Aleksandr CARIOW [1], Galina CARIOWA [1], Marina CHICHEVA [2,3]
[1] WEST POMIERANIAN UNIVERSITY OF TECHNOLOGY, 49 Żołnierska St., 71-210 Szczecin, Poland
[2] SAMARA UNIVERSITY, 34 Moskovskoye sh., 443086 Samara, Russia
[3] IPSI RAS – Branch of the FSRC "Crystallography and Photonics", 151, 443001 Samara, Russia

Hardware-Efficient Schemes of Quaternion Multiplying Units for 2D Discrete Quaternion Fourier Transform Processors

Abstract

In this paper, we offer and discuss three efficient structural solutions for the hardware-oriented implementation of discrete quaternion Fourier transform basic operations with reduced implementation complexities. The first solution – a scheme for calculating *sq* product, the second solution – a scheme for calculating *qt* product, and the third solution – a scheme for calculating *sqt* product, where *s* is a so-called *i* -quaternion, *t* is an *j* - quaternion, and *q* – is an usual quaternion. The direct multiplication of two usual quaternions requires 16 real multiplications (or two-operand multipliers in the case of fully parallel hardware implementation) and 12 real additions (or binary adders). At the same time, our solutions allow to design the computation units, which consume only 6 multipliers plus 6 two input adders for implementation of *sq* or *qt* basic operations and 9 binary multipliers plus 6 two-input adders and 4 four-input adders for implementation of *sqt* basic operation.

Keywords: discrete quaternion Fourier transform, fast algorithms, implementation complexity reduction, FPGA implementation.

1. Introduction

Two dimensional discrete Fourier transform (2D-DFT) have been widely used in image processing ever since the discovery of Fast Fourier transform (FFT) which made the computation of DFT feasible using a computer [1]. However, if we want to apply the classical 2D-FFT to color images, we must perform three separate 2D-FFTs. This is because every color image pixel has three values associated with it: the red, green, and blue components.

Until recently, it was not offered any discrete transform, which would perceive the each pixel of the color image as a whole. Sangwine and Ell defined a new transform, called the discrete quaternion Fourier transform (DQFT) which allows to process simultaneously of all color components of the image [2-4]. The idea of DQFT is based on representation of color image pixels via quaternions – four-dimensional hypercomplex numbers discovered by Hamilton in 1843. Today there are several ways to calculate the 2D-DQFT [5-13]. Another way for calculating the QDFT has been reported in [14]. While it produces the same result as the other approaches, it is more efficient because leads to reducing the computation complexity. During the DQFT implementation using the method proposed in [14], it is necessary to perform three types of the quaternion multiplication operation, namely: left-sided quaternion multiplication, right-sided quaternion multiplication and two-sided quaternion multiplication. What is more, in all three cases only one quaternion is a usual quaternion, and the rest quaternions are constant quaternions, i.e. quaternions, which coefficients are real constants. Next we propose hardware-effective schemes to implement these operations.

2. Statement of the problem

The typical operations of the related two dimensional forward and inverse discrete quaternion Fourier transform are [14]:
- left-sided quaternion multiplication *sq*,
- right-sided quaternion multiplication *qt*, and
- two-sided quaternion multiplication *sqt*.

Where $q = q_0 + q_1 i + q_2 j + q_3 k$ is a conventional quaternion, with three imaginary units $ij = k$, $ji = -k$, $i^2 = j^2 = k^2 = ijk = -1$ and four real variables $\{q_n\}$, $n = 0,1,2,3$; $s = \alpha + \beta i + 0j + 0k$, and $t = \gamma + 0i + \delta j + 0k$ are so-called $i-$quaternion and $j-$quaternion respectively [14], and α, β, γ, δ - are real constants.

During synthesis of the discussed schemes we use the fact that multiplication of two quaternions may be represented as vector-matrix product [15, 16]. The matrix that participates in the product calculating has unique structural properties that allow performing its advantageous factorization [17]. Namely this factorization leads to significant reducing of the computational complexity of quaternion multiplication. Furthermore, since *s* and *t* are truncated quaternions and are in fact complex numbers, which located in the different domains of complex space, the corresponding matrices are sparse. This leads to additional effect in minimization of the computational complexity. Finally, an additional effect can be achieved using the fact that the numbers α, β, γ, and δ are real constants and their products can be calculated and stored in memory in advance.

3. The schemes

Let $\mathbf{X}_{4\times1} = [q_0, q_1, q_2, q_3]^{\mathrm{T}}$ - be a column vector, that contains the all coefficients of quaternion q, and $\mathbf{Y}^{(1)}_{4\times1} = [y^{(1)}_0, y^{(1)}_1, y^{(1)}_2, y^{(1)}_3]^{\mathrm{T}}$ - be a column vector containing the elements of *sq* product.

The first scheme (for implementation *sq* - kernel) can be written with the help of following matrix-vector calculating procedure:

$$\mathbf{Y}^{(1)}_4 = \mathbf{P}^{(1)}_4 \mathbf{A}^{(1)}_{4\times6} \mathbf{D}^{(1)}_6 \mathbf{A}^{(1)}_{6\times4} \mathbf{X}_{4\times1} \tag{1}$$

where

$$\mathbf{A}^{(1)}_{6\times4} = \left[\begin{array}{cc:c:cc} 1 & -1 & & & \\ 1 & 0 & \mathbf{0}_{3\times2} & & \\ 0 & 1 & & & \\ \hdashline & & & 1 & -1 \\ \mathbf{0}_{3\times2} & & & 1 & 0 \\ & & & 0 & 1 \end{array}\right],$$

$$\mathbf{D}^{(1)}_6 = diag(\alpha, d_1, d_2, d_1, d_2, \alpha),\ d_1 = \alpha + \beta,\ d_2 = \alpha - \beta,$$

$$\mathbf{A}^{(1)}_{4\times6} = \left[\begin{array}{ccc:ccc} -1 & 1 & 0 & & & \\ 1 & 0 & 1 & \mathbf{0}_{2\times3} & & \\ \hdashline & & & -1 & 1 & 0 \\ \mathbf{0}_{2\times3} & & & 1 & 0 & 1 \end{array}\right],\ \mathbf{P}^{(1)}_4 = \left[\begin{array}{cc:cc} 0 & 1 & & \\ 1 & 0 & \mathbf{0}_2 & \\ \hdashline & & 0 & 1 \\ \mathbf{0}_2 & & 1 & 0 \end{array}\right].$$

Fig. 1 shows a data flow diagram of the proposed scheme for realization *sq* product kernel. In this paper, data flow diagrams are oriented from left to right. Straight lines in the figures denote the operations of data transfer. The circles in these figures show the operation of multiplication by a real number inscribed inside a circle. Points where lines converge denote summation a dotted lines indicate the sign-change operations. We use the usual lines without arrows on purpose, so as not to clutter the picture.

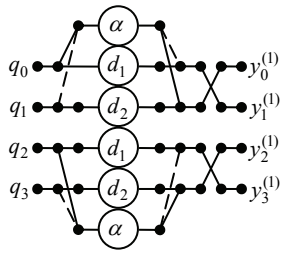

Fig. 1. The data flow diagram of proposed scheme for implementation *sq* product kernel

Let $\mathbf{Y}_{4\times1}^{(2)} = [y_0^{(2)}, y_1^{(2)}, y_2^{(2)}, y_3^{(2)}]^{\mathrm{T}}$ - be a column vector containing the elements of *qt* product.

Then the second scheme (for implementation *qt* - kernel) can be written with the help of following matrix-vector calculating procedure:

$$\mathbf{Y}_4^{(2)} = \mathbf{P}_4^{(2)}\mathbf{A}_{4\times6}^{(2)}\mathbf{D}_6^{(2)}\mathbf{A}_{6\times4}^{(2)}\mathbf{P}_4^{(2)}\mathbf{X}_{4\times1} \tag{2}$$

where

$$\mathbf{P}_4^{(2)} = \left[\begin{array}{cc|c} & & 1 \\ 1 & & \\ \hline & 1 & \\ & & 1 \end{array}\right], \quad \mathbf{A}_{6\times4}^{(2)} = \left[\begin{array}{cc|cc} 1 & 0 & & \\ 1 & 0 & \multicolumn{2}{c}{\mathbf{0}_{3\times2}} \\ -1 & 1 & & \\ \hline & & 1 & -1 \\ \multicolumn{2}{c|}{\mathbf{0}_{3\times2}} & 0 & 1 \\ & & 0 & 1 \end{array}\right],$$

$$\mathbf{A}_{4\times6}^{(2)} = \left[\begin{array}{ccc|ccc} 1 & 0 & 1 & \multicolumn{3}{c}{\mathbf{0}_{2\times3}} \\ 0 & 1 & 1 & & & \\ \hline \multicolumn{3}{c|}{\mathbf{0}_{2\times3}} & 1 & 0 & 1 \\ & & & 0 & 1 & 1 \end{array}\right], \quad \mathbf{P}_4^{(3)} = \left[\begin{array}{cc|c} & & 1 \\ & 1 & \\ \hline 1 & & \\ & & 1 \end{array}\right].$$

$$\mathbf{D}_6^{(1)} = diag(\alpha, d_1, d_2, d_1, d_2, \alpha)\,,\ g_1 = \gamma - \delta\,,\ g_2 = \gamma + \delta\,.$$

Fig. 2 shows a data flow diagram of the proposed scheme for implementation *qt* product kernel.

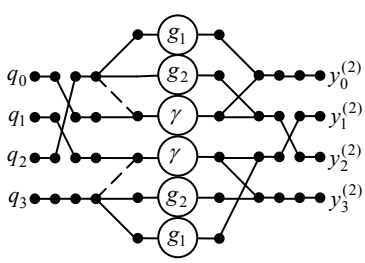

Fig. 2. The data flow diagram of proposed scheme for implementation *qt* product kernel

Let $\mathbf{Y}_{4\times1}^{(3)} = [y_0^{(3)}, y_1^{(3)}, y_2^{(3)}, y_3^{(3)}]^{\mathrm{T}}$ - be a column vector containing the elements of *sqt* product. Then the third scheme (for implementation *sqt* - kernel) can be written with the help of following matrix-vector calculating procedure:

$$\mathbf{Y}_{4\times1}^{(3)} = \mathbf{A}_{4\times9}\mathbf{D}_9\mathbf{W}_{9\times7}\mathbf{W}_7\mathbf{A}_{7\times4}\mathbf{P}_4\mathbf{X}_{4\times1} \tag{3}$$

$$\mathbf{A}_{7\times4} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & 1 & & \\ & 1 & -1 & \\ & & 1 & \\ & & 1 & \\ & & & 1 \end{bmatrix}, \quad \mathbf{W}_7 = \left[\begin{array}{c|c|c} \mathbf{H}_2 & \mathbf{0}_{2\times3} & \mathbf{0}_2 \\ \hline \mathbf{0}_{3\times2} & \mathbf{I}_3 & \mathbf{0}_{3\times2} \\ \hline \mathbf{0}_2 & \mathbf{0}_{2\times3} & \breve{\mathbf{H}}_2 \end{array}\right] =$$

$$= \begin{bmatrix} 1 & 1 & & & & & \\ 1 & -1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & -1 \\ & & & & & 1 & 1 \end{bmatrix},$$

$$\mathbf{W}_{9\times7} = \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & 1 & & & & & & & -1 \\ & & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ 1 & & & & & & -1 & & \\ & & & & & & 1 & & \\ & & & & & & & & 1 \end{bmatrix}, \quad \mathbf{P}_4 = \begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix}$$

$$\mathbf{D}_9 = diag(p_6, p_5, p_2, p_4, p_1, p_4, p_3, p_6, p_5)$$

where

$$p_1 = (\alpha - \beta)\,,\ p_2 = \alpha\delta\,,\ p_3 = \beta\delta\,,\ p_4 = (\alpha - \beta)(\gamma - \delta)\,,$$
$$p_5 = \alpha(\gamma - \delta)\,,\ p_6 = \alpha(\gamma - \delta)\,.$$

Fig. 3 shows a data flow diagram of the proposed scheme for implementation *qt* product kernel. The rectangles indicate the operations of multiplication by the matrices $\mathbf{H}_2$ and $\breve{\mathbf{H}}_2$.

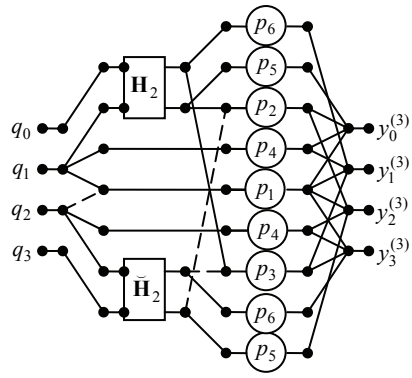

Fig. 3. The data flow diagram of proposed scheme for implementation *sqt* product kernel

4. Conclusions

The article presents three new hardware-efficient schemes for the execution *sq* - product, *qt* - product and *sqt* - product kernels with reduced computational complexities. To reduce the hardware complexity (number of embedded adders and multipliers), we exploit the specific structural properties of the matrix-vector products that represent mentioned basic operations. So, the fully parallel implementation of *sq* - product and *qt* - product kernels require only 6 multipliers by real numbers, and 6 adders. In turn, a fully parallel implementation of *sqt* - product kernel requires only 9 binary multipliers, 7 two-input adders and 4 four-input adders.

Reducing the number of multiplications is especially important in the design of specialized VLSI on-board DSP processors because minimizing the number of necessary multipliers also reduces the power dissipation and lowers the cost implementation of the entire system being implemented. This is because a hardware multiplier is more complicated unit than an adder and occupies much more chip area than the adder. (It is proved that the hardware complexity of an embedded multiplier grows quadratically with operand size, while the hardware complexity of an binary adder increases linearly with operand size). Even if the VLSI chip already contains embedded multipliers, their number is always limited. This means that if the implemented scheme has a large number of multiplications, the projected processor may not always fit into the chip and the problem of minimizing the number of multipliers remains relevant.

This problem becomes extremely challenging for applications requiring real-time processing at high throughput especially in digital signal and image processing. Hence, for meeting the high requirements to throughput and power-consumption constraints of real-time image processing systems, developing hardware-efficient schemes to implement them on the base of application specific integrated circuits (ASICs) or field-programmable gate arrays (FPGAs) is of paramount importance.

5. References

[1] Pratt W. K.: Digital Image Processing, Part III, Chapter 8, pp. 185-212 3rd Edition, John Wiley & Sons, Inc. 2001. ISBNs: 0-471-37407-5.

[2] Sangwine S. J.: Fourier transforms of color images using quaternion or hypercomplex numbers, Electronics Letters, Vol. 32, No 21, 10 Oct 1996), Page(s): 197 – 198 DOI: 10.1049/el:19961331.

[3] Sangwine S. J.: The discrete quaternion Fourier transform. 6th International Conference on Image Processing and its Applications. 1997, p. 790 – 793, DOI: 10.1049/cp:19971004.

[4] Ell T. A., Sangwine S. J.: Decomposition of 2D hypercomplex Fourier transforms into pairs of complex Fourier transforms. In: Moncef Gabbouj and Pauli Kuosmanen, editors, Proceedings of EUSIPCO 2000, Tenth European Signal Processing Conference, volume II, pages 1061–1064. Tampere, Finland, 5–8 September 2000. European Association for Signal Processing.

[5] Bülow T. and Sommer G.: Hypercomplex signals - a novel extension of the analytic signal to the multidimensional case. IEEE Trans. Sign. Proc., vol. SP-49, no. 11, pp. 2844–2852, Nov. 2001.

[6] Schütte H.-D. and Wenzel J.: Hypercomplex numbers in digital signal processing. In Proc. ISCAS '90, New Orleans, 1990, pp. 1557–1560.

[7] Alfsmann D.: On families of 2N-dimensional hypercomplex algebras suitable for digital signal processing. In Proc. European Signal Processing Conf. (EUSIPCO 2006), Florence, Italy, 2006.

[8] 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, pp. 1322-1326.

[9] Sangwine S. J., Bihan N. Le: Hypercomplex analytic signals: extension of the analytic signal concept to complex signals. Proc. EURASIP 15th European Signal Processing Conference (EUSIPCO 2007), Poznań, Poland, 2007, Poznań, pp. 621-624.

[10] Pei S.-Ch., Ding J.-J., Chang J.-H.: Efficient implementation of quaternion Fourier transform, convolution, and correlation by 2-D complex FFT, IEEE Transactions on Signal Processing, 2001, Vol. 49, No 11, pp. 2783-2797.

[11] Ell T. A., Sangwine S. J.: Hypercomplex Fourier Transforms of Color Images. IEEE Transactions on Image Processing, 2007, Vol. 16, No 1, pp. 22-35.

[12] Felsberg M. and Sommer G.: Optimized fast algorithms for the quaternionic Fourier transform. In: F. Solina and A. Leonardis, editors, Computer Analysis of Images and Patterns. Vol. 1689 of Lecture Notes in Computer Science, pp. 209–216. Springer, 1999. 8th International Conference CAIP'99, Ljubljana, September 1–3, 1999 Proceedings.

[13] Said S., Le Bihan N., Sangwine S. J.: Fast complexified quaternion Fourier transform. IEEE Transactions on Signal Processing 2008, Vol. 56, No 4, pp. 1522-1531.

[14] Chichyeva M. A., Pershina M. V.: On various schemes of 2D-DFT decomposition with data representation in the quaternion algebra. Image Processing and Communications, Institute of Telecommunications, Bydgoszcz, Poland, vol. 2, No 1, pp. 13-20, 1996.

[15] Ţariova G., Ţariov A.: Aspekty algorytmiczne redukcji liczby bloków mnożących w układzie do obliczania iloczynu dwóch kwaternionów. Pomiary Automatyka Kontrola, No 7, 2010, pp. 668-690.

[16] Parfieniuk M., Park S. Y.: Sparse-iteration 4D CODRIC Algorithms for multiplying quaternions. IEEE Transaction on Computers, 2016, v. 65, no 9, pp. 2859-2871.

[17] Cariow A.: Strategies for the Synthesis of Fast Algorithms for the Computation of the Matrix-vector Products. Journal of Signal Processing Theory and Applications, 2014, v. 3 No. 1 pp. 1-19.

Received: 24.03.2017 Paper reviewed Accepted: 03.05.2017

Prof. Aleksandr CARIOW, DSc.

He received the Candidate of Sciences (PhD) and Doctor of Sciences degree (DSc or Habilitation) in Computer Sciences from LITMO of St. Petersburg, Russia in 1984 and 2001, respectively. In September 1999, he joined the faculty of Computer Sciences and Information Technology at the West Pomeranian University of Technology, Szczecin, Poland, where he is currently a professor and chair of the Department of Computer Architectures and Telecommunications. His research interests include digital signal processing algorithms, VLSI architectures, data processing parallelization.

e-mail: acariow@wi.zut.edu.pl

Galina CARIOWA, PhD

She received the MSc degree in Mathematics from Moldavian State University, Chişinău in 1978, and PhD degree in computer science from West Pomeranian University of Technology, Szczecin, Poland in 2007. She is currently working as an assistant professor of the Department of Multimedia Systems. She is also an Associate-Editor of World Research Journal of Transactions on Algorithms. Her scientific interests include numerical linear algebra and digital signal processing algorithms, VLSI architectures, and data processing parallelization.

e-mail: gcariowa@wi.zut.edu.pl

Marina CHICHEVA, PhD

She received the MSc and PhD degrees in Computer Science from the Kuibyshev Aviation Institute (now Samara State Aerospace University) in 1987 and in 1998 respectively. She is currently working as assistant professor in the Samara University and as a senior researcher in the Image processing systems institute – branch of the Federal Scientific Research Centre "Crystallography and photonics" of Russian Academy of Sciences. Her scientific interests include image processing, data compression, and fast algorithms of discrete transforms.

e-mail: mchi@geosamara.ru