computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

AN MPI-BASED FRAMEWORK FOR PARALLEL PROCESSING OF INTEGRATED CIRCUITS LAYOUT IMAGES

Aleksej Otwagin, Alexander Doudkin [1)]

[1)] United Institute of Informatics Problem, 6, Surganova st., Minsk, 220012, Belarus,
emails: forlelik@yahoo.com, doudkin@newman.bas-net.by

Abstract: *We consider basic algorithms and processing technologies for integrated circuit layout images. The images represented as a set of frames can regard as a dataflow and the processing are perfectly suited for parallel implementation. We propose a framework architecture for designing parallel systems of image dataflow processing. The framework uses the algorithm of a virtual associative network for increasing processing speed and system throughput during runtime.*

Keywords: *parallel processing, image dataflow, application design, optimization framework, multi-agent architecture.*

1. INTRODUCTION

The modern semiconductor manufacturing needs to control all of the critical process modules that drive IC manufacturing success. An optical inspection is the important part of such control solutions. It implies the presence of some operative analysis system [1] providing image registration, visual information processing and analysis.

The video map of an integrated circuit (IC) layout (metallization, diffusion or polysilicon layers) is obtained with the help of special or standard input devices (scanners). Layout image is represented as a set of raster frames and consists from areas, the boundaries of which are rectangle, polygon, circle or ellipse. They are contact windows, pads, diffusion or metal wires and other items of the IC layout. The areas differ from each other by color and its intensity.

The set of frames can be considered as a dataflow. The type of IC layer defines a type of the frame. The frames can be repeated in dataflow in arbitrary order and the analyzing system must process this dataflow in minimal time.

An image dataflow can be deterministic, which means, that the amount of frames and their types are known before processing. Another case of processing observes a stochastic dataflow, when the types of arrived frames and their overall count are not known in advance. The task of processing this dataflow is more difficult then it is in case of deterministic dataflow. The solution of this task consists of runtime adaptation of processing system to dataflow characteristics.

The effective dataflow processing can be achieved only with use of modern software design technologies, especially parallel processing. However the problem of design of effective parallel architectures and applications is the fundamental obstacle to its wide use in many areas of computing.

A problem of design of parallel processing systems can be solved by use of design automation tools at various development stages. This approach is good applicable to deterministic dataflow processing task, because of its comprehensive definition.

For automation of processing of stochastic image flows the developer can use the technique of load balancing. This technique is based on decomposition of algorithm on separate modules, them solves the part of the problem. The modules are implemented in common programming languages (C, C++, Java) and realized as separate processes, which integrate and work inside a framework for collective interaction support. This approach is more powerful and provides great performance and code reuse. The separate image processing operations are defined as agents, which try to find the joint solution, satisfying some criteria. The agents plan and perform their interaction in such a way to achieve the minimal processing time.

As the planning itself is a very complicated problem, there is a necessity to create the planning methods and tools, which allow obtaining high efficiency of the problem solution with low expenses. There are some systems of computer vision and data processing, and most of these

systems use parallel and distributed processing [2, 3, 4, 5]. These systems use different techniques and architectures, for example CORBA [5] or agent-based architecture [2].

Our contribution consists in development of a framework, which is initially based on the MPI (Message Passing Interface standard) [6] for parallel computations. The framework uses a multi-agent application architecture. Another significant attribute of the framework consists in application of new hybrid algorithms with the purpose of computational optimization. The ability of online schedule optimization allows applications to achieve high speed and utilization of parallel processing system.

2. BASIC OPERATIONS

The technology of integrated circuit layout images processing includes the following image processing and image analysis algorithmic stages:

- Image registration
- Preprocessing and binarization
- Segmentation and vectorization
- Object analysis

Consider each stage in more detail.

Image registration

Two schemes of algorithms are used for quasi-optimal solution of the frames merging problem with the following restrictions: the frames are rectangular and have an identical scale. Three or four fragments matching are used instead of known algorithms to obtain good merging. In the first scheme local criterion is used to estimate a quads of frames located as a square matrix. In the second scheme the common criterion is used together with relative error of an outcome.

Preprocessing and binarization include the following operations:

1) The conversion of the entry color map in a gray scale image with 256 intensity levels.
2) The median filtering with the purpose of anti-aliasing the map. The median filtration is fulfilled with a cross-window that accepts values 3,5,7,9.... "Diffusion" is regulated by an amount of iterations. These parameters are accessible to the operator, however in most cases it is possible to use parameters set by default. Probably, the correction of these parameters will be indispensable at a rescaling of filming.
3) Correction of histogram to remove shadows along object boundaries.
4) Image smoothing with Gauss filter for image jitter – parameters are operator size and number of iteration
5) Filtration taking into account layer type and object size.

The stage of **Segmentation and vectorization** includes the following operations:

6) The threshold sharing and contour detection based on orthogonal transformation. The value of a threshold is selected automatically according to the histogram of allocation of intensities of the initial map, and the user can also adjust it.
7) The threshold sharing and contour detection based on cluster approach.
8) Morphological filtration for quality improving of segmented image by an elimination of blobs and an alignment of contour lines. This stage includes the following set of operations: extension, anabrosis and elimination noise that has not been deleted by two operations mentioned above. The first operation intends for the elimination of blobs on the detected objects. A collateral effect of this operation is the extension of objects; therefore it runs with the anabrosis operation function that is inverse function for the extension one. The erosion operation is intended for thinning of the objects after the extension operation. The third operation fulfils a rectification of boundaries of the objects. It realizes a search of beforehand detected objects by scanning all area of the map with the operator window, in which one the percentage of color, inhering to the object is determined.
9) Finding segments semantic descriptors and semantic filtration.
10) Construction of inner vector description and straight lines which approximation of contours with given accuracy.
11) Transformation of inner vector description into Source or GDSII formats.

Object analysis

12) Creation of the library of layout items.
13) Object identification
14) Design and training of classifiers/
15) Object recognition

All operations run in of a particular sequence (scenario), but the interaction with the operator is stipulated. The operator can choose the executable operation and adjust its parameter, execute sequence of operations and evaluate the quality of the results, change the initial sequence (add operations or change the order), i.e. to produce so-called hand-held tuning of the scenario.

Based on experimental researches two basic techniques were proposed as the most useful for IC images processing:

- the processing technique [7] for images with bimodal histograms of intensity based on operations 1 and 6.
- the processing technique [8] for images with multimodal histograms of intensity based on operation 7.

Within these two techniques there exist some scenarios that differ both operations and parameters according to layout type and image features.

All scenarios, produced by the operator on the sample image from one IC layer, must be repeatedly applied to full set of images of this IC layer. These images are entered into automated processing system, which is capable to process many images of different types simultaneously. The task of design automation consists in development of methods and tools for creation, simulation, analysis and synthesis of parallel processing systems.

3. A PROBLEM DESCRIPTION

The dataflow processing task assumes using of cluster of computers [9]. While most clusters are homogeneous in real world, we consider a case of a heterogeneous cluster that is more general. The heterogeneous cluster of computers is modeled as $P=\{p_1, p_2,..., p_m\}$, where p_i is an autonomous computer (also called node). Each computer p_i is weighted by w_i, which represents the time it takes to perform one unit of computation. The nodes in the heterogeneous cluster are connected by a high performance communication subsystem. Each communication link between computers p_i and p_j, denoted by l_{ij}, is weighted by s_{ij}, which models the time it takes to transfer one unit of message data between p_i and p_j.

The image dataflow is represented as a set of frames $J=\{j_1,\ldots,j_n\}$. Each frame must be processed by separate scenario based on a type of this frame. A processing system performs a set of image processing operations $O=\{o_1,\ldots,o_k\}, k>m$. The scenario for processing separate frame performs a subset of operations $O_j=\{o_{j_1},\ldots,o_{j_k}\}, \bigcup_{j=1}^{n} O_j = O$. The operations in each scenario have a precedence relation $o_{ja} \succ o_{jb}$, that means, that in the scenario for frame type j operation a performs before operation b. Each operation $o_k \in O$ is executed on a dedicated node of cluster P_i^k. Each data processing operation o_i is characterized by execution cost $w_{o_i}^j$, which represents the amount of computation units in operation for specified frame type j. Two operations for different frames, which are performed on some processor, cannot be executed in the same time, and two instances of one operation for different frames also must be executed in different times.

We represent all scenarios for data processing in the form of Directed Acyclic Graph (DAG). DAG is represented as a tuple $G=(V,E,W,C)$, where:

V is a set of graph vertices $v_i \in V, 1 \le i \le N$. Each vertex is associated with data processing operation from an operation set $O=\bigcup O_j$. A set of graph vertices represents decomposition of parallel dataflow processing program on the separated operations;

E is a set of graph edges $\{e_{i,j}=(v_i,v_j)\} \in E, i=\overline{1,N}, j=\overline{1,N}, i \ne j$. An edge represents a precedence relation between operations in scenario. Some edges are included in multiple scenarios;

W is an operation cost matrix $W=\bigcup W_{o_i}^j$;

C is an edge cost set, where $c_{i,j} \in C$ determines the communication volume between two data processing operations, which is transferred by edge $e_{i,j} \in E$. We consider those operations, which are related and connected by the edge, use an identical data format for a predecessor output and a successor input. For all scenarios, particular edge has an equal cost.

A design of parallel processing system for image dataflow processing consists of mapping of the scenario graph onto cluster topology. A parallel program is represented as a decomposition $((O),(P)) \longrightarrow \bigcup_{p \in P} (O_p)$, where $\forall O_k, O_p : O_k \ne O_p \supset O_k \bigcap O_p = \varnothing$. Each operation subset O_p is placed on selected processor node.

For effective data processing, this decomposition must be made in such a way, that the high processing speed and system throughput are achieved. Thus, a design automation system must create a schedule for processing of presented image dataflow, that ensures the goal of minimization of processing time

$$F = \min\{\max\ F_i\}, \qquad (1)$$

where F_i is a completion time of processing for frame i. For evaluation of the schedule the simulation model is used, which is equal to real world parallel computation process.

An example of scenarios and graph denotation for three types of data frames is presented in Fig.1, where each operation is denoted as *Oi* with some cost (on the top), each operation process data frames with types *T1, T2, ..., Tn* and performs processing with cost *C1, C2, ..., Cn*. The edges transfer frames of types *T1, T2, ..., Tn* with cost *C*. We assume that each pair of operations for all scenarios exchanges the same amount of data.

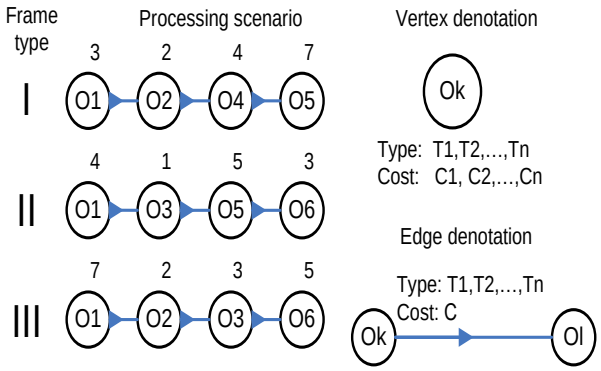

Fig. 1 - An example of scenarios and a graph denotation semantic.

The example of scenario graph for the data scenarios from Fig. 1 is presented in Fig. 2.

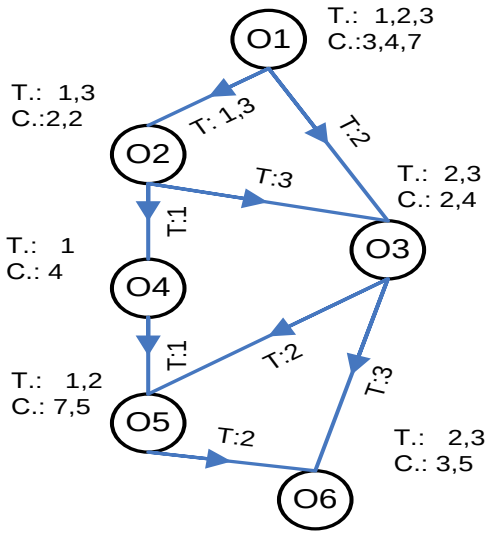

Fig. 2 - The annotated scenario graph.

There exist many algorithms of DAG scheduling that use various optimization techniques and heuristics. The techniques include priority based list scheduling, for example, algorithms HLF (Highest Level First), LP (Longest Path), CP (Critical Path) [10-12]. Another technique is clusterization, and such algorithms, as DSC (Dominating Sequence Clustering) [13], and Sarkar algorithm [14], belong to this technique. However all static scheduling algorithms are constructed for special graph topologies, or use special constraints, such as a zero communication time between nodes or an unbounded number of processors.

Another perspective search techniques use evolutionary optimization. These techniques are based on such algorithms, as a tabu search [15], simulated annealing, and genetic algorithms [16]. The most powerful is a genetic algorithm (GA) technique, and many of algorithms are proposed in this field. However, the classical genetic algorithm is a blind search technique. To speedup genetic algorithms we proposed an algorithm of virtual associative network [17-19], which belongs to a class of hybrid algorithms (also called memetic algorithms).

The algorithm of a virtual network is based on a concept of associations between the particular operations and dedicated processors. Each operation O and a processor P have associated with a virtual link with force $\omega_{O,P}$. The algorithm uses some kind of an associative memory for optimization, which consists of an association forces. This memory is learned by the experience, accumulated in a solution search process. The algorithm is based on a GA representation of solutions in a form of population of chromosomes. Each chromosome represents a variant of scenario graph decomposition.

Each chromosome is evaluated by fitness function, which is based on a simulation model and satisfies the criterion (1). After the stage of evaluation and selection of a best solution candidate, the virtual network is learned by the positive experience. When the selected solution for some stage doesn't outperform previous best model, the virtual network is learned by best previous experience. The learning procedure increases the association forces, which belongs to the best model.

The accumulation of experience allows the realization of a guided search in the solution space. This search is faster and gives better solutions at earliest stages of search. The size of population in the algorithm of the virtual associative network is smaller (5-10 chromosomes), than in classical GA (25-30 chromosomes), and requires less time for evaluation.

The virtual network algorithm introduces a new genetic operator – clusterization, which is performed with use of an experience from an associative memory. This operator allows a faster creation of stable schema in chromosomes, and thus an implementation of a genetic local search strategy. The clusterization operator means grouping of operations on processors with the strongest associations.

The solutions, created by the virtual network algorithm, must be realized as a parallel program. The development of parallel processing system is automated, and we propose the appropriate application development framework. Architecture of framework isolates the behavior logic that is dependent on the data processing schema, from the basic service code, that is common for all agents at modeling or application running.

4. THE FRAMEWORK ARCHITECTURE

The architecture of image dataflow processing framework is presented in Fig. 3.

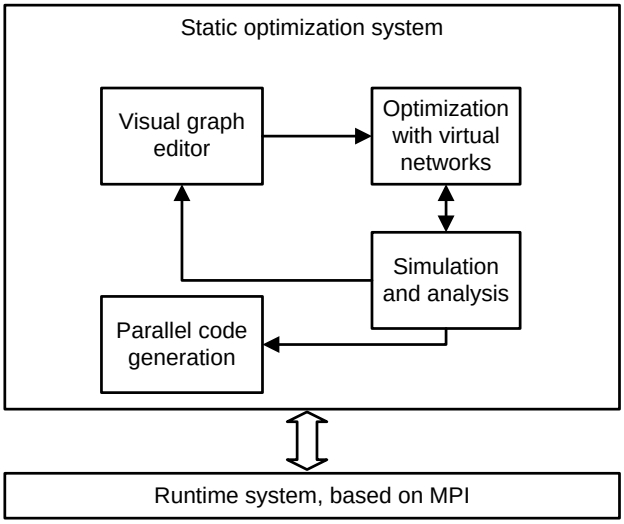

Fig. 3 - The architecture of dataflow processing framework.

The framework consists of two subsystems. The first one is a static optimization subsystem for analysis of dataflow schedules. The second one is a runtime subsystem, which is based on MPI.

The subsystem of static optimization of dataflow processing is used for development and analysis of parallel programs, which process a deterministic dataflow. The input for optimization subsystem is a scenario graph and a dataflow, represented as a vector of object types. The scenario graph is developed in a visual editor, which allows specifying all the characteristics of separate operations for all types of data frames.

After the scenario graph has created, it is optimized with use of a virtual networks algorithm. The algorithm performs a scenario graph decomposition, which is evaluated by simulation tool, to develop a schedule for processing the deterministic dataflow. The optimization algorithm and the simulation subsystem works together to achieve the best possible solution. The decomposition of the dataflow processing operations defines the schedule, which can be displayed as a Gantt chart. The developer can play with various decompositions by manual specification. The simulation subsystem can also obtain performance characteristics (processing time, throughput, mean flow time etc.)

The best decomposition is transformed to a program code for parallel processing. The code contains a switching mechanism, realized as a finite state machine, which determines operation transitions during data processing. The second mechanism, which is called an operation call wrapper, determines an operation that must be called for specified data type. These functions are generated from an annotated scenario DAG.

Each frame of an image flow is represented by a descriptor. The descriptor contains an identifier, type attributes, current operation identificator and some additional information, for example, name of image file, which contains frame information. When operation requires additional data for processing, this descriptor can be extended for specified application in appropriate way.

As the descriptors are transferred between processors of the parallel application, therefore the application code must contain serialization mechanisms. These mechanisms are realized with MPI facilities for registration, packing and unpacking custom data types. The code is included in header file, which is linked with the runtime subsystem.

The runtime subsystem for dataflow processing is designed as distributed multi-agent system [20]. The system consists of two types of program agents: a coordinator and a processor. All agents are realized as MPI processes and use MPI facilities for execution control and data exchange. The principles of system functioning allow to use it for processing both deterministic and stochastic image flows. The architecture of runtime subsystem is presented in Fig. 4.

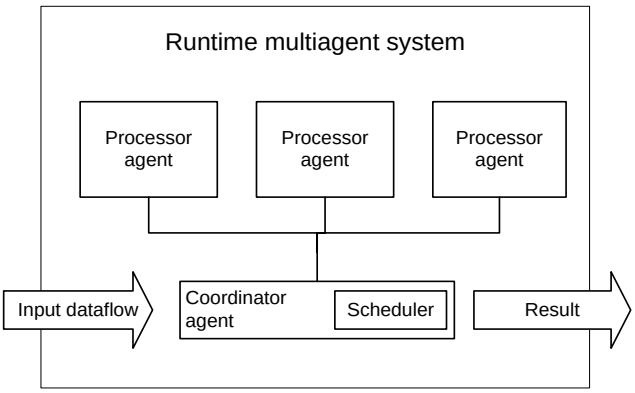

Fig. 4 - The architecture of runtime system.

The main function of the coordinator is a coordination of parallel dataflow processing. All descriptors of processed frames are stored in a frame pool. The scheduler chooses the next processed frame and the coordinator sends its descriptor to an appropriate processor agent. After processing, the coordinator receives descriptor, places it to the frame pool and changes information about next stage of processing. The process repeats while the frame pool is not empty.

The processor agent is linked with a library of image processing operations. Each processor executes a specified subset (cluster) of operations. Information about the operation set decomposition is

stored by the coordinator. All processors work according to the same algorithm. The processor receives the frame descriptor from the coordinator, determines the next operation and executes it using the descriptor data. After completion of data processing, the descriptor is returned to the coordinator. The processor works while stop instruction is not received from the coordinator.

Besides the process coordination, the runtime agents check system state and characteristics. These characteristics are collected in the coordinator and used for runtime optimization. The optimization is based on the measuring of data processing speed. When the dataflow changes its pattern significantly, the system must adapt to this situation. The adaptation performs reconfiguration of the operation subsets for all processors. When this reconfiguration is done, the coordinator applies any new scheme to transfer the descriptors. The system tries to adapt to changed conditions and to achieve a high processing speed.

5. A RESULTS OF EXPERIMENTS

For evaluating of proposed algorithms and a framework two series of experiments have been made. First, we studied the deterministic image flows, which had a fixed number of frames and the types of frames were known before processing. The second group of experiments was run with stochastic image flows, where the input data were generated randomly. All experiments have been done on the massively multiprocessor system K-500, developed by United Institute of Informatics Problems, Minsk, Belarus.

We used the scenario graph Fig.2. The experimental data flows were irregular and generated randomly. Table 1 shows the results of static optimization for deterministic flows and a comparison between classical GA and virtual network (VN) algorithms. In this table we show the relative values (in %), that characterize the improvement of processing time for VN algorithm.

Table 1. The improvement for static VN algorithm

Processors count	Data frames count		
	20	40	80
2	0.67	2.25	4.17
3	1.14	3.06	4.92
4	2.37	4.41	5.78

The results shows, that the algorithm of virtual networks finds better solutions and the performance of the algorithm is increased, when the search space is increased too. The algorithm based on virtual networks finds solutions faster, than classical GA and requires fewer computations.

In the second experiment with stochastic image flows we used a static schema for operation decomposition. This schema was compared with dynamic processing schema, which was controlled by VN algorithm. Table 2 shows the results of processing in relative values (in %) of improvement of processing time for VN algorithm.

Table 2. The improvement for dynamic VN algorithm

Processors count	Data frames count		
	50	100	200
2	1.67	3.56	6.19
3	2.34	4.47	7.27
4	4.03	7.34	8.81

The results show that the VN algorithm, which is embedded in the runtime processing system, can significantly improve the data processing in case of stochastic flows.

6. CONCLUSION

An adaptive optimization improves image dataflow processing and brings a new level of intellectual behavior into systems. On the other hand, the modern technologies of optimization allow the minimization of expenses for design and evaluation of such systems. The suggested approach and framework will find their place at creation of modern dataflow processing systems for industrial applications.

The architecture of framework, based on an algorithmic skeleton approach, is suitable for many applications, which are distributed and use a graph representation. This framework can be extended by new operation sets for developing applications for another distributed processing problem areas.

6. REFERENCES

[1]. M. Voganti, F. Ercal, C. Dagli, S. Tsunekawa. Automatic PCI Inspection Algorithms: A Survey, *Computer Vision and Image Understanding*, **63**, (1996), p. 287-313.

[2]. D. Argiro, S. Kubica, M. Young, and S. Jorgensen. Khoros: An integrated development environment for scientific computing and visualization. Whitepaper, Khoral Research, Inc., 1999.

[3]. M. Zikos, E. Kaldoudi, S. Orphanoudakis. DIPE: A Distributed Environment for Medical Image Processing. *Proceedings of MIE'97*, Porto Carras, Sithonia, Greece, May 25-29, 1997, pp. 465-469.

[4]. M. Guld, B. Wein, D. Keysers, C. Thies, M. Kohnen, H. Schubert, and T. Lehmann, "A distributed architecture for content-based image retrieval in medical applications," in *Proceedings of the 2nd International Workshop on Pattern Recognition in Information Systems*, pp. 299--314, 2002.

[5]. J. Wickel, P. Alvarado, P. Dörfler, T. Krüger, and K.-F. Kraiss. Axiom — a modular visual object retrieval system. In M. Jarke, J. Koehler, and G. Lakemeyer, editors, *KI 2002: Advances in Artificial Intelligence*, LNAI 2479. Springer, 2002, p. 253–267.

[6]. W. Gropp, E. Lusk, and A. Skjellum Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press, 1995.

[7]. A. A. Doudkin, A. V. Inyutin, M. E. Vatkin. Objects identification on the color layout images of the integrated circuit layers. *Proceedings of 3rd IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* 5-7 September 2005, Sofia, Bulgaria Sofia : IEEE, 2005, p. 610-614.

[8]. A. A. Doudkin, D. A. Vershok. Integrated circuit and photomask images processing technology. *J. AMSE*, 2005, p. 81-88.

[9]. K. Hwang, Z. Xu. Scalable Parallel Computing – Technology, Architecture, Programming. McGraw-Hill, USA, 1998.

[10]. B. S. Macey, A. Y. Zomaya. A performance evaluation of CP list scheduling heuristics for communication intensive task graphs. *In Proc. of IPPS/SPDP*, 1998, p. 538-541.

[11]. D. A. Menasce, D. Saha et al. Static and dynamic processor scheduling disciplines in heterogeneous parallel architecture. *Journal of Parallel and Distributed Computing*. Vol. 28, 1995. – pp. 1-18.

[12]. H. Oh, S. Ha. A Static Scheduling Heuristic for Heterogeneous Processors. *Second International EuroPar Conference Proceedings*, Vol II., Lyon, France, 1996, p. 573-577.

[13]. A. Gerasoulis, T. Yang. A comparison of clustering heuristics for scheduling directed acyclic graphs onto multiprocessors. *Journal of Parallel and Distributed Computing*, 4 (16), 1992, p. 276-291.

[14]. V. Sarkar. Partitioning and Scheduling Parallel Programs for Execution on Multiprocessors. The MIT Press. 1989.

[15]. A. S. Porto, A. C. Ribeiro. A Tabu Search Approach to Task Scheduling on Heterogeneous Processors under Precedence Constraints. *International Journal of High-Speed Computing*, 2 (7), 1995, p. 45-71.

[16]. Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Second, Extended Edition. Springer-Verlag. 1994.

[17]. Y. M. Yufik, T. B. Sheridan. Virtual Networks: New framework for operator modeling and interface optimization in complex supervisory control systems // A Rev. Control, vol. 20, p. 179-195.

[18]. R. Kh. Sadykhov, A.V. Otwagin. Solution search algorithm of solution search for systems of parallel processing based on a virtual neural network model. *Automatic Control and Computer Science*, vol. 35 (1), 2001, Allerton Press Inc., New York, p. 25-33.

[19]. R. Kh. Sadykhov, A. V. Otwagin. Algorithm for optimization of parallel computation on the basis of genetic algorithms and model of a virtual network. I*n Proceedings of the International Workshop on Discrete-Event System Design DESDes'01*, Przytok, Poland, June 27-29, 2001, p.121-126.

[20]. S. Poslad, P. Buckle, R. Hadingham. Open Source, Standards and Scaleable Agencies. *International Workshop on Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, June 03-07, 2000, Manchester, UK, p.296-303.

The research is partially supported by the Belarusian Republican Foundation of Fundamental Research, grant T04-219.

Aleksej Otwagin *is graduated in Computer Science in BSUIR at 1998, currently works as junior scientist in Laboratory of System Identification at UIIP. He is interested in parallel computing and multiagent systems for parallel processing.*

Alexander Doudkin *is a leader researcher at the laboratory of systems identification, United Institute of Informatics Problems (former Institute of Engineering Cybernetics). His research interests include methods and algorithm for design of digital integrated circuits, computer aided design of control units, digital signal and image processing, pattern recognition, architectures and models of computer vision systems, synthesis of high-performance processors, implementation of graph theory and scheduling methods in business and industry.*