Distributed image processing based on the same IP-cores in FPGA-architecture

V M Zakharov[1], S V Shalagin[1] and B F Eminov[1]

[1]Kazan National Research Technical University named after A.N. Tupolev, Karl Marks, 10, Kazan, Russia, 420111

e-mail: sshalagin@mail.ru

Abstract. The problem of processing images, i. e., two-dimensional data arrays, was solved through implementing two-dimensional fast Fourier transform (FFT) when using single-type hardware modules – IP-cores in the Virtex-6 FPGA architecture. We have shown the possibility of the parallel implementation of each stage in the two-dimensional FFT, based on four "butterfly"-type transforms (BTr) over four elements of the data array being processed. Estimations were obtained regarding time- and hardware complexity of the IP-core implementing BTrs and used in implementing the one-dimensional FFT. The results obtained can be used in estimating hardware and time consumption when performing a two-dimensional FFT over an array of the pre-defined dimensionality in using existing and forthcoming distributed programmable-architecture systems.

CEUR-WS.org/Vol-2416/paper17.pdf

1. Introduction

The problem of real-time image processing is topical today. Software implementation of algorithms employing this problem on a general-purpose computer are limited by the features of the von Neumann architecture.

A way out of the current situation is using special-purpose computers, particularly those embedding the hardware accelerators, both ASIC and FPGA.

This paper discusses the distributed implementation of two-dimensional fast Fourier transform (FFT) based on single-type IP-cores in FPGA-architecture. Based on the estimates of hardware complexity and IP-cores functioning delay time, estimates of hardware and time complexity have been obtained regarding the execution of two-dimensional FFT for an image of a given dimensionality.

There is a known algorithm for the calculation of a two-dimensional FFT, based on the one-dimensional FFT procedure [1 – 4]. A weak point of that algorithm is the fact that it is executed in two stages. At the first stage, the patterns are computed for rows, while at the second stage for columns. Or vice versa, columns at the first stage and rows at the second one. In any case, until all operations have been performed for the first stage, one cannot go to executing the operations at the second stage. Algorithm proposed in this paper does not have this weak point: Operations are performed over the elements of a two-dimensional data array in parallel, without being divided into stages.

The results obtained can be used to solve the problems of the distributed real-time processing of images via the use of special-purpose graphical accelerators based on both existing and promising FPGAs.

Furthermore, in solving the distributed image processing problems, a promising trend is to apply to them distributed programmable-architecture systems [5], the elements of which architecture are FPGAs, such as in [6]. FPGAs allow organizing the distributed data processing at the level of binary data operations, which makes the above hardware platform match with the distributed implementation of a two-dimensional FFT.

We show this in the present paper.

2. Two-dimensional fast Fourier transform

Let us consider the two-dimensional discrete Fourier transform (DFT) for number array $\{x_{mn}\} = X$ sized $N \times N$, $N = 2^a$.

It can be represented as follows [1, 2]:

$$G_{uv} = N^{-2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x_{mn} \cdot W_N^{mu+nv}, \quad (1)$$

where $W_N^{mu+nv} = \exp\left(-2\pi j \frac{mu+nv}{N}\right)$, $u = \overline{0, N-1}$, $v = \overline{0, N-1}$, and $\{G_{uv}\}_{N \times N} = G$ is a pattern for X. Similarly, the reverse two-dimensional DFT is executed:

$$x_{mn} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} G_{uv} \cdot W_N^{mu+nv}, \quad m = \overline{0, N-1}, \quad n = \overline{0, N-1}.$$

Note 1. In computing a two-dimensional FFT, low frequencies will be concentrated in the corners of the above matrix, which is not very convenient for further processing the information obtained. To get a representation of the two-dimensional FFT, in which low frequencies would be concentrated within the center of the matrix, a simple procedure can be performed, which consists in multiplying the initial data array by the value of $(-1)^{m+n}$.

Figure 1 and 2 show the initial image and its Fourier-pattern computed according to (1), taking Note 1 into consideration.

The system indicated (1) can be represented similarly to one-dimensional FFT, as follows:

$$G_{uv} = N^{-2} \left[S_{2n_1, 2m_1} + W_N^u \cdot S_{2n_1+1, 2m_1} + W_N^v \cdot S_{2n_1, 2m_1+1} + W_N^{u+v} \cdot S_{2n_1+1, 2m_1+1} \right], \quad (2)$$

where $u = \overline{0, N-1}$, $v = \overline{0, N-1}$, and $S_{2n_1, 2m_1} = \sum_{n_1=0}^{N/2-1} \sum_{m_1=0}^{N/2-1} x_{2n_1 2m_1} \cdot W_N^{2(n_1 u + m_1 w)}$.

As a result, the two-dimensional FFT can be implemented based on single-type operations executed over the elements of matrices D sized $2^d \times 2^d$, submatrices $\{x_{mn}\}$, $m = \overline{0, N-1}$, $n = \overline{0, N-1}$, $d = \overline{1, a}$, $a = \log_2 N$:

$$D_{u,v} = \left[D_{u,v} + W_N^u \cdot D_{u+2^{d-1},v} + W_N^v \cdot D_{u,v+2^{d-1}} + W_N^{u+v} \cdot D_{u+2^{d-1},v+2^{d-1}} \right],$$

$$D_{u+2^{d-1},v} = \left[D_{u,v} - W_N^u \cdot D_{u+2^{d-1},v} + W_N^v \cdot D_{u,v+2^{d-1}} - W_N^{u+v} \cdot D_{u+2^{d-1},v+2^{d-1}} \right],$$

$$D_{u,v+2^{d-1}} = \left[D_{u,v} + W_N^u \cdot D_{u+2^{d-1},v} - W_N^v \cdot D_{u,v+2^{d-1}} - W_N^{u+v} \cdot D_{u+2^{d-1},v+2^{d-1}} \right],$$

$$D_{u+2^{d-1},v+2^{d-1}} = \left[D_{u,v} - W_N^u \cdot D_{u+2^{d-1},v} - W_N^v \cdot D_{u,v+2^{d-1}} + W_N^{u+v} \cdot D_{u+2^{d-1},v+2^{d-1}} \right], \quad (3)$$

where $u = \overline{0, 2^{d-1}-1}$ and $v = \overline{0, 2^{d-1}-1}$.

Figure 1. Initial image as number array *X*.

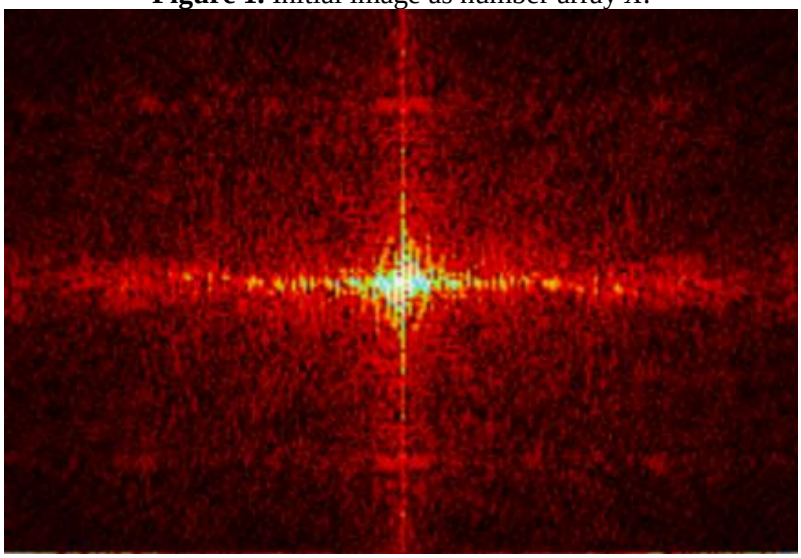

Figure 2. Fourier-pattern of the image shown in figure. 1 above.

Note 2. Number of matrices *D* sized $2^d \times 2^d$ was found to be 4^{a-d}, $d = \overline{1, a}$, $a = \log_2 N$.

Let us represent the system indicated (3) in matrix form as a complex of single-type "butterfly" transforms (BTr) used in computing the one-dimension FFT:

$$\begin{pmatrix} D_{u,v} \\ D_{u,v+2^{d-1}} \end{pmatrix} = \begin{pmatrix} 1 & W_N^v \\ 1 & -W_N^v \end{pmatrix} \cdot \begin{pmatrix} A_1 \\ B_1 \end{pmatrix},$$

$$\begin{pmatrix} D_{u+2^{d-1},v} \\ D_{u+2^{d-1},v+2^{d-1}} \end{pmatrix} = \begin{pmatrix} 1 & W_N^v \\ 1 & -W_N^v \end{pmatrix} \cdot \begin{pmatrix} A_2 \\ B_2 \end{pmatrix},$$

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} 1 & W_N^u \\ 1 & -W_N^u \end{pmatrix} \cdot \begin{pmatrix} D_{u,v} \\ D_{u+2^{d-1},v} \end{pmatrix},$$

$$\begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = \begin{pmatrix} 1 & W_N^u \\ 1 & -W_N^u \end{pmatrix} \cdot \begin{pmatrix} D_{u,v+2^{d-1}} \\ D_{u+2^{d-1},v+2^{d-1}} \end{pmatrix}. \quad (4)$$

As a result, to perform one operation indicated (3) over four elements of number array X, according to (4), four BTrs are required. The above operations must be performed over an array of N^2 elements $\log_2 N$ times. According to (3) and (4), the following statements hold true:

Statement 1. Implementing a two-dimensional FFT over number array X sized $N \times N$ requires executing $a \cdot N^2/4$ operations indicated (3) or $a \cdot N^2$ BTrs.

Statement 2. Single-type operations indicated (4) over number array X sized $N \times N$ can be performed in parallel.

Statements 1 and 2 substantiate the applicability of special-purpose computers implementing a single-type operation indicated (3) or (4).

IP-cores implementing BTrs act as a single-type calculator:

$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} 1 & W \\ 1 & -W \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}.$$

This operation is a basic one in performing an FFT, both one- and multi-dimensional. Therefore, the complexity of performing BTrs determines the complexity of implementing one- and multi-dimensional FFTs. In this study, we confine ourselves with considering the implementation of a two-dimensional FFT based on BTr.

3. Complexity of implementing the single-type butterfly transform on FPGAs

Let us consider the implementation of a hardware module implementing BTr (hereinafter, the "Module") as an IP-core in FPGA-architecture (figure 3). The Module allows handling 18-digit numbers. The functional logic diagram of the Module implemented in ISE Design Suite 14,7, a sheet-oriented CAD editor manufactured by Xilinx, Inc., includes:

- A unit multiplying complex numbers by a constant (ComplexNumMultiplier), shown in figure 4,
- Four adders, two of which perform additions and the resting ones perform subtractions in diminished complements, and
- Registers to ensure pipelined data processing.

An additional register bit serves for storing the sign bit.

Diagram of the ComplexNumMultiplier includes four multipliers (MULT18x18), two 16-bit adders, one of which implements subtractions, and six 16-bit registers to store intermediary results (figure 4).

The Module is implemented on FPGA XC6VLX240t-1FF1156 (Virtex-6 family manufactured by Xilinx, Inc.) that comprises *D*-triggers, generators of Boolean functions of six variables (LUTs), and input/output units (I/O units).

The following is involved in the Module implementation:

- 210 of 301,440 *D*-triggers available (less than 0.1 %);
- 199 of 150,720 LUTs (about 0.13 %), of which 161 were used to implement Boolean functions, while 38 ones as interconnections;
- 83 of 37,680 Slices (about 0.22 %) comprising four LUTs each, implementing Boolean functions of five or six variables and eight *D*-triggers; and
- 130 of 600 I/O units (about 21.7 %), of which 64 are allocated for inputs and 64 for outputs, and 2 more for clock signal and for resetting triggers, respectively.

The maximum function delay time of the Module is 10.438 ns. Values at outputs are computed three clock periods upon the relevant values having arrived at the input.

A structural diagram implementing a BTr is shown in figure 3. The pre-defined transform is implemented in a pipelined manner, within three clock periods, as follows.

At the zeroth clock period, two operands are fed to the Module input, namely complex numbers X_1 and X_2. Operand for multiplying, X_2, is represented by two numbers, eight bits each, while the operand for addition, X_1, is represented as two 16-bit numbers. The first number is for the real part and the second one for the complex part of the number.

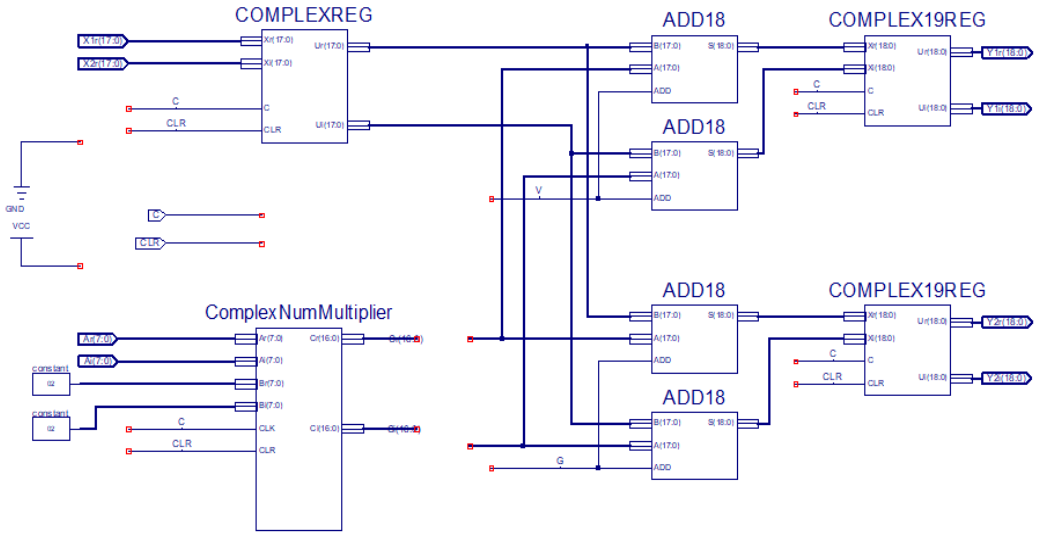

Figure 3. Single-type module diagram to perform a single-type "butterfly" transform.

At the first and second clock periods, the operation of multiplying complex numbers by the pre-defined constant, *W*, is performed with operand X_2, as well as operand X_1 is saved for the complex number addition operation.

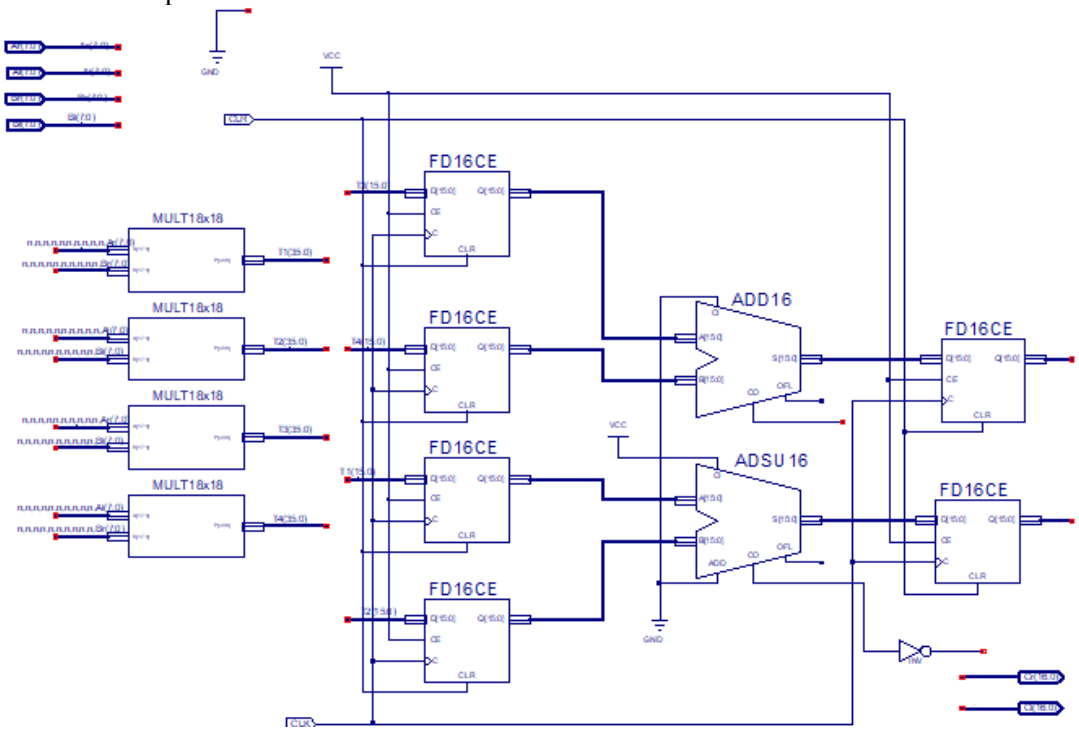

Figure 4. Diagram of multiplying a complex number by constant ComplexNumMultiplier.

At the third clock period, two addition operations are performed: That of operand X_1 and the product of multiplication operation, $W \cdot X_2$, and its values signed minus. Besides, at this stage, the results, Y_1 and Y_2, are saved in registers.

Eventually, the result of BTr is computed in a pipelined manner, within three clock periods, with a frequency that does not exceed 95.8 MHz. The slowest operation is the operation of multiplying complex numbers. Therefore, this operation is performed within two clock periods, intermediary results being saved by organizing the pipelined data processing (figure 4).

4. Discussion

According to [7, 8], a FPGA-based combinational circuit close to the optimal implementation requires involving at most 0.5 of the resources of each type, i.e., *D*-triggers, LUTs, Slices, and input/output units.

According to the complexity estimates obtained, no more than two Modules can be placed on one FPGA of the Virtex-6 family. Limiting factor is the number of I/O units.

$a \cdot N^2/4$ FPGAs of the Virtex-6 family are required to calculate a two-dimensional FFT for a number array sized $N \times N$. Moreover, to calculate the operation described according to (4), one FPGA of the above family is required, each of which accepts at the input and returns by four elements ($D_{u,v}$, $D_{u+2^{d-1},v}$, $D_{u,v+2^{d-1}}$, and $D_{u+2^{d-1},v+2^{d-1}}$), and implements four BTrs. Estimating the operation frequency of a distributed programmable-architecture system (DPAS) [5] that includes this number of FPGAs makes at least 95.8 MHz. Number of delay periods in computing the values of elements $D_{u,v}$, $D_{u+2^{d-1},v}$, $D_{u,v+2^{d-1}}$, and $D_{u+2^{d-1},v+2^{d-1}}$, according to (4), has been $6a$ since the elements to be processed arrived at the input. Generally, true is

Statement 3. To process an array sized $N \times N$ on a DPAS comprising *P* FPGAs that receive for processing *q* elements per clock period and implement a transform indicated (4) per *T* clock periods, *z* time units each, at least $\log N \cdot \left(\left\lceil N^2/(q \cdot P) \right\rceil + T - 1 \right) \cdot z$ time units are required.

For example, let us find the lower estimate of time required to process a number array sized $N \times N$, $N = 2^{10} = 1024$. We will perform the two-dimensional FFT in 10 stages, at each of which $N \times N = 2^{20}$ elements are required. Each Virtex-6 FPGA accepts four elements to be processed, i.e., $q = 4$. If there are *P* of the above-mentioned FPGA in DPAS, then the stage is computed within $\left\lceil N^2/(4 \cdot P) \right\rceil + 5$ clock periods. If we set the number of Virtex-6 FPGAs to 512, as in modern DPASes [5], the one stage of the two-dimensional FFT is implemented within 517 clock periods 10.438 μs each, while 10 stages within 5,170 clock periods, which makes about 5.40 μs to process one stage and about 54 μs to process the entire array. About 18.5 thous of arrays sized 1,024 by 1,024 can be processed within one second.

In case of processing an array sized 2,048 by 2,048 on a DPAS comprising 512 Virtex-6 FPGAs, the lower estimate of processing time, according to Statement 3, is 235 μs, while about 4,255 arrays of the above size can be processed within one second.

Let us compare the two-dimensional FFT implementation proposed, to its known implementation based on "one-dimensional" FFTs [1, 2] denoted as "FFT1," by hardware consumption, i.e., by the number of Virtex-6 FPGAs. In executing *N* FFT1s by the columns of a number array sized $N \times N$, $a \cdot N^2/2$ BTrs are required. The same number of BTrs are required to implement 1FFT to the strings of the number array. Thus, it is necessary to execute $a \cdot N^2$ BTrs, for which $a \cdot N^2/2$ Virtex-6 FPGAs are required, each of which implements two BTrs. As a result, executing a two-dimensional FFT based on the algorithm proposed requires two times fewer Virtex-6 FPGAs due to connecting the algorithmic data to be processed at the FPGA input to the output data according to (4), which allows implementing four BTrs instead of two within the logic resources of one FPGA of the said family, such as *D*-triggers, LUTs, and Slices.

Due to the parallel-serial input of number array *X* into the FPGA, the number of the IP-cores implementing the Module and configured on the Virtex-6 FPGA can be increased significantly.

For the Module implemented on FPGA XC6VLX240t-1FF1156, an additional 64-bit register must be allocated to store the elements of number array *X*. The limiting factor is still the number of Slices involved in implementing the Module. In this case, 227 IP-cores implementing the Module can be placed on the above FPGA. Thus, on one FPGA, $\lfloor 227/4 \rfloor = 56$ operations indicated (4) can be implemented. However, it will additionally require $224a$ clock periods to arrange the serial input of the *X* array elements into FPGA. According to Statement 3, if $P = 512$, $q = 4$, $T = 230$, and

$z = 10.438\ \mu s$, the lower estimate of the processing time of the number array sized $N \times N$, $N = 2^{10} = 1024$, is about 77.4 μs. Within one second, about 12.9 thousand of arrays sized 1,024 by 1,024 can be processed. In case of processing an array sized 2,048 by 2,048, the lower estimate of the operation time is 238 μs, and about 4,202 arrays of that size can be processed within one second.

Note 3. Function delay-time estimate for the operations represented as (4) and performed on a DPAS per a time unit is the lower estimate computed without considering the delay times of communication lines between the FPGA crystals within the DPAS.

5. Conclusion

Currently, much attention is paid to solving a wide variety of problems in using multiprocessor computer systems, the elements of which are the general-purpose processor elements [9-13]. At the same time, the matters of implementing distributed algorithms on DPAS, the elements of which are FPGAs, have been studied insufficiently. Particularly, this is true for the two-dimensional FFT algorithms widely used in image processing, including in the real-time mode. It should also be noted that DPASes are originally intended for the distributed implementation of various algorithms at different times. The present study fills this gap.

Based on estimating the time and hardware complexity of the Module as a single-type IP-core in the FPGA-architecture of the Virtex-6 family, we have evaluated the function delay time and the hardware complexity of a device implementing the pipelined computing of a two-dimensional FFT accompanied by time decimation on DPAS. Relevant estimates for the Module were executed using a special-purpose FPGA CAD, ISE Design Suite 14,7.

A set of single-type IP-cores implementing the two-dimensional FFT allows the implementation of the two-dimensional FFT when placing on the DPAS elements. The estimates of the two-dimensional FFT implementation time have been obtained, depending on the number of FPGAs included in a given DPAS, on the number of Modules implemented on a single FPGA, on the number of timed pulses, within which the Module implements the transform represented as (4), and on the duration of the said timed pulses.

The results obtained in this study allow us to estimate the potential implementation of a two-dimensional FFT on DPASes, both existing and promising ones.

6. References

[1] Gonzalez R C, Woods R E 2007 *Digital Image Processing* (Prentice-Hall) p 976
[2] Dudgeon D, Mersereau R and Merser R 1983 *Multidimensional Digital Signal Processing* (Prentice Hall) p 400
[3] Bailey D H 1988 A High-Performance FFT Algorithm for Vector Supercomputers *International Journal of Supercomputer Applications* **2(1)** 82-87
[4] Cooley J W and Tukey J 1965 An algorithm for the machine calculation of complex Fourier series *Math. Comput.* **19** 297-301
[5] Dordopulo A I, Kalyaev I A and Levin I I 2010 High-Performance Reconfigurable Computer Systems *Supercomputers* **3(3)** 44-48 (in Russian)
[6] Virtex-7. Highest Performance and Integration at 28 nm 2017 (Xilinx, Inc.) URL: https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html
[7] Shalagin S V 2004 Computer evaluation of a method for combinational-circuit synthesis in FPGAs *Russian Microelectronics* **33(1)** 46-54
[8] Shalagin S V 2016 *Implementing digital devices in FPGA architecture when using distributed computing in Galois fields* (Kazan, KNRTU-KAI Press) p 228 (in Russian)
[9] Raikhlin V A, Vershinin I S, Gibadullin R F and Pystogov S V 2013 Reliable Recognition of Masked Binary Matrices. Connection to Information Security in Map Systems *Lobachevskii Journal of Mathematics* **34(4)** 319-325
[10] Gibadullin R F, Vershinin I S and Minyazev R Sh 2018 Development of Load Balancer and Parallel Database Management Module *4nd International Conference on Industrial Engineering, Applications and Manufacturing*

[11] Gibadullin R F, Vershinin I S and Minyazev R Sh 2017 Realization of replication mechanism in PostgreSQL DBMS *International Conference on Industrial Engineering, Applications and Manufacturing*
[12] Raikhlin V A, Vershinin I S, Gibadullin R F and Pystogov S V 2016 Reliable Recognition of Masked Cartographic Scenes During Transmission over the Network *International Siberian Conference on Control and Communications (SIBCON)*
[13] Pavelyeva E A 2018 Image processing and analysis based on the use of phase information *Computer Optics* **42(6)** 1022-1034 DOI: 10.18287/2412-6179-2018-42-6-1022-1034

Acknowledgements
This work was supported by RFBR Grant 18-01-00120a, "Specialized devices for generating and processing data sets in the architecture of programmable logic devices class FPGA."
The authors also acknowledge L.Yu. Bakiev and I.A. Pesoshin, master's students at KNRTU-KAI, for their help in obtaining experimental data.