Proc. of the Intl. Conf. on Advances In Computing, Electronics and Electrical Technology - CEET 2014.

ISBN: 978-1-63248-005-7 doi: 10.15224/ 978-1-63248-005-7-59

Principles of Recursive Residue Number System Computation

A.L. Stempkovsky, V.M. Amerbaev, R.A. Solovyev, T.Y. Isaeva, E.S. Balaka,

D.V. Telpukhov

***Abstract* — New method is proposed which is based on the idea of expressing system of residue numbers of traditional Residue Number System (RNS) using lower dimension system of sub-moduli. This new recursive data representation allows eliminating some of the known drawbacks of RNS arithmetic. Despite the constraints imposed on moduli sets, the proposed method provides speed gain, as experiments show, and it can be used in parallel high speed computing devices.**

***Keywords* — Residue Number System, Residue arithmetic, Chinese Remainder Theorem**

I. Introduction

This paper presents development of some constructional ideas which were presented as generalized schematic model in useful model patent [1]. In computer science history there were cases when required speed performance and reliability [2, 3] could not be efficiently provided for specialized devices design with the means of traditional positional binary arithmetic. At the same time, this problem could be solved by means of RNS computation. RNS arithmetic is not a universal method of calculator design, however, it is irreplaceable for some specialized applications. Therefore, it keeps attracting interest throughout many past decades. Research results, that could help overcome RNS disadvantages and widen its application field, are constantly published, actually, there exists separate research area dealing with this problem [4-7].

There are well-known advantages of RNS arithmetic as applied to digital signal processor (DSP) design:

- natural parallelism of computing;
- self-test and fault correction ability.

Also the following disadvantageous properties of RNS are known:

1) big overhead (usage of forward converters from conventional notation to RNS representation, and reverse converters);
2) representation of modular operations using operations of positional (binary) arithmetic that leads to area redundancy during implementation;
3) non-uniformity of modular calculators concerning their complexity and operation time (performance);
4) absence of due CAD support for design of RNS calculator devices (by systems of structural synthesis)[8].

The first disadvantage can be leveled if comparatively complex calculators are designed. As hardware cost of converters is restricted by design rules (project limits), this overhead can be lessened with the growth of total design complexity. The same is applicable to time expenditures.

The fourth disadvantage can be mastered by the usage of so-called IP-generators – program modules that produce RTL level synthesizable behavioral description of devices carrying out certain RNS (or maybe non-RNS) procedures.

There are no known methods of the essence to jump over the second and the third disadvantages. It is exactly these difficulties that our new approach to RNS calculator design, which is named recursive RNS arithmetic, is aimed to master.

Ideas proposed in this paper are based on the principles of deep paralleling of modular operations of RNS with moduli set $p_1, p_2, \ldots, p_n$ by reducing of modular operations for each of the used moduli p_i $(i \le j \le n)$ to modular computations using the previous used moduli $p_1, p_2, \ldots, p_{i-1}$, which have this or that technological advantage (e.g., small bit width). We will refer to the latter as basic moduli. Note that the mentioned reduction is possible only if the so-called condition of calculation limits matching for each of the used moduli pi with calculation limits of the corresponding sets of basic moduli, is true. The principle of calculation limits matching guarantees, first, isomorphism of ring operations for their corresponding sets of basic moduli; second, possibility of reversion of each step of recursion by conversion of corresponding RNS basic moduli codes to positional code (e.g., basing on Chinese remainder theorem, or by their transformation to Mixed-Radix system).

II. The idea of recursive RNS

Let us explain procedure of recursive conversions using the following simple example. Take two-bit prime integers $p_1 = 2$, $p_2 = 3$ as basic moduli. It is evident that any modulo 5 residue has unique representation by moduli 2 and 3. At the same time, residues for moduli 2, 3 and 5, where modulo 5 residues are expressed via moduli 2 and 3 residues, can uniquely represent any modulo 29 residue. Residues for moduli 2, 3, 5 and 29 can uniquely represent any modulo 863 residue. And so on, until we get the required set of used moduli: 2, 3, 5, 29, 863,... This example clearly illustrates the following 4 facts:

A.L. Stempkovsky , V.M. Amerbaev , R.A. Solovyev, T.Y. Isaeva, E.S. Balaka, D.V. Telpukhov

The Institute for Design Problems in Microelectronics of the Russian Academy of Science
Russia

Proc. of the Intl. Conf. on Advances In Computing, Electronics and Electrical Technology - CEET 2014.

ISBN: 978-1-63248-005-7 doi: 10.15224/ 978-1-63248-005-7-59

- Complexity and time use of number conversion using basic moduli 2 and 3 are approximately equal (bit width equals 2 for both basic moduli);
- Higher degree of paralleling is achieved;
- Regularity is improved (all calculators use moduli 2 and 3);
- Small bit width for basic moduli provides effective combinational implementation of modular operations for these basic moduli.

In reality not all these features are as good as they appear to be at the first glance. Actually, there are several constraints to be applied which add to complexity of devices designed using the proposed methodology. Let us consider these constraints.

Let be the set of basic moduli $p_1, p_2, \ldots, p_m$, and modulo p_{m+1} residues are to be expressed using this set of basic moduli. It is obvious that the maximal modulo p_{m+1} residue is max = $p_{m+1}-1$. Knowing this value and the sequence of operations to do, maximal value MAX of arithmetic operation result can be estimated. Clear that for unique representation of arithmetic operation result it is necessary that MAX <Q, where Q= $p_1 \cdot p_2 \cdot \ldots \cdot p_m$. For the rest of moduli the same estimation procedure applies.

Now consider the previous example for the basic moduli 2 and 3. In this case Q = 2·3 = 6. Minimal prime integer (after 2 and 3) is 5 (max = 4). We cannot implement neither addition because 2·max > Q (8 > 6), nor multiplication, as $\max^2 > Q$ (16 > 6). To implement any of these arithmetic operations (addition or multiplication) value Q should be increased (that is, number or/and value of basic moduli should be increased). Consider the following set of basic moduli: all mutually prime 3-bit integers, i.e. 4, 5 и 7. In this case Q = 4·5·7 = 140. Condition necessary to provide MAX < Q for multiplication is $\max^2 < Q$, in other words, $p_{i-1} < \sqrt{Q}$ ($p_{i-1} < 11{,}8$). Set p_i=11 (prime integer nearest to 7). Generation of basic moduli can be easily continued using the same procedure until the required computational range is achieved.

Finally consider a problem from practice. Let we have to implement Fourier transform for 24-bit arguments and 1024 nodes. To do this, the sum of 1024 products should be calculated, that is, the following constraint should hold: $1024 \cdot \max^2 < Q$. For this task a set of 3-bit basic moduli is not enough. Let us add 4-bit moduli to the basic set: 5, 7, 8, 9, 11 and 13 (Q = 360360). To chose the nearest used modulus it is necessary that $p_i - 1 < \sqrt{\frac{Q}{1024}}$. That is, we get $p_i - 1 < 32$. Let p_i=31. Use the same estimation procedure to generate the whole tree of the used moduli. To implement such device, a block of 6 calculator units (one for each basic modulus) should be designed. 16 of these units are required. That is where regularity comes in useful. Note that all calculators are extremely fast for modular operations (due to super-paralleling) and low area usage due to small values of basic moduli or their closeness to power of 2 numbers.

As such, the proposed method of recursive RNS calculation provides the following advantages:

- Elimination of non-singularity for operating small and large moduli (cost (area) and performance are approximately the same for all moduli, as, ideally, they should have the same bit width).
- Sufficiently higher degree of paralleling resulting in better speed performance.
- Regularity appears (great number of equal moduli are used).
- Small bit width of basic moduli allows for modular operations implementation by combinational circuits optimized in Boolean basis.

III. Data representation and main operations of recursive RNS

Recursive RNS is based on the following idea: express the moduli set in terms of set of smaller dimension sub-moduli.

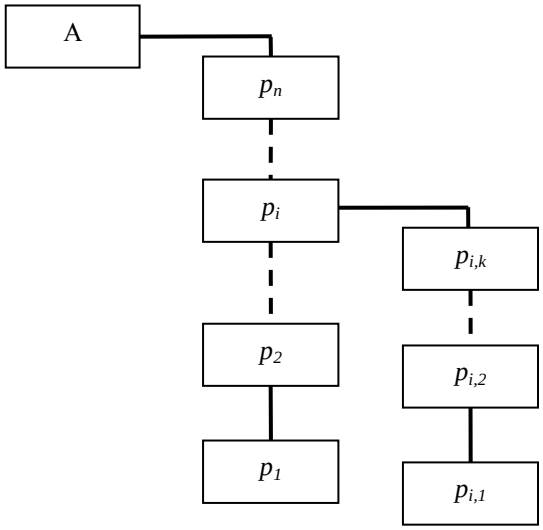

Figure 1. Recursive RNS hierarchy

Consider moduli set $p_1, p_2, \ldots, p_i, \ldots, p_n$ and a vector A=($a_1, a_2, \ldots, a_n$). Let us express a_i in terms of sub-moduli set ($p_{i,1}, p_{i,2}, \ldots, p_{i,k}$) , where $P_i = p_{i,1} \cdot p_{i,2} \cdot \ldots \cdot p_{i,k}$ and $a_i < P_i$: $a_i = (a_{i,1}, a_{i,2}, \ldots, a_{i,k})$. Then vector A can be expressed as follows: $(a_1, a_2, \ldots, a_{i-1}, (a_{i,1}, a_{i,2}, \ldots, a_{i,k}), a_{i+1}, \ldots, a_n)$, see Figure 1. Let us name the set of the first m moduli the basic moduli set, and denote their product Q = $p_1 \cdot p_2 \cdot \ldots \cdot p_m$. Let $p_{i,1} = p_i$, $p_{i,2} = p_2, \ldots, p_{i,i-1} = p_{i-1}$, and $k = i-1$, then for $i = m+1, \ldots, n$ recursion takes place. The i-th element $a_i = (a_1, a_2, \ldots, a_m, a_{m+1}, \ldots, a_{i-1})$ for moduli set $p_1, p_2, \ldots, p_m, p_{m+1}, \ldots, p_{i-1}$ or, expanding recursion, we get: $a_i = (a_1, a_2, \ldots, a_m, (a_{m+1,1}, a_{m+1,2}, \ldots, a_{m+1,m}), \ldots)$. Figure 2 shows a special case of element ai decomposition for n = 6 and m = 3.

Proc. of the Intl. Conf. on Advances In Computing, Electronics and Electrical Technology - CEET 2014.

ISBN: 978-1-63248-005-7 doi: 10.15224/ 978-1-63248-005-7-59

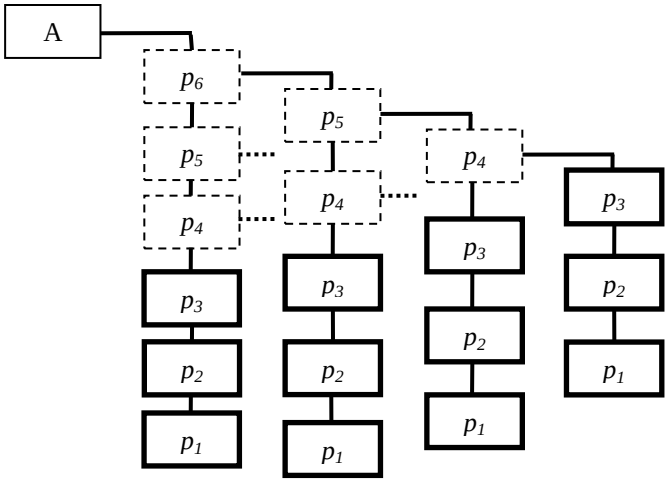

Figure 2. Recursive decomposition of element p6 using sub-moduli set (p_1, p_2, p_3)

A. *Forward conversion of positional number notation to recursive RNS notation.*

While in traditional RNS arithmetic number of vector elements is equal to the number of moduli set elements, in recursive RNS arithmetic number of vector elements grows as function of n and m.

Clear that any of the first m elements is presented by a single number. *m+1*-th element contains m elements, as it is expressed by *m* elements of submoduli set: $a_{m+1}=(a_{m+1,1},a_{m+1,2},\ldots,a_{m+1,m})$. *m+2*-th element contains $2\cdot m$ elements, as it is expressed by m elements of sub-moduli set and m elements of vector a_{m+1}. Thus, we come to conclusion that the number of elements L_i for vector ai can be expressed by the following formula:

$$L_i = \begin{cases} 1, & i \le m, \\ 2^{i-m-1} \cdot m, & m < i \le n. \end{cases} \quad (1)$$

The total number L for vector A can be evaluated using geometric series formula [9]:

$$L = \sum L_i = m + m \cdot \left(2^0 + 2^1 + \ldots + 2^{n-m-1}\right) = m \cdot \left(1 + \frac{2^{n-m}-1}{2-1}\right) = 2^{n-m} \cdot m \quad (2)$$

Consider a numeric example. Consider the moduli set: p_1, p_2, p_3, p_4, p_5 = *(2, 3, 5, 29, 863)*. $P = 2\cdot3\cdot5\cdot29\cdot863 = 750810$. Check the condition: $2\cdot3 > 5$, $2\cdot3\cdot5 > 29$, $2\cdot3\cdot5\cdot29 > 863$. Choose basic moduli set (p_1, p_2). Then Q = $p_1\cdot p_2$ =6, n = 5, m=2. Number of vector elements is: L = $23\cdot2 = 16$.

Decompose number A = 865 written in positional notation to traditional RNS basis vector: A = ($|865|_2$, $|865|_3$, $|865|_5$, $|865|_{29}$, $|865|_{863}$) = (1,1, 0, 24,2).

Now decompose number A for recursive RNS basis:

$$A = \left(|865|_2, |865|_3, \left(|865|_{5|2}, |865|_{5|3}\right), \left(|865|_{29|2}, |865|_{29|3}, \left(|865|_{29|5|2}, |865|_{29|5|3}\right)\right), \left(|865|_{863|2}, |865|_{863|3}, \left(|865|_{863|5|2}, |865|_{863|5|3}\right), \left(|865|_{863|29|2}, |865|_{863|29|3}, \left(|865|_{863|29|5|2}, |865|_{863|29|5|3}\right)\right)\right)\right) = (1,1,(0,0),(0,0,(0,1)),(0,2,(0,2),(0,2,(0,2))))$$

As all numbers in the resulting vector do not exceed 3, the needed storage capacity is 16·2 = 32 bits, instead of 20 bits for positional notation number storage. Amount (coefficient) of redundancy is 1.6. Fig. 3 illustrates this example.

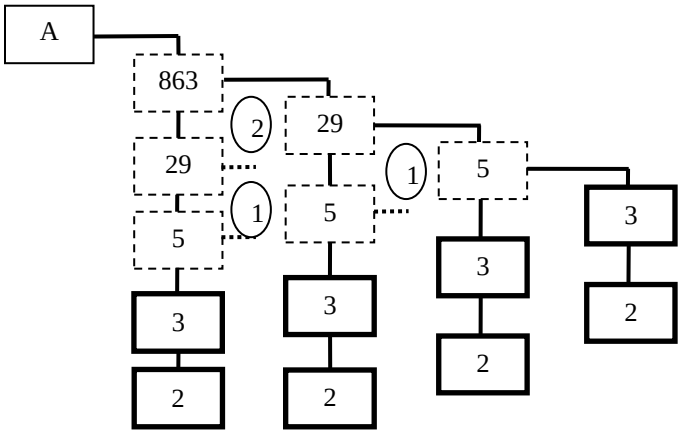

Figure 3. Number decomposition for recursive RNS basis (2, 3, 5, 29, 863) using basic moduli set (2, 3)

B. *Basis choice restrictions*

Maximal value that can be presented in terms of p_{m+1} is max = $p_{m+1}-1$. To enable arithmetic operations for certain numbers operation result for element p_{m+1} should not exceed Q=($p_1 \cdot p_2 \cdot \ldots \cdot p_m$). This value equals 2·max for addition and max2 for multiplication.

Look upon the following example. Consider basis (2, 3, 5). We take the basic moduli set (2, 3). In this case Q = 2·3 = 6, max = 4. As addition requires maximal representable number 8, that is greater than 6, and multiplication requires number 16, that is also greater than 6, this basis is inappropriate for basic arithmetic operations even though it can be used for one-to-one number decomposition. For practical needs value of Q should be increased.

How to choose basic elements? Consider the following example. Consider basic moduli set (4, 5, 7). The element p_4 should be chosen so that the resulting recursive basis enables multiplication.

$$Q > MAX \Rightarrow Q > \max^2 \Rightarrow Q > (p_4 - 1)^2 \Rightarrow p_4 < \sqrt{Q} + 1 \Rightarrow$$

$$\Rightarrow p_4 < \sqrt{140} + 1 \Rightarrow p_4 < 12.8$$

Thus, we can add p_4 = 11 to recursive basis.

Proc. of the Intl. Conf. on Advances In Computing, Electronics and Electrical Technology - CEET 2014.

ISBN: 978-1-63248-005-7 doi: 10.15224/ 978-1-63248-005-7-59

C. *Reverse number conversion from recursive RNS notation to positional notation*

Reverse conversion needs recursive implementation based on the same method that is used for conversion of traditional RNS vector to positional notation number.

Consider arbitrary vector A = (a_1, a_2,..., a_n). It is known for RNS that A = (a_1, a_2,..., a_n) = (a_1,0 ,…,0) + (0, a_2 ,…,0) +…+ +(0,0 ,…, a_n) = $\left|\sum_{i=1}^{n} a_i \cdot B_i\right|_P$, where B_0=(1,0 ,…,0) , B_1=(0,1 ,…,0) ,…, B_n=(0,0 ,…,1) is a system of orthogonal bases [10].

For recursive RNS a set of orthogonal bases should be found for the following residue number systems: (p_1, p_2,..., p_m) , (p_1, p_2,..., p_{m+1}),…, (p_1, p_2 ,..., p_n) .

View one more example. Consider basis (3, 5, 7, 11, 13) and basic moduli set (3, 5, 7). Let us convert vector (1, 2, 1, (0, 3, 3), (0, 1, 6,(0, 1, 6))) to positional notation. For reverse conversion we have to find orthogonal bases for each of the following residue number systems:

$$S_1 = (3, 5, 7) \Rightarrow B_{1,1} = (1, 0, 0) \equiv 70;$$
$$B_{1,2} = (0, 1, 0) \equiv 21;\ B_{1,3} = (0, 0, 1) \equiv 15;$$
$$S_2 = (3, 5, 7, 11) \Rightarrow B_{2,1} \equiv 385;\ B_{2,2} \equiv 231;$$
$$B_{2,3} \equiv 330;\ B_{2,4} \equiv 210;$$
$$S_3 = (3, 5, 7, 11, 13) \Rightarrow B_{3,1} \equiv 5005;$$
$$B_{3,2} \equiv 6006;\ B_{3,3} \equiv 10725;$$
$$B_{3,4} \equiv 1365;\ B_{3,5} \equiv 6930.$$

The process of reverse conversion is shown at fig.4. In positional notation the given vector is presented as 13357.

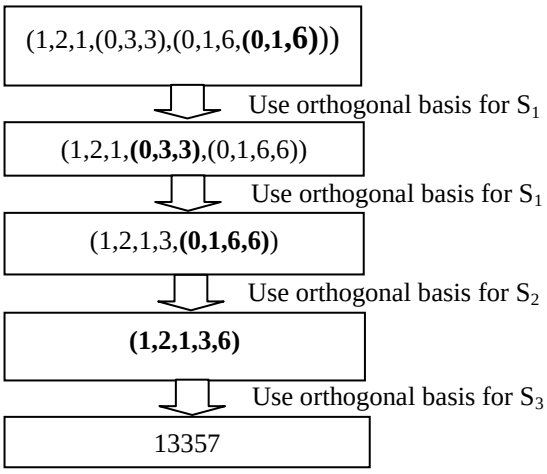

Figure 4. Process of vector converse reversion

D. *Addition and multiplication in recursive RNS arithmetic*

If constraints hold for a basis (see paragraph 3B), then number addition and multiplication are done similar to traditional RNS arithmetic. To add (multiply) two numbers, corresponding vector elements should be added (multiplied) modulo pi. As all vector elements are small and use few digits, parallel addition (multiplication) is very fast.

IV. Experimental results

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

Experiments were done to compare speed performance of scalar multiplication of vectors for three different implementations: first, positional notation; then, traditional RNS basis notation; and, finally, recursive RNS basis notation.

Device models were implemented by means of digital Integrated Circuits design flow using standard cell libraries. The chosen flow includes Verilog HDL behavioral description of the device; Synopsys Design Compiler logic synthesis tools; Synopsys PrimeTime static timing analysis tools; standard cell library Nangate Open Cell library for 45nm design rules.

In the designed recursive RNS device forward conversion is implemented as pipeline, thus, forward conversion delay for given parameters is always less than the delay for the main body of scalar multiplication. Reverse conversion takes rather a long time, however, even for the most difficult cases it is finished long before a new portion of data comes to input of the reverse converter unit, due to the fact that the number of clock cycles assigned for reverse conversion is equal to the number of vector elements.

Thus, the device frequency is defined by the maximal delay unit, i.e., the main body of scalar multiplier.

Let each vector consist of 1024 elements and have 20-bit arguments. Then the sum of 1024 products is to be calculated, i.e., the following condition must hold: $1024 \cdot max^2 < Q$. To provide this, 3-bit basic moduli set is not enough. Let us add 4-bit moduli: 5, 7, 8, 9, 11 and 13 (Q = 360360). Use formulas from p.4 to chose the nearest modulo value. We get $p_7 < 18$. Chose $p_7 = 17$. Construct the whole moduli tree using this algorithm: (5, 7, 8, 9, 11, 13, 17, 73, 659, 16963). To implement device for this moduli tree, a unit (block) containing 6 calculators (one for each basic modulus) should be designed. 16 of these units are required. This is where regularity comes in useful. Note that all calculators are extremely fast (due to super-paralleling) and low cost (area usage) due to small values of basic moduli.

Proc. of the Intl. Conf. on Advances In Computing, Electronics and Electrical Technology - CEET 2014.

ISBN: 978-1-63248-005-7 doi: 10.15224/ 978-1-63248-005-7-59

Consider the combinational part of the synchronous scalar multiplication circuit (see fig.5).

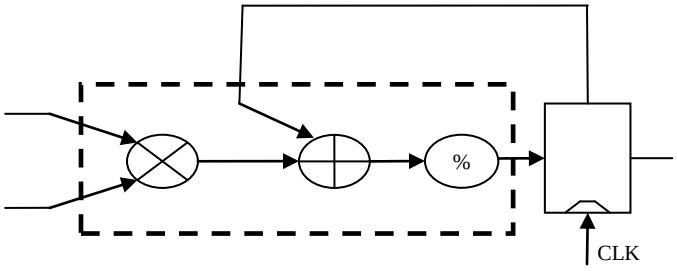

Figure 5. Combinational part of vectors scalar multiplication circuit

Clock frequency of the device depends on the critical path length for this part which contains three operations: multiplication, addition and residue calculation.

Several flow devices for scalar multiplication of vectors were designed. Each RNS device consists of three parts: forward conversion from positional notation, main body for scalar multiplication and reverse conversion from recursive RNS notation to positional notation. Clock frequency is defined by the slowest circuit part. In our case it is scalar multiplication part. Such parameters as maximal clock frequency and total device area were estimated for three different computing methods: traditional implementation using positional notation, RNS implementation and recursive RNS implementation. Tables I, II and III present the results.

TABLE I. BASIC DATA

Vector length	Data bit width	Moduli set	
		RNS	***Recursive RNS***
512	16	7, 11, 13, 16, 17, 19, 23, 27, 29, 31	basic: 5, 7, 8, 9, 13, 17 complementary: 31, 181, 709
1024	20	37, 41, 43, 47, 53, 59, 61, 63, 64	basic: 5, 7, 8, 9, 11, 13 complementary: 17, 73, 659, 16963
2048	24	5, 7, 17, 31, 61, 67, 71, 73, 113, 127, 128	basic: 13, 17, 19, 23, 29, 31 complementary: 199, 2903, 11497

TABLE II. DELAY ANALYSIS

Vector length	Data bit width	Clock frequency (MHz)		
		Positional notation	***RNS***	***Recursive RNS***
512	16	409	726	855
1024	20	346	581	986
2048	24	294	537	794

TABLE III. AREA ANALYSIS

Vector length	Data bit width	Combinational part of the synchronous scalar multiplication circuit		
		Positional notation	***RNS***	***Recursive RNS***
512	16	3119	3928	11443
1024	20	5040	5857	22745
2048	24	6915	6742	26038

v. Disadvantages of recursive RNS arithmetic: possible ways to overcome

Main disadvantages of the proposed method are as follows: 1) area overhead; 2) complication of forward and reverse conversion unit from and to positional notation (complexity of the other non-RNS operations also grows); 3) restrictions on moduli bases choice; 4) restrictions on number of operations in sequence (without recursion applied).

Some of the drawbacks of recursive RNS arithmetic immediately follow from the fact that the used flow includes residue operations for comparatively long numbers. In traditional RNS arithmetic sets of small moduli or some special moduli are used as a rule. So, recursive RNS arithmetic can be further improved by usage of Mercenn numbers [11] or/and special numbers $2^n \pm k$ which require less area for implementation of residue calculation.

This work was partially supported by a grant from RFBR № 13-07-00241a.

References

[1] Residue number system calculation device. Useful model patent No. 103010 Russian Federation, IPC G06F7/72. Assignee: IPPM RAS. App. No. 2010148522; app.: 29.11.2010; reg.: 20.03.2011J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] S.V. Gavrilov, O.N. Gudkova, A.L. Stempkovskiy. The Analysis of the Performance of Nanometer IP-blocks Based on Interval Simulation. // Russian Microelectronics, Vol.42, N7, 2013, P. 396–402. © Pleiades Publishing, Ltd., 2013.

[3] Alexander Stempkovsky, Alexey Glebov, Sergey Gavrilov "Calculation of Stress Probability for NBTI-Aware Timing Analysis" // Proc. of ISQED, 2009, p.714-718.

[4] I. Ya. Akushskij and D. I. Yuditskij, Residue Number System Computer Arithmetic. – Moscow.: Sovetskoe Radio, 1968. – 440pp

[5] N. S. Szabo and R. I. Tanaka, Residue Number System and its applications to Computer Technology, McGraw-Hill, New York, 1967.

[6] M. A. Soderstrand et al, (Eds), Residue Number System Arithmetic: Modern Applications in Digital Signal Processing, IEEE Press, NY, 1986

[7] Amos Omondi, Benjamin Premkumar «Residue Number Systems: Theory and Implementation», 2007.

[8] Gavrilov S., Glebov A., Pullela S., Moore S., Vijayan G., Dharchoundhury A., Panda R., Blaauw D. Library-Less Synthesis for Static CMOS Combinational Logic Circuits // Proc. of IEEE/ACM International

ISBN: 978-1-63248-005-7 doi: 10.15224/ 978-1-63248-005-7-59

Conference on Computer Aided Design (ICCAD-97). - San Jose, CA, USA, November 9-13, 1997. - P. 658-662.

[9] Vygotskij M.Ya. «Elementary mathematics reference guide», Moscow, 2006.

[10] B. Tseng, G. A. Jullien, W. C. Miller. 1992. Implementation of FFT structures using the residue number systems. IEEE Transactions on Computers, 28(11).

[11] http://en.wikipedia.org/wiki/Mersenne_prime.

About Author (s):

A.L. Stempkovsky has degree of doctor of engineering sciences, professor, academician of the Russian Academy of Sciences (RAS) (2006). Dr.Stempkovsky is the winner of the State Awards of the Russian Federation in the field of science (2003); the author of 145 scientific publications including a number of inventions and three monographies. His achievements have been rewarded by the Honor Award and the Friendship Award of the Russian Federation. Dr. Stempkovsky is one of the leading scientists of Russia in the field of Computer Aided Design (CAD) for micro- and nanoelectronic equipment. During only last ten years, 99 research and development programs were carried out by the Institute under his management. The employees of the Institute have published more than 420 articles and reports in proceedings of the Russian and international conferences, nine employees have defended Ph.D. thesis and two - doctoral thesis.

V.M. Amerbaev received his MS degree in Math from Al-Farabi Kazakh National University, Kazakhstan, Almaty, in 1954 and Ph.D degree in Math from Steklov Institute of Mathematics, USSR, Moscow in 1963. He is currently a chief scientific officer in the Electrical and Computer Engineering Department at Institute for Design Problems in Microelectronics, Russia, Moscow. His current research interests are in residue number systems, reliablity and VLSI.

R.A. Solovyev received his MS degree in Computer Systems Engineering from National Research University of Electronic Technology, Russia, in 2004 and Ph.D degree in Electrical Engineering from Institute for Design Problems in Microelectronics, Russia in 2007. He is currently department head in the Electrical and Computer Engineering Department at Institute for Design Problems in Microelectronics, Russia, Moscow. His current research interests are in residue number systems, highspeed hardware architectures, and VLSI.

T.Y. Isaeva received her MS degree in Computer Systems Engineering from Lomonosov Moscow State University, Russia and Ph.D degree in Russia in 2002. is currently a Researcher in the Electrical and Computer Engineering Department at Institute for Design Problems in Microelectronics, Russia, Moscow. Her current research interests are in residue number systems, highspeed hardware architectures, and VLSI.

E.S. Balaka received his master`s degree in Technics and Technology from National Research University of Electronic Technology, Russia, in 2010. She is currently a Researcher in the Electrical and Computer Engineering Department at Institute for Design Problems in Microelectronics, Russia, Moscow. Her current research interests are in residue number systems, highspeed hardware architectures, and VLSI.

D.V. Telpukhov received his master`s degree in Technics and Technology from National Research University of Electronic Technology, Russia, in 2009 and Ph.D degree in Electrical Engineering from Institute for Design Problems in Microelectronics, Russia in 2013. He is currently a Researcher in the Electrical and Computer Engineering Department at Institute for Design Problems in Microelectronics, Russia, Moscow. Diploma For the best Regular Paper on EWDTS'13 Symposium and outstanding contribution in Design & Test (2013). Winner in the fair of scientific and technological ideas and projects of youth "RHYTHM Zelenograd» 2011. His current research interests are in residue number systems, highspeed hardware architectures, and VLSI.