Residue Logarithmic Coprocessor for Mass Arithmetic Computations*

Ilya Osinin

Limited Liability Company "Scientific Production Association Real-Time Software Complexes", 16/2 Krasnoproletarskaya, Moscow 127473, Russia
iposinin@gmail.com

Abstract. The work is aimed at solving the urgent problems of modern high-performance computing. The purpose of the study is to increase the speed, accuracy and reliability of mass arithmetic calculations. To achieve the goal, author's methods of performing operations and transforming data in the prospective residue logarithmic number system are used. This numbering system makes it possible to unite the advantages of non-conventional number systems: a residue number system and a logarithmic number system. The subject of study is a parallel-pipelined coprocessor implementing the proposed calculation methods. The study was carried out using the theory of computer design and systems, methods and means of experimental analysis of computers and systems. As a result of the research and development new scientific and technical solutions are proposed that implement the proposed methods of data computation and coding. The proposed coprocessor has high speed, accuracy and reliability of processing of real operands in comparison with known analogs based on the floating-point positioning system.

Keywords: Logarithmic Number System, Residue Number System, Residue Logarithmic Number System, Performance, Accuracy, Reliability.

1 Introduction

It is well known that the decimal number system significantly simplified the calculations. This served as a revolutionary impetus for technological progress. After the invention of infinite decimal fractions, it acquired the status of a universal number system. The binary number system is the basis of modern technological progress. The areas of its application are computer facilities and information technology. To date, they are implemented on the basis of nanoelectronics.

The development of technological progress stimulates both the growth of the computational need and the improvement of computations in solving new problems. For example, the development of astronomy and navigation in the 16th century stimulated the growth of computational needs. As a result, the logarithm appeared, as a means of reducing the complexity of multiplicative operations.

* The work was supported by the Russian Science Foundation, grant N 17-71-10043

The rapid growth of computing needs in the XX century has aggravated new problems in the field of computing technologies. The key ones are:

- acceleration of calculations due to the parallelism of programs and machine codes, streaming calculations;
- fault tolerance due to self-correction, i.e. detection and correction of errors in the calculation process, including for systems of long-term autonomous existence;
- the accuracy of real calculations in a limited machine bit capacity.

However, their effective solution is almost impossible at the level of computer codes of traditional arithmetic. The reason is the inter-digit relationship of positional numbers. Redundancy is used to improve the reliability of the calculations. Its drawback is a multiple increase in hardware costs. To work with real numbers, the de facto standard is the floating point [1]

$$x = (-1)^{sign} \cdot 2^{E-bias} \cdot (1 + \frac{M}{2^f}) \quad (1)$$

where *sign* is the sign of the number, $2^{E\text{-}bias}$ is the exponent with bias as 127 for single precision and 1023 for double precision, M is the mantissa in the normalized form, and f is the width of the mantissa. In this case, rounding in the calculation process reduces the accuracy of calculations. As a result, this can lead to an erroneous result of the calculations.

The mentioned problems concern all known universal processors available in the market of high-performance computing. Solutions of leading companies Intel, AMD, NVIDIA and several others are based on positional floating-point arithmetic. Thus, to effectively solve the above problems, the development of a new computer account system is urgent.

One of the applicants is the residue logarithmic number system [2]. It combines two-level coding of numbers. One of the levels is the residue number system (RNS) in the basis of modular arithmetic. Another level is the logarithmic number system (LNS). The subject of study is a parallel-pipelined coprocessor implementing the proposed calculation methods. As a result of the research and development new scientific and technical solutions are proposed that implement the proposed methods of data computation and coding. Let us consider them in more detail.

1.1 Residue number system

The foundations of modular arithmetic were proposed more than half a century ago. Scientific papers by I.Ya. Akushsky, D.I. Yuditsky, N.I. Chervyakov, V.S. Knyazkov, Garner H., Omondi A. are devoted to this direction [3-6].

The objectives of transition to RNS are:

- to increase the speed of residue operations (addition, multiplication) due to the parallel processing of each digit of a number from the basis of modules $\{p_1,p_2,...,p_n\}$;
- to increase the reliability of calculations due to self-correction of machine codes when the basis is extended by the control modules $\{p_1,p_2,...,p_{n+k}\}$;

where n and k are the numbers of bases of the main and reference ranges, respectively. To date, there are very few completed technical solutions that operate with the use of RNS. One of them is a modular processor, which operates on the basis of artificial neural networks. It was developed in Stavropol, Russia in 2005 [3].

The limiting factor is the slow (sequential) execution of non-residue operations. They are:

- division, and hence, scaling;
- formation of an overflow indication;
- translation from one number system to another.

The previous work was aimed at the research on high-speed devices for the efficient execution of non-residue operations. They are based on integer parallel-pipelined processing of bit-slices of operands [7].

In general, the presence of non-modular operations makes it difficult to use a floating point. There are some works in which the RNS is adapted to frequent scaling and rounding. One of possible approaches is the use of interval positional characteristics [8]. Its disadvantage is the iterative scaling. This has a negative effect on the speed of arithmetic operations.

However, it is possible to fix the point, providing a wide range of representation of numbers. It will be discussed further.

1.2 Logarithmic number system

In the logarithmic number system (LNS), the real number of the field R is represented by its logarithm along a fixed base. Moreover, the arithmetic operations are isomorphic to the operations of the field R. The interest in the use of LNS in computers first appeared in the late XX century. A fundamental contribution to the study of LSS was made by J. Coleman, M. Arnold, E. Chester, D. Lewis [9-12]. The goal of transition to the LNS is:

- to increase the speed of multiplicative operations (multiplication, division, raising and root extraction) of the R field;
- to increase the accuracy of calculations by fixing a point separating the integer and the fractional part of the number.

The latter allows us to use LNS as an alternative to floating-point calculations. This ensures a similar range of representation of numbers in the allocated bitmap of the machine number

$$\log_2 x = \log_2 2^E + \log_2(1+\frac{M}{2^f}) = E + \log_2(1+\frac{M}{2^f}), \quad (2)$$

where E is the exponent, M is the mantissa in the normalized form, and f is the width of the mantissa.

Despite the rather extensive volume of studies performed, there are only a few completed technical implementations to date. One of them is the European Logarithmic Microprocessor [9]. It has a 32-bit RISC architecture.

The reason for this is the slow performance of additive operations. They are expressed through multiplicative operations

$$\log_2(a+b) = \log_2 a + \log_2(1+2^{\log_2 b - \log_2 a}), \tag{3}$$

where a and b are operands and a < b. The reason for limiting the single precision is the complexity of the conversion from the traditional number system and back. This fact leads to an exponential increase in the hardware costs of code converters.

The methods of converting numbers, based on the author's research [13], helped to eliminate this drawback by using multilevel interpolation. On the other hand, the use of specialized technical solutions contributed to an increase in the speed of additive operations.

Summing it up, fixing a point in the LNS allows us to work with a real operand the same as with an integer. This allows us to combine its advantages with RNS, thus forming a residue logarithmic number system (RLNS). Let us briefly describe its features.

2 Residue logarithmic format and related work

In the introduction it was established that the application of RLNS is actual for solving the problems of modern arithmetic devices in terms of increasing the speed, the accuracy and the reliability of real calculations.

In simplified form, the translation of the real positional number into a residue logarithmic format takes place in three stages:

- the base 2 logarithm is calculated from the original real number;
- the point separating the integer and fractional part of the logarithm is discarded;
- the received integer number is converted to the RNS.

An example of converting a double-precision number is shown in Fig. 1.

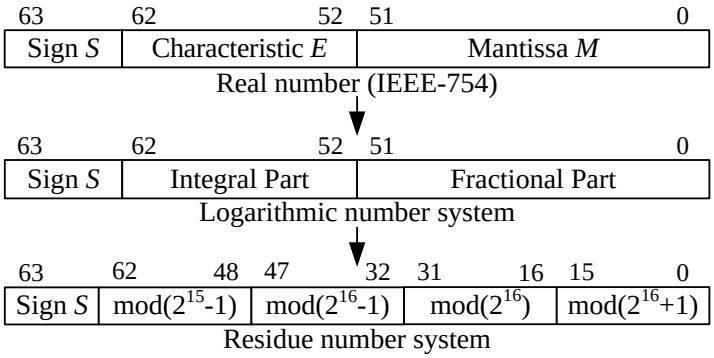

Fig. 1. Converting a double-precision number to RLNS

RLNS integrates implementation advantages both of logarithms:

- multiplicative operations are replaced by additive ones;
- the absence of rounding in the performance of multiplicative operations;

and of the residues of numbers:

- parallel calculation of each digit of a number;
- auto correction of machine codes.

It is obvious that this also accumulates disadvantages:

- the need to convert numbers first in the LNS, then into the RNS;
- the speed of additive operations depends on the speed of the code converters.

Consequently, the classes of computational tasks, where the successes of the RLNS can be achieved are as follows:

- the number of additive operations is commensurable with the number of multiplicative operations, for example, calculation of polynomials, fast Fourier transforms, solution of systems of linear algebraic equations, etc.;
- critical dependence on rounding in ill-conditioned problems, the problems with different-scale coefficients, and also the ones sensitive to the class of equivalent transformations;
- highly reliable real-time systems, for example, missile guidance, management of a nuclear power station, space vehicles operation, etc.

It is worth noting the difference between RLNS and residue logarithmetic [3]. In the latter, the RNS is combined with discrete logarithms. This leads to the preservation of the scaling problem. In contrast, the RLNS allows us to process numbers with a fixed position of a point.

The founder of the RLNS is M.G. Arnold. He described the basic capabilities of this number system and proposed a software implementation [1]. However, RLNS did not receive its further development. The accuracy of the numbers was limited to 32 bits. The calculations were adapted to the implementation on general purpose processors. All these negated the advantages of applying known solutions. It is obvious that theoretically basics of computing in the RLNS are currently investigated insufficiently. There are practically no hardware implementations that operate on its base.

So, the previous work was aimed at creating a set of methods describing residue logarithmic computations a general form [14-16]. These methods are aimed at its high-speed parallel-pipelined technical implementation, also mentioned in [14-16]. The proposed methods and devices based on them relate to high-speed execution of operations in RLNS, as well as conversion of numbers into a residue logarithmic format and back. The maximum speed of operations in them (one clock cycle) is achieved by mass arithmetic processing. This condition is necessary to fill the computational pipeline with a data stream. The conducted research and development made it possible to create a basis for the operation of the arithmetic coprocessor. The features of its organization are discussed below.

6

3 Organization of a residue logarithmic coprocessor

In this section, a device operating on the basis of RLNS is considered. It integrates the entire previous work of the author, allowing us to maximize the benefits of this number system at the hardware level. The use of the device as a coprocessor makes it possible:

- to simplify the internal control device as much as possible;
- to transfer for calculation only a part of the tasks that can be solved most effectively using the residue logarithmic approach.

Fig. 2 shows the structural scheme of the relationship with a universal processor based on shared memory. The advantage of this approach is the lack of a mechanism for memory coherency and simplification of the relationship between the processor and the coprocessor.

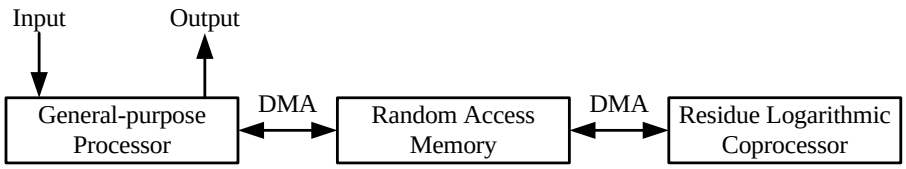

Fig. 2. The structural scheme of the relationship with a universal processor

Fig. 3 shows the structural scheme of the coprocessor in its general form. Each computational core of the coprocessor performs independent processing on a particular residue. Arithmetic operations can be performed in both vector and scalar forms.

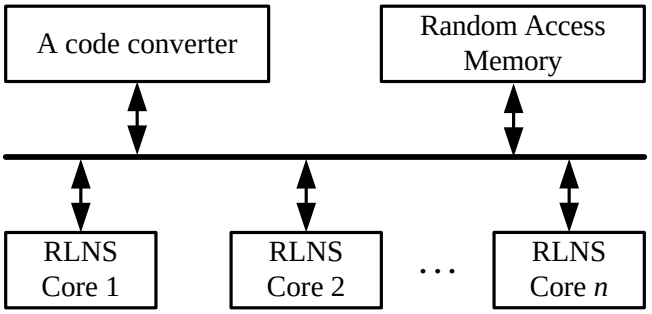

Fig. 3. The structural scheme of the coprocessor in its general form

In general, the number of cores is equal to the number of bases of the main and control range of the RLNS. Their maximum number is limited only by hardware costs. The adder and the multiplier are provided for processing multiplicative operations in

the computational core. Their width corresponds to the width of the base on which the processing is performed.

The architecture of the coprocessor is discussed in more detail in [17]. The cores are connected to a parallel-pipeline code converter. The method and the code converter device are given in a general form in [15, 16].

The difference from the arithmetic devices of known universal processors lies in the following as:

- scalar values without pipelining, which increases the speed of calculations;
- noiseless encoding of operands, which allows automatic error correction;
- when performing multiplicative operations there is no rounding operation, and that increases the accuracy of calculations.

Further the coprocessor is compared with the known analogs.

4 Technical implementation and evaluating the parameters of the coprocessor

The structural schemes described in the previous section are implemented at the functional level using the Verilog hardware programming language. Debugging is performed using Altera Cyclone V programmable logic integrated circuit (FPGA).

At the moment, the residue logarithmic coprocessor is implemented as a standalone intellectual property (IP) block. Such blocks are complete components for creating systems-on-a-chip, for example, microprocessors. Hardware costs of the IP-block are about 20 million transistors, that is comparable to the block expansion of Intel Xeon AVX2 processors Broadwell generation [17].

The following are the estimates of the speed, the accuracy, and the reliability of computations using RLNS in comparison with the traditional approach.

4.1 Performance

Table 1 compares the performance of the residue logarithmic coprocessor (RLC) with the Intel AVX2 extension of the Broadwell generation. Precision of the operand corresponds to a double-precision IEEE-754 standard.

Table 1. Comparison of RLC and Intel AVX2 performance.

Vector operation	RLC, clock cycles		AVX2, clock cycles	
	Latency	Throughput	Latency	Throughput
Addition	6	1	3	1
Subtraction	6	1	3	1
Multiplication	1	1	3	0,5
Division	1	1	16-23	16
Exponentiation	1	1	288	270
Square root	1	1	19-35	16-27

The numbers are packed into vectors with a dimension of 8 elements. It is assumed that the conversion to RLNS and back is performed once at the beginning and in the end of the task. This does not introduce significant time delays into the computational process due to pipelining.

Latency and throughput of the arithmetic operation pipeline in Intel AVX2 (Broadwell generation) is taken from Intel 64 and IA-32 Architectures Optimization Reference Manual [18].

The greatest acceleration is achieved in the proposed coprocessor when using multiplicative operations. For example, for raising to the power operations, the difference is three orders of magnitude in favor of the coprocessor, which is explained by the lack of hardware implementation of these operations in Intel coprocessors.

At the same time, slow additive operations are performed with twice the latency, but the same throughput (1 clock cycle per operation). One-step execution of vector operations multiplication, division and exponentiation makes it possible to obtain a significant advantage in speed of execution of a certain class of tasks, where such operations prevail. These include the calculation of polynomials of *n*-th power, for example, Taylor series expansion, matrix multiplication and a number of others.

4.2 Accuracy of calculations

One of the options for comparing the accuracy of calculations is the problems of numerical analysis. For example, Laguerre's method is a root-finding algorithm tailored to polynomials. In other words, Laguerre's method can be used to numerically solve the equation $p(x)=0$ for a given polynomial $p(x)$. The average relative error of calculations is 25% less compared to the traditional type of computation (see Fig. 4).

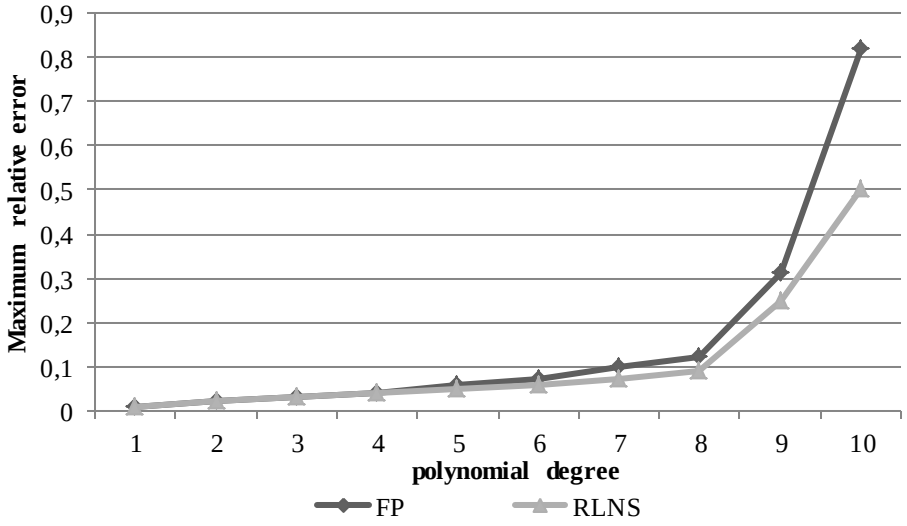

Fig. 4. Accuracy of calculations

This equation has the known analytical solution and, hence, it is possible to calculate a relative error of the solution numerically obtained in the floating-point arithmetic. The experiment is carried out for $n \in [1,10]$, where n is the degree of the polynomial.

The RLNS error does not exceed half the last significant digit when n=10. The result will be exact. While for a floating point, the effect of the error will spread to the last digit of the result, which will cause the error to accumulate. This allows us to conclude that the accuracy of the calculations in the RLNS is higher.

4.3 Reliability assessment

To assess the fault tolerance of RLC, it is advisable to use the availability indicator

$$\delta = \frac{H}{N}, \tag{4}$$

where H is the number of operational states of the system in case of failure and N is the total number of possible states [5].

This indicator allows us to evaluate the effectiveness of the application of corrective codes. Their use makes it possible to increase the coprocessor's fault tolerance in comparison with the quorum - the classic approach of masking failures "2 of 3" [3].

The use of three control bases allows us to correct all double errors with an increase in hardware costs by 75% (4 modules require 3 control modules). The traditional approach has a smaller margin of efficiency (see Fig. 5) with an increase in hardware costs by 300% (quorum need triple hardware costs).

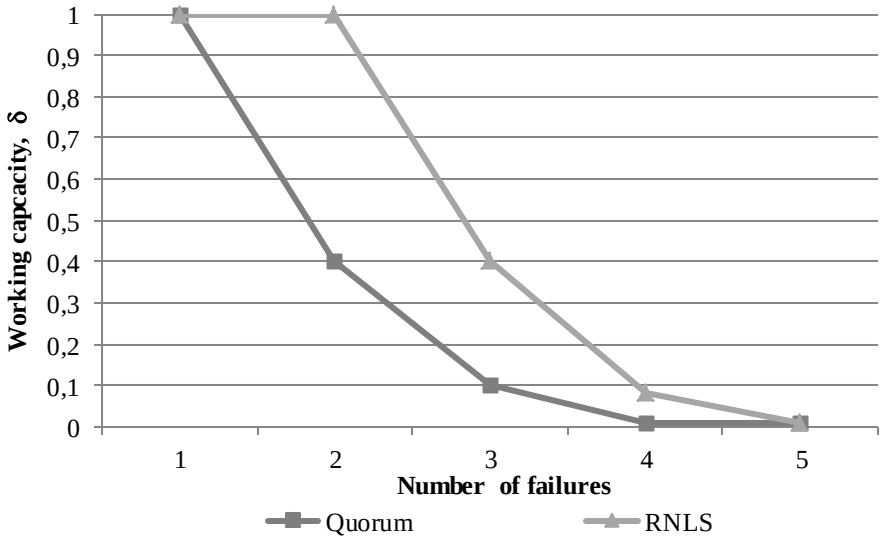

Fig. 5. Reliability assessment

5 Conclusion

The analysis of the problems of modern high-performance computing presented in this article has revealed difficulties in further improving the characteristics without introducing changes on the fundamental level of the principles of computation.

One possible solution is the use of forward-looking residue logarithmic number system.

Its advantages allowed us to improve the technological capabilities of specialized arithmetic devices.

In particular:

- multiplicative operations on each base have the same hardware structure, that is, they consist of the same set of simple blocks;
- the computational element is universal within a set of the same types of bases, which is the basis for building highly reliable schemes;
- the code lag is realizable both at the level of a separate computational element and a modular code in general, which opens prospects for the development of highly reliable computing structures.

The coprocessor considered in this paper functions on the basis of RLNS. To eliminate the disadvantages of non-traditional numbering systems associated with the slow implementation of non-modular (in the part of the RNS) and additive (in the part of the LNS) operations special methods and parallel-pipelined devices based on them were developed.

All this allowed us to increase the speed, the accuracy and the reliability of mass arithmetic calculations at a comparable level of hardware costs.

Future work will be focused on testing the applicability of the coprocessor using for different classes of applied problems. It is possible to provide some evaluated results on synthetic benchmarks and compare it to general purpose processor.

References

1. 754-2008 – IEEE Stander for floating-Point Arithmetic. Revision of ANSI/IEEE Std 754-1985, http:// ieeexplore.ieee.org, last accessed 2018/04/12
2. Arnold M. G.: The residue logarithmic number system: theory and implementation. In: 17th IEEE Symposium on Computer Arithmetic, pp. 196 – 205 (2002)
3. Chervyakov N.I.: Modular parallel computing structures of neuroprocessor systems. Fizmatlit Publishers, Moscow (2003).
4. Knyazkov V.S., Isupov K.S.: Parallell Multiple-Precision Arithmetic Based on Residue Number System. Program system: theory and applications, vol. 7 (28), pp. 61-97 (2016).
5. Garner H.: Number Systems and Arithmetic. In: Advances in Computers, vol. 6, pp. 131-194 (1966).
6. Omondi A.: Residue Number System: Theory and Implementation. Imperial College Press, London (2007).

7. Osinin I.P.: The Organization of a Parallel-Pipelined VLSI-Processor for Direct Conversion of Numbers Based on the Arithmetic of the Bit-Slices. In: News of higher educational institutions. The Volga Region. Technical Science, vol. 4 pp.5-13 (2014).
8. Isupov K.S.: Algorithms for estimating modular numbers in floating-point arithmetic. In: Science. Innovation. Technologies., vol. 4 pp. 44-56 (2016).
9. Coleman J. N., Chester, E. I. Softley C. I., Kadlec J.: Arithmetic on the European Logarithmic Microprocessor. In: IEEE Transactions on Computers, pp. 702-715 (2000). doi: 10.1109/12.863040
10. Arnold M., Bailey T., Cowles J., Winkel M.: Arithmetic Co-Transformations in the Real and Complex Logarithmic Number Systems. In: IEEE Transactions on Computers, vol. 47, pp. 777-786 (1998). doi: 10.1109/12.709377
11. Coleman J. N.: Simplification of Table Structure in Logarithmic Arithmetic. In: Electronic Letters, vol. 31, pp. 1905-1906 (1995). doi: 10.1049/el:19951304
12. Lewis D. M.: An Architecture for Addition and Subtraction of Long Word Length Numbers in the Logarithmic Number System. In: IEEE Transactions on Computers, pp. 1325-1336 (1990). doi: 10.1109/12.61042
13. Osinin I.P.: Optimization of the Hardware Costs of Interpolation Converters for Calculations in the Logarithmic Number System. In: Proceedings of the International Scientific Conference Parallel Computational Technologies (PCT'2018). pp. 153-164 (2018)
14. Osinin I.P.: Method and Device of Parallel-Pipelined Arithmetical Computers in Modular-Logarithmic System. In: Eurasian Union of Scientists, vol. 3 (48), pp. 45-56 (2018).
15. Osinin I.P.: Method and Device of Direct Parallel-Pipelined Transformation of Numbers with Floating Point in Modulr-Logarithmic Format. In: Eurasian Union of Scientists, vol. 3 (48), pp. 56-69 (2018).
16. Osinin I.P.: Method and Device for the Backward Parallel-Pipelined Transformation of Modular-Logarithmic Numbers in a Format with a Floatinng Point. In: Eurasian Union of Scientists, vol. 3 (48), pp. 70-83 (2018).
17. Osinin I.P.: A Modular-Logarithmic Coprocessor Concept. In: Proceedings International Conference on High Performance Computing & Simulation (HPCS-2017). pp. 588-595 (2017). doi: 10.1109/HPCS.2017.93
18. Intel 64 and IA-32 Architectures Optimization Reference Manual, http://www.intel.com, last accessed 2018/04/12.