Aleksandr CARIOW, Galina CARIOWA

West Pomeranian University of Technology, Szczecin

doi:10.15199/48.2015.02.09

An unified approach for developing rationalized algorithms for hypercomplex number multiplication

***Abstract**. In this article we present a common approach for the development of algorithms for calculating products of hypercomplex numbers. The main idea of the proposed approach is based on the representation of hypernumbers multiplying via the matrix-vector products and further creative decomposition of the matrix, leading to the reduction of arithmetical complexity of calculations. The proposed approach allows the construction of sufficiently well algorithms for hypernumbers multiplication with reduced computational complexity. If the schoolbook method requires N^2 real multiplications and $N(N-1)$ real additions, the proposed approach allows to develop algorithms, which take only $[N(N-1)/2]+2$ real multiplications and $3N\log_2N+[N(N-3)+4]/2$ real additions.*

***Streszczenie.** W artykule zostało przedstawione uogólnione podejście do syntezy algorytmów wyznaczania iloczynów liczb hiperzespolonych. Główna idea proponowanego podejścia polega na reprezentacji operacji mnożenia liczb hiperzespolonych w formie iloczynu wektorowo-macierzowego i dalszej możliwości kreatywnej dekompozycji czynnika macierzowego prowadzącej do redukcji złożoności obliczeniowej. Proponowane podejście pozwala zbudować algorytmy wyróżniające się w porównaniu do metody naiwnej zredukowaną złożonością obliczeniową. Jeśli metoda naiwna wymaga wykonania N^2 mnożeń oraz $N(N-1)$ dodawań liczb rzeczywistych to proponowane podejście pozwala syntetyzować algorytmy wymagające tylko $[N(N-1)/2]+2$ mnożeń oraz $3N\log_2N+[N(N-3)+4]/2$ dodawań. (**Uogólnione podejście do konstruowania zracjonalizowanych algorytmów mnożenia liczb hiperzespolonych – tytuł polski artykułu**).*

Keywords: hypernumbers, multiplication, fast algorithms.
Słowa kluczowe: liczby hiperzespolone, operacja mnożenia, szybkie algorytmy.

Introduction

In recent years the hypercomplex numbers [1] (hypernumbers) play an important role during the realization of several tasks of data processing in various fields of science and technology including electrodynamics [2], digital signal and image processing [3-4], computer graphics and machine vision [5, 6, 7], telecommunication [8] and public key cryptography [9]. Among other arithmetical operations in the hypercomplex algebras, multiplication is the most time-consuming one. The reason for this is, because the addition of two N-dimensional hypernumbers requires only N real additions, whereas the multiplication of these hypernumbers already requires $N(N-1)$ real additions and N^2 real multiplications. It is easy to see that the increase of dimension of hypernumber increases the computational complexity of its multiplication. This in turn leads to an increase in the computation time. Therefore, the reduction of the computational complexity of the multiplication of hypercomplex numbers is an important scientific and engineering problem. Efficient algorithmic solutions for the multiplication of hypercomplex numbers already exist [10, 11]. However, the tricks used to obtain them were different in each case. In this paper, we describe a unified approach to the development of such algorithms.

Formulation of the problem

A hypercomplex number is defined as follows [1]

$$S = s_0 + \sum_{l=1}^{N-1} s_i e_i \,, \tag{1}$$

where s_0 and $\{s_i\}, i=1,...,N-1$ are real numbers and $\{e_i\}$ $i=1,...,N-1$ are imaginary units, N – is dimension of hyper-complex number.

Suppose we need to compute the product of two hypercomplex numbers. The following notations are used:

$$A = a_0 + \sum_{l=1}^{N-1} a_i e_i \,,\ B = b_0 + \sum_{l=1}^{N-1} b_i e_i \,,\ C = AB = c_0 + \sum_{l=1}^{N-1} c_i e_i \,,$$

where $\{a_i\}$, $\{b_i\}$, $\{c_i\}$, $i=0,...,N-1$ are real coefficients of the hypernumbers A, B and, C respectively.

We can write the product of two hypercomplex number briefly as a matrix–vector multiplication [12]:

$$\mathbf{C}_{N\times1} = \mathbf{B}_N \mathbf{A}_{N\times1} \tag{2}$$

where

$$\mathbf{A}_{N\times1} = [a_0, a_1, ..., a_{N-1}]^{\mathrm{T}} \,,\ \mathbf{C}_{N\times1} = [c_0, c_1, ..., c_{N-1}]^{\mathrm{T}} \,,$$

and $\mathbf{B}_N$ - is the matrix (consisting of the elements $\{b_i\}$) that establishes one-to-one correspondence between the vector $\mathbf{A}_{N\times1}$ and the vector $\mathbf{C}_{N\times1}$. The direct multiplication of the vector–matrix product in Eq. (2) requires N^2 real multiplications and $N(N-1)$ additions. Nevertheless, we can offer a simple and fairly universal technique for the synthesis of the hypernumber multiplication algorithms which have lower computational complexity.

The main idea

The idea of the proposed approach is based on the possibility of efficient decomposition of the matrix $\mathbf{B}_N$, which would lead to a reduction in the number of calculations required. However, as a rule, the initial matrix in its original form cannot be effectively decomposed. In this case, the matrix must be previously modified, but after modification the result has to be converted, so that it possesses a block-diagonal symmetry is adequate to the product of an unmodified matrix by a vector. It is necessary to say that the matrix $\mathbf{B}_N$ contains both positive and negative elements. If we do not take into account the signs, the matrix would have a structure that possesses a block-diagonal symmetry and then could be efficiently factorized. If this property does not hold once, then the necessary columns and/or rows of the matrix need first to be permuted. It should be remembered that when we permute columns, then it is necessary to permute the elements of a vector $\mathbf{X}_{N\times1}$ the same way. If we permute rows of the matrix, then it is necessary to permute the elements of the vector $\mathbf{Y}_{N\times1}$ in the same way. This is to ensure that the result of the calculation is correct. Thus, our task is to modify the matrix $\mathbf{B}_N$ so that the resulting matrix $\widetilde{\mathbf{B}}_N$ has the desired, block-structural properties.

Hence the primary generalized computational procedure for the multiplication of hypernumbers would look like this:

(3) $$\mathbf{Y}_{N\times 1} = \hat{\mathbf{P}}_N \hat{\mathbf{E}}_N \tilde{\mathbf{B}}_N \breve{\mathbf{E}}_N \breve{\mathbf{P}}_N \mathbf{X}_{N\times 1}$$

or like this:

(4) $$\mathbf{Y}_{N\times 1} = \hat{\mathbf{E}}_N \hat{\mathbf{P}}_N \tilde{\mathbf{B}}_N \breve{\mathbf{P}}_N \breve{\mathbf{E}}_N \mathbf{X}_{N\times 1},$$

where

$$\mathbf{X}_{N\times 1} = [a_0, a_1, ..., a_{N-1}]^{\mathrm{T}},\ \mathbf{Y}_{N\times 1} = [c_0, c_1, ..., c_{N-1}]^{\mathrm{T}},$$

$\hat{\mathbf{P}}_N$ and $\breve{\mathbf{P}}_N$ are columns' and rows' permutation matrices, $\hat{\mathbf{E}}_N$ and $\breve{\mathbf{E}}_N$ are columns' and rows' sign-change matrices.

Data flow diagrams which illustrate the algorithmic structures of computational processes, described by procedures (3) and (4), are shown in Figure 1 and Figure 2.

Straight lines in the figures denote the operations of data transfer. In turn, the rectangles indicate the matrix-vector multiplications by a matrix inscribed inside a rectangle. The circles in these figures show the operation of multiplication by a real number or a variable inscribed inside the circle. In this paper, the data flow diagrams are oriented from left to right. We use the usual lines without arrows on purpose, so as not to clutter the picture.

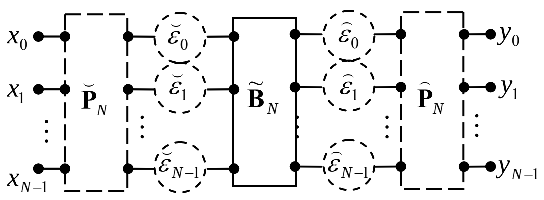

Fig.1. The data flow diagram representing the process of calculating hypernumber product in accordance with the procedure (3).

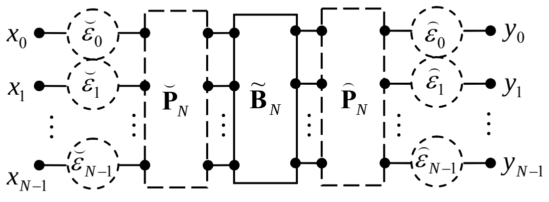

Fig.2. The data flow diagram representing the process of calculating hypernumber product in accordance with the procedure (4).

It must be emphasized that the presence or absence of matrices $\hat{\mathbf{P}}_N$ and $\breve{\mathbf{P}}_N$, in the procedures (3) and (4), depends on the need of the original matrix rows/columns permutation. The presence or absence of matrices $\hat{\mathbf{E}}_N$ and $\breve{\mathbf{E}}_N$ in these procedures depends on the need of change of signs of rows/columns elements in the original matrix. That is why these matrices (or their elements) are shown in the figures by dashed lines.

So, our task is to find such a decomposition of the matrix $\tilde{\mathbf{B}}_N$, which will reduce the computational complexity of the multiplication of this matrix by a vector. As to the specific ways of decomposition of this matrix, they may be different. Nevertheless, we can offer a unified trick for the construction of the fast algorithms for multiplication of the traditional hypernumbers, such as quaternions (*N*=4), octonions (*N*=8), sedenions (*N*=16), trigintaduonions (*N*=32), sexagintaquattuornions (*N*=64), centumduodetrigintanions (*N*=128) and ducentiquinquagintasexions (*N*=256).

The unified computation procedure for the low complexity hypercomplex numbers multiplication

Let us multiply the first row of $\mathbf{B}_N$ by (−1). (We can easily see that this transformation leads to the minimization of the computational complexity of the final algorithm in the future.) This transformation is done in order to present a matrix, modified in this manner, as an algebraic sum of the block-symmetric Toeplitz-type matrix and some sparse matrix, i.e. matrix containing only small number of nonzero elements. Then, a modified version $\tilde{\mathbf{B}}_N$ of the matrix $\mathbf{B}_N$ can be represented as the sum of a symmetric Toeplitz matrix $\breve{\mathbf{B}}_N$ and another matrix $\hat{\mathbf{B}}_N$ which has many zero elements:

(5) $$\tilde{\mathbf{B}}_N = \breve{\mathbf{B}}_N - \hat{\mathbf{B}}_N.$$

Then we can write

(6) $$\mathbf{Y}_{N\times 1} = \breve{\mathbf{B}}_N \mathbf{X}_{N\times 1} - \hat{\mathbf{B}}_N \mathbf{X}_{N\times 1}$$

The Toeplitz-type matrix $\breve{\mathbf{B}}_N$ is shift-structured and there exists a number of algorithms for the fast matrix–vector multiplication. For instance, the matrix can be diagonalized using the Fast Hadamard transform (FHT) and, matrix-vector products can be computed efficiently. Unfortunately, the computational complexity of the product $\hat{\mathbf{B}}_N \mathbf{X}_{N\times 1}$ cannot be reduced and this product is calculated directly, without any tricks. But, since the matrix $\hat{\mathbf{B}}_N$ is sparse, the number of real multiplications is fairly small. Clearly, the smaller the number of negative elements in the matrix $\tilde{\mathbf{B}}_N$, the smaller the real multiplications required for computing the product $\hat{\mathbf{B}}_N \mathbf{X}_{N\times 1}$.

It must be emphasized that the number of non-zero elements in the matrix $\hat{\mathbf{B}}_N$ depends on the number of the negative elements in the matrix $\mathbf{B}_N$. To minimize the number of the negative elements in the matrix $\tilde{\mathbf{B}}_N$ it is necessary to multiply some columns and/or rows of the matrix $\mathbf{B}_N$ by (-1). (In our case, it is multiplied by (-1) only the first row of the matrix $\tilde{\mathbf{B}}_N$, as already noted).

In view of the above arguments, the unified computational procedure that describes rationalized algorithm for hypercomplex numbers multiplication will look as follows:

(7) $$\mathbf{Y}_{N\times 1} = \tilde{\mathbf{E}}_N \mathbf{\Sigma}_{N\times 2N} \tilde{\mathbf{H}}_{2N} \mathbf{D}_{2N} \tilde{\mathbf{B}}_{2N} \mathbf{P}_{2N\times N} \mathbf{X}_{N\times 1}$$

$$\mathbf{P}_{2N\times N} = (\mathbf{1}_{2\times 1} \otimes \mathbf{I}_N),\ \tilde{\mathbf{B}}_{2N} = \mathbf{H}_N \oplus \hat{\mathbf{B}}_N,$$

$$\mathbf{D}_{2N} = diag(s_0, s_1, ..., s_{N-1}, \underbrace{2, 2, ..., 2}_{N\ times}),$$

$$\mathbf{H}_{2N} = \tilde{\mathbf{H}}_N \otimes \mathbf{I}_N,\ \mathbf{\Sigma}_{N\times 2N} = \mathbf{1}_{1\times 2} \otimes \mathbf{I}_N,$$

$\tilde{\mathbf{E}}_N = diag(\underbrace{-1, 1, ..., 1}_{N\ times})$ - is a diagonal matrix, obtained from an identity matrix by changing the sign of its first element, $\mathbf{1}_{M\times N}$ - is a $M\times N$ matrix of ones (a matrix where every element is equal to one), $\mathbf{I}_N$ - is an identity $N\times N$ matrix, signs „$\otimes$” and „$\oplus$” – denote tensor product and a direct sum of two matrices, respectively [13], and $\mathbf{H}_N$ - is $N\times N$

Hadamard matrix [14]. It must be emphasized that the fast calculation of the Hadamard matrix-vector product requires no multiplications of the real numbers.

The elements $\{s_i\}$ of the matrix $\mathbf{D}_{2N}$ can be calculated using the following matrix-vector procedure:

(8) $$\mathbf{S}_{N\times 1} = \frac{1}{N}\mathbf{H}_N\mathbf{B}_N \text{ , where } \mathbf{S}_{N\times 1} = [s_0, s_1, ..., s_{N-1}]^{\mathrm{T}}$$

Fig. 3 shows a data flow diagram representation of the rationalized algorithm for the computation of the two *N*-order hypercomplex numbers product and Fig. 4 shows a data flow diagram of the process for calculation of the matrix $\mathbf{D}_{2N}$ elements. In Fig. 3 points where lines converge denote summation and a dotted line indicates the sign-change operation.

An examination of Fig. 3 and Fig. 4 shows that the algorithm also contains the multiplication by numbers, which are different powers of the two. These operations are reduced to the ordinary shifts to the left (or right) on a number of positions. Because of the ease of implementation, these operations are usually not taken into account when assessing the computational complexity.

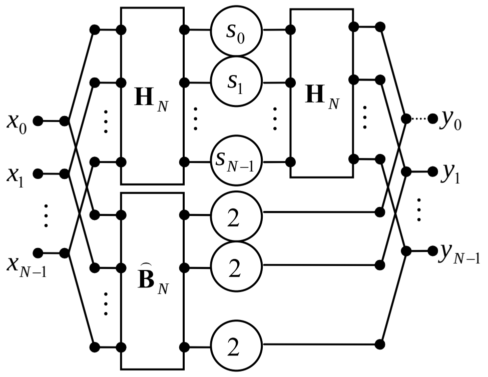

Fig.3. Data flow diagram for the unified rationalized hypernumber multiplication procedure (7).

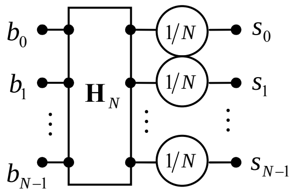

Fig.4. The data flow diagram describing the process of calculating elements of the matrix $\mathbf{S}_{N\times 1}$ in accordance with the procedure (8)

Discussion of computational complexity

The described approach to the computation of the hypercomplex number product requires $[N(N-1)/2]+2$ multiplications and $3N\log_2 N + [N(N-3)+4]/2$ additions of real numbers. Compared to the schoolbook way of computing, it gives more than 50% reduction in the multiplicative complexity. Table 1 and Table 2 show the numbers of multiplications and additions of real numbers for the two compared methods: for the schoolbook method in accordance with the formula (1) and for the proposed approach realization in accordance with the procedure (7). In these tables $\mathrm{O}_1(\times)$, $\mathrm{O}_2(\times)$ - are the amounts of multiplications in the naïve and proposed method, respectively, $\mathrm{O}_1(+)$, $\mathrm{O}_2(+)$ - are amounts of additions in the naïve and proposed method, respectively, O_1, O_2 - are the values of arithmetical complexities for the naïve method and proposed procedure, respectively. We introduce the concept of computational gains in order to be able to evaluate the merits of the solutions discussed.

$$K_1(\times) = \frac{\mathrm{O}_1(\times)}{\mathrm{O}_2(\times)},\ K_1(+) = \frac{O_1(+)}{O_2(+)} \text{ and } K_1 = \frac{\mathrm{O}_1}{\mathrm{O}_2}.$$

Table 3 shows the values of “multiplicative gain” ($K_1(\times)$), “additive gain” ($K_1(+)$) and “full computational gain” (K_1) respectively.

Table 1. Estimates of the hypernumbers multiplication multiplicative complexity for the direct method and proposed approach

No	Number type	O₁(×)	O₂(×)
1	quaternions	16	8
2	octonions	64	30
3	sedenions	256	122
4	trigintadounions	1024	498
5	sexagintaquattuornions	4096	2018
6	centumduodetrigintanions	16384	8130
7	ducentiquinquagintasexions	65536	32642

Table 2. Estimates of the hypernumbers multiplication additive complexity for the direct method and proposed approach

No	Number type	O₁(+)	O₂(+)
1	quaternions	12	28
2	octonions	56	94
3	sedenions	240	298
4	trigintadounions	992	946
5	sexagintaquattuornions	4032	3106
6	centumduodetrigintanions	16256	10690
7	ducentiquinquagintasexions	65280	38529

Table 3. Estimates of the full computational complexity of hypernumbers multiplication for the direct method and proposed approach

No	Number type	O₁	O₂
1	quaternions	28	36
2	octonions	120	124
3	sedenions	496	420
4	trigintadounions	2016	1444
5	sexagintaquattuornions	8128	5124
6	centumduodetrigintanions	32640	18820
7	ducentiquinquagintasexions	130816	71171

Table 4. Estimates of the multiplicative, additive and full computational gain values for the hypernumbers multiplication operation in the direct method and proposed approach

No	Number type	$K_1(\times)$	$K_1(+)$	K_1
1	quaternions	2	0,43	0,8
2	octonions	2,13	0,6	0,96
3	sedenions	2,09	0,81	1,18
4	trigintadounions	2,05	1,04	1,4
5	sexagintaquattuornions	2,02	1,3	1,59
6	centumduodetrigintanions	2,02	1,52	1,73
7	ducentiquinquagintasexions	2,01	1,7	1,84

As can be seen, the developed approach has a lower multiplicative complexity. It should be noted that, in some applications, the matrix elements are constants [14]. In this case, the elements of diagonal matrix $\mathbf{D}_{2N}$ can be calculated and stored in the memory of the computing unit in advance. Then, the number of additions in the implementation of the proposed procedure is even more reduced. As a result, the number of multiplications is reduced by more than half, compared to the naive method of the multiplication of the hypernumbers. The number of additions for the small size hypernumbers is slightly larger than in the naive method, but for the larger sizes it is almost twice less. In this case, the number of real additions is $2N\log_2 N + [N(N-3)+4]/2$.

Let hypernumber $B = b_0 + \sum_{l=1}^{N-1} b_i e_i$ - be constant. This means that $\{b_0\}$ and $\{b_i\}$ - are the real constant numbers. For this case $O(+)$ and $O(\Sigma)$ are the values of "additive complexity" and "full arithmetical complexity" of hypernumbers multiplication operation, respectively. Let also $K_2(+) = \frac{O_1(+)}{O(+)}$ and $K_2 = \frac{O_1}{O(\Sigma)}$ be the values of "additive gain" and "full computation gain" (for this case), respectively. Table 5 and Table 6 show the changes of discussed parameters for the case, when one of the multiplied hypernumbers is constant.

Table 5. Estimates of the additive complexity of hypernumbers multiplication when one of the multiplied hypernumbers is constant

No	Number type	O(+)	O(Σ)
1	quaternions	20	28
2	octonions	70	100
3	sedenions	234	356
4	trigintadounions	786	1284
5	sexagintaquattuornions	2740	4758
6	centumduodetrigintanions	9794	17924
7	ducentiquinquagintasexions	36482	69124

Table 6. Estimates of "additive gain" and "full computation gain" for proposed hypernumbers multiplication method when one of the multiplied hypernumbers is constant

No	Number type	$K_2(+)$	K_2
1	quaternions	0,6	1
2	octonions	0,8	1,2
3	sedenions	1,03	1,39
4	trigintadounions	1,26	1,57
5	sexagintaquattuornions	1,47	1,71
6	centumduodetrigintanions	1,66	1,82
7	ducentiquinquagintasexions	1,78	1,89

Concluding remarks

We presented a unified approach to the development of the computationally effective algorithms for calculating the product of two hypernumbers. The use of such algorithms allows the reduction of the computational complexity of multiplications of hypernumbers, thus, reducing hardware complexity and leading to an effective architecture suitable for VLSI implementation. Additionally, we note that the total number of arithmetic operations in such algorithms is less than the total number of operations in the compared naïve algorithms. Therefore, the algorithms obtained using the proposed approach are better than the naive algorithms, even in terms of their software implementation on a general purpose computer. In conclusion, it should be noted that the proposed approach allows the construction of sufficiently well algorithms for the multiplication of hypernumbers with reduced computational complexity. In our previous work [15-17], we have applied the unified approach proposed here for the synthesis of fast algorithms for the multiplication of quaternions, octonions and sedenions. However, if the specific properties of the matrix are used, even more interesting solutions may be found [18-19]. We will try to develop the ideas raised here in our future publications, as far as possible.

REFERENCES

[1] Kantor, I.L. and Solodovnikov A. S. Hypercomplex numbers. An elementary introduction to algebras. Springer: New York, (1989).

[2] Chanyal B. C., Bisht P. S. and Negi O. P. S., Generalized Octonion Electrodynamics, *Int. J. Theor. Phys.*, 49(6), (2010), 1333-1343.

[3] Alfsmann D., Göckler H. G., Sangwine S. J. and Ell T. A. Hypercomplex Algebras in Digital Signal Processing: Benefits and Drawbacks (Tutorial). *Proc. EURASIP 15th European Signal Processing Conference (EUSIPCO 2007)*, Poznań, Poland, (2007), 1322-1326.

[4] Snopek K. M. The Study of Properties of *n*-D Analytic Signals in Complex and Hypercomplex Domains, *Radioengineering*, (2012), vol. 21, No. 2, 29-36.

[5] Bihan, N. L., Sangwine, S. J. Quaternion principal component analysis of color images. In: *IEEE International Conference on Image Processing (ICIP 2003)*. v.1., Barcelona, Spain, (2003). 809–812.

[6] Moxey C.E., Sangwine S. J., Ell T.A., Hypercomplex correlation techniques for vector images, *IEEE Trans. Signal Process.*, 2003. 51, 1941-1953

[7] Wang Hui; Wang Xiao-Hui; Zhou Yue; Yang Jie Color Texture Segmentation Using Quaternion-Gabor Filters, *Transaction on Image Processing, 2006 IEEE International Conference 8-11 Oct.* (2006), 745 – 748.

[8] Calderbank R., Das S., Al Dhahir N., Diggavi S., Construction And Analysis of A New Quaternionic Space-Time Code For 4 Transmit Antennas, *Commun. Inf. Syst.*, (2005), 5, 97-122.

[9] Malekian E., Zakerolhosseini A., Mashatan A., QTRU: Quaternionic Version of the NTRU Public-Key Cryptosystems, Int. J. Inf. Secur., 3, 29-42, 2011

[10] Makarov O. M. An algorithm for the multiplication of two quaternions, *Zh. Vychisl. Mat. Mat. Fiz.* 17(6) (1977) 1574–1575 (in Russian).

[11] Dimitrov V. S., Cooklev T.,V., Donevsky B.D., On the multiplication of reduced biquaternions and applications, *Inform. Process. Lett.* 43 (3) (1992) 161–164.

[12] Ţariov A., Algorytmiczne aspekty racjonalizacji obliczeń w cyfrowym przetwarzaniu sygnałów, Wydawnictwo Uczelniane ZUT, (2011).

[13] Steeb W.-H., Hardy Y., Matrix Calculus and Kronecker Product: A Practical Approach to Linear and Multilinear Algebra, World Scientific Publishing Company; 2 edition (2011).

[14] Ţariov A., Strategie racjonalizacji obliczeń przy wyznaczaniu iloczynów macierzowo-wektorowych. *Metody Informatyki Stosowanej*, n 1, (2008), 147- 158.

[15] Ţariova G., Ţariov A., Aspekty algorytmiczne redukcji liczby bloków mnożących w układzie do obliczania iloczynu dwóch kwaternionów, *Pomiary, Automatyka, Kontrola*, n 7, (2010), 668-690.

[16] Ţariov A., Ţariova G., Aspekty algorytmiczne organizacji układu procesorowego do mnożenia liczb Cayleya. *Elektronika-Konstrukcje, Technologie, Zastosowania*, n 11, (2010), 137-140

[17] Cariow A., Cariowa G., An algorithm for fast multiplication of sedenions, Information Processing Letters, 113 (2013). 324–331.

[18] Cariow A., Cariowa G., An algorithm for multiplication of Dirac numbers, *Journal of Theoretical and Applied Computer Science*, (2013), vol. 7, no. 4, 26-34.

[19] Cariow A., Cariowa G., An algorithm for fast multiplication of Pauli numbers, Advances in Applied Clifford Algebras, (2014). This article is Publisher with open access at Springerlink.com DOI 10.1007/s00006-014-0466-0.

***Authors**: prof. PhD, D.Sc. Aleksandr Cariow, PhD Galina Cariowa, Department of Computer Architectures and Telecommunications, Faculty of Computer Sciences, West Pomeranian University of Technology, Szczecin, ul. Żołnierska 51, 71-210 Szczecin, E-mail: atariov@wi.zut.edu.pl, gtariova@wi.zut.edu.pl*