its component parts. Uncertainty can be estimated using standard deviation, differential entropy, etc. [5].

Schwiegelshohn et al., 2012 [6] proposed a model for rigid real-time planning. The practical application of algorithms is shown in Tchernykh et al., 2014 [7] and Barquet et al., 2013 [8], where the authors explore its applicability to high-performance computing, which allows it to be adapted for planning in BOINC.

The problem of non-anticipatory planning in P2P networks under uncertainty is discussed in detail in Tchernykh et al., 2014 [9]. The authors compare twenty algorithms from the point of view of Pareto dominance. The main idea of the approach is to select the replication threshold value and dynamically adapt it.

III. Data Processing in RNS

RRNS is a non-positional number system, which is defined by the set of pairwise coprime numbers $\{p_1, p_2, \dots, p_n\}$ –RRNS moduli set. In order to detect, localize and correct errors, RRNS moduli set is divided in two subsets: first k moduli compose the dynamic range of the system, and the rest $r = n - k$ are control moduli. In this case, the system can detect r errors.

A positive number X in RRNS, with a given moduli set, is represented as the tuple $X = \{x_1, x_2, \dots, x_n\}$, where $x_i = |X|_{p_i} = X \bmod p_i$ [8] for all $i = 1,2, \dots, n$. This representation of a number is unique, if $X \in [0, P-1]$, where $P = \prod_{i=1}^{k} p_i$ is the dynamic range of RRNS.

RRNS operations over big numbers are substituted by the operations over small numbers, which are the residues of big numbers divided by the coprime numbers from the moduli set $p_1, p_2, \dots, p_n$. Let

$$A \equiv a_1 \pmod{p_1},\ A \equiv a_2 \pmod{p_2}, \dots, A \equiv a_n \pmod{p_n}.$$

For an integer A, we have the tuple $\{a_1, a_2, \dots, a_n\}$ of least non-negative residues. According to Chinese remainder theorem (CRT), this mapping is bijective and allows error detection and correction, if $A < \prod_{i=1}^{k} p_i$.

Fig. 1 shows the scheme of data processing in RNS.

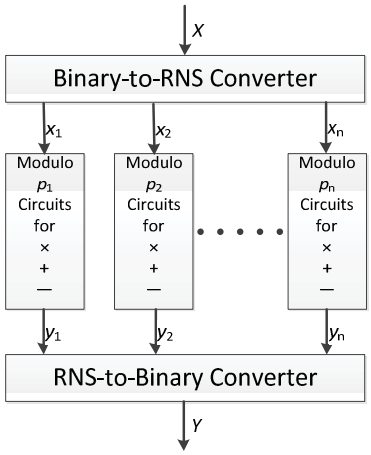

Fig. 1. Data processing in RNS

The advantage of RRNS is the fact that addition, subtraction and multiplication can be done in parallel, and used as the basis for processing in BOINC. We formalize the parallel data processing in RRNS with the following expressions:

$$A \mp B = \{a_1, a_2, \dots, a_n\} \mp \{b_1, b_2, \dots, b_n\} = \{|a_1 + b_1|_{p_1}, |a_2 + b_2|_{p_2}, \dots, |a_n + b_n|_{p_n}\} \quad (2)$$

$$A \times B = \{a_1, a_2, \dots, a_n\} \times \{b_1, b_2, \dots, b_n\} = \{|a_1 \times b_1|_{p_1}, |a_2 \times b_2|_{p_2}, \dots, |a_n \times b_n|_{p_n}\} \quad (3)$$

In order to increase the reliability of the result, we add an additional mechanism based on the approximate computation of the rank of the number in RRNS, from the work [10]. This mechanism verifies the result of computations with smaller values, detects a range overflow, and efficiently compute the result of computations.

On the other hand, RRNS can be considered as a secret sharing scheme, in which each participant does not receive the entire secret data, but only a secret projection of the data. The secret sharing scheme based on RRNS is a fully homomorphic cipher with respect to addition, subtraction and multiplication. These two properties of RRNS can be used to develop a homomorphic encryption function to ensure the security and reliability of data processing in BOINC.

IV. Arithmetic Operations with Moduli of a Special Form

A. Module $2^m - k$

Addition of X and Y modulo $2^m - k$ can be implemented as binary addition of the operands, and finding the residue from division [11-14]. Considering the form of the module and the range of possible values of the sum of X and Y, the addition can be represented as follows:

$$|X+Y|_{2^m-k} = \begin{cases} X+Y, if\, X+Y < 2^m-k, \\ X+Y+k-2^m, \text{if } X+Y \geq 2^m-k. \end{cases}$$

We compute $R_1 = X+Y$ and $R_2 = X+Y+k$, if m-th bit of R_2 is "0", then "R_1" is returned otherwise "$R_2 - 2^m$". "$R_2 - 2^m$" is computed by the following formula: $R_2 - 2^m = R_1 \text{ AND}(2^m - 1)$. Modular subtraction is done in a similar manner with the following formula:

$$|X-Y|_{2^m-k} = \begin{cases} X-Y, \text{if } X-Y \geq 0, \\ X-Y+2^m-k, \text{if } X-Y < 0. \end{cases}$$

For modular multiplication, we compute $R = X \cdot Y$, where $X, Y \in [0,\ 2^m - k)$ and $R \in [0,\ (2^m - k - 1)^2]$, then R can be represented as follows $R = 2^m R_{[2m-1:\ m]} + R_{[m-1:0]}$, therefore

$$|R|_{2^m-k} = \left|k \cdot R_{[2m-1:\ m]} + R_{[m-1:0]}\right|_{2^m-k} \quad (4)$$

Then, we consider how the (4) changes for different values of k proposed in the paper:

- $k = 1$, then (4) becomes $|R|_{2^m-1} = \left|R_{[2m-1:\ m]} + R_{[m-1:0]}\right|_{2^m-1}$ where one modular addition is required.
- $k = 3$, then (4) becomes $|R|_{2^m-3} = \left|3 \cdot R_{[2m-1:\ m]} + R_{[m-1:0]}\right|_{2^m-3} = \left|2 \cdot R_{[2m-1:\ m]} + (R_{[2m-1:\ m]} + R_{[m-1:0]})\right|_{2^m-3}$. Two modular additions and one modular multiplication by 2 are required.
- $k = 2^{\frac{m+1}{2}} - 1$, then (4) becomes

$$|R|_{2^m-2^{\frac{m+1}{2}}+1} = \left|2^{\frac{m+1}{2}} \cdot R_{[2m-1:\ m]} - R_{[2m-1:\ m]} + R_{[m-1:0]}\right|_{2^m-2^{\frac{m+1}{2}}+1} = \left|\left(R_{\left[2m-1:\frac{3m-1}{2}\right]} \middle| \overline{R_{\left[\frac{3m-3}{2}:m\right]}}\right) + \left(R_{[m-1:0]} - 3 \cdot 2^{\frac{m+1}{2}} + 2\right)\right|_{2^m-2^{\frac{m+1}{2}}+1,}$$

where "|" – is concatenation and $\bar{x}$ – is inversion. Two modular additions and one modular subtraction are required.

B. Module $2^m + k$

Addition of X and Y modulo $2^m + k$ is similar to addition modulo $2^m - k$. We make the following modifications [11-14]. Addition can be implemented as addition of two operands and finding the residue from division. Considering the form of the module and the range of possible values of the sum of X and Y, the addition can be represented as follows:

$$|X+Y|_{2^m+k} = \begin{cases} X+Y, \text{if } X+Y < 2^m+k, \\ X+Y-k-2^m, \text{if } X+Y \geq 2^m+k. \end{cases}$$

We compute $R_1 = X+Y$ and $R_2 = X+Y-k$, if m-th bit R_2 is "0", then "R_1" is returned otherwise "$R_2 - 2^m$". "$R_2 - 2^m$" is computed by the formula: $R_2 - 2^m = R_1 \text{ AND}(2^m - 1)$.

Modular subtraction is done in a similar manner with the following formula:

$$|X-Y|_{2^m+k} = \begin{cases} X-Y, \text{if } X-Y \geq 0, \\ X-Y+2^m+k, \text{if } X-Y < 0. \end{cases}$$

For modular multiplication, we compute $R = X \cdot Y$, where $X, Y \in [0,\ 2^m + k)$ and $R \in [0,\ (2^m + k - 1)^2]$, then R can be represented as $R = 2^{2m} R_{[2m+1:\ 2m]} + 2^m R_{[2m-1:\ m]} + R_{[m-1:0]}$, therefore

$$|R|_{2^m+k} = \left|k^2 \cdot R_{[2m+1:\ 2m]} - k \cdot R_{[2m-1:\ m]} + R_{[m-1:0]}\right|_{2^m+k} \quad (5)$$

Then, we consider how the (5) can be changed for different values of k proposed in the paper:

- $k = 1$, then (5) becomes $|R|_{2^m+1} = \left|R_{[2m+1:\ 2m]} - R_{[2m-1:\ m]} + R_{[m-1:\ 0]}\right|_{2^m+1}$, where one modular addition and one modular subtraction are required.
- $k = 3$, then (5) becomes

$$|R|_{2^m+3} = \left|9 \cdot R_{[2m+1:\ 2m]} - 3 \cdot R_{[2m-1:\ m]} + R_{[m-1:\ 0]}\right|_{2^m+3} = \left|2 \cdot (4 \cdot R_{[2m+1:\ 2m]} - R_{[2m-1:\ m]}) + R_{[2m+1:\ 2m]} + R_{[m-1:\ 0]} - R_{[2m-1:\ m]}\right|_{2^m+3,}$$

where two modular additions, two modular subtractions and one modular multiplication by 2 are required.

- $k = 2^{\frac{m+1}{2}} + 1$. Because $(2^m + 2^{\frac{m+1}{2}} + 1) | (2^{2m} + 1)$, then

$$|R|_{2^m+2^{\frac{m+1}{2}}+1} = \left|\left|2^{2m} R_{[2m+1:\ 2m]} +\right.\right.$$

$$R_{[2m-1:\,0]}\big|_{2^{2m}+1}\Big|_{2^m+2^{\frac{m+1}{2}}+1} = \big|R_{[2m-1:\,0]} -$$

$$R_{[2m+1:\,2m]}\big|_{2^m+2^{\frac{m+1}{2}}+1} = \Big|\Big(R_{\left[2m-1:\frac{3m-1}{2}\right]}\Big|\overline{R_{\left[\frac{3m-3}{2}:\,m\right]}} +$$

$$R_{\left[\frac{3m-3}{2}:\,m\right]}\Big|\overline{R_{\left[2m-1:\frac{3m-1}{2}\right]}}\Big) + \Big(R_{[m-1:\,0]} - R_{[2m+1:\,2m]} -$$

$$3\cdot 2^{\frac{m+1}{2}} + 2\Big)\Big|_{2^m+2^{\frac{m+1}{2}}+1.}$$

Two modular additions and one modular subtraction are required.

C. Computational Complexity Analisys

Let T_A , T_S, T_M be the time required to add, subtract and multiply two m bit operands respectively, then the time required to compute modular operations is shown in Table 1.

TABLE I. TIME REQUIRED TO COMPUTE MODULAR OPERATIONS

Moduli	Addition	Subtraction	Multiplication
2^m	T_A	T_S	T_M
2^m-1	$2T_A$	$2T_S$	T_M+2T_A
2^m+1	T_A+T_S	T_A+T_S	$T_M+2T_A+2T_S$
2^m-3	$2T_A$	$2T_S$	T_M+5T_A
2^m+3	T_A+T_S	T_A+T_S	$T_M+4T_A+5T_S$
$2^m-2^{\frac{m+1}{2}}+1$	$2T_A$	$2T_S$	$T_M+4T_A+2T_S$
$2^m+2^{\frac{m+1}{2}}+1$	T_A+T_S	T_A+T_S	$T_M+3T_A+3T_S$

Table 1 shows that the most time consuming is multiplication modulo 2^m+3. Considering the computational complexity of the multiplication algorithm, we can assume the time of modular operations is roughly equal for all moduli apart from 2^m. Modular addition and subtraction take twice more time than usual addition and subtraction, Modular multiplication takes roughly the same time as usual multiplication.

V. ERROR LOCALIZATION AND CORRECTION CODE

Now, let us consider an example with seven participants.

If results are available from all seven client programs, then we can compute the result of applications using moduli

$$\{2^m-2^{\frac{m+1}{2}}+1,\ 2^m-1,\ 2^m+1,\ 2^m+2^{\frac{m+1}{2}}+1\}.$$

The RNS dynamic range in this case is $2^{4m}-1$.

We have projections of the result $\{r_1, r_3, r_5, r_7\}$, and compute the result using CRT:

$$\left|\frac{1}{(2^{2m}-1)(\,2^m+2^{\frac{m+1}{2}}+1)}\right|_{2^m-2^{\frac{m+1}{2}}+1} = 2^{m-2}-2^{\frac{m-5}{2}} \quad (6a)$$

$$\left|\frac{1}{(2^{2m}+1)(\,2^m+1)}\right|_{2^m-1} = 2^{m-2} \quad (6b)$$

$$\left|\frac{1}{(2^{2m}+1)(\,2^m-1)}\right|_{2^m+1} = 2^{m-2} \quad (6c)$$

$$\left|\frac{1}{(2^{2m}-1)(\,2^m-2^{\frac{m+1}{2}}+1)}\right|_{2^m+2^{\frac{m+1}{2}}+1} = 2^{m-2}+2^{\frac{m-5}{2}} \quad (6d)$$

From (6a-6d), it follows that R can be computed in linear time using CRT.

We check if all seven participants returned the true result. To do this, we check tree values $|R|_{2^m-3}-r_2=s_2$, $|R|_{2^m}-r_4=s_4$, $|R|_{2^m+3}-r_6=s_6$.

Without loss of generality, we assume that if $s_i\neq 0$, then $s_i=1$.

TABLE II. ERROR LOCALIZATION AND CORRECTION

#	s_2	s_4	s_6	Position of the error
1	*0*	*0*	*0*	-
2	*0*	*0*	*1*	r_6
3	*0*	*1*	*0*	r_4
4	*0*	*1*	*1*	r_4, r_6
5	*1*	*0*	*0*	r_2
6	*1*	*0*	*1*	r_2, r_6
7	*1*	*1*	*0*	r_2, r_4
8	*1*	*1*	*1*	Not defined

Considering that at one time we can compute three sets of two errors, the total possible amount of errors is 42 [15-17]. The number of computations of R with CRT is $\frac{42}{3}=14$. Taking into consideration that projections method requires computation of 42 projections, the proposed method decreases the amount of computations by 3 times.

The proposed improvement excludes the requirement to have absolutely honest and reliable client programs.

VI. CHALLENGES

The class of tasks that can be solved in BOINC with RNS is related to a number of problems. These problems can be divided into two classes:

1. *Operations in RNS.* RNS with a non-positional number representation allows internal parallelism for addition/subtraction and multiplication with calculation in linear time. But RNS has reduced efficiency for comparisons, modular reductions, modular multiplications, sign determination, and overflow detection. Since the number conversion to a binary representation is not efficient, number approximations in RNS are used to determine the position of the number: core function, diagonal function, the approximate method, etc. As practice shows, solutions that combine several methods of approximation are efficient.

2. *Efficient implementation of basic algorithms in RNS.* The basis of the most computationally complex problems in physics, chemistry, ecology, biology, etc. have typical structures of linear algebra: parallel summation, scalar multiplication of vectors, triangular systems of equations, block-dual-diagonal systems of equations, systems of equations by the Jordan method, difference schemes, etc. Hard problems to solve with RNS in BOINC are: block-dual-diagonal systems of equations, systems of equations by the Jordan method. To solve these problems, the approach from HPC that divides the data into blocks can be used.

VII. Conclusion

Data encoding with RNS decreases the size of the operands asymptotically by the factor of four, which decreases the time of data processing by about 4 times. In order to improve fault tolerance under the conditions of uncertainty, we propose to use three control moduli set. Therefore, the redundancy of the proposed error correction code is $7/4$. The redundancy in BOINC by default is five. Hence, the improvement of the redundancy of the proposed scheme is roughly 20/7, with the same level of security.

Error detection, localization and correction method is improved. It is shown that the proposed approach based on error syndrome does not require reliable computational nodes and honest users. It also decreases the number of computations compared to the method of projections.

In order to expand the class of problems that can be solved efficiently with RNS in BOINC it is required to find solutions of a number of problems. This will be the subject of future work. Moreover, we need a better understanding of user unfairness in real installations. It is another important issue to be addressed.

Acknowledgment

This work is partially supported by Russian Federation President Grant SP-1215.2016, and Act 211 Government of the Russian Federation, contract № 02.A03.21.0011.

References

[1] E. Ivashko, A. Golovin, “Partition Algorithm for Association Rules Mining in BOINC–based Enterprise Desktop Grid,” Springer International Publishing, International Conference on Parallel Computing Technologies, pp. 268–272, 2015.

[2] A. Tchernykh, U. Schwiegelsohn, E. Talbi, and M. Babenko, “Towards Understanding Uncertainty in Cloud Computing with risks of Confidentiality, Integrity, and Availability,” J. Comput. Sci., 2016.

[3] A. Tychinsky, “Innovation management of companies: Modern approaches, algorithms, experience,” 2006 [Online]. Available: http://www.aup.ru/books/m87/ [accessed 29.06.16]

[4] S. Ali, A. A. Maciejewski, H. J. Siegel, and J. K. Kim, “Measuring the robustness of a resource allocation,” IEEE Trans. Parallel Distrib. Syst., vol. 15, no. 7, pp. 630–641, 2004.

[5] L. C. Canon and E. Jeannot, “Evaluation and optimization of the robustness of dag schedules in heterogeneous environments,” IEEE Trans. Parallel Distrib. Syst., vol. 21, no. 4, pp. 532–546, 2010.

[6] U. Schwiegelshohn and A. Tchernykh, “Online scheduling for cloud computing and different service levels,” in Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012, 2012, pp. 1067–1074.

[7] A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. E. Pecero, and S. Nesmachnow, “Bi-objective online scheduling with quality of service for IaaS clouds,” in 2014 IEEE 3rd International Conference on Cloud Networking, CloudNet 2014, 2014, pp. 307–312.

[8] A. L. Barquet, A. Tchernykh, and R. Yahyapour, “Performance Evaluation of Infrastructure as Service Clouds with SLA Constraints Evaluación del desempeño de servicios de infraestructura en nubes con restricciones de acuerdos de nivel de servicio (SLA),” Comput. y Sist., vol. 17, no. 3, pp. 401–411, 2013.

[9] A. Tchernykh, J. E. Pecero, A. Barrondo, and E. Schaeffer, “Adaptive energy efficient scheduling in Peer-to-Peer desktop grids,” Futur. Gener. Comput. Syst., vol. 36, pp. 209–220, 2014.

[10] N. Chervyakov, M. Babenko, A. Tchernykh, N. Kucherov, V. Miranda-López, J. M. Cortés-Mendoza, “Adaptive, Scalable and Reliable Systems for Internet of Things to Ensure Security, Trust, and Privacy,” Future Generation Computer Systems, 2017.

[11] P. M. Matutino, R. Chaves, & L. Sousa, “Arithmetic Units for RNS Moduli {2n-3} and {2n+ 3} Operations,” Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on. IEEE., pp. 243-246, 2010.

[12] S. Ma, J. H. Hu, & C. H. Wang, “A Novel Modulo $2^{n}-2^{k}-1$ Adder for Residue Number System,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 60, no. 11, pp. 2962-2972, 2013.

[13] P. M. Matutino, R. Chaves, & L. Sousa, “An efficient scalable RNS architecture for large dynamic ranges, ”Journal of Signal Processing Systems,” vol. 77, no. 1-2, pp. 191-205, 2014.

[14] S. H. F. Langroudi, & G. Jaberipur, “Modulo-(2^n--2^ q--1) Parallel Prefix Addition via Excess-Modulo Encoding of Residues,” Computer Arithmetic (ARITH), 2015 IEEE 22nd Symposium on. IEEE, pp. 121-128, 2015.

[15] N. I. Chervyakov, P. A. Lyakhov, M. G. Babenko, A. I. Garyanina, I. N. Lavrinenko, A. V. Lavrinenko, & M. A. Deryabin, “An efficient method of error correction in fault-tolerant modular neurocomputers,” Neurocomputing, vol. 205, pp. 32-44, 2016.

[16] H. Xiao, H. K. Garg, J. Hu, & G. Xiao, “New Error Control Algorithms for Residue Number System Codes,” ETRI Journal, vol. 38, no. 2, pp. 326-336, 2016.

[17] P. A. Mohan, “Error Detection, Correction and Fault Tolerance in RNS-Based Designs,” Residue Number Systems, Springer International Publishing, pp. 163-175, 2016..