Bull. Nov. Comp. Center, Comp. Science, 28 (2008), 67–78

Some algorithms of image processing and their reflection onto multiprocessor systems

P. A. Kalinnikov, F. A. Murzin, T. A. Pletneva

Abstract. Problems related to reflection of some algorithms of image processing onto multiprocessor systems are investigated. In particular, algorithms of allocation of contours, such as the gradient method and the combinatorial method (the method of the threshold gradient), are considered. In view of complexity of the algorithm of searching objects on an image, we focus our attention on the parallel variant of only one stage of this algorithm. Namely, we investigate the stage of working with a search tree. Thereby we suppose that architecture containing a switchboard or the multiprocessor Cell developed by IBM Corporation is used in all cases. Data allocation, all data transfers, etc. are described. The formulas for acceleration coefficients are obtained.

1. Introduction

Intensive development of universal parallel computers and special multiprocessor systems gave rise to a number of new challenges. What computer architectures, algorithms and data structures should be used in different situations? How can the quality of algorithms be analyzed? What notions and estimates are useful? Which algorithms are suitable for various selected parallel computer architectures? How should parallel algorithms be programmed? What kinds of programming languages should be used?

In this paper, problems related to reflection of some algorithms of image processing onto multiprocessor systems are investigated. Thereby in all cases we suppose that the architecture containing a switchboard or the multiprocessor Cell developed by IBM Corporation is used.

Note that the architecture containing a switchboard is usually based on parallel memory systems. There are a lot of different parallel memory systems. For example, there exists a computer memory system intended for storing multidimensional arrays. Thereby, this memory system permits parallel access to the cuts distinguished in an array by fixing one of the coordinates and to the large set of parallelepipeds which are subarrays of the same dimension of the given arrays.

This memory may be applied to automatic systems for the analysis of dynamical images containing multiple moving objects.

The main functions of such systems are as follows: transformation of a luminous flux into the two-dimensional matrix of signals intended for further processing; objects discovery and estimation of their coordinates, directions

and velocities relative to the coordinate system of the gauge; tracking objects in a feedback regime; data output in a form convenient for a user. The peculiarity of these systems is the use of parallelism in all stages: data acquisition, storing and processing.

We investigate more simple situations, in particular, the algorithms of contour allocation, such as the gradient method and combinatorial method (the threshold gradient method). We also investigate the algorithm of searching objects on an image. In view of its complexity, we focus our attention on the parallel variant of only one of stages of this algorithm and only for architecture containing a switchboard. Namely, we investigate the stage of work with a search tree. The parallel variant of the algorithm of searching objects on images is based on placing a tree in modules of the two-dimensional memory, giving us the possibility of parallel access to some fragments of this tree. Thereby, the canonical embedding of a binary tree in two-dimensional array is used.

As a result, the formulas for acceleration coefficients are obtained for the gradient and combinatorial methods of contour allocation. In both cases, for the architecture containing a switchboard and for the multiprocessor Cell, acceleration coefficients are approximately equal to the number of processors. We repeat that, for the algorithm of searching objects on an image, we considered parallelization of one of its stages and only for the architecture containing a switchboard. For this case, we obtained the acceleration coefficient on the average, i.e. its mathematical expectation. Some experiments on the basis of IBM Eclipse 3.2 were conducted, allowing us to simulate work of IBM processor Cell.

2. Some image processing algorithms

2.1. Definitions and notations

The color image may be defined by three matrices $S = (S_R, S_G, S_B)$, $S_R = S_R(i,j)$, $S_G = S_G(i,j)$, $S_B = S_B(i,j)$, $0 \leqslant i \leqslant n-1$, $0 \leqslant j \leqslant m-1$. Usually the values of $S_R(i,j)$, $S_G(i,j)$, $S_B(i,j)$ change from 0 to 255. The set of triples $\{\ (r,g,b) : 0 \leqslant r,g,b < 255\ \}$ is called the color cube.

Consider two points of an image $p = (i,j)$ and $p' = (i',j')$ having the following colors (r,g,b) and (r',g',b'). It means that $S_R(i,j) = r$, $S_G(i,j) = g$, $S_B(i,j) = b$, $S_R(i',j') = r'$, $S_G(i',j') = g'$, $S_B(i',j') = b'$.

We will write for brevity $S_R(p)$ instead of $S_R(i,j)$, etc. Let us define the color distance between the points

$$cd(p,p') = \max\left\{\ |S_R(p) - S_R(p')|, |S_G(p) - S_G(p')|, |S_B(p) - S_B(p')|\right\}.$$

Euclidean metrics is denoted by $\rho(p,p') = \sqrt{(i-i')^2 + (j-j')^2}$. In time-sensitive situations, we usually use the square of Euclidean metrics, which is always an integer.

By scanning images and after some transformations (for example, after filtration), some colors may be a little changed. The identical colors become different, though these changes are small.

Therefore, in our further considerations and programs, we fix some constant, which is said to be the color constant and denoted by C_V. This constant characterizes which colors may be identified. More exactly, if $cd(p, p') \leqslant C_V$ then we can suppose that p and p' have the same color.

We denote by $B_n(p) = B_n(i, j)$ a square of size $n \times n$ with the center at a point $p = (i, j)$, where $n = 2k + 1$ is an odd number.

2.2. Combinatorial method of contour separation

Let p' and p'' be two points, $S(p') = (r', g', b')$, $S(p'') = (r'', g'', b'')$.

The color distance $cd(p', p'')$ between these points is defined as follows:

$$cd(p', p'') = \max\left\{ |r' - r''|, |g' - g''|, |b' - b''| \right\}.$$

Now let

$$cd[B_{m(p)}] = \max\left\{ cd(p', p'') : p', p'' \in B_m(p) \right\},$$

$$f(p) = \begin{cases} 0, & if\ cd[B_m(p)] \geqslant C_V, \\ 1, & if\ cd[B_m(p)] < C_V. \end{cases}$$

Resuming, we can say that, if the variation of colors is large, we will consider such points [1]. Otherwise, we delete the points from our consideration, and this fact is fixed with the help of function f.

If a circle was drawn on the initial image, then, using this algorithm to allocate the contour, we will see the same circle, but it will be a bit dim. Namely, the contour will be wider. And if the value of the color constant C_V will be less, then the contour will be wider.

2.3. Gradient method of contour separation

The gradient of the function of brightness is a vector determined as follows:

$$gradS = \left(\frac{\partial S}{\partial x}, \frac{\partial S}{\partial y} \right).$$

Partial derivatives at each point may be approximated, for example, with the help of the central difference

$$\left(\frac{\partial S}{\partial x}\right)_h = \left(\frac{1}{4}\left[S\left(i+1, j+1\right) + 2S\left(i, j+1\right) + S\left(i-1, j+1\right)\right] - \right.$$
$$\left. -\frac{1}{4}\left[S\left(i+1, j-1\right) + 2S\left(i, j-1\right) + S\left(i-1, j-1\right)\right]\right)/2$$

Note, that here it is written 2 instead of $2h$, i.e. h is omitted.

The partial derivative with respect to y is approximated in a similar way.

Thus, the discrete variant of gradient is defined at each point:

$$G(i,j) = grad_h S(i,j) = \left(\left(\frac{\partial S(i,j)}{\partial x} \right)_h, \left(\frac{\partial S(i,j)}{\partial y} \right)_h \right).$$

Further we choose a neighborhood D on the image and consider a pair of points $p = (i,j)$ and $p' = (i',j')$ from D. We will find the gradient of the function of brightness at both points and take the difference of these vectors. If the norm of the difference does not exceed some threshold Q for all pairs of points from D, then we suppose that the contour does not pass through these points [2], i.e. in this case it should be

$$\left\| G(i,j) - G\left(i',j'\right) \right\| = \left\| grad_h S(i,j) - grad_h S\left(i',j'\right) \right\| \leqslant Q.$$

Certainly, the norm can be defined in different ways. It is clear that the result depends on the choice of the norm.

3. Some remarks on architecture

3.1. Parallel memory organization

There are many different memory systems used in computer design. Some of them give us the possibility of parallel access to information. But as a rule, these memory systems are highly specialized and provide users with the limited possibilities.

In particular, some approaches to memory organization considered earlier in scientific literature were generalized in [3–7].

As a result, a memory system is described for storing an arbitrary sequence of multidimensional arrays. Certainly the total size of arrays should not exceed a given memory size. In this memory system, it is possible to implement parallel access to cuts separated in a given array by fixing one of the coordinates, as well as to a large set of parallelepipeds, which are subarrays of the same dimension.

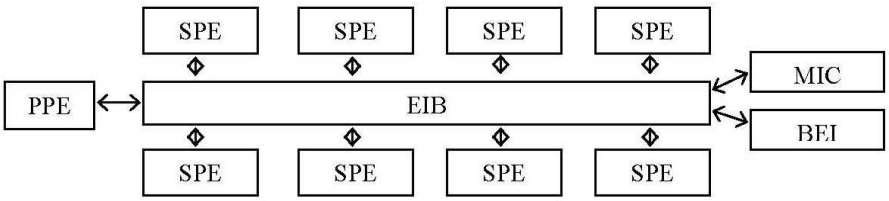
It is interesting and important to consider arrays rather than other data types, because in this case the arising problems have the purest (refined) form that facilitates their decision. Besides, this data type plays the main role in creating computers oriented to image processing and numerical computations.

3.2. IBM multiprocessor cell

The Cell Broadband Engine (IBM Multiprocessor Cell) is the result of collaboration between Sony, Toshiba, and IBM [8–10], known as STI, formally started in early 2001.

The Cell Broadband Engine is a single-chip multiprocessor with nine processors operating on a shared, coherent memory. In this respect, it extends current trends in PC and server processors. The most distinguishing feature

of the Cell Broadband Engine is that, although all processors share main storage (the effective-address space that includes main memory), their function is specialized into two types: the PowerPC Processor Element (PPE), and the Synergistic Processor Element (SPE). The Cell Broadband Engine has one PPE and eight SPEs. The structure of Cell is shown below.



There is Element Interconnect Bus (EIB). The PPE and SPEs communicate coherently with each other and with main storage and I/O through EIB. The memory-coherent EIB has two external interfaces:

1. Memory Interface Controller (MIC) provides the interface between EIB and the main storage.
2. 2. Cell Broadband Engine Interface (BEI)BEI manages data transfers between EIB and I/O devices.

4. Theoretical estimations of acceleration coefficients

At present, theoretical researches related to reflection of algorithms onto architecture containing a switchboard and onto the architecture of IBM processor Cell are conducted. Spadework on program emulation of some algorithms is also being done.

Some algorithms of image processing (more exactly, algorithms of contour allocation) were considered, such as: a gradient method, a combinatory method (the method of a threshold gradient), etc. Their reflection onto the multiprocessor Cell architecture is also described, i.e. allocation of data in memory, all data transfers, etc. The formulas for acceleration are obtained.

For example, in the case of the processor Cell, the estimation of an acceleration coefficient for the gradient method has the following form:

$$k = 8 - \frac{8}{1 + \frac{(\tau_{grad} + \tau_p)}{\tau_{transfer}}}.$$

Whereas the parameters τ_{grad} and τ_p always are much greater, than the parameter $\tau_{transfer}$, we obtain $\frac{(\tau_{grad}+\tau_p)}{\tau_{transfer}} >> 1$, and the value of $\frac{8}{1+\frac{(\tau_{grad}+\tau_p)}{\tau_{transfer}}}$

is small. Therefore, the acceleration coefficient $k \approx 8$, i.e. it is approximately equal to the number of processors.

For the combinatorial method, acceleration can be calculated in a similar way

$$k = \frac{T_{posled}}{T_{parallel}} = 8 - \frac{8}{1 + \frac{\tau_p}{\tau_{transfer} + \frac{1}{l}\tau_{connect}}} = 8 - \frac{8}{1 + \frac{\tau_p}{\tau_{transfer}}} \approx 8.$$

For the combinatorial method, in the case of architecture with a switchboard, the factor of acceleration is

$$k = \frac{T_{posled}}{T_{parallel}} = \\ = \frac{M_1 \cdot M_2 \cdot t_{calc}}{\frac{M_1 M_2}{n^2 N}\left(\tau_{transfer} + t_{calc} + (n-1)^2\left(2\tau_{transfer} + t'_{calc}\right)\right) + 3(N-1)\tau_{transfer}} > \\ > \frac{N t_{calc}}{\frac{1}{N}\left(1 + (n-1)^2\right)t_{calc} + \tau_{transfer}\left(\frac{1}{N}\left(1 + 2(n-1)^2\right) + 3\frac{(N-1)N}{M_1 M_2}\right)} = N - \alpha,$$

where α is small.

Therefore, similarly we have $k \approx N$, where N is the number of processors in the system.

5. Searching objects on images

Another problem which is partly considered is the problem of searching objects on images. It is an important problem that arises in different forms in many areas. For example, satellite object recognition, aerial photography and terrain conjunction. Photoimages can be stored in an archive and then some algorithm can be employed for search in the archive. It can be photos of a crystalline structure of a metal or biological data obtained with the help of microscope, etc. On a production line, the algorithm can select the components and work as machine vision for automated assembling machines. Thus, it can be used in complicated robotic systems.

A lot of specific constructions and concepts appear in this problem: support points; the search trees associated with images and optimization of algorithms processing them; characteristic functions of blocks; coarse palettes; metric characteristics and their use in optimization of search [1].

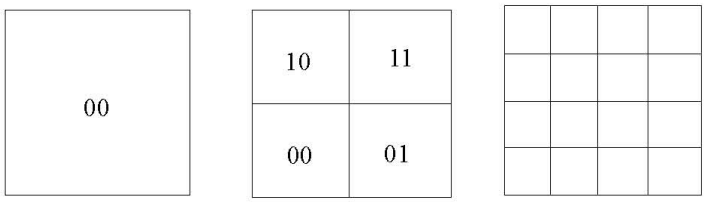
For the algorithm of searching objects on an image, a parallel variant of one of its stages, work with the search tree, is considered. Memory with parallel access to data for storage of characteristic functions and the switchboard are used.

First we describe a structure associated with a given image. It will be shown how it may be used for searching the support points.

We spend some time to build the structure. The time is proportional to the size of image$n \times m$. But after this procedure, we obtain the possibility

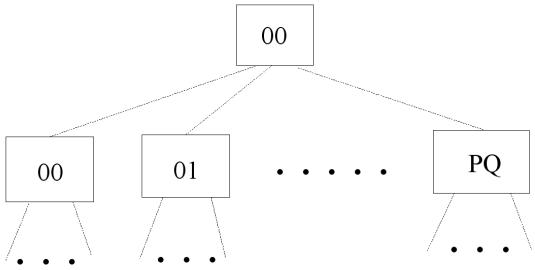
to find the support points in the logarithmic time, i.e. in $C \cdot \log(n \times m)$ steps.

Suppose we have a sequence of partitions of the given image.



Every next partition is finer than the previous one. At the first step, there is no partition. The finest structure has rectangles consisting of one pixel. Some special structure will be associated with every partition.

The previous (more coarse) structure in this sequence will be easily constructed from the current one (more fine). Pairs of integers may enumerate rectangles of the given partition. Note that all these partitions may be represented in the form of a tree.



If (i, j) is a pair used for enumerating, then we denote by $R_{r,s}(i, j)$ the corresponding rectangle of size $r \times s$.

Now let us describe the method of constructing the coarse palettes. The values r, g, b for any point are represented by 8 bits each. Let $0 < C_r \leqslant 8$, $0 < C_g \leqslant 8$, $0 < C_b \leqslant 8$.

We define $\bar{r}$ as follows:

1. leading C_r digits of $\bar{r}$ are equal to the corresponding digits of r,
2. lower $8 - C_r$ digits of $\bar{r}$ are equal to 0.

The values of $\bar{g}$, $\bar{b}$ are defined analogously.

In general, such a palette contains $2^{C_r} \times 2^{C_g} \times 2^{C_b} = 2^{C_r+C_g+C_b}$ colors. Let us denote the set of all these colors by $Pal(C_r, C_g, C_b)$. Also, let us suppose that some coarse palette is fixed.

Further, the characteristic function $\chi_{k,ij}$ associated with $R_{r,s}(i,j)$ defined on the palette is considered.

More exactly, this function depends on a node of a tree. The node is characterized by the triple (k,i,j), where k is the level in the tree, and (i,j) is the pair corresponding to the given rectangle.

The arguments of χ_{kij} are the triple $(\hat{r},\hat{g},\hat{b})$ of colors of the coarse palette. The function χ_{kij} shows whether there exists a point having a color (r,g,b) in the rectangle $R_{r,s}(i,j)$ on the level k, which has color close to $(\hat{r},\hat{g},\hat{b})$.

Let $(\hat{r},\hat{g},\hat{b}) \in Pal(C_r, C_g, C_b)$. We will define $\chi_{kij}(\hat{r},\hat{g},\hat{b})$ as follows:

1. $\chi_{kij}(\hat{r},\hat{g},\hat{b}) = 1$ if there exists a point $p \in R_{r,s}(i,j)$ such that $S(p) = (r,g,b)$ and for some r',g',b' such that $|r-r'| \leqslant C_V$, $|g-g'| \leqslant C_V$, $|b-b'| \leqslant C_V$ we have $\bar{r}' = \hat{r}$, $\bar{g}' = \hat{g}$, $\bar{b}' = \hat{b}$.
2. otherwise, $\chi_{kij}(\hat{r},\hat{g},\hat{b}) = 0$.

In fact, the set of characteristic functions $\chi = \{\chi_{kij} : 0 \leqslant i \leqslant \ldots,\ 0 \leqslant j \leqslant \ldots\}$ is the structure mentioned above.

The more coarse structure may be generated from the more fine structure with the help of disjunction (OR-operation). Thus we obtain the scale of structures $\chi_t, \chi_{t-1}, ..., \chi_0$. We numerate them from right to left.

The searching process of support points is executed with the help of the associated structure described above.

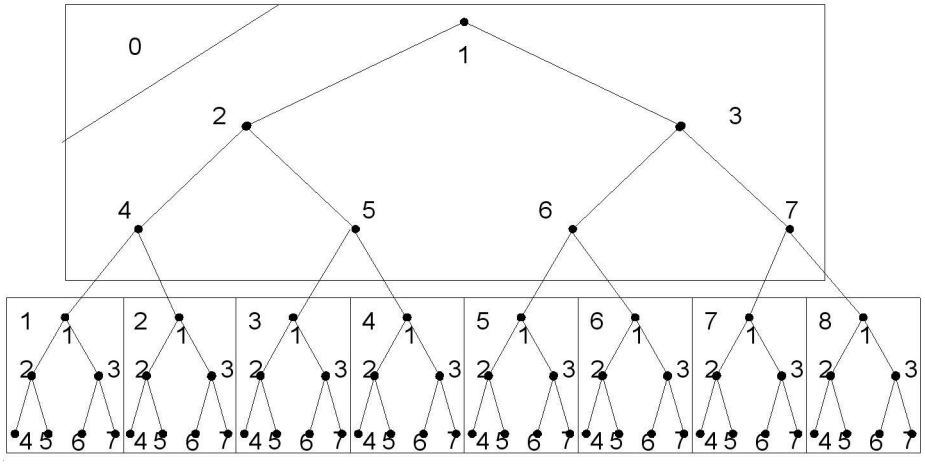
Step 1. We are checking if the points having the same colors as support points exist on Source Bitmap within the precision of the coarse palette. The most coarse structure (when there is no partition) allows us to do it immediately, because existence of the corresponding colors is coded in the structure.

Step 2. Let us consider a more fine partition and try to find the distribution of support points inside the corresponding rectangles, taking into account the distances between support points.

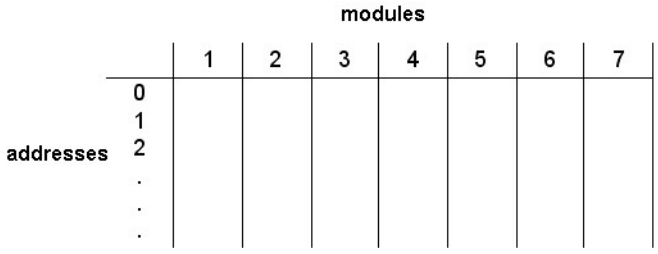
This may be done recursively. In principle, it means that we should pass a tree describing the embedding of rectangles into each other. Thus, using more and more fine structures, the support points will be fixed. Also, different metric characteristics can be used in algorithms of this form. We omit details.

The parallel variant of the algorithm of searching objects on images described above is based on placing the characteristic functions in modules of two-dimensional memory, which gives us the possibility of parallel access. For this purpose, the canonical embedding of a binary tree in a two-

dimensional array can be used [6-7]. An example of this embedding is shown below.

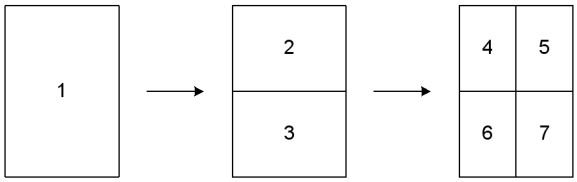


The correspondence between the vertexes of this tree and the cells of two-dimensional memory is rather simple.



In this example, the numbers of modules are shown in the tree vertexes, i.e. there are seven colors in the tree . The corresponding address is specified in the corner of any block. Further we suppose that data associated with the tree vertexes are placed in the corresponding cells of two-dimensional memory.

The image breaks into rectangles as it has been described above. But in this case, binary partition is more convenient.



Every block breaks into 2 blocks at every next step. The partition types are alternated: at the beginning, it is vertical, then it is horizontal, etc.

We can see that numbering of blocks corresponds to numbering of vertexes in a tree, i.e. numbering of memory modules. Further, blocks 4, 5, 6, and 7 can be considered as initial ones, and numbering renews. Certainly, we should remember the level on which the next renovation of numbering is carried out.

The characteristic function $\chi_\Delta(r, g, b)$ corresponds to the tree vertex Δ. We suppose that $\Delta = \langle l, n\rangle$, where l is the number of a block, i.e. an address, and n is the label of the vertex, i.e. the number of a module. Naturally, if $\Delta = \langle l, n\rangle$, then χ_Δ is located in a memory module with the number n and in a cell with the address l. It is convenient for parallel processing.

The parallel variant of searching the support points has a natural form.

The color of the support point $\langle r, g, b\rangle$ is transferred simultaneously to all processors. We consider the case, when the number of processors is equal to $N = 2^k - 1$. Then all processors simultaneously calculate $\chi_\Delta(r, g, b)$, $\Delta = \langle l, n\rangle$, where $l = 0$, $n = 1, \ldots, N$. Thus all processors can simultaneously define the presence of a support point with color $\langle r, g, b\rangle$ in the corresponding blocks, etc.

In this case we obtained an acceleration coefficient on the average, i.e. the mathematical expectation of an acceleration coefficient

$$k = \frac{2^k + k - 1}{2} = \frac{N + \log_2(N) + 1}{2}.$$

6. Conclusion

In all cases here considered, we suppose to use the architecture containing a switchboard or the multiprocessor Cell. As a result, the formulas for acceleration coefficients are obtained for the gradient and combinatorial methods of contour allocation. In both cases, acceleration coefficients are approximately equal to the number of processors.

For the algorithm of searching objects on an image, we considered parallelization of one of its stages. Memory with parallel access to data for storage of characteristic functions is used, i.e. the search tree is stored in parallel memory. Also, the acceleration coefficient for this stage is defined. Note

that this stage is more difficult than others. Therefore, even this partially parallel implementation considerably accelerates the problem decision.

Analogously for the multiprocessor Cell, the formulas for acceleration coefficients are obtained for the gradient and combinatorial methods. We obtain the same situation, i.e. the coefficient is approximately equal to the number of processors. There was made a conclusion that for effective implementation of the algorithm on this architecture, it is necessary to apply the methods, such that all processors perform almost identical operations in some allocated part of data, and data exchange between processors is reduced to minimum.

Installation of the environment for software development on the basis of IBM Eclipse 3.2 is performed, allowing us to simulate [8–10] work of the IBM processor Cell. Dependence of work of a simulator on the installed operating system (on the basis of Ubuntu Version 6.06 LTS and Fedora Core 6) is revealed and analyzed. The possibilities of creating programs in the C language for systems are tested on the basis of the IBM processor Cell, including various models of multiprocessing programming.

The methods of programming for systems on the basis of the IBM processor Cell are investigated with the purpose of further application of acquired experience to implementation of transformation algorithms for mathematical objects, schemes of movement computation, etc.

References

[1] Dunaev A.A., Lobiv I.V., Mekhontsev D.Yu., Murzin F.A., Polovinko O.N., Semich D.F., Chepel A.V., Yarkov K.A. Algorithms of fast search of fragments of photographic images // Modern problems of programs constructing, IIS SB RAS. – Novosibirsk, 2002. – P. 88–109. (In Russian)

[2] Pratt W. Digital Image Processing. – M.: Mir, 1982. – Vol. 1–2. – 738 p.

[3] Van Voorhis D.C., Morrin T.H. Memory Systems for Image Processing // IEEE Trans. on Computers. – 1978. – Vol. C27, N 2. – P. 113–125.

[4] Murzin F.A., Sluev V.A. A Memory Organization for Parallel Computers // New Generation Computing J. – 1988. – Vol. 6, N 1. – P. 3–18.

[5] Bratsev S.G., Murzin F.A., Nartov B.K., Puntus A.A. The conflict of complex systems. Models and control. – M.: Moscow Aviation Inst., 1995. – 118 p. (In Russian)

[6] Wijshoff H.A.G. Storing Trees into Parallel Memories // Parallel Computing 85, (Ed. Feilmeier). – Elsevier Science Pub., 1986. – P. 253–261.

[7] Das S.K., Sarkar F. Conflict-Free Data Access of Arrays and Trees in Parallel Memory Systems // Proc. of the Sixth IEEE Symp. on Parallel and Distributed Processing, Dallas, TX, Oct. 1994. – P. 377–384.

[8] Cell Broadband Engine Programming Handbook Version 1.0, April 2006. IBM Systems and Technology Group. – 878 p.

[9] Cell Broadband Engine Programming Tutorial Version 2.1, March 2007. IBM Systems and Technology Group. – 185 p.

[10] Cell Broadband Engine Software Development Kit 2.1 Installation Guide Version 2.1, March 2007. IBM Systems and Technology Group. – 46 p.