© International Journal of Combinatorial Optimization Problems and Informatics, Vol. 9, No. 1, Jan-April 2018, pp. 62-67. ISSN: 2007-1558.

High Performance Parallel Computing in Residue Number System

Maxim Deryabin[1], Nikolay Chervyakov[1], Andrei Tchernykh[2],
Mikhail Babenko[1], and Mariia Shabalina[1]
North-Caucasus Federal University, Stavropol, Russia[1]
CICESE Research Center, Ensenada, Baja California, México[2]
maderiabin@ncfu.ru, ncherviakov@ncfu.ru, mgbabenko@ncfu.ru,
m.n.shabalina@yandex.ru, chernykh@cicese.mx

Abstract. Residue Number System (RNS) allows performing computation more efficiently. Natural parallelism of representation and processing of numbers makes this number system suitable for applying to high performance computing. We address the main features of application of RNS to high-performance parallel computing. We consider and analyze different stages of data processing in RNS. Based on this analysis, we describe the process of decomposition of algorithms using RNS

Keywords: Residue Number System, High-Performance Computing, number systems, parallel computing.

1 Introduction

High-performance computing has become an important field in modern world. The fact that the volume of data to be processed continuously grows and demands for speed and computing resources are becoming stricter should be taken into account. At the same time, to achieve the maximum efficiency, often, problem-oriented approaches are used. This paper analyzes the features of the implementation of high-performance computing algorithms using a special method of data representation in Residue Number System [1, 2].

Residue Number System (RNS) is a non-positional number system, with the absence of carries between the digits in arithmetic operations. Thus, RNS allows to use parallel computing due to the independence of the computing process for each digit. However, it should be taken into account that such a representation of data requires large amount of additional operations related to the conversion to and from RNS, and a number of other complex operations.

All operations in RNS can be divided into two classes: non-positional (modular) operations, allowing parallelism and non-modular operations related to the need to compute a positional characteristic of a number. The first class includes multiplication and addition. The second class of operations complicates the high-performance computing in RNS. The most important operations of this class are magnitude comparison of numbers and division. In this context, RNS is most often used to solve a certain class of problems in which the number of non-modular operations is minimized. Examples of such problems are digital filters of different nature [3], image processing [4], and big numbers processing [1, 2].

2 Computations in RNS

RNS is based on modular non-positional representation of numbers. Each digit in RNS is a residue of the division by a number called a base. All bases for each digit are form a moduli set. The uniqueness of the representation of numbers in RNS is only guaranteed under the condition that all moduli are pairwise coprime. Let $\{m_1, m_2, \dots, m_n\}$ – be a given moduli set. It determines a unique RNS. A number X in this RNS can be represented as follows: $X = \{x_1, x_2, \dots, x_n\}$, where $x_i = |X|_{m_i}$ for all $i = 1,2, \dots, n$. Note that according to Chinese Remainder Theorem X should belong to the interval $[0, M)$, where $M = m_1 \cdot m_2 \dots \cdot m_n$ is dynamic range of the RNS.

Received Dec 2, 2017 / Accepted Jan 22, 2018

With such a representation, addition and multiplication can be done in parallel. Let $X = \{x_1, x_2, \dots x_n\}$ and $Y = \{y_1, y_2, \dots, y_n)$ be numbers in RNS then their sum $S = X + Y = \{s_1, s_2, \dots, s_n\}$ and product $Q = X \cdot Y = \{q_1, q_2, \dots, q_n\}$ can be computed using following formulas $s_i = |x_i + y_i|_{m_i}$ and $q_i = |x_i \cdot y_i|_{m_i}$, for all $i = 1,2, \dots, n$. These operations do not require carries between digits as in positional number systems. This property allows to perform these operations independently.

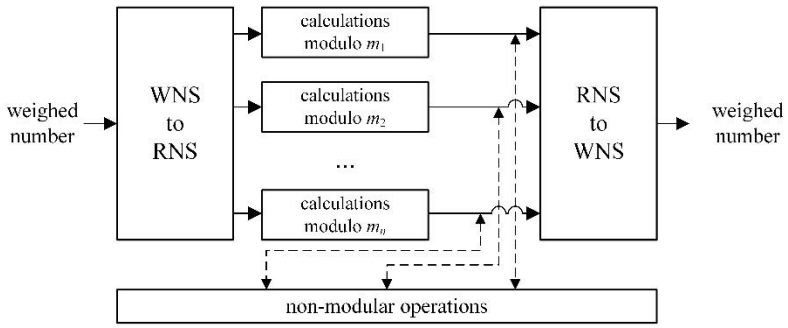

Fig. 1. – Model of computations in RNS

However, computations in RNS require a number of specific operations, without which it is impossible to represent numbers in RNS [3], Figure 1 shows the general model of computations in RNS. It includes the steps of the conversion to RNS and positional notation back from RNS. In addition, class of non-modular operations can be represented as a separate computational structure. These operations require special approaches associated with the estimation of the result. Computations for each digit can be done independently and in parallel. The highest performance is achieved for algorithms, which are based on addition and multiplication.

An example of computational algorithms in which RNS can be applied to dramatically improve performance is linear system solvers that are used for instance in digital filtering [3]. They commonly use operations called accumulation represented in the following form:

$$Y = t_0 + \sum_{j=1}^{k} t_j X_j,$$

where t_j – is a system constant, X_j – system input.. If each numbers X_j and t_j are represented in RNS, the computational time decreases.

Another important application is a cryptographic transformation. Most cryptographic operations are based on computations over a large modulus. They lead to rather complicated operations (finding the residue of division) with big numbers, exceeding the standard size of 64 bits. As studies in [2] show, in some cases, these operations can be efficiently implemented using residue number system.

In general, RNS is very efficient in tasks that require computation with big numbers. However, it is necessary to take into account the features of this number system. For example, problems, in which there are many non-modular operations, obviously, cannot be effectively implemented in RNS. Selecting RNS parameters is one of the most difficult and important issues of the development of problem solvers based on RNS.

3 Analysis of the structure of algorithms in RNS

In this section, we discuss the main features of the implementation of RNS systems, based on the model presented in Figure 1. All operations can be divided into forward conversion of Weighted Number System (WNS) to RNS, and inverse conversion, computations in each digit and non-modular operations. We consider each type of operations separately.

3.1. Parallel operations in RNS and selection of moduli set

The main advantage of RNS in the context of high performance computing is the opportunity to do some operations in parallel. Each number X from the interval $[0, M)$ is substituted by n numbers x_i, each of which belongs to the interval $[0, m_i)$, for $i = 1,2, \dots, n$. Since M is the product of all m_i, then X is larger then each x_i. If X is of N bit length then each x_i (under the condition of balanced moduli set m_i) is of N/n bit length.

Addition and multiplication have asymptotical complexity $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ respectively. In the best case multiplication has complexity $\mathcal{O}(N \log_2 N \log_2 \log_2 N)$ [5]. Using this property and combining addition and multiplication, we can achieve a dramatic increase in speed of computations, which are done in parallel in RNS.

It is worth mentioning that arithmetic operations in positional (binary) number system are done by modulo 2^N, where N is the number of binary digits. Since 2 is the base of the number system, than finding residue modulo 2^N can be seen as reduction of higher bits. Due to the fact that m_i should be pairwise coprime, only one of them can be the power of two. Consequently, computations over any other module from the moduli set require special methods that include finding residue. Requirement of additional resources for non-modular operations influences the performance of the system.

In practice, to deal with this problem, special kind of moduli sets are used. For example, numbers of the form $2^{N \pm r} \pm t$, where t and r – are natural numbers. There are effective algorithms of performing non-modular operations over such kind of moduli set [6].

There are many different options for a moduli set, each of them may be used for certain applications.
In the selection of a moduli set, a number of factors should be taken into account. This factors will influence the further computations in RNS. For a given platform for the implementation, concurrency and independence of computations over each modulus, it is necessary to:

- Select the modules with a minimum computational cost for the given problem, taking into account the implementation of modular operations and considering the difficulty of forward and reverse RNS conversions.
- Select balanced moduli set with respect to their magnitude and complexity of computations. If the moduli set is unbalanced, it may lead to long waiting time for some threads (nodes).
- Minimize the redundancy of RNS representation. If ρ_{WNS} is the number of bits of a number in WNS, and ρ_{RNS} is number of bits required to represent that number in RNS, then, obviously, $\rho_{WNS} < \rho_{RNS}$. Different moduli sets have different redundancy. It should be taken into account that redundancy leads to additional storage, transmitting and processing costs.

Based on the analysis above, we can conclude that only a few algorithms can be effectively implemented in RNS. At the same time, for those algorithms for which RNS is applicable, the benefit from using RNS is essential because it utilizes the special kind of parallel computing parallelism at the level of data representation and arithmetic operations. This is very important under the conditions of development of modern computers with parallel structure such as the GPU and FPGA.

3.1. WNS to RNS and reverse conversion

WNS to RNS and RNS to WNS conversions are crucial to the performance of the system [2]. Moduli sets of special form allow to utilize useful properties of numbers contained in the moduli set, and construct effective algorithms for conversion between number systems.

WNS to RNS conversion comes down to finding residues over each module from the moduli set. There are different approaches to do this depending on the module. It is worth mentioning that m_i is a constant that simplifies the algorithms. For example, a number X can be divided in k parts of l bit length each. In this case, we can use the following formula to find the residue:

$$|X|_m = \left|\sum_{j=0}^{k-1}\left(\sum_{i=0}^{l-1} x_{jl+i}2^i\right)2^{jl}\right|_m = \left|\sum_{j=0}^{k-1} X_i 2^{jl}\right|_m = \left|\sum_{j=0}^{k-1} X_i \left|2^{jl}\right|_m\right|_m$$

This formula allows to substitute the operation with k modular multiplications by a constant $\left|2^{jl}\right|_m$ and $k-1$ modular additions. Moreover, the optimal choice of l implies that the results of partial products can be stored in look-up tables, which simplifies the process.

To achieve maximum efficiency in computations of moduli of special form $2^{N\pm r} \pm t$ can be used. For some moduli of that form, there are special methods to compute the residue of division [7]. Thus, due to special adders and periodicity of computations in the ring of residues, very efficient residue generators can be obtained.

Since computation of residue for each modulus is independent, RNS conversion is one of the parallel operations. However, in spite of this fact and that there are effective approaches for computing residue of the division for some kinds of moduli, the conversion has a significant impact on the performance of the whole system. Unlike RNS, in WNS, there is no need for forward and reverse conversion. Therefore, when using the RNS to obtain efficiency in computations, it is essential to consider what computational costs the conversion requires and how it affects the performance of the algorithm.

Influence of reverse conversion is even more significant due to the fact that unlike forward conversion it is non-modular and requires synchronization of parallel computations. This operation is considered to be the most challenging and important in RNS. A variety of methods are developed to implement it.

Chinese Remainder Theorem (CRT) or its corollaries are usually applied to RNS to WNS conversion. There is a simple algorithm based on CRT which utilizes the formula

$$\mathrm{X} = \left|\sum_{i=1}^{n} x_i B_i\right|_M$$

In this formula, X is represented as $\{x_1, \ldots, x_n\}$ in RNS with moduli set $\{\mathrm{m}_1, \ldots, \mathrm{m}_\mathrm{n}\}$, and $B_i = M_i\left|M_i^{-1}\right|_{m_i}$, $M_i = M/m_i$, are constants of the system. The obvious disadvantage of this conversion method is that M is large. Even with proposed above improvements, it would require much computational resources to find residue over such a large module.

To decrease the computational complexity of CRT, the following approach can be used [8]. Instead of modulus M, that is rather large, we can use the modulus 2^N, for which, obviously, computations can be carried out more efficiently. In this case, we compute the value of X in 2 phases. In the first phase, we approximately estimate the weighted value $X' \approx X/M$, and, in the second phase, we represent X as $X = \lfloor X' \cdot M \rfloor$. In order to make this conversion, we need to change the form of (1), taking into account the number of bits of the number required to do the rounding. Let the rounding be precise N decimals. Then, if we divide each constant by M, we obtain the formula to compute the exact value of X/M from (1). If we multiply each constant by 2^N, and ceil, we obtain constants that are rounded by N decimals precise, and they are integers. Computations modulo 2^N in the same order as in (1) allows to estimate first N decimals of X/M:

$$2^N \frac{X}{M} \approx X' = \left|\sum_{i=1}^{n} x_i k_i\right|_{2^N} = \left|\sum_{i=1}^{n} x_i \left\lceil \frac{B_i 2^N}{M} \right\rceil\right|_{2^N}$$

Taking into account the error that occurs due to rounding of constants, the selection of N to obtain the exact value of X is an important problem. According to [8], the minimal N that is required for conversion for any moduli set can be calculated by the following formula:

$$N = \lceil \log_2 M\mu \rceil - 1$$

where $\mu = m_1 + m_2 + \cdots + m_n - n$.

Selection of moduli set can be done with various methods. Depending on the moduli set, different RNS to WNS conversion methods may be effective.

There are several methods used in practice. The one of them uses mixed radix system (MRS) [9]. This system is weighted and weights are not numbers of the form $1, b, b^2, b^3$ and so on, but arbitrary numbers. It is easy to show that, if we take RNS moduli set as weights, the dynamic ranges of RNS and mixed radix system are the same. This fact can be used for RNS to WNS conversion in 2 phases: conversion to MRS, and conversion to binary system. This method is called Mixed Radix Conversion (MRC). However, in a general case, this method is not the most efficient because it leads to modular computations and increase a number of operations. Nevertheless, with special kinds of moduli sets, MRC can be more efficient than earlier proposed approaches.

Another important method is based on a recursive splitting of the RNS moduli set into two subsets [10]. The recursion is continued while a set contains more than two moduli. Thus the conversion is divided into a series of small steps, in which the number must be converted from 2-moduli RNS, with a significantly lower range then M, and the moduli are partial products of a combination of RNS moduli set. This approach also increases the amount of computation, but can be very effective when the right moduli set and right partition are selected. Some of the moduli in combination may give new moduli, for which computations are effective.

3.2. Non-modular operations

One example of non-modular operations is magnitude comparison of numbers. Obviously, non-positional nature of RNS does not allow to determine which number is less or larger from its representation. Algorithms that imply large number of comparisons or similar operations are not suitable for implementation with RNS. Another example is division of numbers in RNS, which comes down to comparison of numbers. In a general case, to perform non-modular operations, it is necessary to estimate the position of the number represented in RNS compared to other numbers. The direct approach is to convert the numbers to WNS and compare them. But this approach is not effective.

The simpler approach uses positional characteristics [11]. Positional characteristic of a number X in RNS is a value $\pi(X) = \pi(x_1, \ldots, x_n)$, for which $\pi(X) \leq \pi(Y)$ if $X < Y$. The main positional characteristics are:

- The value of X represented in mixed radix system. This system is positional and has all the necessary properties what allows to estimate the positions of numbers.
- Particular cases of core function [12], e.g. diagonal function [13]. Computation of core function is similar to a computation of X with (1). The difference is that the modulus is smaller than M. However, in a general case, the efficiency of using core function is the same as in case of using CRT [14].
- The value X' estimated by (2). This approach is very efficient since it comes down to small amount of operations done modulo 2^N.

However, non-modular operations are making computations in RNS difficult. The complexity of computing any positional characteristic is of the same order as reverse conversion. Most efficiency from RNS can be obtained in the case when the implemented algorithms have small number of non-modular operations.

4 Conclusion

Residue Number System is a numeral system that has parallel operations. Despite the obvious advantages of RNS as a method to improve the performance of computing, it has limitations. The above analysis does not cover all features of using RNS, but serve as the main base for any algorithm. From the performed analysis, we can conclude that in order to get maximum use of RNS, it is necessary to minimize the number of conversions, reduce non-modular operations, and make an optimal choice of moduli set. Parallel computing in RNS is different from parallel computing in WNS because after the conversion all operations become modular and require the implementation of special algorithms for addition and multiplication.

References

1. Tchernykh, A., Schwiegelsohn, U., Talbi, E.-G., Babenko, M.: Towards Understanding Uncertainty in Cloud Computing with risks of Confidentiality, Integrity, and Availability. J Comput Sci. Elsevier. DOI: 10.1016/j.jocs.2016.11.011 (2016)
2. Deryabin, M.A., Zaytsev, A.A.: Using of the Modular Arithmetic for Acceleration of Execution Operations on Big Numbers [in Russian]. Vestnik UGATU (scientific journal of Ufa State Aviation Technical University). 17 (5), 245-251 (2013)
3. Chang, C.-H., Molahosseini, A.S., Zarandi, A.A.E., Tay, T.F.: A New Paradigm to Datapath Optimization for Low-Power and High-Performance Digital Signal Processing Applications. IEEE Circuits and Systems Magazine. 15 (4), 26-44 (2015)
4. Moharrami, S., Taleshmekaeil, D.K.: The Application of the Residue Number System in Digital Image Processing: Propose a Scheme of Filtering in Spatial Domain. Research Journal of Applied Sciences. 7, 286-292 (2012)
5. Karatsuba, E.A.: Fast Algorithms and the FEE Method, http://www.ccas.ru/karatsuba/algen.htm
6. Navi, K., Molahosseini, A.S., Esmaeildoust, M.: How to Teach Residue Number System to Computer Scientists and Engineers. IEEE T. Educ. 54 (1), 156-163 (2011)
7. Piestrak, S.J.: Design of residue generators and multioperand modular adders using carry-save adders. IEEE T. Comput. 43 (1), 68-77 (1994)
8. Chervyakov, N.I., Molahosseini, A.S., Lyakhov, P.A., Babenko, M.G., Deryabin, M.A.: Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem. Int. J. Comput. Math., DOI: 10.1080/00207160.2016.1247439 (2016)
9. Chakraborti, N.B., Soundararajan, J.S., Reddy, A.L.: An implementation of mixed-radix conversion for residue number applications. IEEE T. Comput. 35 (8), 762-764 (1986)
10. Wang, Y.: Residue-to-binary converters based on New Chinese Remainder Theorems. IEEE T. Circuits-II. 47 (3), 197-205 (2000)
11. Chervyakov, N.I., Babenko, M.G., Deryabin, M.A., Nazarov, A.S., Shabalina, M.N.: Computation of positional characteristics of numbers in RNS based on approximate method. Proceedings of the 2016 IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference, EIConRusNW 2016. 177-179 (2016)
12. Akushskii, I.J., Burcev, V.M., Pak, I.T.: A new positional characteristic of non-positional codes and its application [in Russian]. Coding Theory and the Optimization of Complex Systems. Alma-Ata, Kazakh. SSR: Nauka (1977)
13. Dimauro, G., Impedovo, S., Pirlo, G.: A new technique for fast number comparison in the residue number system. IEEE T. Comput. 42 (5), 608-612 (1993)
14. Piestrak, S.J.: A note on RNS architectures for the implementation of the diagonal function. Inf. Process. Lett. 115 (4), 453-457 (2015)